UNIVERSITÀ DEGLI STUDI DI PISA
FACOLTÀ DI SCIENZE MATEMATICHE, FISICHE E NATURALI
CORSO DI LAUREA SPECIALISTICA IN INFORMATICA

Tesi di Laurea Specialistica

UNA INFRASTRUTTURA SERVER PER APPLICAZIONI WEBGIS SU DATI GEOGRAFICI ETEROGENEI E DISTRIBUITI

Candidato
Massei Giulio

Relatore
Ing. Luciano Fortunati

Contro Relatore
Prof. Franco Turini

ANNO ACCADEMICO 2004/2005

"Proteggimi da quel che non ho bisogno di sapere. Proteggimi anche dal sapere che bisognerebbe sapere cosa che non so. Proteggimi dal sapere che ho deciso di non sapere le cose che ho deciso di non sapere. Amen."

Adam Douglas

Indice

Glossario

Acronimi e Sigle

ASP : *Active Server Page*. Tecnologia per definire pagine web dinamiche.

BBOX : *Bounding Box*. Rettangolo di contenimento utilizzato per feature geografiche;

CAD : *Computer Aided Design*: con questo termine si intende generalmente uno strumento software che consente sia la progettazione che il disegno industriale.

CAT : *OGC Catalogue Service*. Servizio web del consorzio OpenGIS, utilizzato per realizzare un catalogo interrogabile al fine di individuare le istanze dei servizi OpenGIS.

CGI : *Common Gateway Interface*. Tecnologia che consente lo sviluppo di applicazioni lato server e la generazione di pagine dinamiche mediante l'uso di vari linguaggi di programmazione quali C/C++, PERL, PHP.

CGM : *Computer Graphic Metafile*. Formato grafico vettoriale utilizzato per illustrazioni tecniche in ambito industriale.

COM : *Component Object Module*. Protocollo sviluppato da Microsoft orientato agli oggetti per poter mettere in comunicazione varie parti di applicazioni

CORBA : *Common Object Request Broker Architecture*. Standard per consentire la comunicazione via rete fra componenti software, indipendentemente dalla piattaforma su cui si trovano e dalla tecnologia di implementazione.

DCOM : *Distribuited COM*. Insieme di interfacce sviluppate da Microsoft per consentire ad applicazioni client di invocare servizi ad applicazioni server attraverso la rete. DCOM si basa su COM.

EbXML : L'iniziativa ebXML fu concepita grazie alla diffusa necessità di consentire alle aziende di qualsiasi dimensione ed in qualsiasi posizione geografica di condurre delle transazioni elettroniche in modo semplice, affidabile e a buon mercato[1].

[1] Centro Informazioni Europeo di Informazione su ebXML: http://www.ebxml.eu.org/It/cosa_e'ebXML.htm

EDI : *Electronic Data Interchange*. Indica un modo di dialogare tra partners commerciali attraverso le reti di telecomunicazione.

EJB : *Enterprise Java Beans*. Si tratta di una componente architetturale lato server, per la piattaforma Java 2 Enterprise Edition (J2EE). La tecnologia EJB consente di sviluppare rapidamente e semplicemente applicazioni (basate sulla tecnologia Java) distribuite, transazionali, sicure e portabili[2].

FTP : *File Transfer Protocol*. Protocollo della pila protocollare di internet progettato appositamente per lo scambio di file attraverso la rete internet.

GIF : *Graphic Interchange Format*: Formato grafico bitmap compresso (basato sull'algoritmo di compressione *lzw*, di tipo conservativo) pensato per l'ambiente web.

GML : *Geographic Markup Language*. Dialetto XML per modellare dai geografici.

GWM : *GeoMedia WebMap*. Applicazione server commerciale, prodotta da Intergraph, per la manipolazione di dati geografici e la loro distribuzione attraverso la rete internet.

HTML : *HyperText Markup Language*. Linguaggio per la pubblicazione di ipertesti su web [W3C].

HTTP : *HyperText Transport Protocol*. Protocollo della pila protocollare di internet utilizzato per trasferire ipertesti su internet[3].

IETF : *Internet Engineer Task Force*. E' una grande Comunità internazionale aperta ai progettisti di reti, agli operatori, ai fornitori e ai ricercatori interessati allo sviluppo dell'architettura di Internet e al regolare funzionamento di Internet[4].

IIS : *Internet Information Server*. Software prodotto da Microsoft per la gestione di un *server* internet (*web server*).

JAVA RMI : *Java Remote Method Invocation*. Sinteticamente, permette ad un oggetto in esecuzione su una Java Virtual Machine (JVM) di invocare un metodo di un altro oggetto in esecuzione su di una seconda JVM attraverso la rete-

[2] Sun Developer Network, http://java.sun.com/products/ejb/
[3] W3C: http://www.w3.org/
[4] The Tao of IETF :http://www.ietf.org/tao.html#what

JPG : Formato per immagini raster, compresse attraverso l'algoritmo JPEG (*Joint Photographic Expert Group*).

JSP : *Java Server Page*. Equivalente di ASP ma basato su JAVA.

KVP : *Key Value Pair*. Modo di codificare coppie <nome, valore>, nella forma *nome=valore*, tipicamente impiegato in richieste HTTP-GET;

MBR : *Minimum Bounding Box*. Rettangolo che delimita l'estensione di un oggetto grafico.

MIME : *Multipurpose Internet Mail Extension.* Protocollo internet per estendere il protocollo SMTP in modo da consentire l'aggiunta, all'interno di e-mail, di contenuti multimediali quali immagini suoni, video od oggetti binari.

ODBC : *Open Database Connectivity*. Si tratta di API standard, indipendenti dal linguaggio di programmazione, per accedere a vari DBMS

OGC : *OpenGIS Consortium*. Un consorzio di aziende che ha come obiettivo la definizione di un insieme di raccomandazioni e specifiche tecniche per massimizzare lo sfruttamento di dati territoriali, in particolare per lo scambio di dati e per la realizzazione di servizi in ambiente distribuito.

PHP : *PHP: Hypertext Preprocessor.* Linguaggio di programmazione interpretato per lo sviluppo di applicazioni server e pagine Internet dinamiche.

PNG : *Portable Network Graphics*. Formato standard per immagini raster caratterizzato da un sistema di compressione dati senzaperdita di dati.

SMTP : *Simple Mail Transfer Protocol*. Protocollo del livello trasporto per e-mail.

SOA : *Service-Oriented Architecture*. E' un'architettura per consentire l'interoperabilità e l'integrazione fra agenti software eterogenei attraverso la rete.

SOAP : *Simple Object Access Protocol*. Protocollo basato su XML per lo scambio di informazioni fra applicazioni; è tipicamente utilizzato assieme ad HTTP per effettuare richieste per *Remote Procedure Call* (RPC).

SRS : *Spatial Reference System*. Indica un sistema di riferimento delle coordinate dei dati spaziali;

SVG : *Scalable Vector Graphics*. Linguaggio per descrivere grafica bidimensionale ed applicazioni grafiche in XML[5].

ISO/TC211 : E' responsabile per l'organizzazione ISO, di stabilire un insieme strutturato di standard legati alle informazioni relative ad oggetti o fenomeni che sono direttamente od indirettamente associati ad una locazione relativa alla Terra. Questi standard possono specificare, per le informazioni geografiche, metodi, strumenti e servizi per la gestione dei dati, acquisizione elaborazione, analisi, presentazione e trasferimento in formati digitali.[6]

TCP : *Transmission Control Protocol*. Protocollo dello strato "trasporto" della rete Internet, in grado di fornire alle applicazioni un servizio affidabile ed orientato alla connessione.

URL : *Uniform Resource Locator*. E' un sottoinsieme di URI (Universal Resource Identifier, [IETF rfc1630]) per identificare risorse attraverso i loro meccanismi primari di accesso[7].

W3C : *World Wide Web Consortium*. Si occupa di sviluppare tecnologie per l'interoperabilità (specifiche, linee guida, software, strumenti) al fine di portare il web al suo massimo potenziale[5].

WFS : *Web Feature Service*. Servizio remoto destinato alla distribuzione e manipolazione di feature class geografiche.

WMS : *Web Map Service*. Servizio remoto destinato alla distribuzione di mappe mediante documenti grafici raster e vettoriali.

WSDL : *Web Service Description Language.* Specifica utilizzata per **descrivere** un web service attraverso una grammatica XML

WS-I : *Web Service Interoperability Organization*. Organizzazione composta da aziende, il cui scopo è di promuovere l'interoperabilità dei web servici fra piattaforme, sistemi operativi e linguaggi di sviluppo[8].

XML : *eXtensible Markup Language*. Formato testuale largamente utilizzato; è impiegato ad esempio per lo scambio di dati[5].

---

[5]W3C: http://www.w3.org/
[6] ISO/Tc211 : http://www.isotc211.org/
[7] IETF: rfc239
[8] WS-I: http://www.ws-i.org/

XSD : *XML Schema*. E' una grammatica XML utilizzata per descrivere la struttura dei documenti XML; è il diretto successore di Document Type Definition (DTD).

Acronimi e sigle appartenenti al progetto:

MIR : *Meta Information Repository*.

SDES : *Start-up/Data-access/Execution/Shut-down*.

GO-WS : *Gis Operations - Web Service*.

LCS : *Local Catalogue Service*.

WCM : *Web Catalogue Manager.*

Terminologia e Notazione

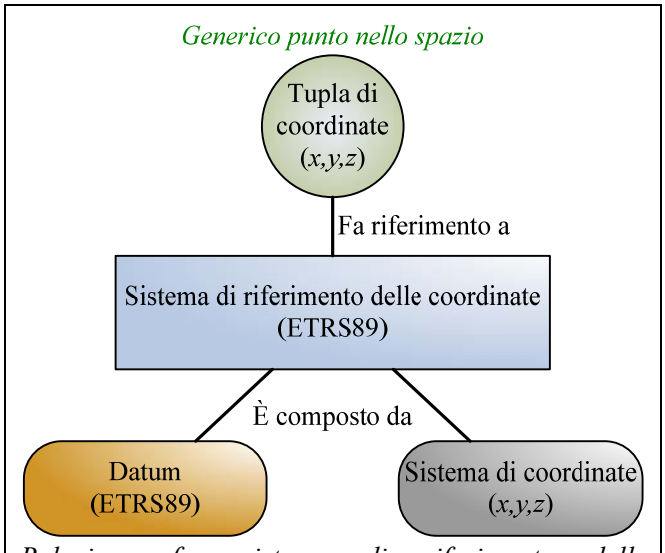
Terminologia

All'interno del documento sono utilizzati i seguenti termini:

- **Dataset**: insieme di dati strutturati.
- **Dati geografici**: definisce un insieme di dati che rappresentano gli oggetti fisici del territorio, composti da:
  - **componente spaziale (dati spaziali)**: primitive geometriche quali punto, linea e poligono.
  - **Componente alfanumerica (dati alfanumerici)**: attributi (caratteristiche) quantitativi e qualitativi associati ai dati spaziali.
- **Dati Territoriali**: sinonimo di *dati geografici*.
- **Datum**: descrive l'origine e l'orientamento del *sistema di coordinate*. Il Datum geodetico individua l'uso di uno specifico *ellissoide* orientato nel centro di emanazione secondo date condizioni. Ad esempio, ED50 (European Datum 50) definito mediante l'ellissoide di Heyford, centro di emanazione posto nelle vicinanze della città di Potsdam (Germania) e calcolato nel 1950.
- **Geographical Information System (GIS)**: "un potente insieme di strumenti in grado di acquisire, immagazzinare, recuperare, trasformare, analizzare e riprodurre dati spaziali riferiti al territorio"[9].
- **Geoide**: Date le irregolarità della superficie terrestre, esso è rappresentato mediante un geoide o sferoide terrestre; si tratta di una superficie normale in ogni punto alla direzione della forza di gravità.
- **Ellissoide**: Data la complessità di modellazione del *geoide* (ad esempio, per la rappresentazione di angoli e distanze), si approssima con un ellissoide. I punti del geoide sono riportati sull'ellissoide in dipendenza di un punto specifico, detto centro di emanazione (o punto di emanazione). che individua punto di tangenza tra il geoide ed ellissoide. Alcuni esempi di ellissoidi sono: Heyford (noto come ellissoide internazionale), WGS84 () e Bessel.

[9] P.A. Borrough, 1986.

- **EPSG4326**: sistema di riferimento geografico, definito dallo European Petroleum Survey Group, con *datum* WGS 84.
- **Layer**: insieme di dati geografici che rappresentano elementi geografici omogenei.
- **Mappa**: Rappresentazione grafica di un insieme di temi, ogni *tema* ha una propria posizione all'interno della gerarchia della mappa. La gerarchia dei *temi* stabilisce l'ordine di vista dei *temi*.
- **Recordset**: indica un insieme di tuple di attributi; possiede quindi la stessa semantica di *dataset*, ma è impiegato in contesti prettamente implementativi o di prodotti software. Tipicamente un *recordset* modella in memoria un sotto insieme di una tabella afferente ad una base di dati.
- **Sistema di coordinate**: Insieme di riferimenti, individuati da assi orientati, per individuare un punto nello spazio.
- **Sistema di riferimento**: è utilizzato come abbreviazione per la dicitura più esatta "Sistema di riferimento delle coordinate".
- **Sistema di riferimento delle coordinate (Coordinate Reference System, CRS)**: è un sistema di coordinate associato alla superficie terrestre attraverso un *datum.* Esistono sottotipi di CRS quali: geografico (basato sul modello ellissoidale della Terra con coordinate angolari) e proiettato (la superficie della Terra rappresentata su di unpiano).



Relazione fra sistema di riferimento delle coordinate, sistema di coordinate e datum.

- **Sistema di Riferimento Spaziale (Spatial Reference System, SRS)**: vedi *sistema di riferimento delle coordinate*.
- **Sorgente dati**: indica un'area di memorizzazione (locale o remota) in cui sono presenti *dataset*; ad un livello più concreto una sorgente dati identifica una base di dati. Una sorgente dati possiede un formato di codifica che caratterizza l'organizzazione dei dati (indici, tabelle di appoggio, …).
- **Sistema Informativo Territoriale (SIT)**: è spesso utilizzato come sinonimo di GIS, ma in realtà un SIT fa riferimento al territorio, a tutto ciò che afferisce al territorio indipendentemente dall'aspetto geografico. Si può dire che un GIS è fa parte sicuramente un SIT, ma non è vero il viceversa.
- **Strato Informativo**: sinonimo di *layer*.
- **Tema**: generalmente i termini *layer* e "tema" sono sinonimi; in questo lavoro di tesi il termine "tema" è impiegato per indicare la rappresentazione grafica di un *layer* di una *mappa*.
- **UTM**: Universal Trasversal Mercatore, sistema di riferimento proiettato in cui le coordinate dell'ellissoide sono riportate su di un piano. Si tratta di una proiezione cilindrica in cui gli angoli tra le direzioni spiccatisi dai singoli punti, risultano inalterati (proiezione conforme o ortomorfa).

Notazione

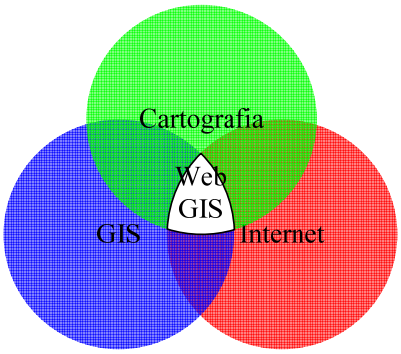
- I termini anglosassoni sono evidenziati in corsivo la prima volta che sono incontrati nel testo.
- Il carattere `Lucida Console` è utilizzato per evidenziare elementi quali:
  - namespace,
  - tag XML,
  - nomi di elementi software fuori dal contesto del codice del linguaggio di sviluppo.
- Il carattere `Courier New` evidenzia porzioni di codice del linguaggio di sviluppo.

Introduzione

Con il processo di consolidamento e diffusione della rete internet, avvenuto nell'arco di venti anni, si è avuto sia una crescente disponibilità di tecnologie quali ad esempio *web service*, sia della capacità di banda trasmissiva; questo ha portato ad un rapido incremento delle infrastrutture in grado di trarne vantaggio. Il mondo dei GIS ha visto nell'ambiente del *world wide web* un mezzo estremamente valido per l'ampia divulgazione di contenuti geografici.

Il connubio fra i GIS e l'ambiente world wide web ha portato alla definizione di una nuova disciplina, trasversale nell'ambito dei sistemi informativi territoriali (SIT): **WebGIS**.

La disciplina WebGIS si occupa non solo degli aspetti strettamente informatici legati alla distribuzione dei dati geografici, ma anche ai processi di preparazione ed elaborazione dei dati per poter essere rappresentati correttamente utilizzando le tecnologie di visualizzazione grafica legate al web (*Web Cartography*).



Relazione fra cartografia, ambiente Internet e sistemi GIS; l'intersezione fra cartografia ed Internet è denominata Web Cartography.

In ambito WebGIS si parla di "sistemi WebGIS" ed "applicazioni WebGIS".

Per sistema WebGIS si intende un GIS caratterizzato da un insieme di tecnologie che lo rendono in grado, attraverso la rete internet, di distribuire contenuti geografici (ad esempio sotto forma di mappe) e di rendere disponibili funzionalità GIS.

Un'applicazione WebGIS è invece una componente del sistema WebGIS e costituisce il mezzo con cui un'organizzazione da' la possibilità ad un utente remoto di accedere a funzionalità e dati geografici mediante un semplice software eseguito sull'elaboratore dell'utente.

Siamo quindi di fronte una classico paradigma *client-server* dove lato server è presente un'entità in grado di pubblicare dati geografici che l'entità client è può utilizzare.

Inizialmente i primi sistemi WebGIS erano progettati per visualizzare mappe in formato raster, tramite un *web browser*, con la possibilità di effettuare operazioni di ingrandimento (zoom) e spostamento dell'area visualizzata sulla mappa (pan). Tali operazioni erano effettuate solo sulla rappresentazione dei dati geografici e comportavano solamente l'invio della porzione di mappa relativa all'area di visualizzazione richiesta dal client.

Questi sistemi hanno poco a che vedere con la definizione di sistema WebGIS in quanto un WebGIS è un sistema GIS e quindi deve essere in grado di fornire, oltre ai dati geografici, operazioni per elaborarli.

Lato server è quindi necessaria la presenza di un'applicazione che possa accedere a dati geografici, manipolarli e restituirli all'entità client; tale applicazione è indicata con il termine **motore GIS**.

Esistono molti paradigmi di applicazioni WebGIS a seconda della "localizzazione" dell'esecuzione delle operazioni GIS (sul client, sul server oppure soluzioni *ibride* che prevedono la distribuzione dell'elaborazione su entrambi i lati) e l'insieme di funzionalità che il sistema fornisce.

Lato client è presente un'interfaccia utente per interagire sia con i dati inviati dal server, che con il server stesso. Tipicamente si hanno due tipi di applicazioni client:

- *Net-Enabled*: sono applicazioni GIS standard (ad esempio Win32) residenti sul sistema dell'utente (eseguiti od interpretati) in grado di manipolare dati provenienti da internet;
- *GIS-Online*: a differenza dei client Net-Enabled, l'applicazione è scritta usando un linguaggio di script ed è interpretata ed eseguita lato client dal *web browser* mentre, lato server, da processi specializzati per la pubblicazione di contenuti geografici ed HTML.

Un tema di primo piano, nell'ambito dei WebGIS, riguarda come i dati geografici possono venir restituiti sul client. Le possibili rappresentazioni sono mediante immagini (raster) codificate nei formati GIF, JPG e PNG oppure tramite formati vettoriali come SWF, PDF, WebCGM, SVG o soluzioni proprietarie basate su *applet* Java.

Ognuno di questi formati ha i propri pregi e difetti per quanto riguarda velocità di trasferimento, accuratezza rappresentativa e interattività; ultimamente i formati vettoriali si stanno diffondendo largamente in quanto permettono sofisticate manipolazioni dei dati geografici sul client.

Analizzando i sistemi WebGIS attualmente esistenti ed accessibili al pubblico, si nota che fondamentalmente rispetto ai primi sistemi sviluppati non sono stati fatti molti passi avanti in quanto manca la componente tipica di un GIS: l'**analisi geografica** (**topologica**).

Le operazioni di analisi geografica (in questo lavoro sono riferite per brevità come *operazioni GIS*) sono finalizzate alla gestione ed all'analisi delle relazioni spaziali esistenti fra gli elementi geografici in esame. Ne segue che, a differenza delle operazioni effettuabili

sulla rappresentazione grafica di una mappa, le operazioni di analisi topologica operano sui dataset geografici che definiscono la mappa. Il risultato di un'operazione GIS è anch'esso un dataset geografico contenente la rappresentazione della relazione spaziale, fra le entità geografiche, estratta mediante l'esecuzione dell'operazione.

Attualmente le applicazioni WebGIS disponibili offrono funzionalità di tipo *WebMap* in cui sono previste due categorie di operazioni di:

- *visualizzazione*: consente la rappresentazione di dati geografici in formato grafico (vettoriale o raster), aggiunta o rimozione di temi, modifica degli stili di visualizzazione dei temi;
- *navigazione*: permettono di interagire con la mappa, rappresentata sul client, al fine di consultarne i contenuti; ad esempio funzionalità di zoom e pan.

Alcune applicazioni di recente concezione permettono di eseguire interrogazioni sugli attributi dei dati geografici e limitare i dati ad una specifica area geografica di interesse, oppure fornire funzionalità per il calcolo di percorsi stradali.

Per quanto riguarda i dati, le applicazioni disponibili si basano prevalentemente su dati geografici codificati secondo formati proprietari, legati al motore GIS che li elabora. Manca quindi il vincolo di eterogeneità dei formati di codifica che limita l'**integrazione** dei dati all'interno dell'applicazione stessa.

Obiettivo

Il presente lavoro di tesi si pone in uno scenario in cui sono presenti server di dati geografici e sistemi WebGIS composti da un sottosistema server ed un sottosistema client, quest'ultimo realizzato mediante un'applicazione di tipo GIS-Online. I server di dati possono essere raggruppati in relazione all'interfaccia di accesso offerta: "server proprietari" (ad esempio ArcIMS, GeoMedia WebMap, MapInfo) e "server *OpenGIS*" (*Web Feature Services* e *Web Map Services*). Un sistema WebGIS deve consentire sia la visione, l'accesso e l'integrazione dei dati geografici resi disponibili dai due tipi di server; tali dati sono eterogenei in quanto caratterizzati da formato di codifica, sistema di riferimento e tipologia distinti. Perché siano consentite attività multi disciplinari, il sistema deve garantire l'interoperabilità sui dati geografici integrati; questo significa consentire l'esecuzione di operazioni di analisi topologica come: analisi di prossimità, *overlay*, calcolo di percorsi, interrogazioni sia sulla componente spaziale che sugli attributi dei dati geografici.

Per realizzare un sistema WebGIS con le caratteristiche sopra elencate, il lavoro di tesi è concentrato sulla progettazione e realizzazione di un'infrastruttura software *server side* in grado di porsi verticalmente al sottosistema lato server di una generica applicazione WebGIS. L'infrastruttura permette di **accedere** ed **integrare** dati geografici eterogenei provenienti da diverse tipologie di sorgenti dati, remote e locali.

Sui dati geografici integrati, l'infrastruttura garantisce l'**interoperabilità** funzionale e fornisce un insieme di operazioni tipiche dei sistemi GIS destinate all'uso in ambito di analisi geografiche.

Sommario dei Capitoli

Parte Prima

I capitoli contenuti in questa parte del documento, procedono secondo la filosofia *top-down*, in modo da introdurre il contesto in cui il progetto si vuol porre e conducono il lettore lungo il cammino che ha portato ad effettuare le scelte che caratterizzano il lavoro di tesi.

Capitolo 1 La Direttiva INSPIRE

In questo capitolo sono presentate le iniziative Europee inerenti le infrastrutture di dati spaziali (SDI) ed una sintesi dello stato dell'arte, all'interno sia della Comunità Europea che dell'Italia.

Capitolo 2 Standard E Tecnologie

Sono introdotti gli elementi principali di due filoni di tecnologie necessarie per poter intraprendere la strada dell'interoperabilità e conseguire l'obiettivo del lavoro di tesi: OpenGIS e Web Service Architecture.

Capitolo 3 Strumenti

Il capitolo è diviso in due parti principali, in cui sono discussi due elementi software: l'infrastruttura di sviluppo/esecuzione (framework .NET) e l'applicazione (GeoMedia WebMap). La trattazione ha lo scopo di fornirne una visione d'insieme in modo tale da poter comprendere le motivazioni che hanno portato a ritenere gli elementi software adeguati per implementare le tecnologie discusse nel capitolo precedente ed a raggiungere gli scopi del lavoro di tesi.

Parte Seconda

Sulla base del lavoro di revisione svolto nella sezione precedente, vengono affrontate le scelte di progetto effettuate per la realizzazione dell'infrastruttura software che questo lavoro ha per oggetto.

Capitolo 4 Architettura Della Infrastruttura

Viene data al lettore una visione d'insieme iniziale sull'infrastruttura e successivamente, sono presentate le scelte di progetto adottate per ogni elemento costituente l'infrastruttura.

Capitolo 5 Applicazione D'Esempio

Al fine di valutare l'uso dell'infrastruttura, sono illustrate due applicazioni web che fanno uso dei componenti sviluppati.

Capitolo 6 Conclusioni

Sono riportate sia le considerazioni sul lavoro svolto ed una valutazione sulle tecnologie utilizzate, sia gli sviluppi futuri.

Appendici

Le appendici trattano in modo dettagliato alcuni temi specifici sia della prima parte, sia gli elementi costituenti l'infrastruttura.

Capitolo 1

La Direttiva INSPIRE

1.1. Un'Infrastruttura per Informazioni Spaziali in Europa

Le informazioni geografiche hanno da sempre un elevato valore socio-politico in quanto necessarie al fine di poter attuare politiche di gestione del territorio sia da parte di enti pubblici che di aziende private. Fenomeni come terremoti, alluvioni, incidenti, non sono limitati dai confini amministrativi; un'analisi finalizzata alla pianificazione di un intervento che si limiti ai soli dati territoriali di tale area, risulterebbe lacunosa ed inefficace.

La politica ambientale Europea ha dato vita ad un ampio sistema di monitoraggio ambientale: il "Quinto programma di azione per l'ambiente" [1] (1992-1999). Il successivo piano, "Sesto programma di azione per l'ambiente" [2], sottolinea la necessità di fondare politiche ambientali al fine di rendere le informazioni facilmente disponibili e di migliorarne la qualità.

I dati territoriali rivestono un ruolo importante dato che permettono di integrare informazioni provenienti da varie discipline in modo da poterle utilizzare in diversi settori.

Una descrizione del territorio, coerente, accessibile e di qualità, permette la coordinazione nella fornitura di informazioni ed il monitoraggio in tutta la comunità Europea. Per questi motivi è stata presentata la proposta di direttiva INSPIRE (*INfrastructure for SPatial InfoRmation in Europe*) sia al Parlamento Europeo che al Consiglio dell'Unione Europea. La proposta mira a rendere disponibile **dati territoriali interoperabili** a sostegno delle politiche nazionali e comunitarie.

La proposta ha lo scopo di creare un piano giuridico per la realizzazione e l'attivazione, in Europa, di un'infrastruttura per l'informazione territoriale (ESDI, *European Spatial Data Infrastructure*) a sostegno sia delle amministrazioni pubbliche che del singolo cittadino.

La proposta INSPIRE segue i seguenti principi [3]:

- I dati devono essere raccolti e mantenuti ad un livello tale da consentire la massima efficacia.
- Deve essere possibile combinare in modo continuo informazioni territoriali provenienti da diverse fonti in Europa e condividerle tra più utilizzatori ed applicazioni.
- Deve essere possibile condividere con tutti gli altri livelli[10] le informazioni raccolte ad un dato livello.
- L'informazione geografica necessaria per un buon governo a tutti i livelli, deve essere abbondante, secondo condizioni che non ne limitino un ampio e diversificato uso.
- Deve essere facile l'individuazione delle informazioni geografiche disponibili per valutarne l'adeguatezza ad un uso specifico e conoscere quali condizioni applicare per il loro uso.
- I dati geografici devono essere di facile comprensione ed interpretazione, si devono poter visualizzare facilmente nel contesto adeguato.

Uno dei principali obiettivi dell'iniziativa INSPIRE è di rendere disponibile una quantità maggiore di dati di elevata qualità ai fini dell'elaborazione delle politiche comunitarie e della loro attuazione. Per questo motivo, la proposta mira all'ottimizzazione dello sfruttamento dei dati esistenti ed a dotarli di una valida documentazione (metadati) senza prevedere campagne di raccolta di nuovi dati territoriali.

L'interoperabilità è di rilevante importanza perché consente agli stati membri dell'UE di non apportare modifiche ai formati dei dati originali, conservando così i propri sistemi e schemi organizzativi; l'interoperabilità è raggiungibile solo grazie all'integrazione. Per far questo, la proposta prevede che siano create delle interfacce comuni per la conversione dei dati territoriali.

E' compito degli stati membri quindi, creare infrastrutture spaziali a livello nazionale (NSDI, National Spatial Data Infrastructure) mentre è compito della direttiva europea gestire il coordinamento di tali infrastrutture affinché possano essere fra loro compatibili. Tali infrastrutture di dati geografici nazionali devono permettere di:

[10] Nella terminologia di INSPIRE, con il termine *livello* si fa riferimento ad un ente/ufficio/dipartimento all'interno della gerarchia amministrativa.

- condividere le informazioni territoriali fra i vari livelli dell'amministrazione pubblica;
- combinare le informazioni territoriali provenienti da più fonti.

Con l'adozione di questa direttiva, gli stati membri istituiscono e gestiscono servizi di rete per il trasferimento di dati e metadati ed i servizi ad essi correlati.

Tali servizi permettono di effettuare:

- ricerche: sia di dati, sia di servizi relativi ai dati territoriali;
- consultazione: visualizzazione, navigazione, variazione della scala (zoom) e variazione della porzione di territorio visualizzato (pan);
- conversione: di trasformazione di coordinate dei dati;
- invocazione di servizi sui dati territoriali.

E' inoltre garantita la creazione di metadati sui dati territoriali ed i servizi ad essi legati.

1.2. European Spatial Data Infrastructure

Come detto, l'infrastruttura ESDI è realizzata dallo sforzo congiunto degli stati afferenti alla Comunità Europea, dove ogni stato costruisce un proprio NSDI.

Un NSDI è un'applicazione di Spatial Data Infrastructure (SDI); uno SDI permette la ricerca, il recupero e la valutazione di dati geografici da parte di enti governativi, commerciali, privati ed accademici. Non comprende solamente l'insieme di dati ed i servizi ad essi legati, ma anche tutte le infrastrutture legislative ed amministrative di coordinamento necessarie affinché lo SDI risulti effettivamente funzionale su tutti i livelli: locali, regionali, nazionali ed internazionali.

L'infrastruttura Europea ha il compito di fornire un'infrastruttura legislativa ed architetturale al fine di armonizzare i vari SDI nazionali.

L'infrastruttura ESDI è genericamente denominata come Global Spatial Data Infrastructure (GSDI). A livello internazionale, oltre ad ESDI di INSPIRE, esistono le seguenti infrastrutture GSDI:

- NSDI, USA,
- SNIG, Portogallo,
- ASDI, Australia,
- NaLIS, Malesia,
- NSIF, sud Africa.

1.3. Il Quadro Generale Europeo

Per avere un quadro generale sullo stato dell'arte degli NSDI negli stati dell'Unione Europea, si può far riferimento al documento di valutazione "Spatial Data Infrastructure in Europe: state of play, Spring 2004" [4].

Il rapporto divide i 32 stati[11] in due tipologie a seconda che il ruolo di coordinatore del NSDI sia rivestito dal National Mapping Agency (NMA, da non confondere con il Produttore Nazionale di Dati, NDP), oppure da un ministero o dipartimenti amministrativi. Di questi stati, sono stati valutati un insieme di portali web da cui è stata estratta una lista di dataset seguendo cinque temi prioritari:

1. Trasporti,
2. Idrografia,
3. Elevazione,
4. Catasto,
5. Indirizzi.

I criteri di scelta dei dataset candidati all'inclusione all'interno di INSPIRE (ed all'uso nella valutazione del documento), si basano su:

1. disponibilità ed accessibilità dei metadati;
2. copertura del territorio nazionale;
3. dettaglio spaziale (un intervallo di scale fra 1:10.000 e 1:50.000);
4. carattere analitico.

L'esame effettuato sulle infrastrutture nazionali, ha rivelato alcuni casi in cui i contenuti degli SDI regionali risultano molto più accurati di quelli dello NDSI[12]; in simili situazioni sono stati considerati gli SDI.

Il rapporto è composto da due tabelle [5], una rappresenta la situazione nel 2004, mentre la seconda evidenzia i miglioramenti fatti dagli stati membri dall'ultimo rapporto risalente al 2003.

Comparando le due tabelle (limitando l'analisi ai 15 stati membri), si nota omogeneità a livello di organizzazione e coordinamento fra i vari livelli, in quanto tutti gli stati hanno mediamente una situazione conforme alla proposta INSPIRE.

Per quanto riguarda le leggi sulla protezione dei dati e privacy si ha una situazione mediocre che, nell'arco di un anno (fra il 2003 e 2004), non è migliorata.

Miglioramenti sostanziali sono stati fatti nell'ambito tecnico legato ai dati, mentre nei settori relativi alla qualità dei metadati, all'accessibilità dei dati e dell'armonizzazione degli NSDI, la situazione è pressoché stazionaria.

---

[11] 15 Stati membri; 10 nuovi membri; 3 stati candidati; 4 stati appartenenti alla EFTA (*European Free Trade Association*, composta da: Svizzera, Norvegia, Islanda e Liechtenstein).

1.4. La Situazione Italiana

In Italia, i maggiori produttori di informazione geografica (NDP) sono:

- L'Agenzia del Territorio: nata all'interno della riforma del Ministero dell'Economia e delle Finanze. Ha il compito di assicurare al cittadino ed agli enti pubblici e privati, una corretta ed efficace gestione dell'anagrafe dei beni immobiliari attraverso l'offerta di servizi relativi al catasto, alla pubblicità immobiliare ed alla cartografia.
- Istituto Geografico Militare: ha il compito di fornire supporto geotopografico alle Unità e Comandi dell'Esercito Italiano oltre che svolgere la funzione di Ente Cartografico di Stato (relativo alle terre emerse).
- Istituto Idrografico della Marina: ente della Marina Militare, ha il compito di eseguire il rilievo sistematico dei mari al fine di produrre la documentazione nautica ufficiale e diffondere l'informazione nautica, per la sicurezza della navigazione e la salvaguardia della vita umana in mare.
- Centro Informazioni Geotopografiche Aeronautiche: soddisfa sia la necessità degli utenti dell'Aeronautica Militare sia quelle delle altre Forze Armate e, nei limiti delle leggi e dei regolamenti in vigore, le necessità civili della Nazione.

Per quanto concerne le altre informazioni territoriali, si fa riferimento agli organi amministrativi delle Regioni, Province e Comuni. Su questo sfondo si muove Intesa GIS[13], l'intesa fra Stato, Regioni e Province per i sistemi informativi geografici. Intesa GIS è stata approvata dalla Conferenza Stato Regioni e Province Autonome nella seduta del 26 settembre 1996 e coinvolge diverse Amministrazioni Centrali ed organismi statali, come il Centro Nazionale per la Pubblica Amministrazione (CNIPA), i Comuni (ANCI) e Province (UPI)[14].

L'obiettivo di Intesa GIS è "lo sviluppo di interventi coordinati per realizzare in Italia entro 6-8 anni le basi informative territoriali gestite su elaboratore a copertura dell'intero territorio nazionale, necessarie per l'esercizio delle funzioni di interesse locale, regionale e nazionale".

---

[12] Spagna e Belgio.
[13] Intesa GIS non è una legge.
[14] Inoltre, Regioni e province Autonome, Comunità Montane (UNCEM) e le Aziende per la gestione dei pubblici servizi (Confservizi).

Questo si traduce nella produzione di specifiche tecniche comuni, produzione di dati compatibili con le specifiche e la messa in opera di attività finalizzate a rendere pubbliche informazioni geografiche tramite la produzione di cataloghi cartografici.

L'infrastruttura di rete è pensata in modo da poter scambiare dati geografici fra il nodo centrale, National Digital Mapping Portal (NDMP, un portale web), ed i nodi periferici della rete, le amministrazioni federate.

Lo NDMP mantiene gli strati di informazioni nazionali relativi al sistema di riferimento cartografico, mentre le amministrazioni federate offrono informazioni locali presenti nei loro archivi.

A livello italiano il sistema di riferimento cartografico è detenuto dal Portale Cartografico Nazionale (per opera del Ministero dell'Ambiente e della Tutela del Territorio [6]) mentre la costruzione dei livelli base delle mappe topografiche, basata sulle sorgenti regionali, locali e sui cataloghi di metadati, è affidata al Centro Interregionale di coordinamento e documentazione per le informazioni territoriali [7].

Facendo riferimento ad alcuni temi su cui si basa il rapporto di valutazione degli NSDI Europei (Building Block) [8], si notano i seguenti fatti:

- non si ha nessuna informazione per quanto riguarda la qualità dei dati di riferimento e dei dati tematici. Intesa GIS sta preparando delle specifiche tecniche (Capitolato tecnico) perché siano adottate da tutte le autorità pubbliche; alcune regioni utilizzano già tali specifiche.
- le linee guida per l'interoperabilità dei dati iniziano a seguire quelle dettate dallo standard ISO 19111[15] ed dal consorzio OpenGIS. Nella maggior parte dei casi, per il trasferimento dei dati sono utilizzati formati proprietari pubblici (come shape file di ESRI o DWG di AutoCAD), malgrado Intesa GIS stia spingendo verso il formato di codifica open introdotto dal consorzio OpenGIS: GML.
- l'italiano costituisce prevalentemente l'unica lingua in cui è presentata la documentazione degli SDI. Solo alcuni portali supportano documentazione in più lingue (l'unico esempio in tal senso è costituito dal sito delle rete civica dell'Alto Adige che offre contenuti in inglese, tedesco, italiano, francese e ladino [9]).

[15] ISO19111:2003 definisce lo schema concettuale per la descrizione di riferimenti spaziali tramite coordinate.

- i metadati sono quasi assenti e se esistenti, sono molto eterogenei nel formato e nella qualità (un esempio interessante è costituito dal sito della regione Emilia-Romagna [10]).
- Servizi di accesso a metadati fanno riferimento ai seguenti siti:
  - portale Cartografico Nazionale [5];
  - Centro Interregionale [11],
  - portale della regione Emilia-Romagna basato sul software ArcIMS;
  - sito delle rete civica dell'Alto Adige, Ufficio informatica geografica e statistica [12].

1.4.1. SIGMA TER

In Italia, oltre ad Intesa GIS, esistono vari progetti laterali in ambito GIS, uno di questi è il progetto SIGMA TER (Servizi Integrati catastali per il Monitoraggio Amministrativo Territoriale, Σ^3).

Il progetto SIGMA TER propone la costruzione di un'infrastruttura informatica che consenta l'interscambio di informazioni catastali e territoriali fra "Regione - Agenzia del Territorio" e "Regione – Enti Locali" [13].

Prevede inoltre allo sviluppo di un ampio numero di servizi, basati sui dati catastali e territoriali, rivolti sia ai singoli cittadini che alle imprese.

Il ruolo dell'Agenzia del Territorio sarà di realizzare la componente d'interscambio e revisione dei dati (servizi infrastrutturali), mentre le regioni andranno a sviluppare applicazioni destinate all'interazione con la componente d'interscambio (servizi orientati all'utenza finale).

1.5. Conclusioni

In sintesi, il sorgere di questi progetti ed iniziative nasce dalla presa di coscienza, da parte di enti governativi, che le informazioni territoriali sono di fondamentale importanza per una corretta amministrazione del territorio. Tale necessità ha portato a focalizzare gli sforzi per migliorare la qualità dei dati e dei servizi ad essi legati al fine di costruire un'infrastruttura flessibile che permetta l'accesso alle informazioni territoriali in modo semplice, immediato ed omogeneo.

Il panorama italiano, come quello europeo, mostra che questi sforzi stanno producendo risultati, anche se in maniera lenta ma progressiva.

Bibliografia

- Relazione sulla "*Proposta di Direttiva del Parlamento Europeo e del Consiglio che istituisce un'infrastruttura per l'informazione nella comunità (INSPIRE)*", Bruxelles 23/07/2004 COM(2004) 516 Definitivo 2004/0175 (COD);
- "*INSPIRE,* INfrastructure for SPatial InfoRmation in Europe", http://www.ec-gis.org/inspire/home.html.
- Douglas D. Nebert, "*Developing Spatial Data Infrastructures, The SDI Cookbook v2.0*": 25/01/2004;
- "*Global Spatial Data Infrastructure Association*", http://www.gdsi.org;

Capitolo 2

Standard E Tecnologie

2.1. Open Geospatial Consortium, OGC

Per quanto detto nel capitolo precedente, la direttiva INSPIRE non fornisce raccomandazioni di tipo implementative, tanto meno specifiche riguardanti determinate piattaforme o tecnologie di sviluppo da utilizzare; al contrario, Intesa GIS indirizza verso le specifiche **OpenGIS** per raggiungere il grado di interoperabilità ed integrazione richiesto da INSPIRE.

Open Geospatial Consortium (OpenGIS) è un consorzio, composto da oltre 250 membri fra aziende ed enti del settore GIS, che ha come obiettivo la definizione di un insieme di raccomandazioni e specifiche tecniche per massimizzare lo sfruttamento di dati territoriali, in particolare per lo scambio di dati e per la realizzazione di servizi in ambiente distribuito.

Nell'ambito di INSPIRE è spesso sottolineata la necessità di interfacce comuni e di interoperabilità; questo è lo scopo primario di OGC: **sviluppare e promuove interfacce comuni ed aperte** che permettano l'integrazione e la cooperazione fra sistemi eterogenei via Web.

In prima analisi, il compito del consorzio OpenGIS è di sviluppare **Open Standard** che permettano l'interoperabilità tra i GIS commerciali[16].

Uno standard è indicato come **open** (secondo OpenGIS) se:

- è creato in un processo internazionale ed aperto a cui partecipano varie aziende, quindi uno standard **non proprietario**;
- non è sottoposto a diritti di autore od a forme di royalty;
- l'accesso alle specifiche è consentito gratuitamente e pubblicamente;
- è privo di discriminazioni verso persone o gruppi;
- le specifiche sono indipendenti dalle tecnologie di implementazione.

Attenzione a non confondere *Open Source* con *Open Standard*: la dicitura *Open Source* indica che il codice prodotto deve rimanere di pubblico dominio. Il fine di un *Open Standard* è di far interoperare vari prodotti, commerciali e *Open Source*.

L'uso di interfacce, conformi alle specifiche OGC, significa semplificare attività come:

- ricerca di dati, ignorando la loro posizione fisica;

[16] Gli *Open Standard* iniziano ad essere adottati anche da software GIS non commerciali.

- integrazione, combinazione ed uso di informazioni spaziali provenienti da sorgenti eterogenee;
- effettuare sovrapposizione, di dati spaziali aventi sistemi di riferimento diversi;
- permettere operazioni distribuite su dati distribuiti;
- lavorare in un ambiente indipendente dalla piattaforma, *open platform*.

La possibilità di usufruire di un *open platform* è data dall'uso di interfacce *open* definite tramite le specifiche OGC.

L'ideazione, lo sviluppo, l'armonizzazione ed il mantenimento delle interfacce, è formalizzato dal consorzio attraverso un insieme di documenti, detti **specifiche**:

- ***Abstract Specification*** (AS), composto da due modelli: il **modello essenziale** (*essential model*) che stabilisce un collegamento concettuale fra il software ed il mondo reale. Il **modello astratto**, (*abstract model*) che descrive il software indipendentemente dall'implementazione. La specifica astratta è composta da un insieme di argomenti, *topics*, (fra loro correlati) che definiscono l'architettura di base ed il comportamento delle interfacce. Ne segue che tutte le interfacce ed i protocolli sviluppati, fanno riferimento ai concetti fondamentali descritti nella specifica in modo da risultare interoperabili fra loro. La AS definisce modelli di riferimento per lo sviluppo delle specifiche di implementazione OpenGIS (*Implementation Specifications*).
- ***Implementation Specification*** (IS), queste specifiche sono indirizzate ad un "pubblico" tecnico (produttori di software) in quanto definiscono la struttura delle interfacce fra componenti software, secondo quanto stabilito nelle specifiche AS.

Il ciclo di vita delle specifiche è formalizzato dai seguenti documenti[17]:

- ***Discussion Paper***, il suo scopo è di creare una discussione nell'ambiente dell'informazione geospaziale su argomenti specifici. Questi documenti non riflettono la posizione ufficiale del consorzio.
- ***Raccomandation Paper***, contiene una discussione su tecnologie da rilasciare al pubblico. Questi documenti riflettono la posizione ufficiale del consorzio.
- ***Request for Comments*** (RFC), contengono tutte le proposte candidate a divenire IS. Su tali proposte sono richiesti commenti da parte dei membri del consorzio.

[17] Ne vengono riportati solo alcuni, per la lista completa, consultare http://www.opengeospatial.org/.

L'elaborazione dei documenti e delle specifiche è effettuata dai membri del consorzio, divisi in tre gruppi, detti **comitati**:

- Il ***Technical Committee***, si occupa di coordinare lo sviluppo e la modifica delle AS oltre che a coordinare lo sviluppo e l'adozione delle IS. E' diviso in gruppi di lavoro (*Working Group*, WG), dove ognuno si occupa di sviluppare specifiche di implementazione per vari temi (CAD, metadati, linguaggi di interrogazione, sistemi di riferimento, …). È compito dei gruppi di lavoro proporre le richieste RFC.
- ***Planning Committee*** (PC), ha il compito di:
  - approvare le raccomandazioni per l'adozione ed il rilascio di specifiche da parte del comitato tecnico, TC;
  - accettare le specifiche candidate a divenire IS, proposte dal TC;
  - valutare, ed eventualmente adottare, le revisioni eseguite dal TC sulle IS adottate (*Adopted Implementation Specification*).
- ***Strategic Management Advisory Committee*** (SMAC), permette ai membri di partecipare ai processi strategici di pianificazione del consorzio e di sostenere le operazioni necessarie al raggiungimento della missione del consorzio. Questo comitato lavora in parallelo con il TC ed il PC.

Il fatto che un produttore assicuri che il suo software sia **conforme** alle specifiche OpenGIS®[18] significa che supporta l'accesso trasparente a dati geospaziali ed a risorse di *geoprocessing* distribuite in un ambiente di rete. Perché un software sia considerato conforme deve superare dei processi di test relativi alle rispettive specifiche; il programma *Conformance Testing & Interoperability Evaluation* descrive tale processo di attestazione di conformità. Se un prodotto software **implementa** le specifiche OpenGIS, non implica che sia conforme alle specifiche che implementa.

Riassumendo, OpenGIS definisce un insieme di *Open Standard*, non proprietari e soggetti a revisione; tali standard operano su un'infrastruttura architetturale che consiste in un modello tecnico di riferimento, un insieme di componenti standard, specifiche e relazioni fra i vari elementi dell'architettura (secondo quanto definito nella AS).

[18] I marchi OpenGIS®, Open GIS® e open GIS™ sono registrati in modo da proteggere legalmente gli standard.

Il consorzio si "limita" ad organizzare standard riconosciuti con **specifiche guida** e promuoverne l'adozione; per questo, OGC può essere paragonato al *World Wide Web Consortium* (W3C) od al *Internet Engineering Task Force* (IETF).

Il compito di definire standard nel campo dell'informazione spaziale digitale è della commissione TC211 dello *International Standards Organization, ISO/TC211*. La commissione stabilisce un insieme strutturato di standard[19] (internazionali) relativi ad oggetti e fenomeni associabili direttamente od indirettamente al territorio. Tali standard specificano, per l'informazione geografica, metodi, strumenti e servizi per la gestione dei dati.

Esiste una stretta collaborazione fra ISO/TC211 e OGC (spesso le persone coinvolte sono le stesse) in modo tale che il lavoro dei due gruppi sia coerente; molte specifiche OGC sono divenute standard ISO o sono in attesa di esse approvate.

2.1.1. Specifiche di Implementazione

Il consorzio OGC ha definito molte specifiche di implementazione: saranno presentate solo quelle inerenti al linguaggio di codifica dei dati vettoriali (GML), ai servizi per lo scambio di dati (*Web Feature Service* e *Web Map Service*) e di interrogazione sui metadati (*Catalog Service*). Questi ultimi tre servizi sono generalmente riferiti come "*OGC Web Service*, OWS".

2.1.2. Geographic Markup Language, v2.1.2

Fino a qualche tempo fa, per poter utilizzare dati geospaziali, era di rilevante importanza conoscerne il formato: in base all'applicazione con cui erano stati codificati si avevano diversi formati proprietari (SHAPE, DXF, MDB, …). Questo comporta dei problemi al momento in cui nasce una forte necessità di interscambio di dati e di elaborazione su insiemi di dati, codificati in formati eterogenei.

Uno dei capisaldi dell'opera di armonizzazione iniziata da OGC, consiste nella creazione di un linguaggio *open* per descrivere i dati geospaziali, che:

- separi contenuto e rappresentazione;
- sia sufficientemente estendibile per supportare vari scenari di rappresentazione;
- permetta una codifica efficiente per le geometrie;
- sia pensato per lo scambio e la memorizzazione delle informazioni spaziali ed in ambienti Internet;

[19] Attraverso i progetti definiti dai documenti ISO: ISO 6709 e da ISO19101 sino a ISO19143.

- permetta una semplice integrazione fra sorgenti di dati geospaziali e non (specialmente per dati rappresentati in XML);
- fornisca un insieme comune di oggetti geografici per la modellazione al fine di ottenere interoperabilità ed indipendenza dalle applicazioni.

Il lavoro ha portato alla definizione di un dialetto XML, chiamato ***Geographic Markup Language*** (GML), specifico per le informazioni geospaziali.

2.1.2.1. Feature

GML basa la rappresentazione dei dati geospaziali definendo un'unità fondamentale dell'informazione geospaziale: ***feature***. Nella AS relativa a GML [14], è definita come "un'astrazione di un fenomeno o entità del mondo reale", mentre una ***geographic feature*** è una feature associata ad una locazione relativa al territorio;

Ad una feature è associato uno stato costituito da un insieme di **proprietà**[20] descritte da triple <*nome, tipo, valore*>, dove:

- *tipo*::= *simple* | *complex*
- *simple*::=*integer* | *bool* | *double* | *string* | …
- *complex*::= *date* | *time* | *address* | ...

È prevista, inoltre, la possibilità di combinare assieme tipi appartenenti a *simple* e *complex*.

Una geographic feature ha un modello che permette di rappresentare solo dati vettoriali (v2.1.2), anch'essa ha un insieme di proprietà dove almeno una assume "valori" geometrici. Tali valori sono la rappresentazione in coordinate, secondo un dato sistema di riferimento bidimensionale[21], di oggetti geometrici aventi come tipo punto, spezzata (*line string*) e poligono; in questo caso si parla di **feature semplici**. Sono previsti inoltre collezioni di elementi geometrici omogenei od eterogenei: gruppi di oggetti geometrici quali punti (*multi-point*) spezzate (*multi-line string*) e poligoni (*multi-polygon*).

Quanto detto è riportato nel modello geometrico, mostrato in figura 2.1; la classe astratta "*Geometry*" associa ad ogni oggetto geometrico un sistema di riferimento spaziale (SRS, *Spatial Reference System*) che descrive lo spazio di coordinate in cui l'oggetto è definito.

---

[20] Nella comunità dell'informazione geospaziale (GI) è di comune uso riferire la proprietà di una *feature* con il termine "attributo".

[21] Dalla versione 3 di GML è stata formalizzata la terza dimensione.

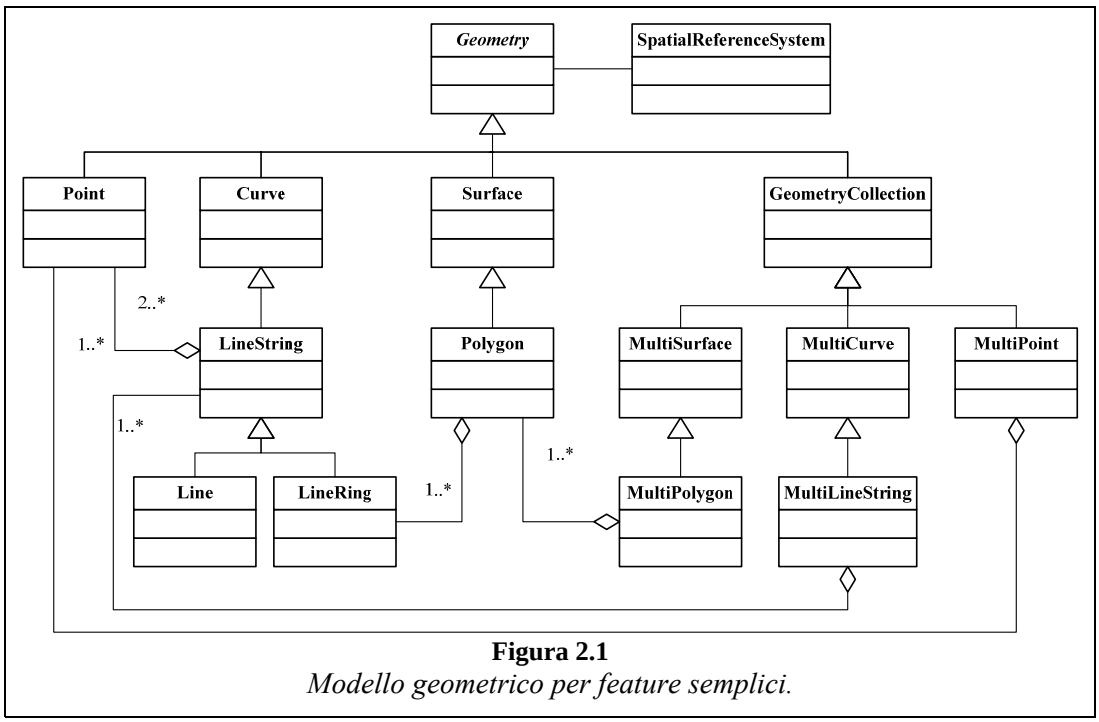


Figura 2.1
Modello geometrico per feature semplici.

Tramite una collezione di feature (che può essere considerata una feature) si esegue una classificazione del mondo reale andando ad identificare un insieme di *feature type*; questo introduce gli aspetti classici di rappresentazione dei fenomeni in ambiente GIS: scegliere la corretta rappresentazione geometrica per un dato fenomeno secondo un contesto preciso.

2.1.2.2. Rappresentazione Di Una Feature

Una feature type è rappresentata come un elemento XML in cui sono annidate le proprietà anch'esse rappresentate in XML.

Trattandosi di una rappresentazione di dati basata su XML, sono previsti tre schemi XML (e relativi namespace) per la definizione dei tipi degli elementi standard:

feature.xsd: definisce il modello generale di una feature con attributi;

geometry.xsd: descrive dettagliatamente il modello geometrico;

xlinks.xsd: fornisce gli attributi del linguaggio XML *Linking Language*, XLink, utilizzati per collegare risorse (immagini, altri documenti GML, altre feature).

Assieme costituiscono un meta-schema contenente tipi base e strutture ma è parzialmente utilizzabile in quanto la rappresentazione della feature type dipende dal contesto del fenomeno rappresentato. E' definito quindi un ulteriore schema XML, detto *application schema*, che incorpora i tre citati e definisce lo schema adatto al dominio della feature type (figura 2.2).

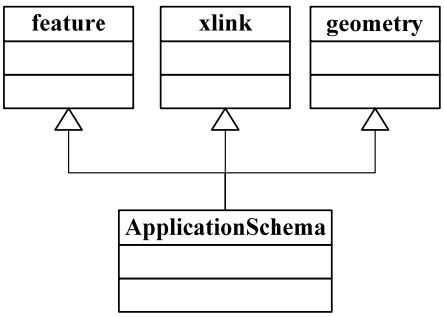


Figura 2.2
Composizione di un application schema.

La costruzione dello *application schema* è realizzata seguendo un insieme di linee guida e regole (**normative** [15]) che portano ad avere uno schema **GML conforme**.

L'esempio sotto riportato, mostra la rappresentazione GML di un'istanza di feature con tre proprietà ed il relativo application schema.

Esempio 2.1	Codifica di una feature senza proprietà geometrica

Si supponga di avere una feature type che rappresenti dei negozi, una sua istanza può essere così definita:

```
<Negozio>
     <nomeAttività[22]>BirraShop</nomeAttività>
     <nomeProprietario>Jhon Doe</nomeProprietario >
     <età>28</età>
     <partitaIva>100022587425</partitaIva >
</Negozio>
```

Se si crea uno schema XSD per questo elemento XML, si ottiene:

```
<element name = "Negozio" type = "ex:TipoNegozio"/>
<complexType name = "TipoNegozio">
     <sequence>
          <element name = "nomeAttività" type = "string"/>
          <element name = "nomeProprietario" type = "string"/>
          <element name = "età" type = "integer"/>
          <element name = "partitaIva" type = "string"/>
     </sequence>
</complexType>
```

Questo schema è corretto, ma non tiene conto della conformità con GML; applicando le normative di definizione si ottiene lo application schema:

```
<element name = "Negozio" type = "ex:TipoNegozio"
         substitutionGroup = "gml:AbstractionFeatureType"/>
<complexContent>
          <extension base = "gml:_Feature"> <sequence>
               <element name = "nomeAttività" type = "string"/>
               <element name = "nomeProprietario" type = "string"/>
               <element name = "età" type = "integer"/>
               <element name = "partitaIva" type = "string"/>
```

[22] Convenzione: in GML i nomi delle feature hanno sempre il primo carattere maiuscolo, mentre le proprietà in minuscolo. In caso i nomi contengano più termini, i termini successivi al primo hanno il primo carattere maiuscolo.

```
        </sequence>
    </extension>
</complexContent>
```

L'uso dello schema *feature.xsd* consente di utilizzare attributi predefiniti, come ad esempio l'identificatore di feature, "FID".

2.1.2.3. Codifica della Geometria

GML fornisce una rappresentazione per le seguenti classi di geometrie:

- *Point,*
- *LineString,*
- *LinearRing,*
- *Polygon,*
- *MultiPoint,*
- *MultiLineString,*
- *MultiPolygon,*
- *MultiGeometry.*

La primitiva *linear Ring* definisce un percorso chiuso dove le coordinate del primo ed ultimo punto devono coincidere; essa è utilizzata per definire la primitiva "poligono".

La primitiva "poligono" è caratterizzata da un bordo interno e da zero o più bordi esterni, con l'unico vincolo che un bordo interno non può intersecare od essere contenuto in un altro bordo. I bordi sono definiti tramite la primitiva *linear ring*

Per rappresentare le coordinate delle geometrie, GML prevede due elementi contenuti all'interno dello schema *geometry.xsd*:

- <*coordinates*>: definisce in una singola stringa le tuple delle coordinate.

```
<element name = "coordinate" type = "gml:CoordinatesType"/>
<complexType name = "CoordinatesType">
     <simpleContent>
          <extension base = string">
               <attribute name = "decimal" type = "string"
                          use = "default" value = "."/>
               <attribute name = "cs" type = "string"
                          use = "default" value = ","/>
               <attribute name = "ct" type = "string"
                          use = "default" value = "&#x20"/>
          </extension >
     </simpleContent>
</complexType>
```

Significa che il delimitatore decimale delle coordinate è il carattere "**.**", mentre "**,**" separa le coordinate all'interno di una tupla; infine il carattere "spazio" (codice ASCII 20) separa le tuple all'interno della stringa.

L'esempio 2.2 mostra le tre principali primitive GML rappresentate tramite l'elemento <*coordinates*>.

- <*coord*>: permette di specificare una successione di tuple di coordinate.

```
<element name = "coord" type = "gml:CoordType"/>
<complexType name = "CoordType">
      <sequence>
            <extension base = "string">
                  <attribute name = "X" type = "decimal"/>
                  <attribute name = "Y" type = "decimal"
                              minOccours = "0" />
                  <attribute name = "Z" type = "string"
                              minOccours = "0"/>
            </extension >
      </sequence>
</complexType>
```

L'esempio 2.3 mostra un punto rappresentato in GML tramite l'elemento *<coord>*.

Esempio 2.2 | Primitive geometriche definite tramite *<coordinates>*

Primitiva punto:

```
<Point srsName = "http://www.opengis.net/gml/srs/epsg.xml#4326">
     <coordinates>15.2,22.4</coordinates>
</Point>
```

Primitiva spezzata (*line string*):

```
<LineString srsName = "http://www.opengis.net/.../epsg.xml#4326">
    <coordinates>15.0,25.0 30.0,50.0 60.0,100.0</coordinates>
</Point>
```

Primitiva poligono con un bordo interni ed uno esterno:

```
<Polygon srsName = "http://www.opengis.net/.../epsg.xml#4326">
     <outerBounduaryIs>
        <LinearRing>
           <coordinates>0.0,0.0 0.0,40.0 40.0,40.0 40.0,0.0 0.0,0.0</coordinates>
        </LinearRing>
     </outerBounduaryIs>
     <innerBounduaryIs>
        <LinearRing>
           <coordinates>10.0,10.0 30.0,30.0 40.0,10.0 10.0,10.0</coordinates>
        </LinearRing>
     </innerBounduaryIs>
</Point>
```

Esempio 2.3 | Primitiva GML punto definita mediante *<coord>*

```
<Point srsName = "http://www.opengis.net/gml/srs/epsg.xml#4326">
       <coord><X>15.2</X><Y>22.4</Y>
       </coord>
</Point>
```

Una volta introdotto il concetto di feature e di rappresentazione delle primitive geometriche tramite coordinate, il passo per costruire feature type geometriche è fatto; nell'esempio 2.1, l'istanza di feature type viene georeferenziata:

Esempio 2.4	Georeferenziazione della feature type negozio

Se si decide di modellare geometricamente un negozio, mostrato nell'esempio 2.1, come un punto, allora l'istanza sarà:

```
<Negozio>
     <nomeAttività>BirraShop</nomeAttività>
     <nomeProprietario>Jhon Doe</nomeProprietario>
     <età>28</età>
     <partitaIva>100022587425</partitaIva >
     <gml:location>
          <gml:Point>
               <gml:coord>
                    <gml:X>15.2</gml:X>
                    <gml:Y>22.4</gml:Y>
               </gml:coord>
          </gml:Point>
     </gml:location>
</Negozio>
```

Il relativo *application schema* sarà il seguente:

```
<element name = "Negozio" type = "my:TipoNegozio"
         substitutionGroup = "gml:AbstractionFeatureType"/>
<complexContent>
     <extension base = "gml:_Feature">
          <sequence>
               <element name = "nomeAttività" type = "string"/>
               <element name = "nomeProprietario" type = "string"/>
               <element name = "età" type = "integer"/>
               <element name = "partitaIva" type = "string"/>
               <element ref = "gml:location" />
          </sequence>
     </extension>
</complexContent>
```

L'elasticità di XML e XSD permettono di personalizzare la proprietà geometrica di una feature type. Nell'esempio è possibile definire un tipo di locazione specifica per la feature type dei negozi:

```
<element name = "PosizioneNegozio" type = "gml:PointPropertyType"/>
```

da sostituire all'elemento `<element ref … />`. Quello che viene fatto è una alias dell'elemento globale `gml:PointPropertyType`.

L'ultimo tassello per fornire una visione generale di GML è costituito dalla **collezione di feature**; una collezione può contenere un numero arbitrario di istanze di feature type eterogenee.

Si supponga di aver due feature type, una già discussa nell'esempio 2.4 e la seconda definita come segue:

```
<element name = "PostoDiBlocco" type = "ex:PostoDiBlocco"
         substitutionGroup = "gml:AbstractionFeatureType"/>
<complexContent>
     <extension base = "gml:AbstractFeatureType">
          <sequence>
               <element name = "idPattugliaPolizia" type = "string"/>
               <element ref = "gml:location"/>
```

```
            </sequence>
        </extension>
    </complexContent>
```

Tramite `<extension base = "gml:AbstractFeatureType">` è possibile ereditare gli attributi per le feature definiti nello schema *feature.xsd*. Nell'esempio 2.5 viene utilizzato l'identificatore univoco di feature "FID" (i valori di tale attributo devono essere unici all'interno del documento GML).

Tramite una collezione è possibile raggruppare le istanze di feature type che modellano i negozi assieme alle istanze che modellano i posti di blocco:

Esempio 2.5	Collezione di feature type eterogenee

```
<Collezione>
     <proprietàCollezione> un valore </proprietàCollezione>
     <gml:featureMember>
          <PostoDiBlocco fid = "pdb01"> ... </PostoDiBlocco>
     </gml:featureMember>
     <gml:featureMember>
          <Negozio> ... </Negozio>
     </gml:featureMember>
     <gml:featureMember>
          <Negozio> ... </negozio>
     </gml:featureMember>
</Collezione >
```

Il rispettivo *application schema* della collezione sarà:

```
<element name = "Collezione" type = "my:Collezione"
         substitutionGroup = "gml:FeatureCollection"/>
<complexType>
     <complexContent>
          <extension base = "gml:AbstractFeatureCollectionType">
               <sequence>
                    <element name = "prioprietàCollezione"
                             type = "string"/>
               </sequence>
          </extension>
     </complexContent>
</complexType>
```

L'elemento `featureMember` permette di includere sia un'istanza di feature type sia un puntatore ad una istanza contenuta in un altro documento GML.

Il meccanismo di collegamento è fornito dal linguaggio XLink tramite lo schema *xlink.xsd*. L'esempio 2.6 mostra una collezione contenente feature assieme a collegamenti.

Esempio 2.6	Uso di XLink in una collezione

Si supponga di avere un documento GML, *postiBlocco.gml*, contenente una collezione di istanze di feature type.

```
<Collezione>
     <proprietàCollezone> un valore </proprietàCollezone>
     <gml:featureMember xlink:type = "simple"
                        xlink:href =
                                  "http://.../postiBlocco.gml#pdb01">
```

```
    </gml:featureMember>
        <Negozio> ... </negozio>
    </gml:featureMember>
</Collezione >
```

E' possibile inoltre esplicitare lo schema di una feature type collegata, facendo riferimento al suo application schema definito in un documento separato mediante l'attributo `remoteSchema` del namespace `gml`.

Facendo riferimento all'esempio 2.6, si supponga di aver definito un *application schema* per la feature type "PostidiBlocco" in un file *postiBlocco.xsd*, allora si avrà:

```
<gml:featureMember xlink:type = "simple"
                   gml:remoteSchema = "http://.../doc/postiBlocco.xsd"
                   xlink:href = "http://.../doc/postiBlocco.gml">
```

Mediante XLink è possibile modellare due tipologie di relazioni fra feature:

- *lightweight*, la relazione è descritta senza identità e non è possibile associarvi delle proprietà;
- *heavyweight*, a differenza delle precedenti, è possibile associarvi delle proprietà ed ogni relazione è identificata; come mostrato nell'esempio 2.7, si presenta come una feature.

Esempio 2.7	Esempi di relazione binaria fra feature

Esempio di relazione binaria *lightweight*:

```
<ParcellaCatastale fid = "pc001">
    <proprietà> un valore </proprietà>
    <relazione xlink:type = "simple" xlink:href = "#pc112"/>
</ParcellaCatastale >

<ParcellaCatastale fid  = "pc112">
    <proprietà> un valore </proprietà>
    <relazione xlink:type = "simple" xlink:href = "#pc001"/>
</ParcellaCatastale >
```

Esempio di relazione binaria *heavyweight* identificata tramite l'attributo "fid"; si consideri le due parcelle catastali senza la proprietà `relazione`:

```
<RelazioneBinaria fid  = "rb001">

    <proprietàRelazioneB> un valore </proprietàRelazioneB>
    <relazione xlink:type = "simple" xlink:href = "#pc001"/>
    <relazione xlink:type = "simple" xlink:href = "#pc112"/>
</ParcellaCatastale >
```

2.1.3. Introduzione a OpenGIS Web Service

I servizi OpenGIS Web Service (OWS), o servizi OGC, sono di rilevante importanza per l'**interoperabilità** fra sistemi GIS; con interoperabilità si indica la "capacità di sistemi eterogenei di scambiarsi dati ed istruzioni in tempo reale e di fornire servizi".

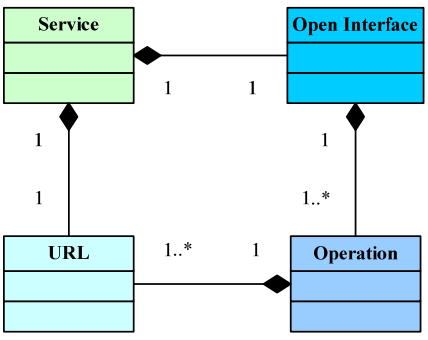


Figura 2.3
Elementi che compongono un servizio OWS.

Un servizio OGC è composto da un insieme d'**operazioni** che ne compongono l'**interfaccia** (figura 2.3). Un servizio comunica con un client attraverso un protocollo di rete che trasporta le richieste per accedere alle operazioni.

Ne segue che i servizi OGC sono **risorse remote**, utilizzabili tramite interfacce *open*, capaci di effettuare elaborazioni invocate tramite richieste e di restituire dati. Possono essere considerati i precursori degli attuali *web service* (Capitolo 2.2).

La figura 2.4 riporta l'architettura OWS in cui sono evidenziati i servizi per l'accesso ai dati geospaziali.

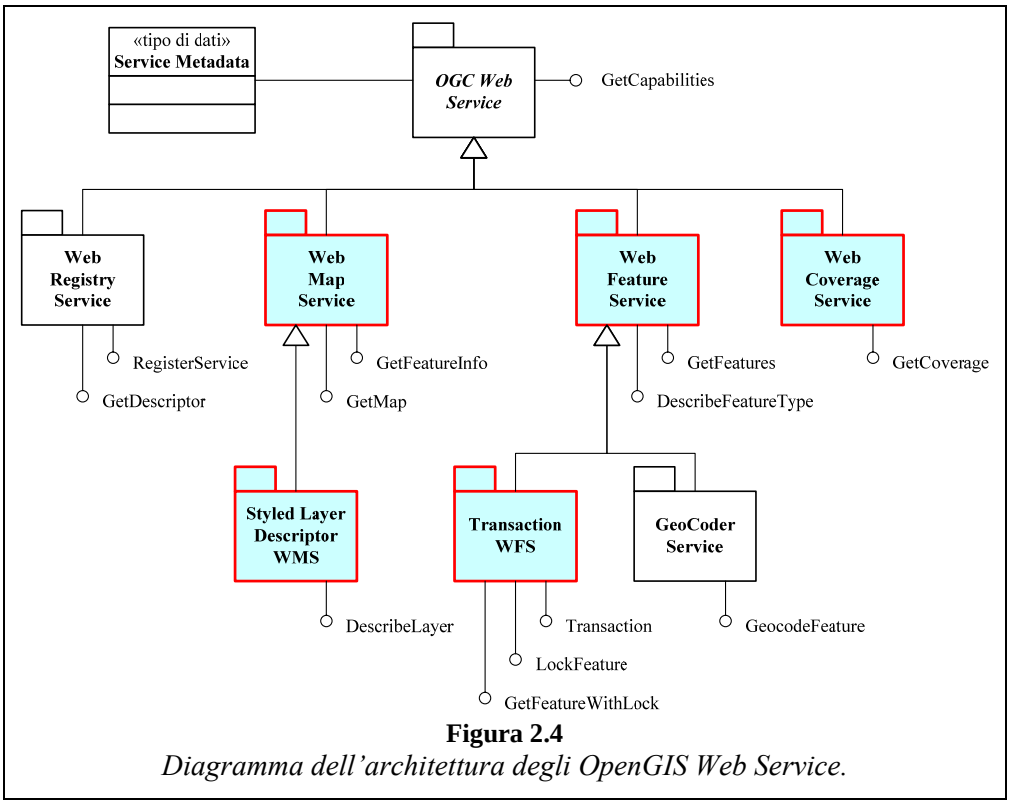


Figura 2.4
Diagramma dell'architettura degli OpenGIS Web Service.

Attualmente, la specifica di implementazione OWS indica il protocollo HTTP come la piattaforma distribuita di riferimento (*Distributed Computing Platform*, DCP) per l'accesso ai servizi, ma non indica quale tecnologia utilizzare per l'implementazione. Si possono trovare quindi implementazioni che sfruttano varie tecnologie (ASP, JSP, PHP, CGI) e

diversi linguaggi di programmazione (C, Python, Java, C#, Vb, VbScript, JScript), a riprova del grado di apertura ed interoperabilità forniti dalle specifiche degli OWS. L'utente finale si troverà di fronte ad una scatola chiusa (*Black Box*) e non si dovrà minimamente preoccupare dei dettagli di implementazione.

Essendo i servizi risorse remote che sfruttano HTTP come DCP, sono identificati tramite un *Unified Resource Locator* di HTTP (HTTP URL); la specifica permette che le URL delle operazioni possano essere diverse rispetto a quella dell'istanza di servizio. Le interfacce dei servizi OWS sono accessibili sfruttando richieste di tipo HTTP-GET e/o HTTP-POST In base a quale metodo si utilizzi, cambia la composizione della URL:

- **HTTP-GET**: la richiesta per invocare l'operazione è costruita nella URL secondo la codifica KVP (*key value pair*). All'indirizzo del servizio è appesa una stringa che inizia con il carattere "?" seguito da un insieme di coppie <chiave,valore> delimitate dal carattere "&". Si avrà quindi una URL composta da `protocollo://host[:porta]/percorso?{key[=value]&}`. Ad esempio:
  `http://webgisserver.isti.cnr.it/mypath/dosomething.aspx?key1=value1&...`
- **HTTP-POST**: il metodo POST è utilizzato in prevalenza, quando si vuol mandare dati contenuti in una form HTML oppure per scavalcare le limitazioni sulle richieste HTTP-GET poste dai web server; i dati sono incapsulati all'interno del *entity body* della richiesta HTTP.

Il servizio risponde con un documento contenente ciò che è stato richiesto (accompagnato sempre dal tipo MIME) oppure con un'eccezione se, ad esempio, il contenuto della richiesta inviata non è valido per l'istanza di servizio.

Come detto, un servizio espone un insieme di operazioni ma secondo il diagramma dell'architettura dei servizi OWS (figura 2.4), una è comune a tutti: ***GetCapabilities***, che restituisce al chiamante un documento che descrive le "capacità" dei servizi OGC.

2.1.3.1. Il Documento Delle Capacità Di Un OWS

La richiesta *GetCapabilities* permette di recuperare il documento dei metadati (meta informazioni) che descrivono la capacità di un server che implementa un servizio OWS; i metadati rendono il server OWS parzialmente auto descrittivo.

Ogni servizio ha una versione che fa riferimento alla revisione della rispettiva specifica di implementazione; quando viene effettuata la richiesta al server OWS, questi confronta la versione specificata nella richiesta con la propria. Se le due versioni non coincidono, è intrapresa una negoziazione secondo versioni alternative; OGC incoraggia implementazioni di più versioni della stessa specifica.

Il documento delle capacità è codificato in XML secondo un appropriato schema XML definito da OGC; il documento delle capacità è diviso in quattro sezioni che ogni istanza di OWS deve avere:

- ***ServiceIdentification***: contiene metadati relativi al servizio, alcuni elementi sono descritti nella tabella 2.1;
- ***ServiceProvider***: riporta informazioni generiche sul gestore del server ove risiede il servizio OWS (istanza di servizio), come: nome del contatto, recapiti del provider, … ;
- ***OperationsMetadata***: per ogni operazione sono definiti l'insieme di URL ed i rispettivi protocolli per invocarle. Un'operazione può essere resa disponibile su altri server e tramite più protocolli, ad esempio HTTP e SOAP (figura 2.3);
- ***Content***: contenuti specifici del dominio del servizio OWS.

Nei paragrafi successivi saranno presentati i servizi OGC presi in esame in questo lavoro di tesi ed i dettagli più importanti delle rispettive specifiche.

Metadati	Descrizione
Name	Nome assegnato all'istanza del OWS.
Title	Titolo rivolto all'utenza umana per identificare il servizio.
Abstract	Descrizione narrativa del servizio.
KeywordList	Parole chiavi utili a fini di ricerca nel catalogo dei servizi.
OnlineResource	URL HTTP di livello superiore del servizio: tipicamente il portale relativo al sito Internet che ospita servizio o dell'azienda proprietaria.
ContactInformation	Informazioni riguardanti una persona fisica che riveste il ruolo di contatto.
Fees	Indica il costo di utilizzo del servizio oppure per il recupero dati. Se il valore è "NONE" significa che non esiste nessuna forma di pagamento.
AccessConstraints	Descrive le restrizione imposta dal gestore del servizio. Se il valore è "NONE" significa che non esiste alcuna restrizione.

Tabella 2.1
Alcuni metadati relativi alla sezione ServiceIdentification.

2.1.3.2. Web Map Service (WMS) v1.1.1

Un WMS produce mappe da dati georeferenziati; una **mappa** è "una rappresentazione visiva dei dati" e non i dati stessi. Tipicamente una mappa è rappresentata da un WMS in vari formati raster (come JPEG, GIF e PNG[23]) e alcuni WMS sono in grado di restituire

[23] Joint Photographic Experts Group, Portable Network Graphics.

mappe in formato vettoriale come Scalable Vector Graphic (SVG) e Web Computer Graphic Metafile (WebCGM).

Un WMS permette di personalizzare una mappa includendo nella richiesta i seguenti parametri:

- informazioni da mostrare nella mappa, uno o più layer;
- stile di visualizzazione da associare ai layer;
- porzione dei dati che costituiscono la mappa (estensione), *bounding box*;
- sistema di riferimento spaziale delle coordinate (*Spatial Reference System*, SRS). Il servizio WMS può fornire mappe sia in sistemi di riferimento proiettati che geografici;
- dimensioni, in pixel, della mappa specificando altezza e larghezza;
- sfondo trasparente.

Si possono identificare tre tipi di *Web Map Service*:

- ***basic* WMS**: permette il semplice recupero di mappe da una banca dati a cui ha diretto accesso. Espone le seguenti operazioni:
  - *GetCapabilities* (obbligatorio): ottiene i metadati relativi al servizio;
  - *GetMap* (obbligatorio): restituisce la mappa richiesta;
  - *GetFeatureInfo* (opzionale): fornisce informazioni sui layer della mappa.
- ***Cascading Web Map Server***: funziona da centro di aggregazione di dati geospaziali provenienti da altri WMS. Rispetto agli altri WMS si comporta come un client, mentre rispetto ad un client funziona come un WMS standard. Questo modello è riferito come *chaining web service*, figura 2.5.

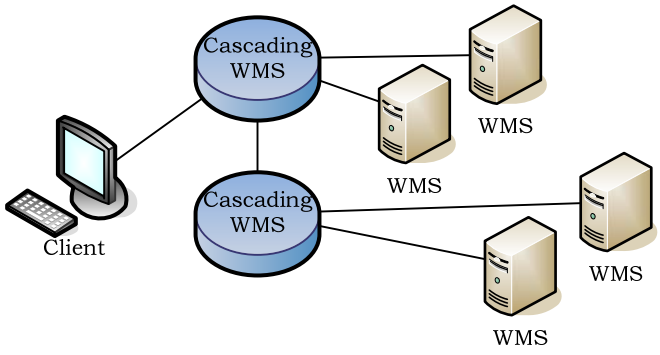


Figura 2.5
Esempio di un Cascading WMS.

- **WMS con specifica *Styled Layer Descriptors*** (SLD): un WMS semplice permette di utilizzare uno specifico stile di visualizzazione, per uno o più layer, scelto da un insieme predefinito. La specifica SLD si applica a quei WMS che si limitano a produrre mappe e non possiedono una propria banca dati. La mappa

viene creata utilizzando i dati provenienti da un servizio OGC *Web Feature Service* e lo stile di visualizzazione applicato è definito dall'utente al momento della richiesta. Le operazioni disponibili sono:

- *DescribeLayer*: restituisce la feature type corrispondente al layer indicato;
- *GetLegendGraphic*: restituisce i simboli di legenda tramite una URL;
- *GetStyle*: utilizzato per recuperare stili definiti dall'utente dal WMS;
- *PutStyle*: utilizzato per memorizzare stili definiti dall'utente nel WMS.

2.1.3.2.1. Operazione GetCapabilities

La prima operazione di un *basic WMS* presa in esame è *GetCapabilities* che restituisce i metadati inerenti al servizio. La parte peculiare del servizio è quella delimitata dal tag XML `<Capability>`; al suo interno vi sono quattro sottosezioni:

1. ***Request***: in cui sono definite le risorse per le operazioni del servizio; stando alla specifica per OWS, le operazioni possono essere disponibili presso altre URL e mediante protocolli distinti (esempio 2.8).

Esempio 2.8	Meta informazioni relative all'operazione GetCapabilities

In questo caso per il protocollo HTTP sono presenti i metodi GET e POST. L'elemento `OnlineResource` indica la URL dove l'operazione è disponibile.

```
<GetCapabilities>
  <Format>text/xml</Format>
    <DCPType>
      <HTTP>
         <Get>
            <OnlineResource
                      xmlns:xlink = "http://www.w3.org/1999/xlink"
                      xlink:type = "simple"
                      xlink:href ="http://hostname/path?"/>
         </Get>
         <Post>
             <OnlineResource
                       xmlns:xlink = "http://www.w3.org/1999/xlink"
                       xlink:type ="simple"
                       xlink:href = "http://hostname/path?"/>
         </Post>
      </HTTP>
```

2. ***Exceptions***: al suo interno sono definiti i formati delle eccezioni restituite dal servizio in caso si verifichino degli errori.
3. ***Layer***: insieme di elementi che rappresentano i layer disponili per creare una mappa; l'inizio di una mappa è preannunciato dall'elemento `<Layer>` (secondo OGC, una mappa dovrebbe contenere almeno un layer). L'elemento `<Layer>` possiede sia proprietà sia attributi (tabella 2.2), inoltre è possibile costruire una struttura gerarchica di layer con livelli di ereditarietà per alcune proprietà ed

attributi. Una gerarchia di layer è detta **categoria** se l'elemento `<Layer>` radice ha la proprietà `<Title>`, ma non `<Name>`. Una categoria si limita a raggruppare layer con attributi e proprietà comuni. Un caso particolare di gerarchia si ha quando l'attributo `<Name>` è specificato, allora è possibile utilizzare l'intero insieme di layer specificando solo il nome della categoria (per un esempio di sezione `<Layer>`, vedere appendice A.1).

4. ***Style***: all'interno di un elemento `<Layer>`, vi possono essere più stili (`<Style>`) di visualizzazione predefiniti e descritti in XML; uno stile contiene inoltre la URL alla legenda associata al layer. Se gli stili sono definiti per la radice di una gerarchia, allora i figli potranno ereditarli.

Metadati	Descrizione
Name	Identifica il layer all'interno del servizio.
Title	Titolo rivolto all'utenza umana per identificare il servizio.
Abstract	Si tratta di una descrizione narrativa del layer.
KeywordList	Delimita un insieme di termini chiavi utili alla ricerca all'interno del catalogo.
SRS	*Spatial Referecnce System*, indica il sistema di riferimento del layer.
LatLonBoundingBox	Definisce il *Minimum Bounding Box*, MBR (estensione), della mappa in EPSG:4326[24] (sistema di riferimento delle coordinate geografiche con datum WGS84).
BoundingBox	Per ogni layer vi può essere un insieme di elementi che definiscono lo MBR secondo un dato sistema di riferimento spaziale.
MetadataUrl	URL che punta alla definizione dei metadati, opzionale.
Style	Uno o più stili di visualizzazione utilizzabili per il layer
FeatureListURL	Definisce un collegamento ad una lista di feature rappresentate nel layer.

Tabella 2.2
Alcuni metadati relativi all'elemento layer.

2.1.3.2.2. Operazione GetMap

Per poter creare ed ottenere una mappa bisogna invocare l'operazione *GetMap*. L'invocazione comporta la specifica di tutti i parametri necessari alla definizione della mappa, oltre che ad esplicitare la versione dell'interfaccia che si intende utilizzare. L'esempio 2.9 mostra una possibile invocazione dell'operazione tramite una richiesta HTTP-GET.

[24] *European Petroleum Survey Group* (EPSG): definisce un sistema di riferimento geodetico per coordinate.

Esempio 2.9	Richiesta *GetMap*

La richiesta indica all'operazione *GetMap* (*REQUEST*) di produrre una mappa composta dai dati di tre layer (LAYER*S*), limitati dalla *bounding box* (*BBOX*) e con gli stili di visualizzazione indicati (*STYLES*).

```
http://www.dummyurl.com/WMS?
                              VERSION = 1.1.0&
                              REQUEST = GetMap&
                              BBOX = 0.0,0.0, 1.0,1.0&
                              LAYERS = Rivers,Roads,Houses&
                              STYLES = CenterLine,CenterLine,Outline
```

Non è obbligatorio specificare la versione del protocollo (*VERSION*), viene utilizzata la più recente.

In questo esempio non è indicato a quale servizio si sta facendo la richiesta in quanto si assume che alla URL specificata sia implementata solo l'istanza di WMS.

2.1.3.2.3. Operazione GetFeatureInfo

Si tratta di un'operazione opzionale e permette di ricevere informazioni sulla feature della mappa se il layer corrispondente è marcato come "interrogabile".

L'operazione consente di ottenere informazioni relative ad un punto indicato tramite le sue coordinate sulla mappa. La specifica da piena libertà su come e cosa il servizio deve restituire [16]. In generale è restituita un'immagine oppure una pagina HTML; l'oggetto restituito è accompagnato dal suo tipo MIME [17].

WMS è un protocollo *stateless* e quindi non tiene traccia delle mappe prodotte; l'esecuzione di un'operazione *GetFeatureInfo* richiede che ogni volta siano forniti i parametri necessari alla ricostruzione della mappa.

2.1.3.3. Web Feature Server v1.0.0 (WFS)

Un WFS è un'interfaccia per permettere ad un client di recuperare, interrogare e modificare dati geospaziali codificati in GML.

I WFS si dividono in due tipi:

- ***Basic WFS***: permette il recupero e l'interrogazione delle feature. Espone le seguenti operazioni (oltre alla *GetCapabilities*):
  - *DescribeFeatureType*: fornisce la descrizione di ogni feature type che il servizio espone;
  - *GetFeature* (obbligatorio): consente di recuperare le feature codificate in GML.
- ***Transaction WFS***: per la modifica di dati esistenti e l'inserimento di nuove feature (codificate in GML). Le operazioni disponibili sono:

- *Transaction*: per definire le operazioni di trasformazione sulle istanze di feature type.
- *LockFeature*: pone un blocco (effettuazione di un'operazione di *lock*) sulle istanze di feature type;
- *GetFeatureWithLock*: si comporta come l'operazione *GetFeature*, solo che il WFS pone un blocco sui dati richiesti.

2.1.3.3.1. Operazione GetCapabilities

In questo caso le sezioni peculiari relative al WFS sono due:

1. *FeatureTypeList*: riporta la lista delle feature type che l'istanza di servizio espone. A differenza del servizio WMS, le feature type non sono strutturate secondo una gerarchia. In testa alla lista sono riportate le feature type e tutte le operazioni su di esse eseguibili, la specifica prevede inoltre che ad ogni feature type sia possibile assegnare operazioni specifiche (vedere appendice A.1). La tabella 2.3 mostra alcuni metadati associati ad ogni feature type;
2. *FilterCapabilities*: Questa sezione non è obbligatoria e riporta un insieme di operazioni che il servizio supporta; tali operazioni, indicate dall'elemento `<Filter>`, sono definite in una specifica OGC [18]. Si hanno principalmente tre tipi di operatori:
   - Operatori Spaziali: permettono di esprimere condizioni in base alle relazioni spaziali che intercorrono fra le feature, come ad esempio intersezione e contenimento. Degno di nota è l'operatore BBOX (*Bounding Box*) impiegato per definire una zona d'interesse rettangolare sui dati. Nonostante possa essere derivata dagli altri operatori,è stata implementata in quanto è di uso frequente in quanto aiuta a ridurre la mole di dati da trasferire;
   - Operatori di Confronto: utilizzati per costruire espressioni basate sugli attributi;
   - Operatori Logici AND, OR, NOT: permettono di comporre espressioni complesse combinando gli altri operatori.

   La specifica prevede operatori aritmetici e la possibilità di invocare funzioni offerte dal server.

Elemento	Descrizione
Name	Nome qualificato da un namespace del feature type.
Title	Titolo, leggibile da persone, per identificare la feature type.
Abstract	Descrizione approfondita del feature type.
Keyword	Termini chiave utili per la ricerca all'interno del catalogo.
SRS	Indica il sistema di riferimento da utilizzare per esprimere lo stato della feature.
Operation	Contiene le operazioni supportate dalla singola feature.
LatLongBoundingBox	Indica il *bounding box* della feature type, espresso secondo il SRS.
MetadataURL	Ha cardinalità "0..*" e definisce un URL relativo ai metadati relativi ai dati geospaziali.

Tabella 2.3

Metadati relativi alle feature type elencate nel documento Capability per servizi WFS.

2.1.3.3.2. Operazione DescribeFeatureType

Genera lo application schema relativo alle feature type fornite dal WFS; nella richiesta vengono specificate di quali feature type si è interessati riceve lo schema XML.

2.1.3.3.3. Operazione GetFeature

Permette il recupero di un insieme feature offerte dal servizio WFS; nella richiesta oltre al nome della feature type (identificato dal *tag* XML `<Name>`) è possibile indicare il numero massimo di istanze di feature type da ricevere. Se è presente la sezione *FilterCapabilities*, sarà possibile indicare restrizioni sui dati. I dati restituiti, per questa versione del servizio WFS (v1.0.0), saranno resi in GML v2.1.2.

Nel caso di un *transaction WFS* è possibile effettuare modifiche alla base di dati; essendo in un ambiente distribuito e sottoposto a tempi di latenza dovuti al protocollo della piattaforma, è inevitabile l'insorgere di problemi di aggiornamento. Lo scenario più frequente prevede due client che aggiornano lo stesso insieme di feature in modo concorrente, portando quindi ad avere dati non consistenti. Un secondo scenario prevede un malfunzionamento sul server, dove risiede l'istanza di WFS, mentre è in esecuzione un insieme di operazioni di modifica sui dati

E quindi necessario un meccanismo che garantisca sia la **serializzabilità** che l'**affidabilità** delle operazioni sui dati.

L'affidabilità permette di prevenire il secondo scenario; si definisce il meccanismo delle **transazioni** come una sequenza di operazioni di lettura e scrittura che iniziano con l'operazione di *begin* e terminano con un'operazione di *commit* se la transazione è andata a

buon fine, altrimenti con un'operazione di *rollback* per riportare la base di dati allo stato precedente all'inizio della sequenza di operazioni. La **serializzabilità** consente di poter eseguire transazioni intercalate come se avvenissero in modo sequenziale. Per introdurre questi meccanismi, la specifica relativa a WFS prevede le operazioni *LockFeature* e *Transaction*.

2.1.3.3.4. Operazione LockFeature

La strategia classica per implementare la serializzabilità delle transazioni è di permettere solo ad una transazione di modificare il proprio insieme di dati. Questo è realizzato ponendo un **blocco** sui dati; si tratta di una marcatura interpretabile dal DBMS. La specifica d'implementazione WFS prevede un **blocco a lungo termine** per ovviare a problemi di latenza della rete. La richiesta per l'operazione di *LockFeature* permette inoltre di definire un tempo di scadenza obbligatoria del blocco specificando nella richiesta il parametro EXPIRY seguito dal tempo espresso in minuti.

Il blocco sulle feature può essere posto in due modi:

1. **esplicito**, definendo la lista di istanze di feature type da bloccare;
2. **condizionale**, vengono bloccate tutte le feature che soddisfano le condizioni di un filtro (restrizione) specificato all'interno dell'elemento `<Filter>`.

Questi due metodi possono coesistere in una richiesta.

Il meccanismo della serializzabilità per servizi WFS prevede molti altri aspetti che esulano questo lavoro, per maggiori dettagli si rimanda a "*Panagiotis A. Vretanos, Web* Feature *Service Implementation Specification v1.0.0*" disponibile tramite il portale del consorzio OpenGIS.

2.1.3.3.5. Operazione Transaction

Questa operazione permette di definire una transazione; è preceduta da un'operazione di *LockFeature* che restituisce l'identificatore di blocco, LOCKID su cui opera la transazione. Nella richiesta, oltre al parametro LOCKID, per ogni tipo di operazione saranno indicate le feature coinvolte.

Le sezioni, di una transazione, che definiscono le operazioni di modifica sono:

- ***Insert***: contiene tutte le nuove feature da inserire; sono codificate in GML e devono essere validate tramite lo application schema relativo alla feature type, ottenuto tramite l'operazione *DescribeFeatureType.* L'operazione restituisce gli identificatori delle nuove feature secondo l'ordine di inserimento;
- ***Update***: questa operazione è simile all'operazione SQL di *update* in quanto si specificano le proprietà da aggiornare (tramite <*nomeProprietà*, *nuovoValore*>,

rispettando il tipo della proprietà) potendo utilizzare un'espressione condizionale espressa tramite l'elemento `<Filter>`;

- ***Delete***: utilizza l'elemento `<Filter>` per eliminare le feature.

Alla fine della richiesta di transazione, è indicato come il blocco deve essere rilasciato; l'operazione di transazione restituisce un documento XML dove è riportato lo stato di successo ed informazioni aggiuntive nel caso siano state aggiunte nuove feature.

2.1.3.4. I Servizi di Catalogo

I servizi di catalogo permettono di recuperare dati geospaziali o servizi basandosi sui metadati che li descrivono (ad esempio i metadati identificati in XML con `<Keyword>` e `<KeywordList>`). OGC ha previsto rispettivamente due servizi:

- *Catalogue Service* (CAT),
- *Web Registry Servic*e (WRS).

2.1.3.4.1. Catalogue Service

Fornisce un'interfaccia per effettuare operazioni di accesso, manutenzione, ricerca e organizzazione su cataloghi di dati spaziali. L'interfaccia consente ad un utente (umano e software) di ricercare informazioni all'interno basi di dati remote e distribuite.

Secondo il modello dell'architettura di riferimento, Figura 2.6, OGC CAT può fare richieste ad un altro OGC CAT (ricerca distribuita tramite interfaccia OGC) oppure verso un *metadata repository* (una base di metadati) tramite un'interfaccia proprietaria.

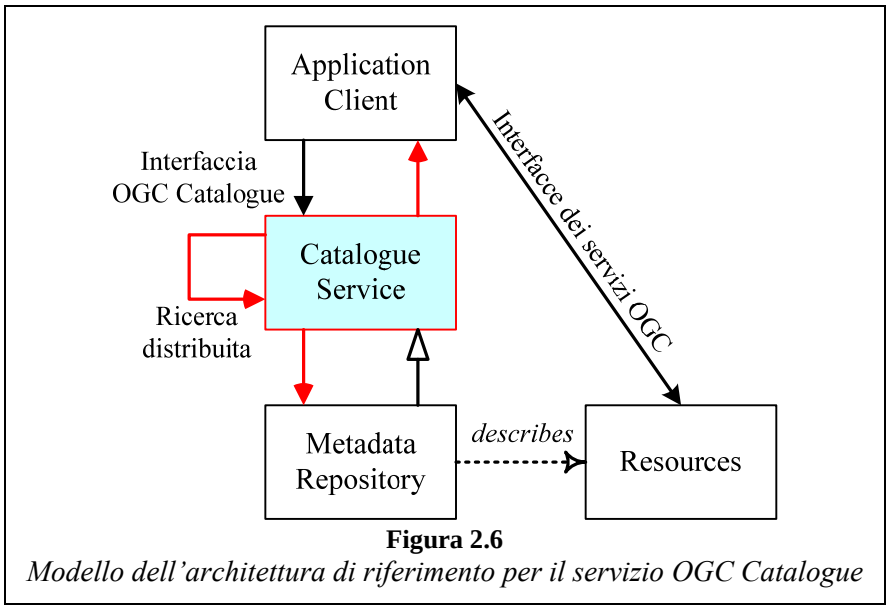


Figura 2.6
Modello dell'architettura di riferimento per il servizio OGC Catalogue

Il servizio OGC CAT espone un proprio linguaggio di interrogazione, *OGC Common catalogue query language* (stile *SQL Select-From-Where*), per il recupero delle informazioni. Tale linguaggio costituisce un'astrazione per interrogare il repository, questo garantisce una sua libera implementazione.

Se l'interrogazione del catalogo riporta dati ritenuti utili, il client ottiene tutte le informazioni per raggiungere le risorse di cui ha bisogno; tali risorse sono altri servizi OGC che espongono l'interfaccia standard definita da OWS.

Il servizio è schematizzato, figura 2.7, mediante 4 classi che offrono le funzionalità necessarie al catalogo:

- ***Discovery*** (obbligatorio): fornisce l'interfaccia necessaria alla ricerca di risorse all'interno del catalogo;
- ***Session***: permette di creare sessioni interattive fra i client e server*;*
- ***BrokeredAccess***: permette di richiedere una risorsa che è registrata all'interno del catalogo ma non visibile all'utente.;
- ***Manager***: permette di creare transazioni per effettuare operazioni di aggiornamento del catalogo.

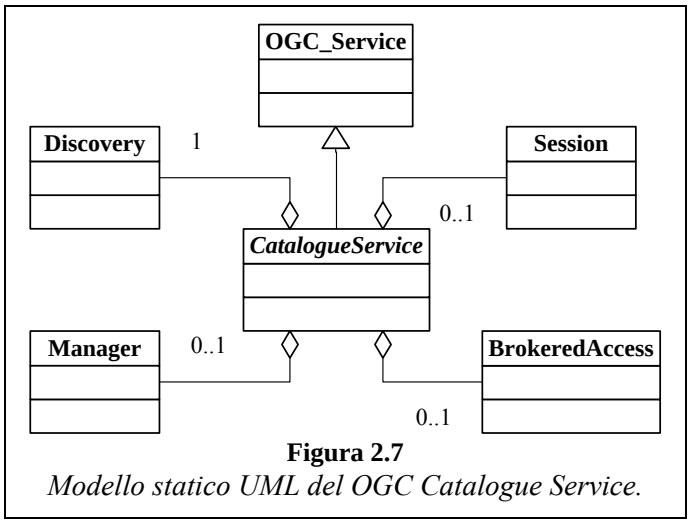


Figura 2.7
Modello statico UML del OGC Catalogue Service.

2.1.3.4.2. Web Registry Service (WRS)

Il servizio WRS definisce un meccanismo per classificare, registrare, descrivere e mantenere informazioni riguardanti risorse basate sulla specifica OWS.

Questo servizio è proposto in un discussion paper del 20 febbraio 2001[19] di cui attualmente non esiste una specifica di implementazione.

Con l'emergere di nuove tecnologie, questo è comprensibile; esistono due discussion paper, uno del 2003 [20] ed un secondo più recente (2005) [21] in cui è proposto l'adozione della specifica "*Universal Description, Discovery and Integration*" (UDDI), affrontata nel capitolo 2.2.4.

2.2. Introduzione ai Web Service

Un'architettura consolidata per l'interoperabilità e l'integrazione di sistemi eterogenei, è costituita da *Service-Object Architecture* (SOA). Si tratta di un modello a componenti per mettere in relazione differenti unità funzionali, detti servizi, attraverso interfacce ben definite. Tali interfacce sono definite in maniera neutra ed indipendente dalla piattaforma hardware, dal sistema operativo e dal linguaggio di programmazione.

La ***Web Service Architecture*** (WSA) risulta essere un'architettura da utilizzare per raggiungere l'obiettivo di SOA.

WSA è composta da tre protocolli fondamentali (ma generici in quanto non legati a ciò che esprimono) sviluppati da Microsoft ed IBM e soggetti a revisione da parte del consorzio W3C. Orizzontalmente, l'organizzazione *Web Service Interoperability Organization* (WS-I) ha il compito di organizzare, integrare, armonizzare e promuovere tale tecnologia.

Secondo la definizione data dal consorzio W3C, a livello concettuale, un web service è "un sistema software che consente un'interazione interoperabile fra macchine attraverso la rete ed ha un'interfaccia descritta in un formato processabile da una macchina (WSDL). I sistemi interagiscono con un web service mediante messaggi descritti tramite una grammatica derivata da XML e tipicamente trasportati tramite HTTP " [22].

In seconda analisi si può affermare che un web service è un'interfaccia verso un insieme di funzionalità costituenti un **servizio** indirizzabile attraverso una URL http. Un servizio è una risorsa astratta che descrive la capacità di eseguire un insieme operazioni [23].

Ad un'interfaccia è associata una descrizione contenuta in un documento, *service description*, espresso in un linguaggio comprensibile dalle applicazioni. Si definisce inoltre la **semantica** di un web service come il comportamento che ci si aspetta dal servizio interagendo con esso. Esprime, quindi, l'effetto dell'invocazione del servizio. L'invocazione delle operazioni di un servizio e l'eventuale risposta, avvengono tramite messaggi codificati secondo un dato formalismo; attualmente i più usati sono i seguenti:

- *XML Remote Procedure Call* (XML-RPC),
- *Simple Object Access Protocol* (SOAP).

L'abbinamento di questi formati con l'uso di HTTP, come piattaforma distribuita, fa sì che i web service costituiscano una piattaforma *open* in quanto non vi è alcun legame né con un determinato sistema operativo né con i linguaggi di sviluppo.

Il comune uso della rete Internet prevede l'interazione fra un'entità server in grado di fornire contenuti ed un web browser capace di visualizzarli; questa è una visione del Web umano-centrica (*Human-Centric Web*). Nell'architettura web service si ha una visione *Application-Centric* del Web, dove lo scambio di contenuti avviene fra applicazioni.

Si può obiettare che la WSA non introduce alcuna innovazione in quanto da sempre si sono sviluppate soluzioni basate su CGI o Java Servlet per la condivisione di dati e procedure. Tali soluzioni, dette *Tight-Coupling*, in quanto sono caratterizzate da interfacce strettamente legate alle funzionalità realizzate. L'architettura web service offre un modello *loose coupling* capace di adattarsi alle evoluzioni delle tecnologie e delle infrastrutture utilizzate per realizzare i servizi.

Se si confronta l'architettura web service con quella consolidata di CORBA (od anche DCOM, EJB)[25], vediamo che questi ultimi forniscono un'architettura orientata agli oggetti di tipo *Tight-Coupling* pensata per creare un ambiente di calcolo distribuito, mentre la tecnologia WSA è pensata per l'interoperabilità per.

Nell'architettura web service, un servizio è una notazione astratta implementata da un **agente**, dove un agente è una porzione di software in grado di inviare e ricevere messaggi. Gli agenti vanno ad implementare le funzionalità che il servizio descrive; ne segue che due agenti distinti possono implementare uno stesso web service.

Lo scambio di messaggi, previsto dall'architettura, avviene fra gli attori descritti in figura 2.8:

- ***Service requester***: è un'entità (*requester entity*: un'applicazione od una persona) che utilizza le funzionalità del servizio[26]. Esso prevede un *requester agent* per lo scambio di messaggi;
- ***Service provider***: un'entità (*provider entity*, un server) in grado di distribuire su Internet uno o più servizi. Tali servizi sono realizzati da *provider agent*;

[25] CORBA, *Common Object Request Broker Architecture;* DCOM, *Distributed Component Object Module*; EJB, *Enterprise Java Bean*.
[26] Nulla vieta che sia un altro *web service*.

- ***Service registry***: può essere considerato come un catalogo dove gli sviluppatori di web service pubblicano il service description relativo al web service. Questo servizio è controllato da un operatore che decide quali informazioni possono esser memorizzate, chi può accedervi e quali operazioni può effettuare sulle registrazioni. Un esempio è il *Microsoft UDDI Business Registry Node* [24];

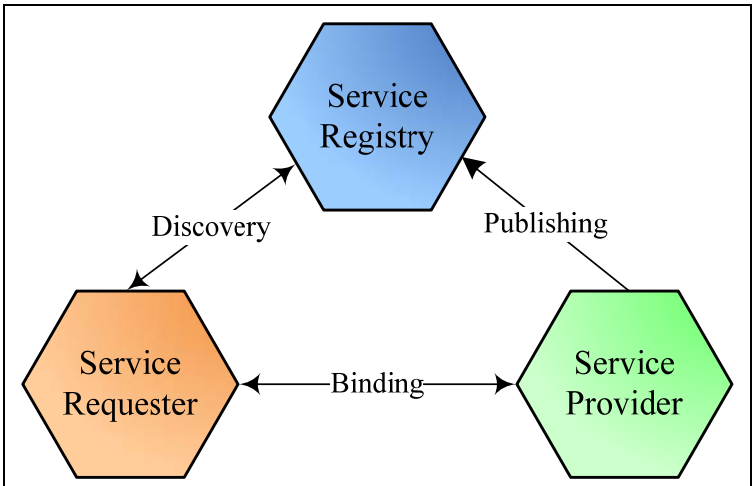


Figura 2.8
Attori ed operazioni dell'architettura dei web service.

Supponendo noto l'indirizzo del servizio, l'integrazione di un web service da parte di un *service requester* avviene tramite l'operazione di *Binding*:

1. sia *requester entity* che *provider entity* concordano (riconoscono) la descrizione del servizio e la semantica che andranno a governare lo scambio di messaggi. Il riconoscimento può avvenire in vari modi: entrambe le entità utilizzano uno standard noto; il *requester entity* espone sia la semantica che il documento di descrizione del servizio, … ;
2. entrambi gli agenti devono implementare/includere sia semantica che descrizione del servizio prima che possa avere inizio l'interazione vera e propria;
3. l'interazione avviene tramite lo scambio di messaggi definiti secondo una grammatica da entrambi nota.

Facendo riferimento alla terminologia utilizzata nel paradigma ad oggetti, si può definire questa procedura come *early binding* in quanto il riferimento al service provider è definito all'interno del requester agent a tempo di compilazione.

Il caso più interessante vede l'assenza dell'indirizzo del servizio; è intrapresa un'operazione preliminare di *Discovery*, da parte del service requester verso il service

registry. Questa operazione permette di ricercare all'interno di un catalogo, il servizio che soddisfa le necessità del *requester entità*.

Questo tipo di integrazione del servizio è definito come *Just-In-Time Integration* e può essere considerata come un *late binding:* essa permette l'integrazione del servizio a tempo di esecuzione.

L'operazione di *Publishing* consente ad un service provider di inserire lo schema e la semantica del servizio nel catalogo, gestito dal service registry.

2.2.1. Lo Stack Protocollare

Come detto, l'architettura WSA è una implementazione di SOA che prevede l'uso di specifici elementi definiti mediante una pila protocollare (figura 2.9) in cui ogni strato risolve un determinato compito:

- ***Discovery***: fornisce il meccanismo che consente ad un service requester di recuperare la descrizione di un servizio; il meccanismo più utilizzato è *Universal Description, Discovery and Integration* (UDDI);
- ***Description***: implementa la descrizione del servizio (service description) facendo riferimento agli strati sottostanti. Al suo interno sono riportati i formati ed i protocolli utilizzati per lo scambio di messaggi, il protocollo di "impacchettamento" e di trasporto. Per far questo, è utilizzato un linguaggio per la creazione del documento *service description*. Il linguaggio utilizzato è *Web Service Description Language* (WSDL);
- ***Packaging***: uno dei passi fondamentali per raggiungere l'interoperabilità è poter comunicare tramite un linguaggio comprensibile da tutte le entità. L'uso di HTML come formato da passare allo strato trasporto non è una scelta indicata in quanto è un linguaggio strettamente legato alla rappresentazione dei dati. La scelta più naturale ricade su un protocollo di comunicazione basato su XML: *Simple Object Access Protocol* (SOAP). SOAP non è l'unico protocollo utilizzato, XML *Remote Procedure Call* è completamente basato su HTTP e consente lo scambio di messaggi codificati in XML tramite richieste HTTP-GET / HTTP-POST e risposte poste nell'intestazione HTTP-RESPONSE di HTTP;
- ***Transport***: questo strato include i protocolli utilizzati per mettere in comunicazione molteplici applicazioni. Facendo riferimento alla pila protocollare di Internet, questo strato corrisponde al livello *Application*.

Troviamo quindi un insieme di protocolli quali HTTP, FTP, SMTP, TCP e *Block Extensible Exchange Protocol* (BEEP);

- ***Network***: coincide con lo strato di Trasporto ed instradamento della pila protocollare di Internet: TCP/IP.

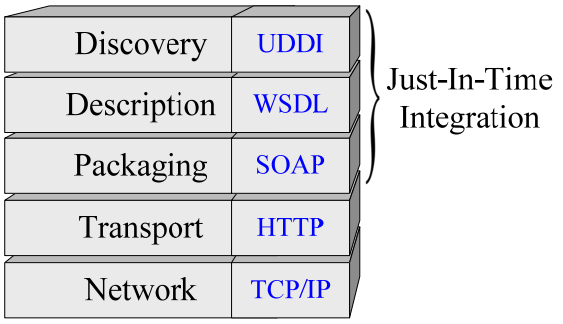


Figura 2.9
Pila protocollare dei web service.

I primi tre strati sono responsabili del meccanismo di *Just-In-Time-Integration*, come mostra la figura 2.9.

Verranno di seguito presentati solo gli strati più alti della pila protocollare dei *web service*, tralasciando quelli affini alla pila protocollare di Internet.

2.2.2. Simple Object Access Protocol

Simple Object Access Protocol, SOAP, è un protocollo basato su XML per lo **scambio di informazioni** fra applicazioni; tipicamente è utilizzato assieme ad HTTP per effettuare richieste per *Remote Procedure Call* (RPC).

A differenza delle infrastrutture CORBA, Java RMI e DCOM, SOAP è completamente basato su XML; questo garantisce l'assoluta indipendenza dalla piattaforma.

2.2.2.1. Specifiche SOAP

Per rendere possibile un'interazione fra due agenti, è necessario stabilire delle regole per l'invocazione di un metodo di web service, lo scambio di messaggi e la codifica dei dati. Per far ciò, SOAP prevede tre specifiche:

- ***Envelope Specification***: definisce le regole d'incapsulamento dei dati da trasferire, come ad esempio: nome di metodi, parametri formali e valori da restituire;
- ***Data Encoding Rule***: regole di codifica dei tipi di dato. Ad esempio per rappresentare un valore, una data od una valuta;

- **RPC *Convention***: definisce le convenzioni per la messaggistica unidirezionale e bidirezionale (RPC) fra agenti.

2.2.2.2. Note sul Paradigma di Comunicazione

SOAP può essere utilizzato in due contesti:

- ***Electronic Document Interchange***, EDI: modello di messaggistica impiegato in abito commerciale per scambio di documenti finanziari. Tipicamente questo modello impiega la specifica *electronic business XML* (ebXML) che adotta SOAP come protocollo di messaggistica [25]. In questo caso si ha un modello detto *SOAP document-style;*
- ***Remote Procedure Call***, RPC: modello standard d'interazione fra applicazioni, basato su XML, dove un agente può invocare un metodo (operazione) esposto da un altro agente software. Il modello, in questo contesto, è definito *SOAP RPC-style*.

2.2.2.3. Il Messaggio SOAP

Un messaggio SOAP, creato secondo la specifica, è composto principalmente da tre sezioni (figura 2.10):

1. ***Envelope***: elemento obbligatorio e costituisce la radice del documento. Fornisce un sistema d'impacchettamento del messaggio in quanto lo delimita tramite gli elementi XML `<Envelope>`. Da notare che la versione della specifica SOAP non è indicata come per XML tramite un numero, bensì tramite namespace, es:

SOAP 1.1:	http://schemas.xmlsoap.org/soap/encoding/.
SOAP 1.2[27]:	http://www.w3.org/2001/09/soap-encoding.

2. ***Header***: è paragonabile alle intestazioni di HTTP, si tratta di una sezione opzionale che consente di accludere un insieme di elementi come firme digitali, account per sistemi di pagamento ed altro;
3. ***Body***: obbligatorio, incapsula il *payload* del messaggio SOAP. Tipicamente richieste e risposte di invocazioni di RPC. In caso si presentino errori, SOAP prevede una sezione *Fault* per riportare una descrizione dettagliata degli errori.

[27] Riferito anche come SOAP 2.0 per le sostanziali differenze da SOAP 1.1 .

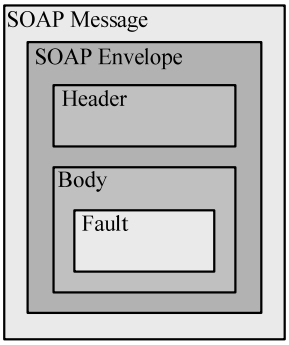


Figura 2.10
Schema di un tipico messaggio SOAP

Supponendo di avere un web service che espone un metodo "DummyOp", l'esempio 2.10 mostra la relativa richiesta SOAP effettuata da un agent requester.

Esempio 2.10 | Messaggio SOAP per l'invocazione del metodo "DummyOp"

```
<soap:Envelope
      xmlns:xsi  = "http://www.w3.org/2001/XMLSchema-instance"
      xmlns:xsd  = "http://www.w3.org/2001/XMLSchema"
      xmlns:soap = "http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <DummyOp xmlns="http://www.dummyUrl.eu/DummyWebservice">
      <parameter1>ValoreParametro1</parameter1>
      <parameter2>ValoreParametro2</parameter2>
    </DummyOp>
  </soap:Body>
</soap:Envelope>
```

Al centro è stata evidenziata in verde l'invocazione vera e propria del metodo appartenente al servizio; gli elementi XML "`parameter1`" e "`parameter2`" contengono i valori dei parametri formali del metodo.

SOAP prevede due insiemi di tipi per i parametri formali e per i valori restituiti:

- *Simple*: sono tutti quei tipi che rappresentano un solo valore, come ad esempio un intero, un float od una valuta;
- *Compound*: tipi che rappresentano un raggruppamento di tipi *Simple*, ad esempio: vettori, record, tipo data.

La loro codifica segue la convenzione utilizzata in XML e definita dallo XML *Schema Part* 2 [26].

2.2.2.4. Note su SOAP & HTTP

Come già accennato, SOAP non è strettamente legato ad un protocollo di trasporto, ma attualmente la sua specifica prevede solo HTTP in quanto risulta il più utilizzato.

Una richiesta SOAP è inviata al server web solo tramite il metodo HTTP-POST, all'interno della richiesta HTTP deve essere presente l'intestazione "`SOAPaction`". Si

tratta di un URI, utilizzato per indicare lo scopo della richiesta senza dover esaminare l'intero contenuto del messaggio.

2.2.3. Web Service Description Language

Web Service Description Language, WSDL, contribuisce all'interoperabilità fra applicazioni indipendentemente dal protocollo che queste utilizzano per codificare i messaggi. E' una specifica utilizzata per **descrivere** un web service attraverso una grammatica XML. Tale descrizione rappresenta un "patto" fra il service provider ed il service requester.

Attraverso WSDL, un client può localizzare un servizio, conoscerne ed invocarne metodi offerti. Per far questo, la specifica prevede sei elementi principali (figura 2.11):

1. ***Definition***: rappresenta la radice del documento e,
   - contiene il nome del web service,
   - dichiara i namespace utilizzati all'interno del documento.
2. ***Type***: descrive come SOAP utilizza la codifica per i tipi di dato, prevista da XML Schema Part 2;
3. ***Message***: descrive la messaggistica unidirezionale, definisce il nome del messaggio ed uno o più elementi che fanno riferimento ai "valori" che il messaggio trasporta;
4. ***portType***: elemento di fondamentale importanza poiché permette di definire l'interfaccia del *web service*. Combinando più elementi *Message* è possibile creare un metodo (**operazione**). WSDL supporta quattro tipi di operazioni:
   - *One-Way*: ricezione di un messaggio da parte del servizio, messaggistica unidirezionale;
   - *Request-Response*: il servizio riceve una richiesta ed inoltra una risposta, messaggistica bidirezionale;
   - *Solicit-Response*: il servizio invia un messaggio ed attende un messaggio di risposta;
   - *Notification*: il servizio invia un messaggio verso molteplici destinatari, per una comunicazione stile *broadcast*. Non si aspetta risposte.
5. ***Binding***: indica come il servizio sarà implementato;
6. ***Service***: contiene il riferimento (mediante URL) per invocare il servizio.

Oltre agli elementi sopra elencati, n'esistono due secondari, detti "di utilità":

1. ***Documentation***: al cui interno è riportata la documentazione destinata all'utenza umana;

2. ***Import***: consente di importare documenti WSDL o schemi XML.

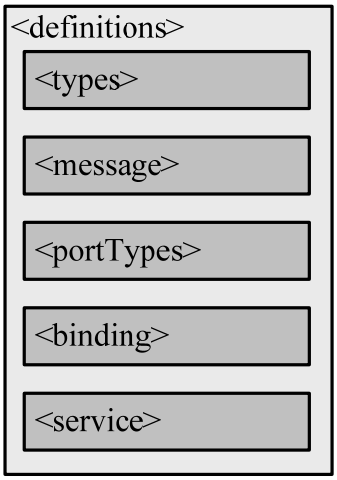


Figura 2.11
Struttura del documento WSDL.

WSDL fornisce un modo per raggruppare elementi XML semplici in modo da rappresentare l'interfaccia di un web service; la figura 2.12 mostra questa strutturazione. L'interfaccia (*portType*) di un web service è composta da operazioni ed ognuna di queste è composta da uno o più messaggi (tipicamente per RPC sono due).

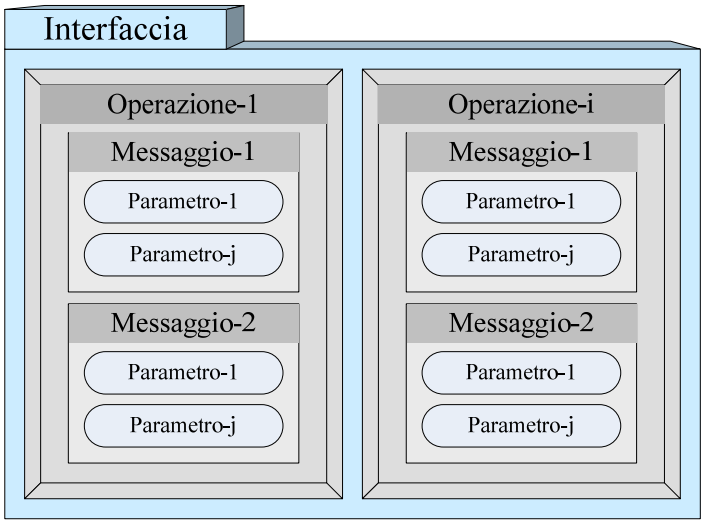


Figura 2.12
Composizione dell'interfaccia di un web service tramite elementi descritti in WSDL.

L'esempio 2.12 illustra un frammento del documento, contenente il service description WSDL relativo al web service `GeoProcessingService` discusso nel capitolo 4.

Esempio 2.11	Frammento WSDL del web service `GeoProcessingService`

Lo schema sotto riportato è stato semplificato eliminando i namespace non direttamente pertinenti a WSDL.

Elemento radice *Definition*:

```
<wsdl:definitions xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
                  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
                  xmlns:tns="http://.../GeoSpatialOperationsService/"
                  xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
                  targetNamespace="http://.../GeoSpatialOperationsService/"
                  xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
```

Type: è definito il tipo dei parametri formali ed il valore restituito del metodo; in questo caso il parametro formale è un tipo composto da due elementi *simple* (tipo *compound*):

```
      <s:element name="IntersectionLayer">
        <s:complexType>
          <s:sequence>
            <s:element minOccurs="0" maxOccurs="1"
                       name="request" type="s:string" />
            <s:element minOccurs="0" maxOccurs="1"
                       name="layerName" type="s:string" />
          </s:sequence>
        </s:complexType>
      </s:element>
      <s:element name="IntersectionLayerResponse">
        <s:complexType>
          <s:sequence>
            <s:element name="IntersectionLayerResult"
                       minOccurs="0" maxOccurs="1" type="s:string" />
          </s:sequence>
        </s:complexType>
      </s:element>
```

Message: i due messaggi, in ordine, di richiesta e risposta che compongono il metodo del web service:

```
  <wsdl:message name="IntersectionLayerSoapIn">
    <wsdl:part name="parameters" element="tns:IntersectionLayer" />
  </wsdl:message>
  <wsdl:message name="IntersectionLayerSoapOut">
    <wsdl:part name="parameters" element="tns:IntersectionLayerResponse" />
  </wsdl:message>
```

portType: definizione dell'operazione tramite i due messaggi:

```
  <wsdl:portType name="GeoProcessingServiceSoap">
    <wsdl:operation name="IntersectionLayer">
      <documentation xmlns="http://schemas.xmlsoap.org/wsdl/">
         Performs the Intersection between two layers.
      </documentation>
      <wsdl:input message="tns:IntersectionLayerSoapIn" />
      <wsdl:output message="tns:IntersectionLayerSoapOut" />
    </wsdl:operation>
```

Binding: l'attributo `type` indica la "porta" del collegamento (l'interfaccia). L'attributo `style` di `<operation>` può valere "*rpc*" o "*document*". Questo dipende dal contesto applicativo del protocollo SOAP, rispettivamente RPC o EDI:

```
<wsdl:binding name="GeoProcessingServiceSoap"
```

```
                type="tns:GeoProcessingServiceSoap">
 <wsdl:operation name="IntersectionLayer">
      <soap:operation soapAction="http://.../IntersectionLayer"
                      style="document" />
      <wsdl:input>
        <soap:body use="literal" />
      </wsdl:input>
      <wsdl:output>
        <soap:body use="literal" />
      </wsdl:output>
       </wsdl:operation>
```

Service: riporta la URL per localizzare il servizio ed il nome del metodo per invocarlo:

```
<wsdl:service name="GeoProcessingService">
    <documentation xmlns="http://schemas.xmlsoap.org/wsdl/" />
    <wsdl:port name="GeoProcessingServiceSoap"
               binding="tns:GeoProcessingServiceSoap">
      <soap:address location="http://.../GeoProcessingService.asmx" />
    </wsdl:port>
   </wsdl:service>
```

2.2.4. Universal Description, Discovery and Integration

Universal Description, Discovery and Integration (UDDI) è una specifica tecnica che formalizza operazioni di descrizione, **ricerca** ed **integrazione** di web service, presentata nel 2000 come risultato della collaborazione fra tre aziende del settore informatico (Microsoft, IBM ed Ariba) che hanno poi invitato altre aziende a collaborare.

L'architettura di UDDI prevede tre elementi principali:

- *UDDI Data Model,*
- SOAP-API,
- *UDDI Business Registries.*

2.2.4.1. UDDI Data Model

Si tratta di uno schema XML per descrivere imprese e *web service* dove, al centro del modello, vi sono quattro tipi di informazioni:

- ***Business Entity***: include informazioni relative dell'azienda, quali: nome descrizione, indirizzo, contatti. Ogni azienda è identificata da un valore unico[28];
- ***Business Service***: per uno o più web service riporta nome, descrizione ed una lista opzionale di *binding template*;
- ***Binding Template***: indica al requester agent come e dove accedere al web service;

- ***tModel***: significa modello tecnico (*technical model*) e fornisce puntatori a specifiche tecniche esterne. Queste specifiche indicano come interfacciarsi con i metodi del servizio. Ciò porta a concludere che tali specifiche altro non sono che documenti WSDL. Questo modello è applicabile anche ad altre risorse come pagine web, documenti, altri tipi di servizi (CORBA).

2.2.4.2. SOAP-API

Per interagire coi dati relativi ai servizi, all'interno dei registri, la specifica prevede un insieme di API basate su SOAP. Un registro è una base di dati che rappresenta lo *UDDI Data Model*. Le API vanno a costituire due interfacce:

- *Publishing*: prevede sedici operazioni per consentire ad un'azienda di registrare i propri servizi;
- *Inquiry* (o *Discovery*): dieci operazioni per ricercare ed ottenere, dai registri, informazioni relative ad un servizio od un'azienda.

Un servizio, all'interno del registro, è indicato come *UDDI business service*.

2.2.4.3. UDDI Business Registry

Noto come *UDDI cloud services*, rappresenta un insieme di "siti operatori" che implementano la specifica UDDI. Un esempio di operatori che offrono pubblicamente il servizio sono Microsoft, NTT COM ed IBM mentre Hewlett-Pakard sta pianificando l'adesione [27].

Un *UDDI Business registry* può essere considerato come un web service che espone un'interfaccia SOAP per interagire con una base di dati.

Il *cloud service* fornisce una banca dati logicamente centralizzata ma fisicamente distribuita; le informazioni memorizzate, una volta pubblicate, sono periodicamente replicate sugli altri nodi (ogni 24 ore). Si tratta quindi di una rete globale di registri UDDI federati (*Federated UDDI Registries*) che posseggono la stessa interfaccia SOAP per pubblicare e ricercare informazioni, figura 2.13.

---

[28] Ad esempio, per Microsoft il valore è *0076b468-eb27-42e5-ac09-9955cff462a3*.

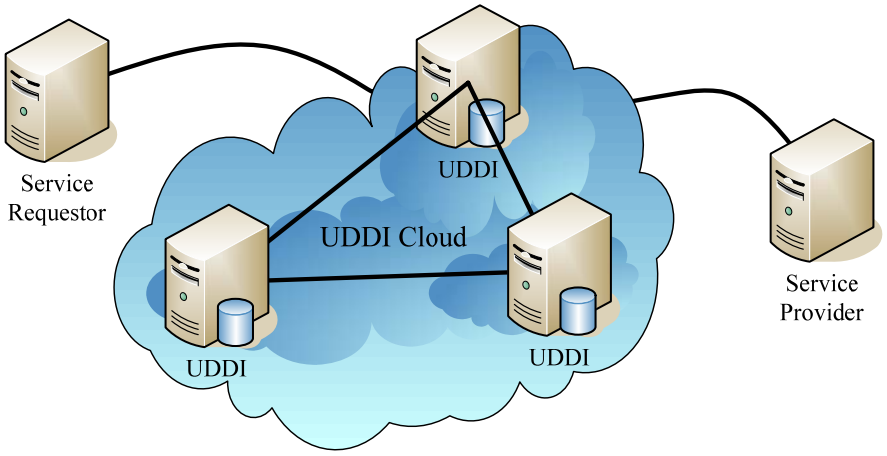


Figura 2.13
Esempio di rete globale di registri UDDI federati.

I dati mantenuti all'interno dei registri UDDI sono tipicamente classificati in tre categorie:

- *White Pages*: tutte le informazioni generiche relative alle compagnie registrate (nome, recapiti, indirizzi, contatti);
- *Yellow Pages*: classificazione generale dei dati secondo la compagnia od il servizio offerto;
- *Green Pages*: informazioni tecniche relative al web service, generalmente un puntatore ad una specifica esterna (*tModel*) ed un indirizzo per invocare il *web service* (URL).

Nel caso in cui UDDI mantiene informazioni relative ad un web service SOAP, nella categoria dei *Green Pages*, il *tModel* sarà costituito da il description service definito in WSDL.

Da notare che UDDI può esser utilizzato per pubblicare informazioni relative ad altri servizi web come:

- pagine web,
- indirizzi e-mail
- servizi basati su CORBA, Java RMI, DCOM, …

2.2.4.4. Publishing & Discovery

Un service provider, per pubblicare web service presso un sito operatore UDDI (service registry), deve seguire i seguenti cinque passi:

1. registrarsi come UDDI *business entity*, quindi come entità aziendale in grado di offrire servizi, presso un operatore UDDI. Generalmente ogni operatore offre l'accesso secondo i propri termini d'uso. La registrazione permette di avere un account per accedere al registro;

2. immettere informazioni riguardanti l'azienda. Più le informazioni saranno precise e dettagliate e più saranno utili al momento della fase di *Discovery*;
3. registrare i servizi offerti dall'azienda;
4. per ogni *UDDI business service* specificare informazioni quali la categoria, il campo di applicazione (finanziario, statistico, meteorologico, …) del servizio;
5. registrare il documento di service description di ogni servizio specificando la URL per poterlo riferire.

Questi passaggi sono possibili grazie all'interfaccia di *publishing*; esistono molti strumenti per sviluppare applicazioni di registrazione, ad esempio *UDDI for Java* (UDDI4J) di IBM, ma tipicamente i siti operatori offrono interfacce basate su *form* HTML (come IBM e Microsoft) [28]. Da notare che, l'autenticazione e la comunicazione verso gli operatori avviene secondo un protocollo sicuro (HTTPS).

La ricerca di un servizio o di un'azienda avviene compilando una richiesta SOAP ed inviandola ad un operatore UDDI. Se la richiesta esplicita un preciso servizio, l'operatore UDDI invierà il service description. Altrimenti, se la richiesta è un'interrogazione, saranno restituite le informazioni relative alle compagnie ed i servizi che soddisfano la richiesta.

E possibile ricercare servizi e compagnie senza utilizzare direttamente l'interfaccia di *discovery* esposta dagli operatori; Microsoft offre un'interfaccia basata su *form* HTML per consultare i registri, ma esistono altri strumenti come *UDDI Browser* di SOAPClient.org [29].

2.2.5. OpenGIS e Web Service

Attualmente le specifiche del consorzio OGC non prevedono l'adozione dell'architettura web service, ma non è nemmeno ipotizzabile che possa essere ignorata dal lavoro del consorzio.

Con la stesura del *OpenGIS Web Service* 2.0 (OWS2), è stato redatto un *Discussion Paper* [7] in cui sono presi in esame i tre elementi principali dell'architettura WSA.

2.2.5.1. SOAP

Uno dei temi principali riguardanti l'adozione di SOAP consiste nell'affrontare le problematiche di incapsulamento di dati binari di grandi dimensioni (*Binary Large Object*, BLOB) all'interno del messaggio.

Essendo SOAP basato su XML, per includere dati di tipo BLOB , si possono prevedere due strategie:

- Rappresentazione del dato secondo la codifica BinHex o Base64. Questo porta all'introduzione di due passi forzati di codifica e decodifica con un conseguente degrado delle prestazioni ed aumento della quantità di banda necessaria per trasferire il messaggio.
- Inserimento del dato BLOB come **allegato** nel messaggio SOAP, utilizzando lo standard MIME. Questa soluzione è contemplata come possibile caratteristica di SOAP 2.0 [30]. È definito un *Compound SOAP Structure*, composto dal messaggio SOAP (*Primary SOAP Message Part*) ed una parte secondaria (*Secondary Part*) costituito dall'allegato. Come mostrato in figura 2.14, il messaggio SOAP è delimitato da *SOAPMessage,* mentre gli allegati sono contenuti all'interno di *SecondaryBagPart*.

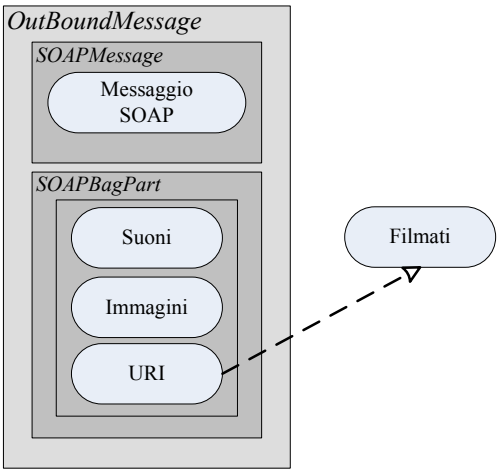


Figura 2.14
Compound SOAP Structure per immagini, suoni e filmati. I filmati in questo caso sono riferiti tramite una URI.

2.2.5.2. WSDL

OWS2 segue le raccomandazioni date dal WS-I per quanto riguarda la definizione degli schemi, ma non considera necessaria la completa compatibilità. La specifica per i web service non contempla l'operazione di *binding* tramite HTTP-POST ed HTTP-GET (KVP *binding*), al contrario di OWS. Esistono alcuni problemi di adattamento dallo standard di codifica KVP caso per caso.

2.2.5.3. UDDI

Per quanto riguarda UDDI non vi è niente da segnalare in quanto una volta che OWS2 rispecchia le raccomandazioni di WS-I, i servizi possono essere pubblicati all'interno dei registri UDDI.

Bibliografia

- Open Geospatial Consortium Inc. (OGC) , http://www.opengeospatial.org/;
- Carl Reed, "*Technical Committee Policies and Procedures*": 12/05/2005;
- John Herring, "*The OpenGIS™ Abstract Specification Topic 1:* Feature *Geometry (ISO 19107 SpatialSchema), Version 5*": 10/05/2001;
- Cliff Kottman , "*The OpenGIS™ Abstract Specification Topic 8: Relationships Between* Feature*s, Version 4*": 26/03/1999;
- Wenjue Jia, Yumin Chen, Jianya Gong, Aixia Li; State Key Laboratory for Information Engineering in Surveying, Mapping and Remote Sensing, Wuhan University, "*Web Service Based Web* Feature *Service*"
- Simon Cox, Adrian Cuthbert, Ron Lake, Richard Martell, "*OpenGIS® Geographic Markup Language (GML) Implementation Specification, v2.1.2*" 17/09/2002;
- Jeff de La Beaujardière, "*Web Map Service Implementation Specification v1.1.1*": 16/01/2002;
- William Lalonde, "*Styled* Layer *Descriptor Implementation Specification v1.0.0*": 19/09/2002;
- Panagiotis A. Vretanos "*Web* Feature *Service Implementation Specification v1.0.0*": 19/09/2002;
- Douglas Nebert, Arliss Whiteside "*OpenGIS® Catalogue Services Specification v2.0*": 11/05/2004;
- Louis Reich, "*Web Registry* Server *Discussion Paper v0.0.2*":26/01/2001;
- Josh Lieberman, Lou Reich, Peter Vretanos, "*OGC Web Services UDDI Experiment*": 17/02/2003;
- Jérôme Sonnett, "*OWS 2 Common Architecture: WSDL SOAP UDDI*" 17/02/2005.
- Ethan Cerami, "*Web Services Essentials Distributed Applications with XML-RPC, SOAP, UDDI & WSDL*": prima edizione Febbraio 2002;

📖 Eric NewComer, "*Understanding Web Services- XML, WSDL, SOAP and UDDI*": Maggio 2002;

Capitolo 3

Strumenti

3.1. GeoMedia WebMap

Per poter creare un insieme di web service in grado di svolgere operazioni GIS è necessario disporre di un motore GIS in grado di offrire un insieme di API per eseguire operazioni di manipolazione sui dati geospaziali.

Dato che il consorzio OpenGIS ha definito alcuni servizi (oltre a WFS, WMS, CAT), alcune operazioni come la trasformazione di coordinate da un sistema di riferimento ad un altro, sono disponibili sotto forma di servizi OWS. In realtà la quantità di specifiche implementate e rese pubbliche sono ancora poche; le implementazioni delle specifiche OWS1 attualmente rese disponibili, possono essere classificate come:

- **compatibili**, come nella maggior parte dei casi e quindi non si ha la piena certezza dell'aderenza con la specifica OWS1;
- **conformi**, in generale sono quelle per i servizi di distribuzione di dati, mostrati in figura 2.4.

Soprattutto non esistono specifiche di implementazione relative alla distribuzione via web di funzionalità ed operazioni geografiche.

Esistono varie soluzioni software GIS proprietarie ed Open Source; in questo lavoro di tesi è stato impiegato *GeoMedia Web Map Professional* 5.01 (GWM) di Intergraph, in quanto si tratta di un ambiente GIS per l'elaborazione di dati spaziali e la pubblicazione dei risultati su web. E' stato scelto perché offre:

- un motore GIS: un insieme di meccanismi e funzionalità che supportano la gestione dei dati geografici ed operazioni su di essi;
- un ambiente di sviluppo organizzato ad oggetti;
- supporto per formati di sorgenti dati proprietari come MGE, MicroStation, ARC/INFO, ArcView, MapInfo, AutoCAD, Oracle8i Spatial Object Model, SQL Server, MS Access, ODBC;
- la conformità con le specifiche per WFS v1.0, WMS v1.1.1 e GML v2.1.2.[29] che lo rendono interoperabile con altre soluzioni software;

[29] Intergraph è un membro storico del consorzio OGC.

- la composizione di mappe da dataset geografici con formati e sistemi di riferimento spaziali eterogenei;
- vari formati di restituzione cartografica fra cui due vettoriali: *Scalable Vector Graphic* (SVG) e *Computer Graphic Metafile* (CGM).

L'ambiente di sviluppo fornito da GWM si basa sulla tecnologia COM e quindi progettato per piattaforme Win32; tale tecnologia permette di accedere alle API di GWM sfruttando tutti i linguaggi compatibili COM quali Visual Basic, Visual C++, Delphi. Nello specifico, GWM per distribuire contenuti geospaziali è combinato assieme alla tecnologia *Active* Server *Page* (ASP).

Il modello ad oggetti consente di:

- gestire il modello **relazionale** dei dati. Sono previste un insieme di classi per rappresentare tutti gli aspetti e meccanismi per la manipolazione relazionale dei dati geografici;
- gestire la componente spaziale dei dati geografici, permettendo di manipolare:
  - la **geometria** dei dati;
  - i **sistemi di riferimento delle coordinate** (proiezione e trasformazione): impostazione, modifica, creazione di sistemi di riferimento spaziale degli oggetti spaziali;
- effettuare la **rappresentazione cartografica**: copre tutti gli aspetti legati alla creazione di una mappa: impostazione del sistema di riferimento di mappa, vestizione degli strati informativi, creazione di *hotspot* e *tooltip*[30], selezione del formato grafico di restituzione (raster o vector);
- eseguire **trasformazioni**: effettuare operazioni su più dataset geografici, basate sulla geometria o sugli attributi, su più dataset;

Come detto, GWM è una piattaforma di sviluppo e la distribuzione di informazioni geografiche su web e quindi si appoggia ad un web server.

Un'applicazione, composta ad esempio da pagine ASP, utilizza le classi che il sistema rende disponibile mediante il web server che ha il compito di istanziarle.

Per quanto riguarda GWM, il web server istanzia solo una classe principale con cui è possibile accedere al servizio di sistema "*MapServer.exe*" che costituisce il motore GIS (figura 3.1). Tale servizio, rappresentato da un processo, è il motore di GWM; le

[30] Un *hotspot* è un punto attivo sulla mappa; tale punto può reagire ad un evento di *mouse* tramite un determinato comportamento (aprire pagine HTML, mostrare immagini, eseguire script,…). Un *tooltip* è un'etichetta, associata alle feature della mappa, che riporta le proprietà descrittive della feature.

successive classi vengono istanziate direttamente dal servizio in modo che gli oggetti creati appartengano allo stesso ambiente dell'istanza di servizio.

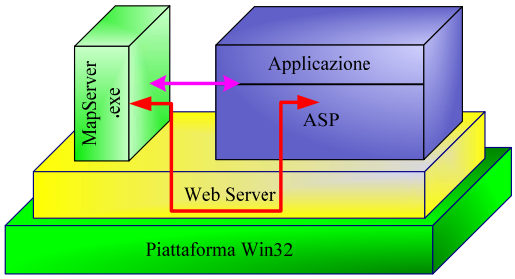


Figura 3.1

Architettura di una tipica applicazione ASP che sfrutta API di GWM. La freccia rossa indica la prima interazione fra l'applicazione e GWM tramite web server, mentre l'altra rappresenta le successive interazioni dirette.

3.1.1. Introduzione alle Classi di GWM

Per illustrare il funzionamento e le relazioni fra le classi fornite da GWM è stato utilizzato il diagramma[31] in figura 3.2; di tutte le classi disponibili, la figura illustra solo la porzione minimale per accedere, manipolare e restituire graficamente i dataset.

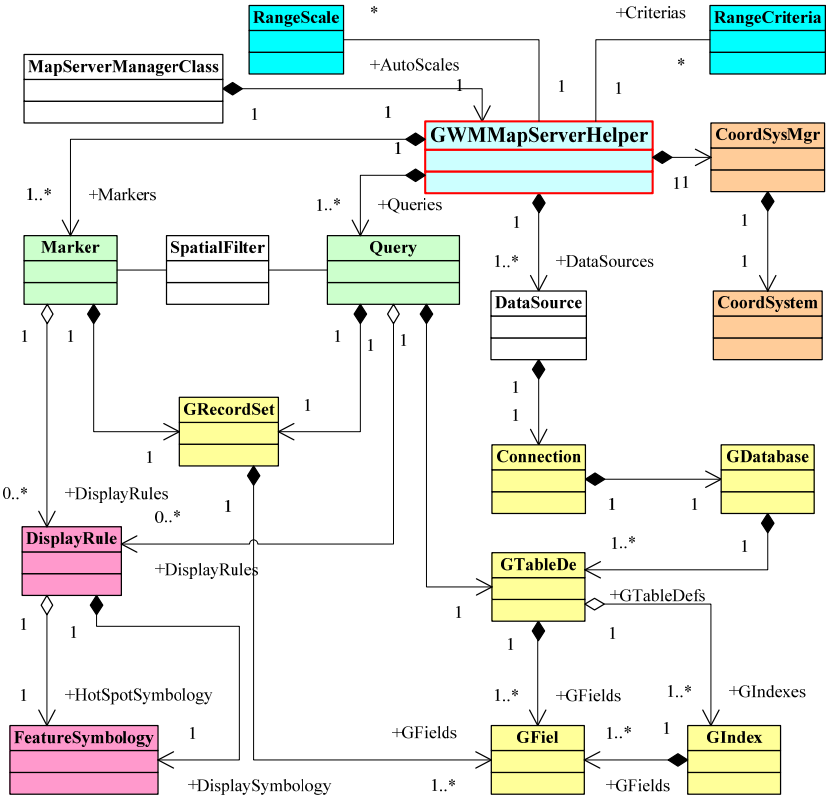


Figura 3.2

Diagramma UML sintetico delle classi di GeoMedia WebMap. In verde sono evidenziate le classi per la definizione dei temi della mappa, in giallo quelle per la manipolazione relazionale dei dataset, in arancio le classi destinate alla gestione del sistema di riferimento ed in fucsia le classi necessarie alla vestizione grafica dei temi.

[31] Non si tratta di uno schema ufficiale.

La classe `GWMMapServerHelper` permette di interagire con il servizio di sistema "`MapServer.exe`", tale classe viene istanziata tramite `MapServerManagerClass` che è creata mediante il web server.

Tale classe fornisce l'infrastruttura necessaria alla creazione di una mappa in tutti i suoi aspetti, oppure all'esecuzione di operazioni geografiche su di un insieme di dataset geografici.

Per la gestione delle connessioni ai dati geografici, sono previste delle estensioni (*driver*) cha hanno il compito accedere ai dataset indipendentemente dal formato di codifica, figura 3.3.

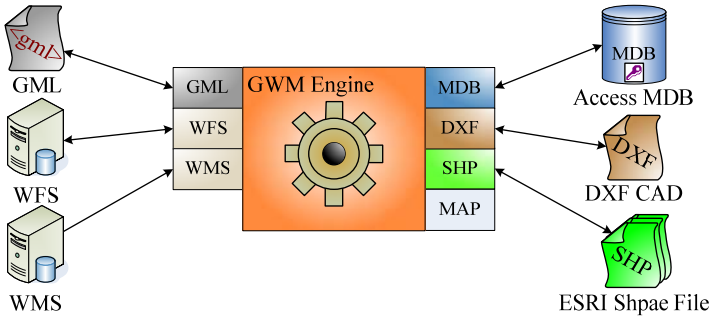


Figura 3.3
Alcuni driver di GWM per l'importazione di sorgenti dati.

I dataset geografici sono strutturati secondo un preciso formato di codifica, proprietario oppure OpenGIS, e raggruppati all'interno di una sorgente dati. La classe `DataSource`, descrive le caratteristiche di una connessione verso una sorgente dati e consente di manipolare, dove è consentito, sia lo schema della base di dati che i dataset. La classe `GDatabase` fornisce il supporto per la modellazione dello schema della base di dati: relazioni, campi ed indici tramite le classi `GTableDef` , `GField` e `GIndex`.

Creando più connessioni è possibile integrare un insieme di dati eterogenei in formato e sistema di riferimento; queste due caratteristiche dei dataset geografici sono modellate attraverso un insieme di classi che fanno riferimento all'istanza di `CoordSystem`. Se si hanno *n* dataset geospaziali[32], in generale è possibile avere *n* sistemi di riferimento spaziali distinti; GWM li uniforma utilizzandone uno specificato precedentemente all'operazione di connessione ai dati, in modo che le mappe ed i nuovi dataset prodotti siano coerenti a tale sistema di riferimento spaziale;

L'estrazione dei dati, che andranno a costituire i temi della mappa, avviene tramite due classi: `Marker` e `Query`. Entrambe permettono di definire un dataset geografico (indicato

[32] GWM è in grado di manipolare anche *dataset* non spaziali.

come *recordset* ed implementato mediante la classe `GRecordset`) attraverso una restrizione ottenuta tramite criteri che operano sull'estensione spaziale oppure sugli attributi delle feature (è possibile combinare assieme i due tipi di criteri). Ogni oggetto `Query` e `Marker`, che fa riferimento alla rispettiva connessione, definisce un tema della mappa; a tale tema è possibile associare un hotspot ed un tooltip tramite la classe `DisplayRule`. Attraverso quest'ultima, è assegnata la vestizione grafica sia del tema, sia dello hotspot utilizzando la classe `FeatureSymbology`.

La figura 3.4 schematizza gli elementi impiegati per la produzione di una mappa.

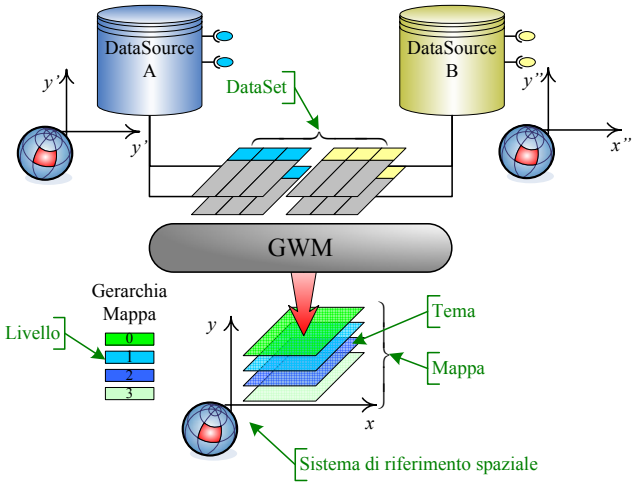


Figura 3.4
Composizione di una mappa a partire da due sorgenti dati definite in formati e sistemi di riferimento diversi.

Le classi `Marker` e `Query` hanno la stessa semantica ma diverso impiego: la classe `Query` risulta essere più efficiente della classe `Marker`, ma quest'ultima offre funzionalità addizionali come la creazione di un layer in base ad un recordset oppure estrarre una porzione di layer in base ad un poligono (box) od un cerchio.

Per la visualizzazione di una mappa è possibile associare un intervallo di scala[33], tramite la classe `RangeScale`; perché ciascun tema possa essere visualizzato deve possedere l'intervallo di scala in cui esso è visibile all'interno della mappa (tramite la classe `RangeCriteria`). Questo significa che un dato tema può essere visibile solo all'interno di un intervallo di scala; questo meccanismo è utilizzato per mostrare temi secondo vari livelli di dettaglio in seguito ad operazioni di zoom.

Oltre all'operazione di connessione ai dati e restituzione di mappe, GWM permette di effettuare sui dati operazioni geografiche, descritte nel capitolo successivo.

[33] Valore di scala, minimo e massimo, in cui la mappa od il tema sono visibili.

Le operazioni geografiche sono effettuate su recordset (creati ad esempio tramite oggetti `Marker` o `Query`) e ne restituiscono uno nuovo; le classi che modellano alcune di queste operazioni sono dette `Pipe`.

Un oggetto `Pipe` riceve in ingresso uno o più recordset assieme ai parametri che controllano l'operazione. Le operazioni avvengono nell'ambiente definito dall'istanza della classe `MapServer`, che caratterizza il recordset creato.

In figura 3.5, sono riportate le classi `Pipe` utilizzate per implementare le operazioni GIS esposte dai web service e presentate nel prossimo capitolo:

- *SpatialIntersectionPipe*: dati due recordset (`LeftRecordSet`, `RightRecordSet`), restituisce la loro intersezione;
- *SpatialDifferencePipe*: dato un recordset in input (`InputRecordSet`) ed uno come maschera (`MaskRecordSet`), l'operazione restituisce il recordset di input meno le feature in comune definite dalla maschera;
- *SpatialSubSetPipe*: dati due recordset (`FirstRecordSet` e `SecondRecordSet`), restituisce tutte le feature del primo recordset che soddisfano una relazione di confronto spaziale con le feature del secondo recordset;
- *UnionPipe*: effettua l'operazione di unione fra feature appartenenti ad un insieme di recordset;
- *BufferPipe*: permette di implementare operazioni destinate alla creazione di arre di rispetto (buffer).

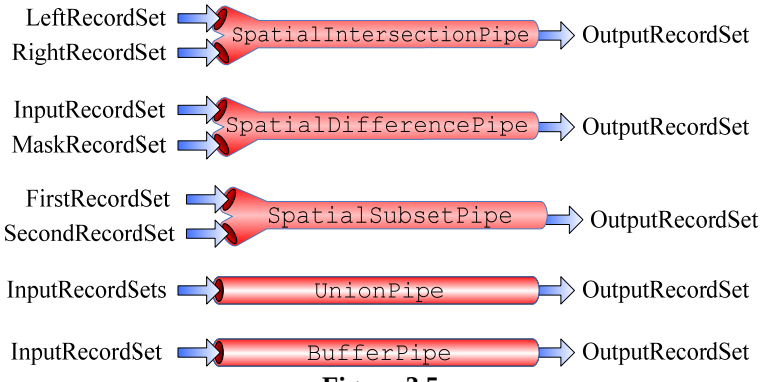


Figura 3.5
Esempio di alcuni oggetti `pipe`, forniti da GWM.

3.1.2 GeoMedia Web Map e OpenGIS

Il supporto ai servizi definiti da OWS1, è fornito da Intergraph attraverso librerie ed applicativi, distribuiti separatamente da Intergraph

Per la pubblicazione di dati geografici mediante i servizi WFS e WMS sono presenti due "kit". Per interagire con i servizi WFS e WMS e per poter trattare dati codificati in

GML, sono previsti tre adattatori (figura 3.3) che consentono di accedere in modo trasparente alle sorgenti dati geografiche.

3.2. Framework .NET

Lo sviluppo del lavoro di tesi richiede un'infrastruttura che sia in grado di fornire l'implementazione dell'architettura web service e che permetta l'uso delle classi definite all'interno delle librerie di GWM.

Le infrastrutture più diffuse che offrono pieno supporto all'architettura web service, appartengono ai due produttori di software che storicamente hanno iniziato il lavoro di creazione degli standard SOAP-WSDL-UDDI:

- **IBM**: AXIS + *IBM Web service ToolKit*;
- **Microsoft**: *framework .NET*.

Un'altra interessante soluzione viene da Sun Microsystem[34] con il progetto *Java Web service Developer Pack* (Java WSDP) [31].

IBM basa la sua infrastruttura su *Java Virtual Machine* (JVM) ed un'evoluzione del web server Apache: Axis[35]. Axis è un'infrastruttura basata su Java per la creazione di applicazioni che necessitano di un motore SOAP, inoltre offre pieno supporto per WSDL ed UDDI.

L'infrastruttura di Microsoft, *framework* .NET, è stata storicamente la prima a fornire completo supporto alla tecnologia dei web service (framework .NET 1.0); si tratta di un'applicazione di sistema che offre supporto a tempo di esecuzione e di sviluppo.

Questo è reso possibile grazie ad un vasto insieme di librerie di classi, interfacce e meccanismi (sicurezza, connettività, …) completamente integrati con la piattaforma Win32.

Entrambe le soluzioni di IBM e Sun sono basate su JVM+Apache e quindi indipendenti dalla piattaforma, mentre framework .NET è legato strettamente a sistemi Windows+IIS. In realtà esiste un progetto parallelo, sponsorizzato da Novel, chiamato **Mono**, che fornisce un supporto quasi completo a tutte le specifiche[36] del framework .NET e permette di eseguire applicazioni .NET su vari sistemi come Linux, Unix, Mac OS X e Windows [32].

L'infrastruttura supporta l'**integrazione** fra linguaggi di sviluppo oltre che l'indipendenza dal linguaggio usato. Si intende quindi la possibilità di far interagire una porzione di codice scritto in linguaggio *A*, con altro codice scritto in un diverso linguaggio,

---

[34] IBM, Microsoft e Sun Microsystem fanno parte delle otto aziende componenti il WS-I.
[35] Axis è il successore di Apache SOAP.
[36] Si tratta di un'ambiente in continuo sviluppo, quindi ogni release copre una nuova porzione di .NET.

B. In concreto, significa poter ereditare classi, catturare eccezioni, usufruire del polimorfismo fra differenti linguaggi.

Dell'architettura del framework .NET, figura 3.6, saranno esaminati:

- *Common Language Runtime* (CLR),
- ASP.NET,
- Web Service Architecture.

IL CLR caratterizza l'infrastruttura ed aiuta a capirne la filosofia, mentre i restanti due sono stati impiegati per la realizzazione dei web service GIS (capitolo 4) e dell'applicazione di esempio (capitolo 5).

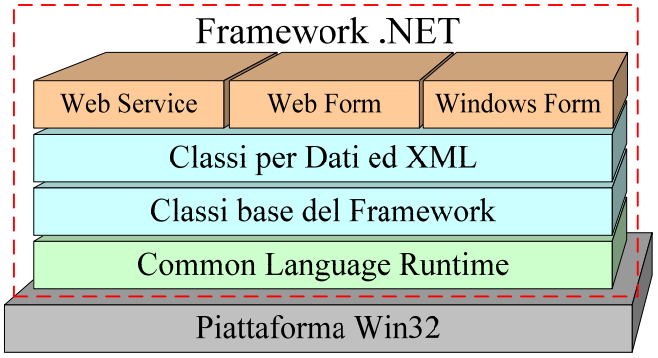


Figura 3.6
Architettura del Framework *.NET.*

3.2.1. Common Language Runtime

Il *Common Language* Tempo di esecuzione (CLR)[37] costituisce la base dell'architettura del framework .NET, in quanto ha il compito di **gestire** ed eseguire il codice compatibile .NET (tipicamente si parla di ambiente *managed*).

Può essere paragonato ad una JVM, dato che entrambi sono infrastrutture di esecuzione che astraggono la piattaforma sottostante. Il CLR si occupa (in sintesi) di caricare in memoria le classi, amministrare la sicurezza e gestire il *garbage-collection*. A differenza di JVM, CLR non limita l'uso ad un solo linguaggio, bensì qualsiasi linguaggio rappresentabile secondo il linguaggio intermedio ECMA *Common Intermediate Language* (ECMA-CIL), l'equivalente del *bytecode* di JVM.

Per rendere interoperabili i linguaggi sono state definite altre due specifiche:

- *Common Type Specification* (CTS): indica la semantica dei tipi che deve essere comune a tutti i linguaggi;
- *Common Language Specification* (CLS): regole minime comuni che ogni linguaggio deve rispettare perché possa essere considerato un linguaggio .NET.

---

[37] Il CLR di .NET è quello che ECMA definisce *Common Language Infrastructure* (CLI).

Un eseguibile (EXE o DLL) è tipicamente composto da codice ed opzionalmente da dati, mentre in ambiente .NET sono presenti inoltre dei metadati. Questa strutturazione è detta *Portable Executable File* (*PE File*) e deriva dalla specifica *Common Object File Format*[38] (COFF). Il file PE è composto da molte sezioni, quella più interessante è denominata *CLR-Data* e contiene un insieme di metadati ed il codice *Microsoft Intermediate Language*, (MIL).

3.2.1.1. Metadata

Un ***assembly*** .NET è una collezione di uno o più file che sono sottoposti a versionamento e rilascio sotto forma di un'entità unica; costituisce le fondamenta di un'applicazione .NET.

L'insieme dei metadati costituisce il meccanismo con cui memorizzare le informazioni riguardanti i tipi che lo assembly espone. Il CLR utilizza tali metadati a tempo di esecuzione e garantisce l'interoperabilità, in quanto le informazioni sui tipi sono memorizzate allo stesso modo secondo la specifica CTS.

I metadati consentono a tempo di esecuzione di recuperare qualsiasi informazione necessaria all'integrazione dello assembly; una volta che un assembly è stato creato, è subito disponibile all'uso secondo la strategia *Plug&Play*.

Oltre all'interoperabilità fra linguaggi, i metadati consentono:

- l'ispezione a tempo di esecuzione del tipo di un oggetto (*reflection* di .NET);
- creazione di strumenti di navigazione su classi e librerie;
- operazioni di *debug*;
- ottenere l'interoperabilità fra linguaggi.

All'interno del framework .NET, sono presenti due strumenti molto utili che si basano sui metadati:

- *Type Library Importer*: maschera un componente COM da assembly .NET, effettuando un'operazione di *wrapping*[39]. Senza scendere troppo nel dettaglio, viene creata una **classe *proxy*** (sotto forma di *Dynamic Linked Library*, DLL) (*Runtime Callable Wrapper*, RCW) che assume il ruolo di interfaccia fra l'ambiente gestito (*managed*) di .NET con quello non gestito (*unmanaged*) di COM (figura 3.7);

---

[38] Formato di file oggetto in ambiente *UNIX System V release* 3, sostituito dal formato *Executable Linking Format* (ELF) nella *UNIX System V release* 4. COFF è pienamente supportato in ambiente Linux.

[39] Un *wrapper* può essere considerato una classe che inscatola un componente software in modo da poter essere trattato più agilmente.

- *XML Schema Definition Tool*: da' la possibilità di convertire una classe in uno schema XML e viceversa.

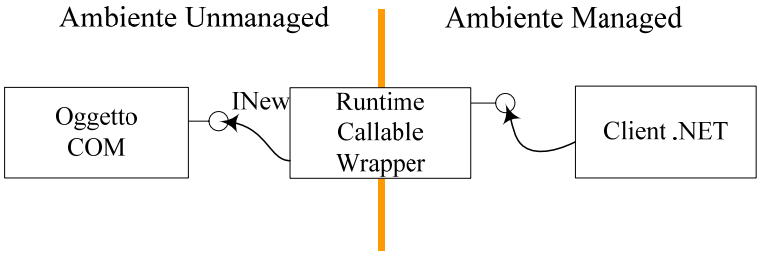


Figura 3.7
Esempio di classe proxy RCW fra un oggetto COM ed un'applicazione .NET.

Associato allo assembly esiste un insieme di metadati, *manifest*, che ne descrive ad esempio l'identità, i file che lo compongono ed i permessi di esecuzione. Il file manifest è impiegato a tempo di esecuzione per risolvere i riferimenti ad altri assembly e validare l'integrità dello assembly.

3.2.1.2. Microsoft Intermediate Language

EMCA-CIL è una specifica per un linguaggio orientato agli oggetti di livello intermedio, in grado di astrarre il sistema sottostante. *Microsoft Intermediate Language* (MIL), conosciuto anche come *Language Intermediate* (IL), è l'implementazione del ECMA-CIL. Oltre a supportare tutte le caratteristiche del paradigma ad oggetti come ereditarietà e polimorfismo, introduce altri meccanismi come eccezioni, eventi ed enumeratori.

IL è il punto di convergenza di tutti i linguaggi compatibili .NET. Da questo punto in poi il CLR utilizza il componente *Just-In-Time Compiler* (JIT-*Compiler*) per tradurre da IL a linguaggio nativo, permettendo di ottenere codice ottimizzato per la piattaforma in uso.

La compilazione avviene la prima volta che un metodo è invocato, il codice compilato è successivamente memorizzato in uno spazio di memoria attivo per tutto il tempo di vita dell'applicazione. Esiste la possibilità di evitare la compilazione a runtime, eseguendola a una sola volta al momento dell'installazione dell'applicazione (*pre-JITing*).

3.2.2. ASP.NET

Active Server Page (ASP) permette di creare pagine dinamiche lato server. Questo significa poter instaurare un'interazione dinamica fra un browser HTTP sul client, tramite un'interfaccia utente HTML, ed un server in grado di elaborare le richieste ed inviare risposte al client (figura 3.8).

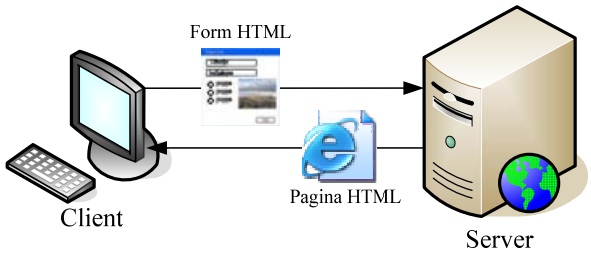


Figura 3.8
Tipica interazione browser-Server *tramite form HTML su HTTP.*

Una pagina ASP è caratterizzata da contenuto HTML misto a codice, usualmente scritto in linguaggi di *scripting* come JavaScript o VbScript.

Application Service Provider .NET (ASP.NET) è una piattaforma di sviluppo, in grado di fornire un'infrastruttura per applicazioni web.

E' il successore della tecnologia ASP, ma a differenza di questa permette di separare il contenuto HTML dall'automazione descritta tramite codice. Inoltre, grazie a IL, permette di utilizzare i linguaggi .NET come C#, Vb.NET, JScript.NET[40].

Il framework .NET (v1.1) è composto da oltre 9000 classi organizzate in namespace; il namespace "`System.Web`" raggruppa le classi, le interfacce ed altri namespace per abilitare la comunicazione fra web browser e server.

Una pagina ASP.NET (*aspx*) è una classe derivata dalla classe "`Page`" definita all'interno del namespace "`System.Web.UI`".

Lato server, ASP.NET consente di manipolare gli elementi dell'interfaccia di un *form* HTML (rappresentati dai *tag* HTML, come: bottoni, caselle di testo, link, tabelle, …) attraverso un insieme di classi definite all'interno del namespace "`System.Web.UI.HTMLControls`".

Dato che tali controlli hanno un insieme limitato di funzionalità, ASP.NET introduce un nuovo insieme di controlli denominati *controlli web* (*web controls*) Come per gli elementi standard di HTML, esiste una corrispondenza con delle classi del framework *.NET* ma appartengono ad un diverso namespace: "`System.Web.UI.WebControls`". I controlli web sono caratterizzati dal fatto di essere stati riprogettati completamente in modo tale da

[40] Con l'introduzione della versione 1.1 del *framework* .NET, è possibile sfruttare il linguaggio J#.

fornire un insieme più ampio e sofisticato di funzionalità. La sintassi di questi controlli è basata su XML ma all'atto pratico il web browser riceve comunque codice HTML standard.

Esistono altri namespace che gestiscono la sessione utente e la sicurezza, oltre ad un elevato numero di aspetti legati al web [33].

Tutti questi aspetti vanno a comporre quello che Microsoft definisce *web forms*; la tecnologia web forms permette di introdurre il modello di interazione ad eventi, tipica delle applicazioni desktop, nell'ambiente web.

Perché tale modello possa sussistere, è necessario tenere traccia sia dello stato dell'applicazione client, tramite il tag HTML `<form>`, sia della pagina appartenente alla sessione sul server.

Tener traccia dello stato del server non è di semplice realizzazione in quanto HTTP è un protocollo *stateless*, ne segue che ogni richiesta è risolta in un ambiente separato dalle altre. ASP.NET serializza lo stato della pagina (l'insieme degli stati di ogni oggetto istanziato) e lo incapsula all'interno della pagina tramite campi nascosti (*tag* HTML standard). Quando una richiesta arriva al web server, esso deserializza lo stato della pagina, lo modifica secondo la richiesta ed infine lo serializza per inviarlo nuovamente al client (figura 3.9).

Da notare che ASP.NET consente di separare l'aspetto della pagina (il codice HTML) dal codice che definisce l'automazione (*code behind*).

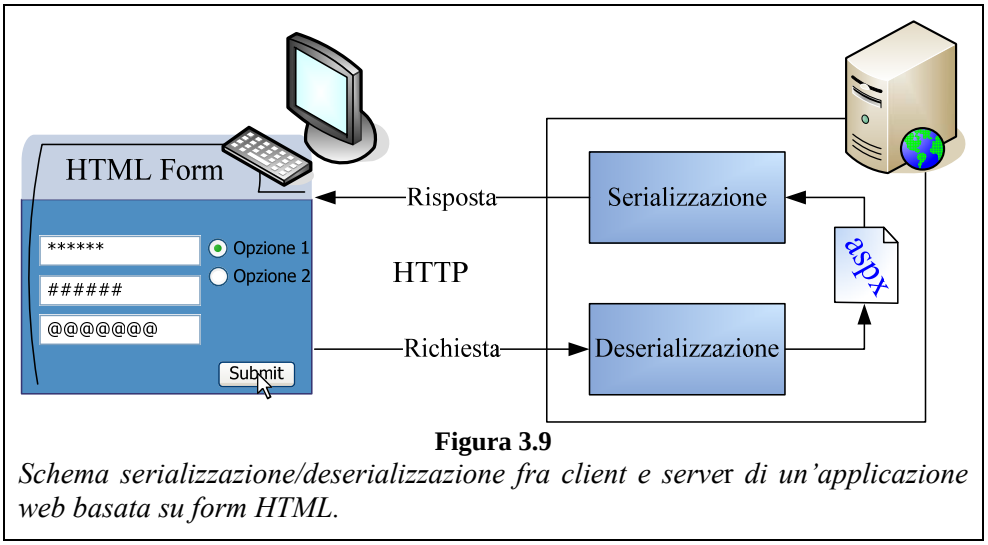


Figura 3.9
Schema serializzazione/deserializzazione fra client e server di un'applicazione web basata su form HTML.

3.2.2.1. Risoluzione di Pagine ASP.NET

ASP.NET è ufficialmente supportato dal web server *Internet Information Service* (IIS), di Microsoft[41], mentre in ambiente Mono si ha XSP (sviluppato in C# in ambiente Mono) e Apache tramite un modulo di XSP [2].

IIS gestisce le richieste alle risorse in base al loro contesto: ogni tipologia di risorsa viene tratta da un'estensione apposita, detta ISAPI (*Internet Server Application Programming Interface*) definita all'interno di una libreria dinamica (DLL).

Quando IIS riceve una richiesta per una pagina *aspx*, carica la ISAPI appropriata; questa funziona da *dispatcher*, in quanto ottiene tutte le informazioni relative alla risorsa richiesta, ed attiva un processo separato (non managed) detto *ASP.NET Worker* che rappresenta l'ambiente runtime di ASP.NET ed ospita il CLR.

ASP.NET Worker attiva un insieme di classi, detto *HTTP Pipeline*, che ha il compito di compilare ed eseguire lo assembly relativo alla pagina, inclusa la serializzazione e deserializzazione (figura 3.10).

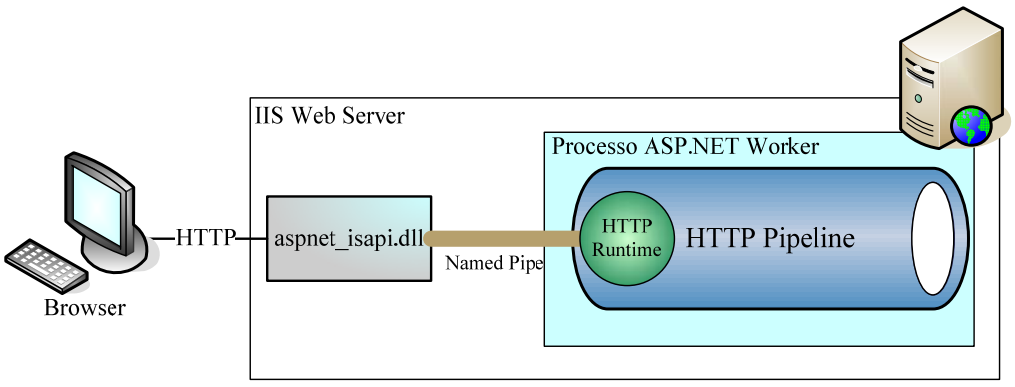


Figura 3.10
Semplificazione dell'architettura di esecuzione di una pagina aspx,; la classe `HTTPRuntime` *ha il compito di gestire la HTTP Pipeline*

Per una completa trattazione si rimanda a MSDN Library [3].

3.2.3. Web Service

Il framework .NET offre un pieno supporto sia per XML che per i protocolli WSDL, SOAP ed UDDI tramite classi definite all'interno del namespace `System.Web.Service`.

L'implementazione e l'uso dei web service all'interno del framework .NET non è molto dissimile da quanto detto per le pagine ASP.NET, in quanto è trattato come una particolare pagina ASP.NET.

All'interno della *HTTP Pipeline* è prevista una classe, `WebServiceHandler`, che ha il compito di consumare le richieste SOAP (figura 3.11).

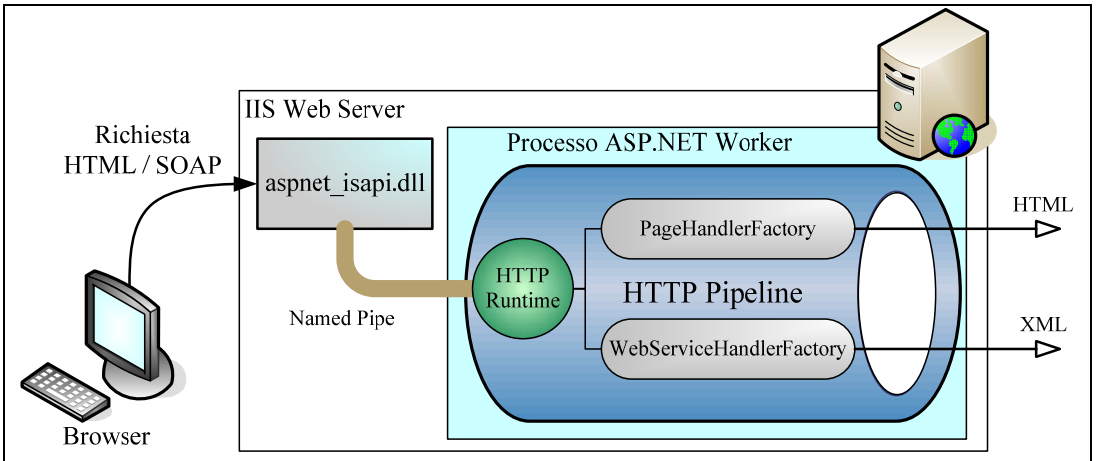


Figura 3.11

Architettura di ASP.NET con la componente destinata alla gestione delle richieste SOAP per i web service.

Come per le pagine *aspx*, un web service è una classe derivata dalla classe base "`Webservice`" definita all'interno di un namespace di "`System.Web`"; oltre ai metodi, attributi e proprietà (pubblici e privati) è possibile definire i metodi accessibili via HTTP: **metodi web**. Tali metodi sono definiti (marcati) tramite il meccanismo degli **attributi**[42].

L'attributo `WebMethod` è utilizzato per indicare che quei metodi sono invocabili via HTTP; in generale gli attributi danno informazioni sul codice a tempo di esecuzione.

Per quanto riguarda il *marshalling* dei tipi, è impiegato massicciamente il serializzatore XML, fornito dalla classe `XMLSerializer`. Tale classe, produce un'efficace rappresentazione dei tipi di dato tramite XSD e SOAP.

Combinando il serializzatore con classi *proxy*, il *marshalling* risulta completamente trasparente al client; in questo caso una classe *proxy* rappresenta un'interfaccia con cui invocare il web service tramite SOAP o HTTP-POST in modo sincrono od asincrono. Questa classe, derivata da una classe base, può essere creata tramite uno strumento compreso nel framework .NET.

---

[41] *Cassini* è un valido web server compatibile con ASP.NET. Per l'ambiente Mono, XSP

[42] In generale, gli attributi danno informazioni sul codice a tempo di esecuzione.

3.2.4. Conclusioni su framework .NET

Per l'implementazione del progetto, illustrato nel prossimo capitolo, il framework .NET è stato ritenuto un valido strumento in quanto:

- permette una semplice integrazione con GeoMedia Web Map (compatibilità certificata da GeoMedia);
- fornisce pieno supporto all'architettura web service;
- offre meccanismi sofisticati per la manipolazione di dataset e documenti XML;
- il CLR offre elevate prestazioni;
- è una tecnologia consolidata;
- è completamente gratuito.

Bibliografia

- 🕮 Dino Esposito, "*Programmare Microsoft ASP.NET*": Settembre 2003;
- 🕮 MSDN Library: http://msdn.microsoft.com/library/

Capitolo 4

Architettura Della Infrastruttura

4.1. Introduzione alle Operazioni GIS

Qualsiasi strumento GIS non si limita solo ad organizzare i dati secondo un modello prestabilito, bensì offre supporto al processo di analisi di fenomeni di natura spaziale.

A tal fine, sono previste un insieme di operazioni che **estraggono** nuove informazioni dai dati in base agli aspetti intrinseci legati ai fenomeni fisici rappresentati. Viene effettuata una riorganizzazione dei dati per evidenziare aspetti specifici. Tali operazioni sono dette "operazioni GIS", "operazioni geografiche" od anche "operazioni topologiche".

In questo lavoro di tesi sono stati considerati quattro gruppi di operazioni geografiche:

- *Geoprocessing*, note anche come operazioni di *overlay*;
- Selezione per tema;
- *Buffering*;
- Restituzione grafica dei dati, *web mapping*;

Gli operandi delle operazioni sopra elencate sono costituiti da insiemi di feature (geografiche).

4.1.1. Geoprocessing

Le operazioni di *geoprocessing*[43] sono generalmente caratterizzate da due insiemi di feature, uno di input ed uno detto di *overlay*, ed utilizzano criteri di confronto basati sia sulla topologia che sugli attributi degli operandi per produrre un nuovo insieme di feature.

Di seguito sono elencate e descritte le operazioni di *geoprocessing* affrontate:

- **Unione**: dati due insiemi di feature vettoriali omogenee, l'operazione restituisce un nuovo insieme composto dalle feature generate eseguendo l'*incrocio* sugli insiemi di input. La tupla degli attributi è composta da tutti gli attributi degli insiemi di feature, dati in ingresso (figura 4.1);

[43] Le operazioni di geoprocessing sono note in letteratura anche come "operazioni di overlay" o "incrocio topologico".

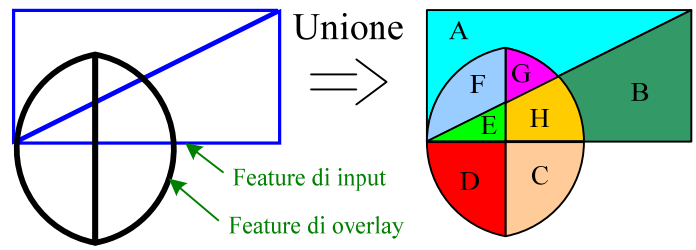


Figura 4.1

Operazione di Unione fra due insiemi di feature.

- **Intersezione**: da due insiemi di feature vettoriali omogenee, ne crea uno nuovo composto da tutte le feature comuni ai due insiemi di input, individuate dall'operazione di incrocio (figura 4.2). Gli attributi delle feature risultanti, sono composti dalla concatenazione di quelli degli insiemi di partenza.

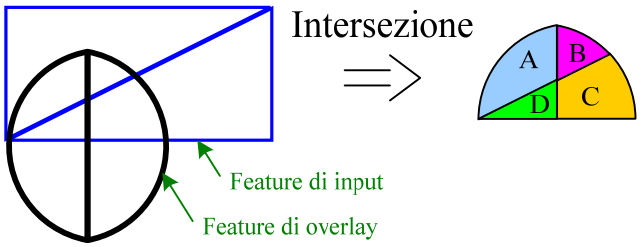


Figura 4.2

Operazione di Intersezione fra due insiemi di feature.

In base all'operazione di Intersezione fra differenti tipi di geometrie, per la tipologia di feature class dell'insieme prodotto, si hanno i casi riassunti dalla tabella 4.1.

Input	Overlay	Risultato
Poligono	Poligono	Poligono
Linea	Poligono	Linea

Tabella 4.1

Risultato dell'operazione di Intersezione in base al tipo di feature contenute nell'insieme di input ed in quello di overlay.

- **Clip** (taglio): questa operazione utilizza una "maschera geometrica" per estrarre una porzione di dati da un insieme di feature. La maschera consiste in una geometria poligonale impiegata per delimitare un'area geografica. Può essere considerata come un insieme di overlay costituito solo da un'unica feature priva di attributi (figura 4.3).

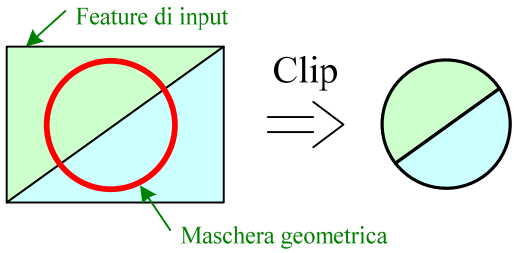


Figura 4.3
Il cerchio riveste il ruolo di maschera geometrica, mentre le due feature triangolari costituiscono l'insieme di input.

L'operazione di Clip preserva sia la tipologia di feature, sia gli attributi delle feature, indipendentemente dalla geometria della maschera (tabella 4.2).

Input	Risultato
Poligono	Poligono
Linea	Linea
Punto	Punto

Tabella 4.2
Risultato dell'operazione di Clip in base alla tipologia di feature contenuta nell'insieme di input.

- **Merge**: appende le feature di due insiemi in uno unico e gli attributi sono tenuti quelli in comune (figura 4.4). L'operazione è concettualmente simile all'unione, tranne che gli attributi risultanti sono quelli in comune agli insiemi di feature di partenza.

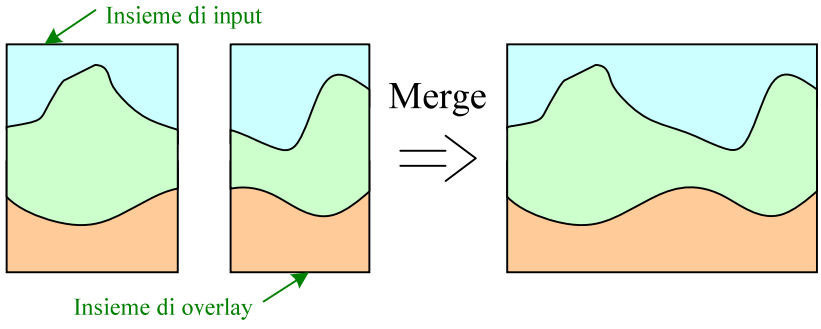


Figura 4.4
Esempio di Merge eseguito su due insiemi di feature poligonali.

Per implementare l'operazione di Unione definita precedentemente, GWM fornisce la classe `UnionPipe` che riceve in input un insieme di recordset e permette di impostare il criterio con cui scegliere l'insieme degli attributi che apparterranno al recordset prodotto. I criteri possibili sono i seguenti:

- *first-recordset*: gli attributi dell'insieme di output sono composti da quelli del primo recordset dato in input all'oggetto `Pipe`;

- *union-of-all*: un attributo è incluso nel recordset di output se esiste in almeno un recordset di input;
- *intersection*: un attributo è incluso nel recordset di output se è comune a tutti i recordset di input.

Anche l'operazione di Merge è stata implementata mediante la classe `UnionPipe`, ma con una sostanziale differenza: mentre per l'unione è utilizzato il criterio *union-of-all*, per l'operazione di Merge è stato utilizzato il criterio *intersection* per ottenere solo gli attributi in comune ai recordset di input.

L'operazione di Intersezione è realizzata tramite la classe `SpatialIntersectionPipe`, di GWM, che consente di specificare l'operatore spaziale (criterio di selezione) da utilizzare per decidere quali feature dell'insieme di overlay contribuiranno alla produzione del risultato. Alcuni operatori disponibili (figura 4.5), sono:

- contenimento;
- contenimento completo;
- sovrapposizione;
- adiacenza.

Per realizzare l'operazione di intersezione è stato utilizzato l'operatore di sovrapposizione.

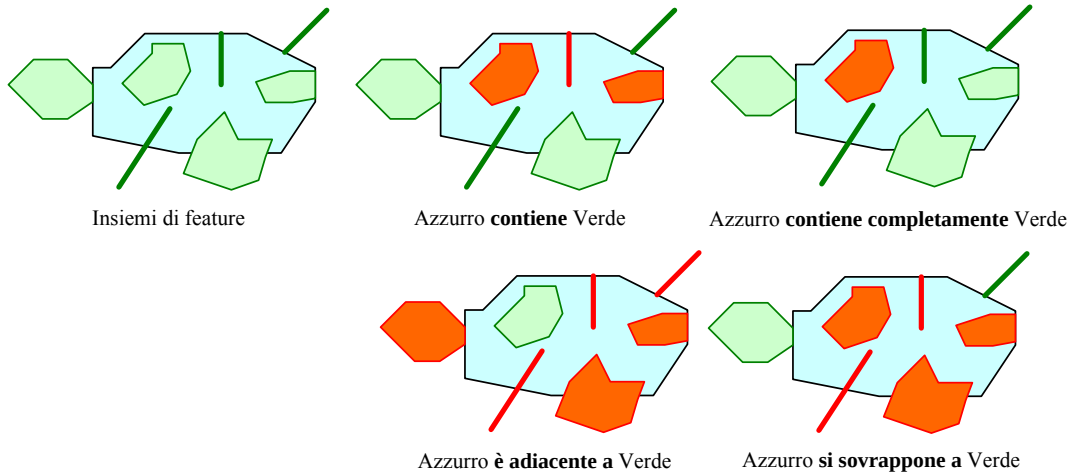


Figura 4.5
Data la feature di input azzurra e quelle di overlay in verde, è illustrato il risultato dell'applicazione dei quattro operatori spaziali (in arancio).

4.1.2. Selezione per Tema

Il modello georelazionale consente di eseguire interrogazioni sugli attributi per selezionare un sottoinsieme di feature.

Le operazioni di selezione per tema utilizzano operatori spaziali per confrontare geometricamente due insiemi di feature vettoriali: l'*insieme di feature attive* e l'*insieme di feature di selezione*. Le operazioni prese in esame si basano sui seguenti criteri spaziali:

- **Intersezione**: sono restituite tutte le feature dell'insieme attivo che intersecano quelle dell'insieme di selezione;
- **Contenimento**: confronta i due insiemi di feature e restituisce le feature dell'insieme attivo che contengono completamente quelle del secondo;
- **Distanza** (lineare): sono restituite le feature dell'insieme attivo che, rispetto alle feature dell'insieme di selezione, si trovano ad una distanza prefissata. La distanza è fornita come valore.

4.1.3. Buffering

Le operazioni di *buffering* sono impiegate per determinare **aree di rispetto** attorno alle entità geometriche. Un'area di rispetto è un poligono che circonda l'oggetto geografico, indipendentemente dalla sua tipologia (puntuale, lineare o poligonale).

Le operazioni di buffering implementate differiscono per com'è determinata la distanza fra l'oggetto geografico ed il bordo dell'area di rispetto:

- **distanza fissa**: la distanza è espressa tramite un valore costante. È possibile specificare una sequenza di intervalli per avere più aree di rispetto associate ad una feature (figura 4.6);
- **distanza parametrica**: la distanza è determinata dal valore di un attributo numerico della feature.

Un'area di rispetto può essere caratterizzata da spigoli smussati oppure squadrati; nel caso in cui le aree di rispetto delle feature si intersechino, è possibile fonderle in un'unica (figura 4.5).

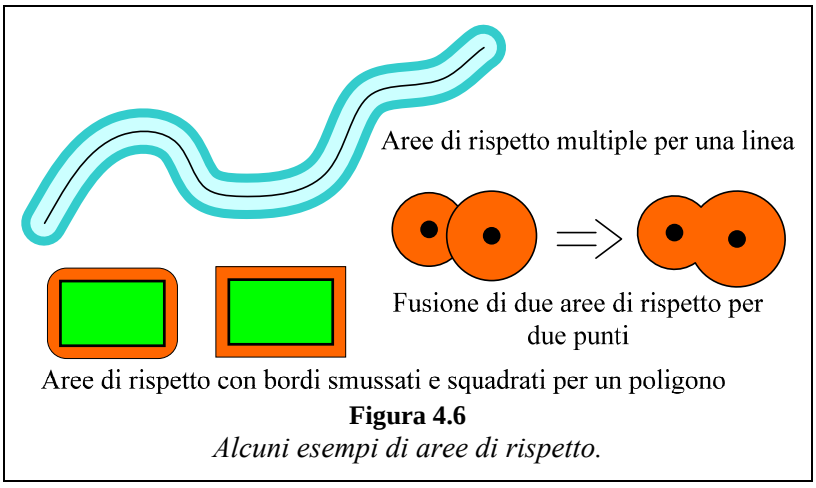


Figura 4.6
Alcuni esempi di aree di rispetto.

4.1.4. Restituzione Grafica

Una mappa è la rappresentazione grafica, secondo un sistema di riferimento spaziale, della sovrapposizione di più temi secondo un ordine prestabilito (livello, o priorità, del tema). Un tema è un layer corredato di un insieme di parametri che comprendono:

- colore di riempimento: si applica ai punti ed ai poligoni;
- *pattern* di riempimento: retinatura di riempimento per i poligoni;
- colore del tratto: applicato ai bordi di un poligono od una linea;
- aspetto del tratto: tratteggio delle linee: punti, linea-punto, …;
- simbologia del punto: un punto può esser rappresentato mediante varie forme, rettangoli, cerchi, disegni come ad esempio bandiere o spilli.

L'insieme di questi parametri è riferito come stile di visualizzazione (vestizione) del layer. La mappa così costituita, è restituita in due possibili tipologie di formati grafici: raster o vector.

4.2. Architettura dell'Infrastruttura

L'infrastruttura creata è composta di un insieme di web service, accessibili pubblicamente tramite HTTP, in grado di fornire le operazioni GIS precedentemente elencate. Tali web service saranno indicati come *GIS Operations - Web Service* (per brevità, GO-WS).

La figura 4.7 mostra i possibili casi d'uso dei servizi GO-WS. Qualsiasi dispositivo in grado di accedere alla rete Internet e di inviare richieste XML può usufruire dei servizi di elaborazione su dati geospaziali. É stato scelto di centralizzare i servizi GO-WS su di una sola macchina server, in quanto necessitano del motore GIS di GWM (per semplicità sarà utilizzato l'acronimo GWM per indicare il motore GIS anziché la soluzione software).

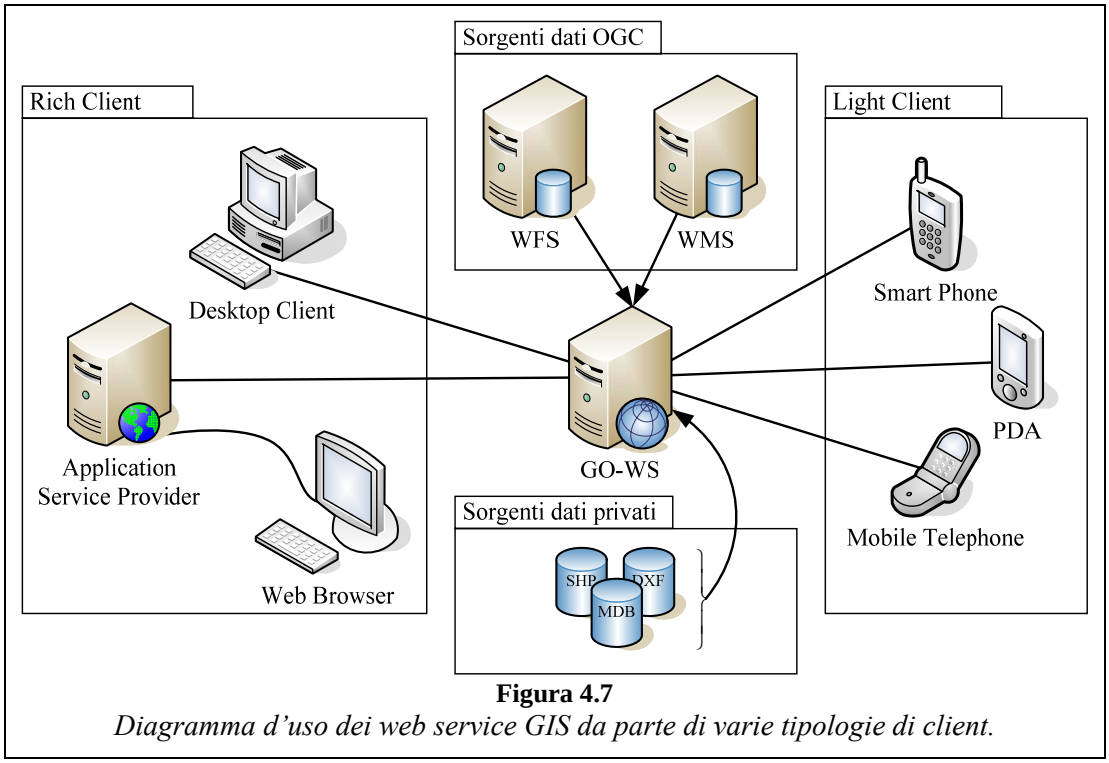


Figura 4.7
Diagramma d'uso dei web service GIS da parte di varie tipologie di client.

Oltre ai GO-WS è stato previsto un servizio di catalogo per WFS e WMS che sopperisce all'assenza di un modulo di interfaccia per accedere ai servizi OGC CAT.

L'architettura dei servizi GO-WS, illustrata in figura 4.8, evidenzia i due elementi portanti dell'infrastruttura (discussi nel capitolo 3):

- il web server: l'elemento software in grado di pubblicare contenuti su internet. In questo lavoro è stato impiegato Microsoft *Internet Information Server*;
- GeoMedia Web Map: è il motore GIS scelto per questo lavoro di tesi.

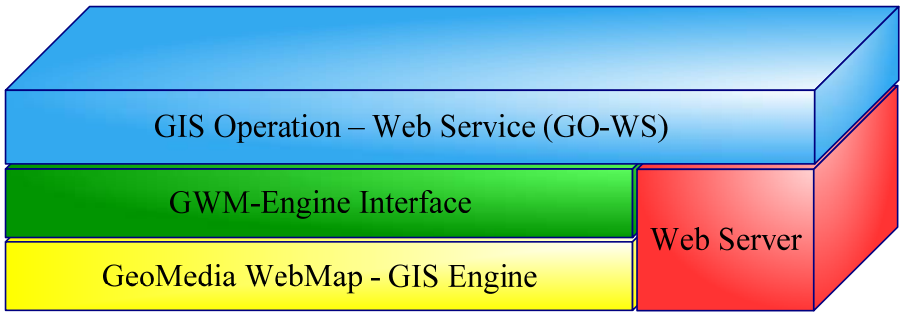


Figura 4.8
Architettura dei servizi GO-WS.

Sui due elementi portanti, è stata realizzata una libreria di classi, GWM-Engine Interface, in cui sono implementate le operazioni geografiche che i GO-WS andranno ad esporre via Internet.

4.2.1. Identificazione dei Layer

Prima di descrivere la libreria è necessario illustrare come identificare i layer su cui le classi della libreria operano.

Indipendentemente dal formato della sorgente dati in cui i dataset sono memorizzati, i *driver* per la creazione delle connessioni dati di GWM utilizzano un modo unico per identificare un layer (figura 4.9).

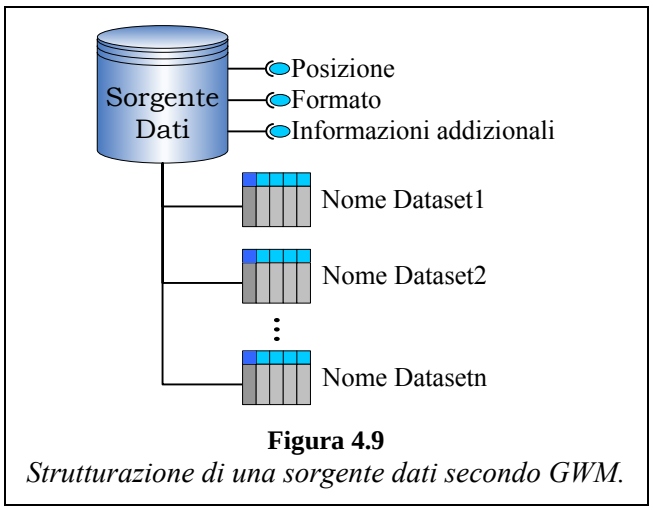


Figura 4.9
Strutturazione di una sorgente dati secondo GWM.

Perché GWM possa accedere al dataset geospaziale relativo ad un layer, necessita di quattro parametri:

1. la **posizione** della sorgente dati relativa al layer, che può essere il percorso su filesystem oppure una URL. Una sorgente dati descrive l'organizzazione in cui sono materializzati i dati su supporto fisico.
2. il **formato** del tipo di organizzazione della sorgente dati, tipicamente il formato dipende dall'applicazione utilizzata per codificare i dati (ad esempio

ESRI/ArcView o AutoCAD). Le interfacce dei servizi WFS e WMS di OGC sono considerate come un particolare formato di sorgente dati;

3. **nome**: nome del dataset all'interno della rispettiva sorgente dati;
4. **informazioni addizionali**: riportano parametri specifici relativi alle sorgenti dati. Ad esempio, nel caso di *shape file* (ESRI/ArcView), indicano il sistema di riferimento per georeferenziare i dataset.

È possibile estendere questa visione, per rappresentare un tema di una mappa aggiungendo la posizione nella gerarchia dei temi e lo stile di visualizzazione da associare al layer.

Le informazioni necessarie per identificare un layer ed un tema sono state modellate mediante due semplici classi definite all'interno del namespace `GenericLayerIdentifier` (appendice B):

- `LayerIdentifier`: identificatore del layer;
- `MapThemeDescriptor`: descrittore di tema. Riporta le informazioni necessarie a definire un tema all'interno di una mappa, a partire da un layer.

4.2.1.1. L'identificatore di Layer

La classe `LayerIdentifier` riporta i quattro parametri fondamentali sopra elencati, modellandoli mediante le seguenti quattro proprietà:

1. `dataSource`: posizione della sorgente dati;
2. `dataSourceType`: formato sorgente dati;
3. `layerName`: nome del dataset relativo al layer;
4. `additionalInfo`: informazioni generali sulla sorgente dati.

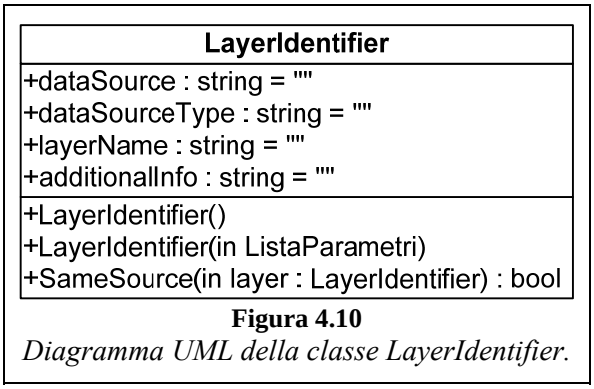


Figura 4.10
Diagramma UML della classe LayerIdentifier.

Come mostrato in figura 4.10, oltre a due costruttori, è presente il metodo `SameSource` utilizzato per stabilire se due "identificatori di layer" si riferiscono alla stessa sorgente dati. È stato introdotto per creare agevolmente insiemi di layer afferenti alla stessa sorgente dati in quanto GWM non consente più connessioni distinte alla stessa sorgente dati.

La caratteristica di integrazione richiesta, è garantita dai quattro parametri suddetti, inoltre il parametro "posizione" (rappresentato nella classe `LayerIdentifier` con la proprietà `dataSource`) può riferire sia ad un URL HTTP che ad un percorso su filesystem. Questo comporta la possibilità di riferire sia a sorgenti di dati di tipo OGC sia a sorgenti di dati proprietarie, poste in un server remoto.

4.2.1.2. MapThemeDescriptor

Questa classe è impiegata per rappresentare un tema all'interno di una mappa. Un tema è caratterizzato da un livello all'interno della gerarchia dei temi e da uno stile di visualizzazione. Quest'ultimo è stato considerato composto, per semplicità, dai seguenti elementi:

- colore di riempimento;
- colore del tratto;
- dimensione del tratto.

Come mostrato in figura 4.11, la classe deriva da `LayerIdentifier` in quanto ne estende il significato. Organizzando i descrittori di temi in una lista, è stato possibile modellare la gerarchia di temi appartenente ad una mappa, tramite la classe `ThemeList`.

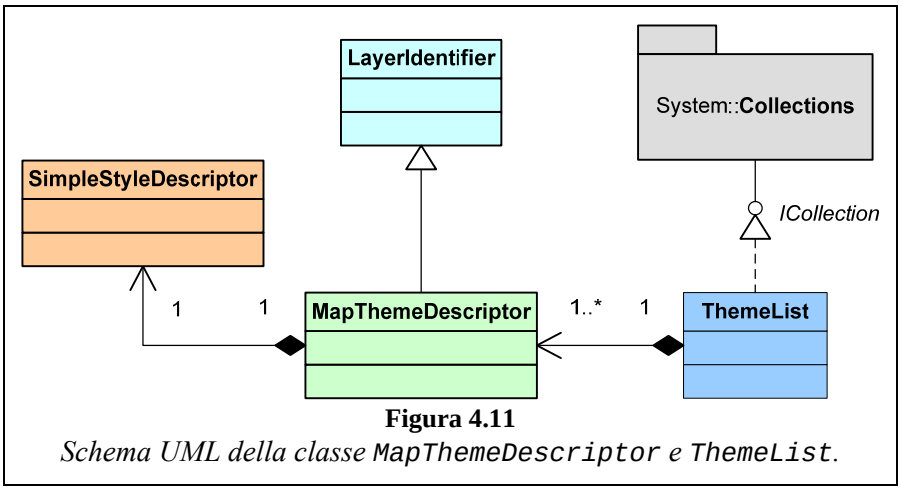


Figura 4.11
Schema UML della classe `MapThemeDescriptor` e `ThemeList`.

4.2.2. GWM-Engine Interface

L'interazione fra i servizi GO-WS e GeoMedia WebMap avviene tramite un'entità intermedia chiamata *GWM-Engine Interface*, progettata ed implementata appositamente con il fine di astrarre le componenti ed i meccanismi di GWM fornendo semplici chiamate alle operazioni GIS. L'impiego della libreria `GWMEngineInterface` per implementare i servizi GO-WS costituisce un particolare caso d'uso, in quanto è stata realizzata come libreria di uso generico.

La libreria contiene il namespace `GWMEngineInterface` al cui interno sono definite le classi che implementano le quattro tipologie di operazioni GIS presentate precedentemente.

Tutte le classi hanno in comune alcuni metodi e proprietà; due proprietà in particolare permettono di descrivere gli errori che si sono verificati durante l'invocazione di un metodo:

- `GeoEngineErrorMsg`: riporta la descrizione di un fallimento attribuito al motore GIS;
- `SystemErrorMsg`: descrive un errore di sistema scaturito da altre condizioni non legate a GWM;

4.2.2.1. Uso delle Classi

L'uso di una delle classi del namespace `GWMEngineInterface` avviene secondo una procedura, detta SDES, composta da quattro passi:

1. *Start-up*: attivazione del motore GIS;
2. *Data access*: accesso ai dati geografici;
3. *Execution*: esecuzione dell'operazione GIS;
4. *Shut-down*: disattivazione motore GIS.

Seguendo questo schema si è certi che alla fine dell'esecuzione lo stato di GWM sarà ripristinato. Ciò è dovuto al fatto che gli oggetti COM istanziati, relativi alle librerie costituenti il motore GIS, sono rilasciati in modo controllato e del tutto invisibile all'utente. Se questo non accade, il comportamento di GWM ad ogni esecuzione è influenzato dalle precedenti.

4.2.2.1.1. Attivazione del motore GIS

Tutte le classi contenute all'interno del namespace `GWMEngineInterface` sono accomunate dal meccanismo di attivazione di GWM che avviene tramite i costruttori delle classi, in modo:

- diretto: al costruttore della classe interessata viene fornito il riferimento (*handler*) alla classe `GWMMapServerHelper` di GWM;
- indiretto: la classe che implementa una tipologia di operazioni geografiche, abilita il motore GIS usando il riferimento alla classe `HTTPServerUtility` del framework .NET passatogli tramite il costruttore. Tale classe è utilizzata per processare le richieste web.

4.2.2.1.2. Accesso ai Dati

L'accesso ai dati avviene tramite il metodo `DataAccess`, i dataset geografici sono identificati tramite istanze della classe `LayerIdentifier` discussa in precedenza. Oltre ai layer è possibile specificare un'area di interesse descritta tramite la classe `BoundingBox`, definita all'interno del namespace `GenericLayerIdentifier`.

Il metodo restituisce i dataset richiesti, modellati tramite istanze della classe `GRecordset` di GWM, se non si sono presentati errori. Nel caso contrario, il metodo restituirà un valore logico corrispondente a "falso" e la descrizione dell'errore sarà riportata attraverso le due proprietà preposte.

4.2.2.1.3. Esecuzione dell'Operazione GIS

In generale le operazioni GIS accettano due dataset come operandi e restituiscono un nuovo dataset contenente il risultato. È stato scelto di limitare il numero di dataset a due solo per semplificare l'implementazione dell'infrastruttura, dato che GWM non impone limiti al riguardo.

Un'altra semplificazione è stata effettuata scegliendo di operare su interi dataset, limitati da un'area di intereresse, anziché su un insieme di feature. Specificare un insieme di feature afferenti ad un layer implica dover indicare un criterio di restrizione sui dati. Per poterlo identificare un insieme basta aggiungere una proprietà, alla classe preposta all'identificazione di un layer, in cui riportare la restrizione sotto forma di costrutto "SQL-WHERE".

4.2.2.1.4. Disattivazione del motore GIS

L'architettura di GWM pone un forte vincolo sull'uso del motore GIS in quanto una sua istanza può servire solo una richiesta alla volta; dopo che l'operazione è stata effettuata è necessario disattivare GWM deallocando l'oggetto `MapServer` istanziato dalla classe che implementa l'operazione. L'eliminazione di tale oggetto COM può avvenire in due modi:

- diretto, tramite il metodo `ShutDownGeoEngine` comune a tutte le classi;
- indiretto, tramite il distruttore della classe.

Da notare che qualsiasi richiesta rimane pendente se effettuata prima del rilascio dell'istanza del motore GIS.

4.2.2.2. Classi di GWM-Engine Interface

Il modello delle classi che implementano l'interfaccia GWM-Engine Interface è molto semplice: esso prevede la classe astratta `CommonInterfaceElements` in cui sono definiti gli elementi necessari ad implementare la procedura SDES. In figura 4.12 è

illustrato il modello semplificato delle classi, per una descrizione completa vedere l'appendice C.

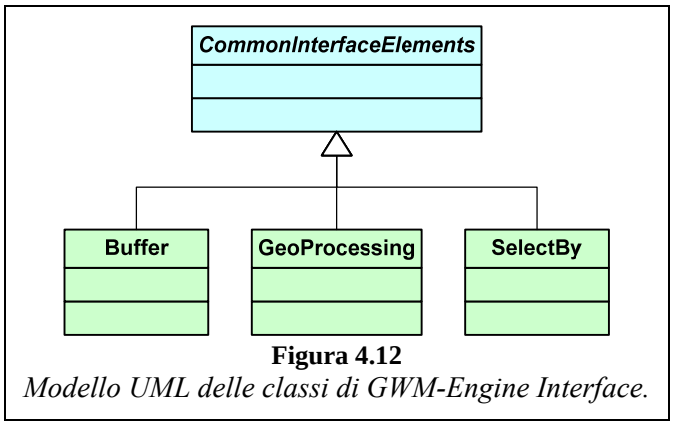


Figura 4.12
Modello UML delle classi di GWM-Engine Interface.

Le classi che ereditano da `CommonInterfaceElements`, raggruppano le operazioni geografiche nelle quattro tipologie presentate.

La classe `GeoProcessing` implementa le seguenti operazioni:

- **`Clip`**: effettua l'operazione di Clip di un dataset utilizzando la geometria definita mediante coordinate nello stesso sistema di riferimento del dataset di input;
- **`Intersect`**: esegue l'operazione di Intersezione fra due dataset non nulli;
- **`Merge`**: esegue l'operazione di Merge;
- **`Union`**: effettua l'operazione di Unione fra due dataset;

La classe `SelectBy` implementa le operazioni di selezione per tema. Tali operazioni restituiscono una lista contenente gli identificatori delle feature selezionate. I metodi esposti sono i seguenti:

- **`Distance`**: dati due dataset ed una distanza espressa nell'unità di misura dei dataset, seleziona le feature del dataset attivo (relativo al layer attivo) che si trovano alla data distanza dalle feature del dataset di selezione (relativo al layer di selezione).
- **`Overlap`**: dati due dataset D_1 e D_2 (rispettivamente, attivo e di selezione) e due generiche feature f_1 e f_2 tale che $f_1 \in D_1$ e $f_2 \in D_2$. Allora, il metodo seleziona tutte le feature di D_1 che rispettano uno dei seguenti criteri spaziali:
  - f_1 contiene f_2;
  - f_1 interseca f_2;
  - f_1 contiene f_2.

La classe `Buffer` espone i metodi necessari ad implementare due tipi di operazioni di buffering, da eseguire su di un dataset:

- **`BufferByDistanceConst`**: per le feature del dataset, crea una o più fasce di rispetto definendole come successione di intervalli di distanze della forma "*inf…sup*", ad esempio "0..100;101..200". Il metodo necessita dell'unità di misura con cui la distanza è indicata.
- **`BufferByDistanceField`**: permette di costruire un'area di rispetto attorno ad una feature in base al valore di un suo attributo numerico.

Per entrambi i metodi è possibile indicare se il buffer deve avere i bordi squadrati od arrotondati. E' possibile inoltre considerare i buffer che si intersecano come un'unica feature.

4.2.2.3. GWMEngineInterface.Service

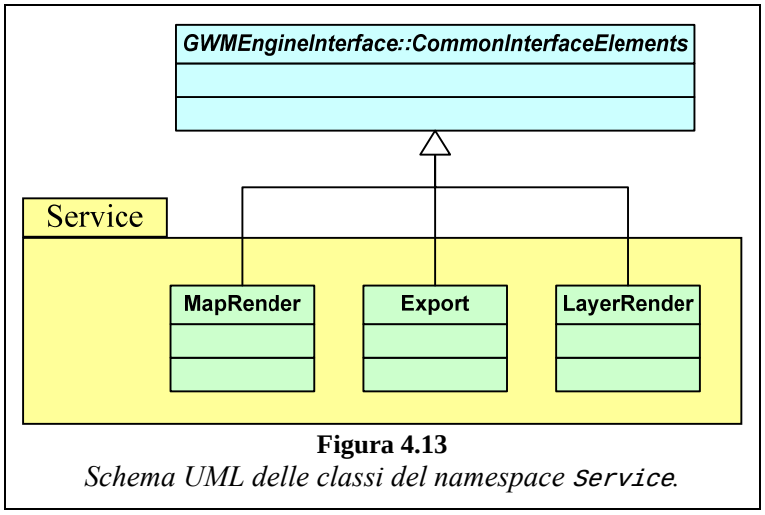


Figura 4.13
Schema UML delle classi del namespace `Service`.

All'interno di questo namespace, figura 4.13, è definita una classe per materializzare i dataset e due classi per creare mappe a partire da dataset geografici e temi:

- **`Export`**: permette di materializzare un dataset in GML od in formati proprietari (attualmente solo *shape file)*. I metodi di esportazione accettano un dataset, un nome con cui riferirlo ed un percorso (sul filesystem) dove materializzare i dati;
- **`LayerRender`**: consente di restituire un dataset in un formato grafico. Perché questo possa essere fatto, è necessario fornire un sistema di riferimento ed uno stile di visualizzazione;
- **`MapRender`**: per poter creare una mappa sono necessari un insieme di temi, un sistema di riferimento ed eventualmente un'area di interesse. Il metodo `RenderMap` accetta questi parametri assieme alla dimensione in pixel ed il formato grafico in cui restituirla. A differenza delle precedenti classi, l'accesso

ai dati è eseguito indirettamente in quanto al metodo sono indicati i temi tramite la collezione di tipo `ThemeList`, contenente la lista di temi descritti tramite i parametri di identificazione (figura 4.8). Il sistema di riferimento di mappa è indicato tramite il percorso su filesystem ad un file, proprietario di GWM, che lo definisce. Il metodo restituisce, in caso non vi siano fallimenti, una URL alla cache di GWM dove è contenuto il file grafico. Sono previsti sia formati raster sia vettoriali, quali rispettivamente:

- JPG, PNG, GIF;
- SVG, CGM.

Le prime due classi sono state progettate in modo tale da poter sfruttare gli ambienti, creati dall'uso delle classi `GeoProcessing` e `Buffer`. per restituire il risultato delle operazioni geografiche.

4.2.3. GIS Operation - Web Service

I servizi GO-WS, implementati basandosi sulla libreria di classi GWM-Engine Interface, comprendono i seguenti web service:

1. **BufferingService**: espone operazioni per creare aree di rispetto. La distanza fra l'oggetto geografico ed il bordo dell'area di rispetto è definita mediante:
   - valori costanti;
   - un valore specificato da un attributo numerico del dataset.
2. **GeoProcessingService**: espone operatori per eseguire le seguenti operazioni:
   - Unione,
   - Intersezione,
   - Clip,
   - Merge.
3. **GeoSelectionService**: fornisce funzionalità per effettuare operazioni di selezione per tema;

Per la creazione di mappe è stato creato un servizio simile a WMS SLD di OGC:

4. **MapRenderService**: consente di creare e restituire graficamente una mappa;

Inoltre è stato previsto un servizio secondario, non basato sulla libreria GWM-Engine Interface, per trasformare un insieme di coordinate in diversi sistemi di riferimento:

5. **SystemCoordTransform**: i sistemi di riferimento spaziali su cui il servizio opera, sono un numero finito e preordinato; il servizio dispone di un metodo, `GetCoordSystemList`, per fornirli sotto forma di lista di nomi secondo la convenzione EPSG.

La trasformazione[44] fra sistemi di riferimento avviene in due passaggi: sia il sistema di coordinate di partenza (ad esempio di una feature) SRS_A, SRS_B quello di arrivo (ad esempio il sistema di riferimento di mappa) e SRS_G il sistema di riferimento spaziale in coordinate geografiche. Allora, la trasformazione da SRS_A a SRS_B è realizzata in due passaggi:

1. da SRS_A a SRS_G,
2. da SRS_G a SRS_B[45].

Naturalmente questa operazione è resa possibile sfruttando le classi appartenenti a GWM.

I primi quattro servizi principali utilizzano l'interfaccia GWM-Engine Interface per implementare le operazioni GIS ed i primi tre seguono lo schema SDES. Il risultato di un'operazione relativa al servizio di selezione per tema, consiste in una lista strutturata di identificatori di feature appartenenti al layer attivo. I restanti servizi restituiscono una o più URL che puntano al dataset materializzato, secondo il formato di codifica richiesto, oppure al documento grafico che rappresenta la mappa.

I servizi di geoprocessing e buffering possono restituire graficamente il risultato dell'operazione, eseguita senza che sia materializzato, mediante la URL al documento grafico.

L'implementazione del servizio `MapRenderService` non utilizza la procedura SDES in quanto per la filosofia della classe `MapRender`, la mappa richiesta è creata "*on the fly*" a partire dalla descrizione dei temi. Per maggiori dettagli sulla realizzazione dei servizi, vedere appendice D.

L'invocazione di un GO-WS avviene specificando quale operazione deve essere effettuata, su quali dati (il contenuto) e come codificare il risultato (formato del contenuto della risposta).

Per quanto detto nel paragrafo 4.2.1, perché GWM possa localizzare un dataset, deve possedere un insieme di informazioni. Non è pensabile quindi implementare metodi di web service con un numero elevato di parametri formali, tanto più che nel caso della creazione di una mappa il numero di temi non è prefissato. Quindi è necessario un modo semplice e

---

[44] L'uso del termine "trasformazione" è parzialmente esatto in quanto si parla di "trasformazione" quando il sistema di riferimento delle coordinate di partenza e di arrivo hanno datum differente, altrimenti l'operazione è indicata con "conversione".

[45] La concatenazione di trasformazioni ha come effetto negativo la propagazione di errori analitici dovuti al passaggio intermedio di trasformazione. Per i fattori di scala utilizzati, tali errori possono essere considerati accettabili.

flessibile per veicolare tutti i parametri senza perdere di vista l'obiettivo di integrazione ed interoperabilità fra sistemi eterogenei, proprio dell'architettura SOA.

4.2.3.1. Codifica Dei Parametri

Un client richiede l'esecuzione di un'operazione per poter ottenere dei contenuti come mappe, dataset oppure identificatori per selezionare le feature di una mappa.

Per poter richiede tali contenuti bisogna specificare un insieme di parametri per indicare ad un web service GO-WS quali dati utilizzare e come codificarli. Una volta che l'operazione è stata eseguita il servizio deve poter comunicare all'entità richiedente il risultato dell'operazione eseguita, oppure comunicare la presenza di un fallimento.

Nel contesto dei servizi GO-WS, un metodo che implementa un'operazione geografica, riceve un **messaggio di richiesta** composto da un insieme di parametri strutturati. L'esecuzione del metodo comporta la restituzione al client di un **messaggio di risposta** in cui sono riportate una serie di informazioni strutturate che rappresentano il risultato dell'operazione.

Un messaggio è caratterizzato da un **contenuto** che indica cosa trasporta. Un messaggio può avere come scopo la richiesta[46] di una mappa; la mappa quindi costituisce il contenuto del messaggio di richiesta.

Per aggiungere maggiore dettaglio a ciò che un messaggio trasporta, si introduce il **formato** del contenuto; una mappa può essere richiesta in un determinato formato graficoa (SVG, JPG, …), un dataset in un formato di materializzazione ben preciso come GML. Il formato ed il contenuto indicano di conseguenza come la risposta verrà codificata, a meno di fallimenti.

I possibili contenuti previsti per un messaggio sono:

- **ERR**: indica che un'operazione ha causato un fallimento;
- **IDS**: insieme di identificatori relativi a feature;
- **LYR**: indica un dataset che rappresenta un layer;
- **MAP**: mappa;

Ad esempio, il contenuto LYR in un messaggio di richiesta, indica che il client con l'operazione invocata si aspetta un dataset; in una risposta, lo stesso contenuto, indica al client che il massaggio trasporta un dataset.

[46] Il messaggio ha uno scopo che consiste nel "cosa richiedere" e nel "rispondere".

I messaggi di richiesta e di risposta possono essere caratterizzati dallo stesso contenuto, ad eccezione del contenuto “ERR” impiegato solo nei messaggi di risposta per indicare la presenza di un errore, figura 4.14.

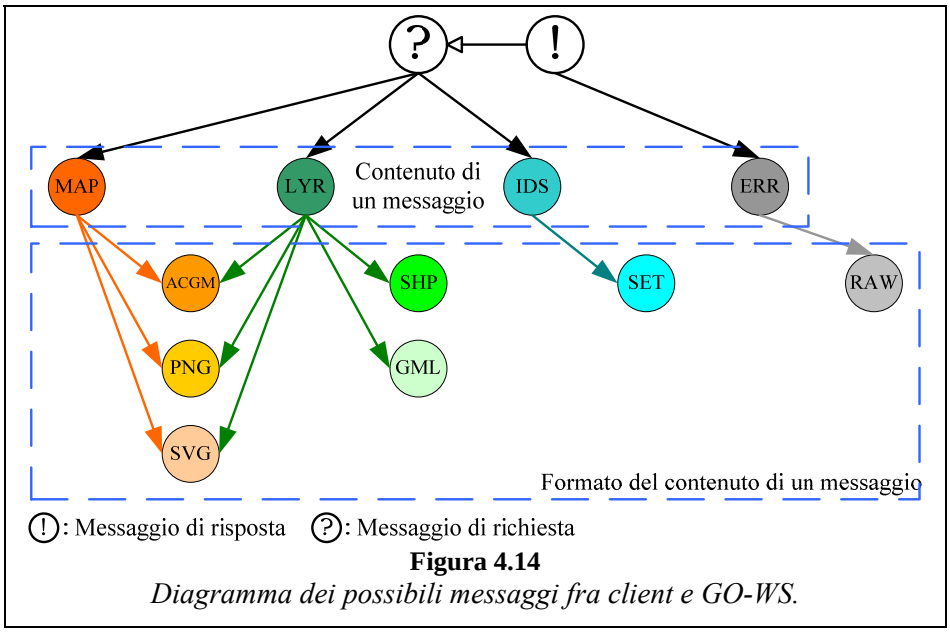


Figura 4.14

Diagramma dei possibili messaggi fra client e GO-WS.

Sono previsti i seguenti formati dei contenuti:

- ACGM, SVG, PNG, …: formati grafici in cui una mappa oppure un dataset possono essere rappresentati;
- SHP, GML: formati di materializzazione previsti per un dataset;
- SET: indica che il risultato è un insieme di elementi di vario tipo;
- RAW: formato “grezzo” è utilizzato esclusivamente per i messaggi di errore per indicare che sono codificati come stringhe di testo.

I percorsi dalla radice alle foglie, nell’albero in figura 4.14, mostrano i possibili messaggi supportati dai servizi. L’implementazione effettuata evidenzia una corrispondenza fra contenuto del messaggio di richiesta ed i servizi GO-WS:

- LYR : `GeoProcessingService` / `BufferingService`;
- MAP : `MapRenderService`;
- SET : `GeoSelectionService`.

Va sottolineato che è possibile combinare ulteriormente contenuti e formati. Niente vieta di esprimere un insieme di identificatori di feature in GML oppure restituire le feature selezionate in SVG.

Resta da descrivere come strutturare i parametri costituenti un messaggio. Dato che SOAP può trasportare informazioni testuali, è possibile creare un messaggio in XML ed

inviarlo come parametro attuale dei metodi dei quattro “GIS Operations - Web Service” principali.

Sulla base dei possibili messaggi XML, sono stati creati i rispettivi schemi XML che ne definiscono la struttura.

A livello pragmatico creare messaggi XML via codice e valicarli mediante gli schemi XML creati, non è agevole. Al contrario, risulta più naturale modellare un messaggio come un insieme di classi e poi produrre una stringa XML valida.

Il framework .NET offre il meccanismo di serializzazione/deserializzazione che traduce una classe in un file binario oppure in un documento XML a partire da uno schema XSD.

Per costruire un metodo flessibile che permetta di creare a tempo di esecuzione messaggi XML, è stato applicato il seguente procedimento:

1. usando gli schemi XML dei messaggi, mediante lo strumento *XML Schema Definition Tool* del framework .NET, sono state create le classi che modellano ogni singolo messaggio;
2. in base alle classi create, ne sono state progettate di nuove in modo tale da poter disporre di un meccanismo di creazione e definizione messaggi flessibile ed espandibile.

Il risultato ha portato alla definizione dei due modelli illustrati in figura 4.15a e 4.15b.

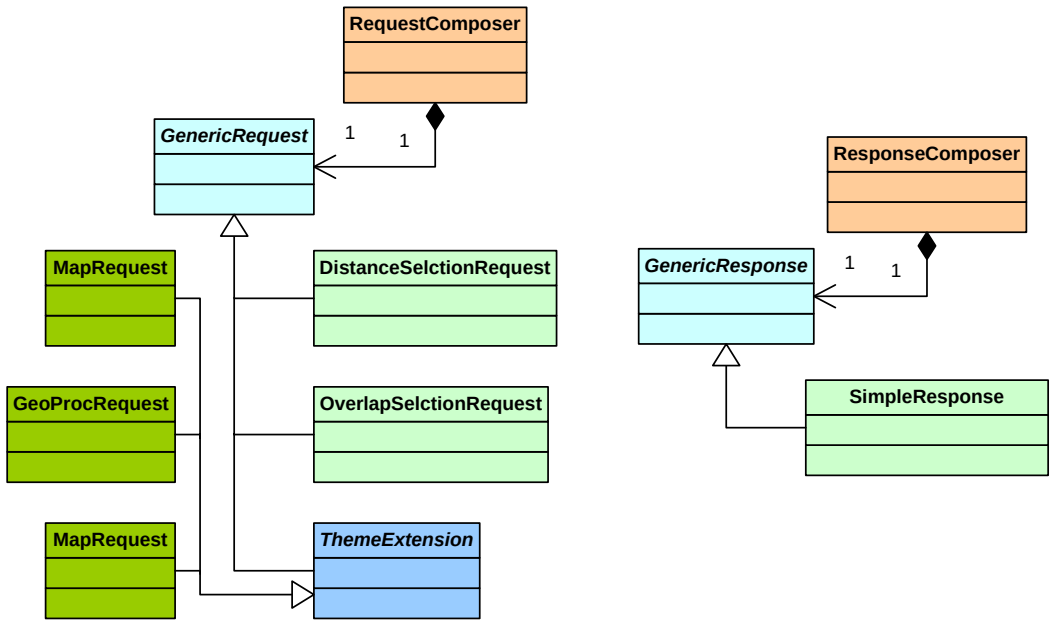


Figura 4.15a **Figura 4.15b**

In figura 4.11a è illustrato il modello delle classi per comporre le richieste, mentre in figura 4.11b le classi per comporre le risposte. In azzurro sono state evidenziate le classi astratte che modellano i messaggi generici mentre in verde le classi derivate dalle astratte, per modellare i messaggi specifici.

Per visionare gli schemi XML relativi ai messaggi, vedere appendice E.

4.2.3.1.1. Messaggi di Richiesta

Analizzando l'insieme di tutti i parametri necessari all'esecuzione di tutte le operazioni, se ne può estrarre un sotto insieme comune a tutti, che contiene:

- identificazione di un layer;
- una box di contenimento definita sul layer;
- un sistema di riferimento spaziale in cui eseguire le operazioni.

Facendo riferimento a quanto illustrato per la figura 4.14, è necessario indicare inoltre il contenuto del messaggio ed il formato del contenuto.

La classe astratta `GenericRequest` definisce questi elementi, i parametri necessari ad identificare il dataset, sono modellati utilizzando la classe per l'identificazione di un layer (`LayerIdentifier`).

In seconda analisi si può identificare un secondo insieme di attributi da utilizzare per poter trasformare un dataset in un formato grafico. Tali attributi sono:

- lo stile di visualizzazione;
- l'altezza e la larghezza in pixel dell'immagine.

Questa caratterizzazione è applicabile solo per operazioni di geoprocessing, buffering e restituzione cartografica. La classe astratta `ThemeExtension` (che eredita da `GenericRequest`, figura 4.13a) permette quindi di modellare le seguenti classi:

- **`GeoProcRequest`**: modella la richiesta per le operazioni di geoprocessing. Al suo interno è definito il secondo operando, di tipo `LayerIdentifier` e lo stile di visualizzazione del layer risultante;
- **`MapRequest`**: definisce la richiesta per l'operazione di creazione di una mappa a partire da una lista di temi, tramite il servizio "`MapRenderService`". la lista di temi è modellata mediante la classe `ThemeList`;
- **`BufferingRequest`**: consente di definire una richiesta per un'operazione di buffering al web service "`BufferingService`". Gli attributi che fanno parte di questa classe sono l'ampiezza dell'area di rispetto, la rispettiva unità di misura, l'aspetto dei bordi (smussati o meno) e lo stile di visualizzazione in caso di restituzione grafica. L'attributo `distanceCriteria` indica l'ampiezza del buffer che può essere espressa sia come valore costante oppure come attributo numerico di una feature class. La semantica dell'attributo del messaggio viene determinata in base al metodo invocato:
- **`CostantBuffering`**: crea aree di rispetto in base a valori costanti;

- **`FieldBuffering`**: l'area di rispetto è definita attraverso i valori di un attributo numerico.

Mentre queste tre classi ereditano da `ThemeExtension` per poter includere proprietà legate alla restituzione grafica di un layer; le seguenti invece ereditano direttamente da `GenericRequest`:

- `OverlapSelectionRequest`: definisce i parametri necessari al metodo `SelectByOverlap` del web service `GeoSelectionService` ed indica il parametro "distanza" necessario al criterio di confronto spaziale "distanza-da";
- `DistanceSelectionRequest`: richiesta destinata al metodo `SelectByDistance` del web service `GeoSelectionService` ed indica un criterio di confronto spaziale non basato sulla distanza.

Gli ultimi due messaggi di richiesta sono destinati allo stesso servizio, ma a metodi diversi in quanto nel primo caso è indicato un criterio di confronto spaziale mentre nel secondo, una distanza mediante un valore numerico.

4.2.3.1.2. Messaggi di Risposta

La classe astratta `GenericResponse` definisce la struttura di base di un messaggio di risposta, che prevede l'indicazione del contenuto del messaggio e del formato ad esso legato.

Un messaggio di risposta trasporta:

1. un riferimento (URL) per accedere il risultato, questo è il caso di un dataset materializzato oppure di una mappa;
2. il risultato stesso nel caso di un insieme di identificatori o per i messaggi di errore.

Nel primo caso sopra elencato, svincolando il trasporto del risultato dal protocollo con cui è restituito il messaggio di risposta, si hanno alcuni vantaggi:

- trasferire un dataset via FTP dato che spesso hanno dimensioni consistenti (superiori ad 1Mbyte);
- utilizzare protocolli di trasporto sicuri come HTTPS.

La classe `SimpleResponse` modella una "risposta semplice" e non aggiunge nessun attributo rispetto allo schema della classe base. Questa costruzione è stata adottata per mantenere la simmetria con le classi dedicate alla definizione di richieste.

4.2.3.1.3. Compositori di Messaggi

Lo scambio di messaggi che avviene fra un generico client ed un GO-WS, illustrato in figura 4.16, prevede una fase di serializzazione della classe che modella la richiesta, il

documento XML prodotto, è trasmesso via HTTP nella sezione *Body* di SOAP. Il servizio riceve il documento XML, lo deserializza ed infine utilizza i parametri inviati accedendo alle proprietà che li rappresenta.

Lo stesso accade per la risposta che il servizio crea.

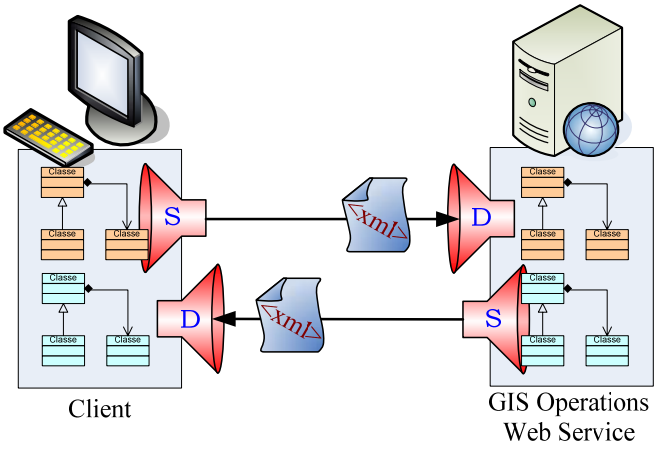


Figura 4.16
Meccanismo di scambio messaggi tramite serializzazione/deserializzazione ("S" e "D") fra un generico client ed un servizio GO-WS.

Le classi `RequestComposer` e `ResponseComposer` automatizzano la creazione dei messaggi e la loro trasformazione da oggetti *runtime* a XML e viceversa. Il costruttore di entrambe le classi istanzia automaticamente una delle classi sopra esposte in base al messaggio da creare. Una volta che il messaggio è stato costruito, tramite il metodo `FormalizeRequestToXml` (`FormalizeResponseToXml` per le risposte) la classe relativa al messaggio è serializzata in XML e restituita come stringa di testo. Viceversa, il metodo `FormalizeRequestFromXml` (`FormalizeResponseFromXml` per le risposte) deserializza un messaggio ed istanzia la classe relativa al messaggio.

Le due classi appena presentate, per quanto riguarda i client che interagiscono con i GO-WS, sono da considerarsi uno strumento opzionale per semplificare la composizione dei messaggi di richiesta e risposta.

4.2.3.2. Note sui GIS Operations - Web Service

L'uso delle classi per modellare i messaggi di richiesta non limita l'impiego dei servizi GO-WS solo in ambiente .NET; ogni web service implementato espone un metodo per recuperare via HTTP lo schema XML (XSD) dei messaggi. Le classi sopra illustrate sono da considerarsi solo uno strumento per semplificare la creazione dei messaggi sfruttando il meccanismo di serializzazione/deserializzazione del *framework* .NET.

Per maggiori dettagli sulle classi precedentemente descritte, vedere appendice E.

4.2.4. Local Catalogue Service

Tutti i servizi sopra presentati utilizzano meta informazioni[47] per poter operare sui dati offerti da uno o più servizi OGC. Tali informazioni, riportate nei documenti di capacità dei servizi, sono accessibili tramite il servizio CAT di OGC.

Dato che non è stato possibile né reperire un'interfaccia per interagire con i servizi OGC CAT, né implementarne una in tempi brevi, è stato realizzato un servizio di catalogo, *Local Catalogue Service* (LCS). Tale servizio è pubblicamente accessibile ed in grado di fornire semplici operazioni per memorizzare le meta informazioni provenienti dalle istanze dei servizi WFS e WMS.

A differenza dei servizi GO-WS, questo non necessita la presenza di un motore GIS in quanto interagisce direttamente con i servizi tramite il protocollo HTTP. L'unico elemento software necessario è un DBMS in grado di garantire sia la serializzabilità che l'affidabilità per prevenire inconsistenze all'interno della base di dati.

L'architettura del servizio LCS prevede (figura 4.17), oltre ad un DBMS:

- un'interfaccia verso il catalogo per offrire dei metodi di manipolazione sulla base di dati, *Core Catalogue Interface*;
- un web service che espone operazioni per interagire con il catalogo.

Per quanto riguarda il DBMS, Microsoft JET SQL è stato ritenuto valido, data l'esigua entità dei dati coinvolti. L'impiego di ADO.NET consente di usufruire di un insieme di classi, organizzate all'interno del namespace `System.Data`, necessarie per interagire con il DBMS.

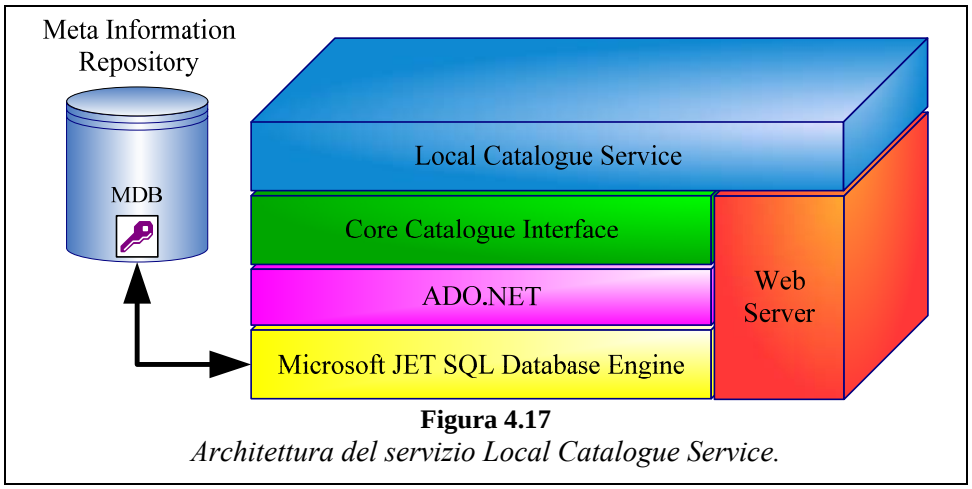


Figura 4.17
Architettura del servizio Local Catalogue Service.

[47] Notazione: in questo contesto è stato appositamente utilizzato il termine "meta informazioni" per non confonderlo con il concetto di "metadati" relativi ai dati geografici..

4.2.4.1. La Base di Dati

Il nucleo del catalogo è una base di dati in cui sono memorizzate le meta informazioni riportate all'interno dei documenti delle capacita delle istanze dei servizi OGC.

Il catalogo è stato strutturato per mantenere meta informazioni riguardanti:

- l'istanza di servizio OGC: dove si trova, di quale tipo di servizio OGC si tratta, i termini chiave ed il nome assegnatogli (per i dettagli relativi a questi attributi, vedere la tabella 2.1 di capitolo 2.1.3.1);
- gli strati informativi: per ogni istanza di servizio OGC sono riportare informazioni relative ai layer, quali: nome, tipo di geometria, termini chiave, descrizione, posizione dei metadati e posizione del dataset;
- il sistema di riferimento e box di contenimento di uno strato: per ogni strato informativo sono riportati i sistemi di riferimento spaziali supportati dal servizio OGC.

Lo schema illustrato in figura 4.18, mostra come è stato modellato il catalogo (*Meta Information Repository*, MIR). La relazione `Tbl_SRSBox` riporta descrive le coppie <"*sistema di riferimento spaziale*","*bounding box*"> (<SRS,BBox>) in cui un servizio fornisce gli strati informativi.

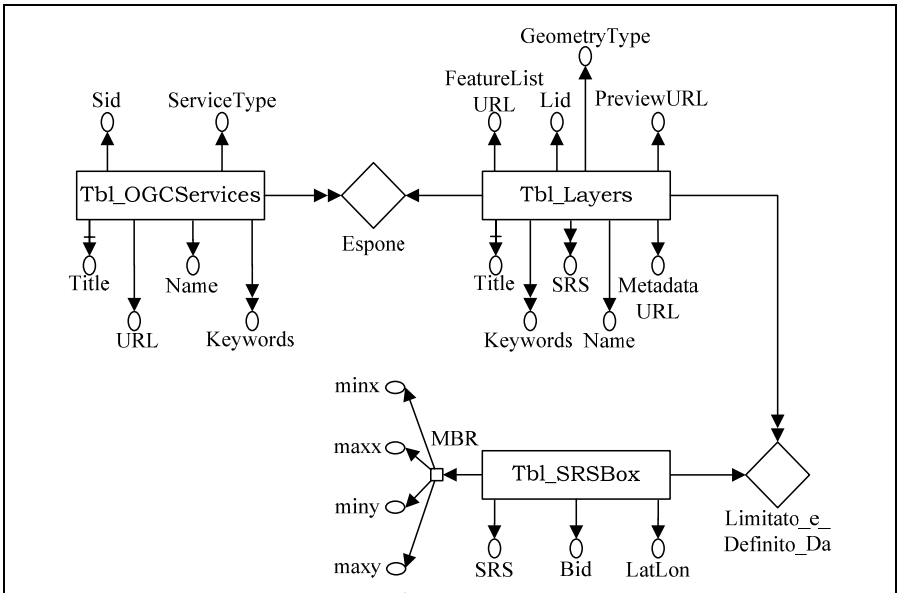


Figura 4.18

Schema ad oggetti del Meta Information Repository; per il relativo schema entità-relazioni vedere appendice F.

Gli attributi di ogni relazione fanno riferimento alle meta informazioni del documento delle capacità di un WFS o WMS; tranne i seguenti:

- `Sid`: identifica univocamente un'istanza di servizio all'interno del MIR;
- `ServiceType`: indica il tipo di servizio: WFS o WMS;

- `Lid`: identifica univocamente uno strato informativo;
- `Bid`: identifica le tuple che modellano le coppie <SRS,BBox>;
- `LatLon`: indica se la coppia <SRS,BBox> corrisponde alla meta informazione *LatLongBoundingBox*.

4.2.4.2. OWS-ServiceCrawler

Per poter riportare le meta informazioni contenute nel documento delle capacità all'interno del catalogo, è stato progettato un insieme di classi che costituiscono la libreria (ed il namespace) `OWSServiceCrawler`, che consente di creare una corrispondenza con il documento delle capacità di un istanza di servizio WFS e WMS.

All'interno della libreria, sono definite tre classi astratte che si occupano di modellare gli elementi del documento delle capacità comuni a tutti i servizi:

- `ServiceClass`: contiene caratteristiche comuni ai servizi, come: URL HTTP, versione, tipo di servizio;
- `ServiceInfosClass`: riporta le meta informazioni legate al servizio;
- `LayerClass`: contiene le meta informazioni relative ad un layer esposto dal servizio.

Da queste classi base, ne sono state derivate altre specializzate per i servizi WFS e WMS. La creazione della corrispondenza avviene istanziando una classe derivata da `ServiceClass` che ha il compito avviare il processo di estrazione delle meta informazioni istanziando la classe derivata da `ServiceInfosClass`. Questa estrae le meta informazioni relative al servizio ed infine crea tante istanze della classi derivate da `LayerClass`, quanti sono i gli strati informativi esposti dai servizi.

Per i servizi WMS (le classi evidenziate di giallo in figura 4.19) le categorie sono state modellate con una classe supplementare `WMS_LayerHierarchyClass` che può contenere istanze delle classi `WMS_LayerClass` oppure ulteriori istanze di se stessa.

Per maggiori dettagli sulle classi del namespace `OWSServiceCrawler`, vedere l'appendice F.

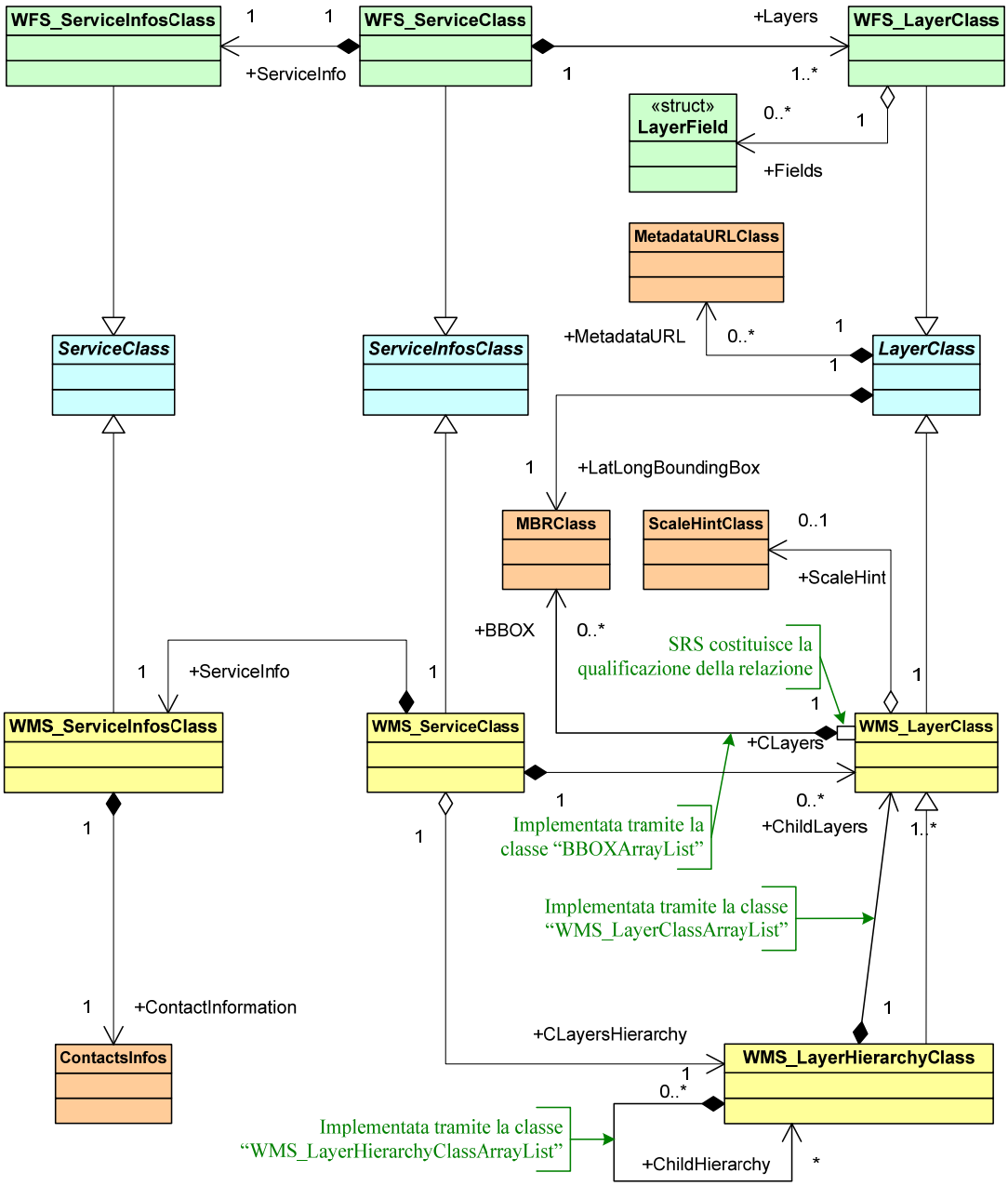


Figura 4.19

Modello ad oggetti della libreria OWSServiceCrawler. In azzurro sono state evidenziate le tre classi astratte, in verde le classi derivate per gestire la corrispondenza con i servizi WFS ed in giallo per i servizi WMS; in arancio sono indicate le classi di utilità.

4.2.4.3. Core Catalogue Interface

L'interfaccia con la base di dati è costituita da un'unica classe, `CoreManager`, che espone i metodi necessari a memorizzare e gestire le meta informazioni riportate all'interno dei documenti delle capacità. Il framework .NET offre supporto alla manipolazione dei documenti XML attraverso il namespace `System.Xml`, mentre ADO.NET per l'interazione con il DBMS; figura 4.20.

Il flusso di dati fra i due tipi di documenti delle capacità ed il MIR, è realizzato attraverso le classi del namespace `OWSServiceCrawler` che veicolano le meta informazioni dal documento delle capacità, verso il catalogo.

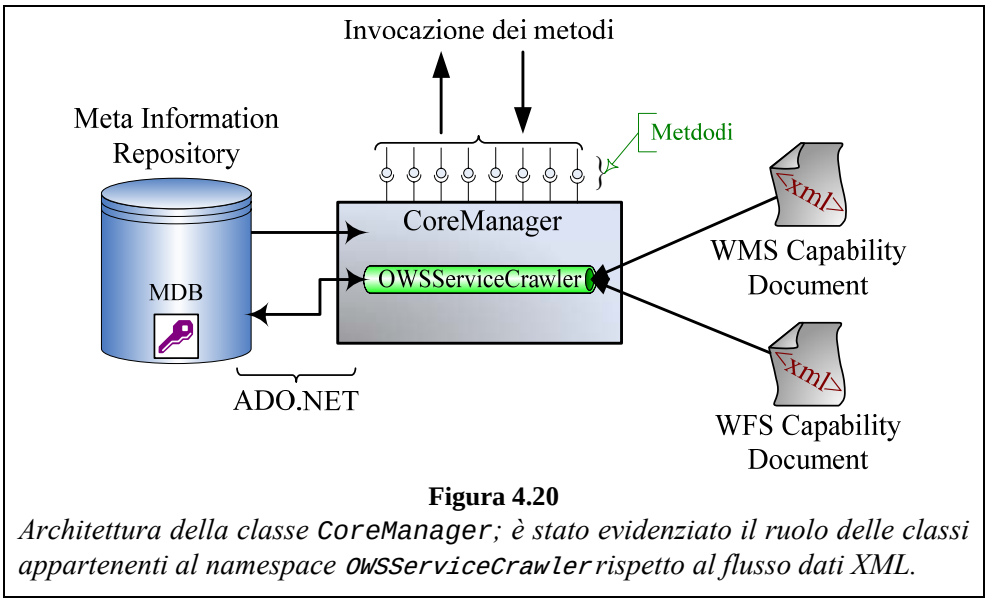


Figura 4.20
Architettura della classe `CoreManager`; è stato evidenziato il ruolo delle classi appartenenti al namespace `OWSServiceCrawler` rispetto al flusso dati XML.

I metodi che la classe `CoreManager` espone, sono raggruppati in due insiemi:

- **manipolazione del catalogo**: sono i metodi che permettono di aggiungere, eliminare e sincronizzare le informazioni contenute nel catalogo in base alle meta informazioni del documento delle capacità:
  - `AddService`: memorizza le meta informazioni, ottenute istanziando una classe derivata da `ServiceClass`, all'interno del MIR;
  - `UpdateService`: sincronizza le meta informazioni relative ad un'istanza di servizio, memorizzate all'interno del catalogo, utilizzando la rappresentazione tramite `ServiceClass` della stessa istanza di servizio disponibile in rete;
  - `DeleteService`: elimina le meta informazioni relative ad un'istanza di servizio tramite:
    - la URL dell'istanza;
    - attraverso la chiave che lo identifica univocamente all'interno del catalogo.
- **recupero meta informazioni**: metodi per recuperare meta informazioni memorizzate nelle relazioni del MIR:
  - `GetWFSServiceFromCatalog`: permette di rappresentare un servizio memorizzato all'interno del catalogo attraverso la classe `WFS_ServiceClass`;

- `GetWMSServiceFromCatalog`: permette di rappresentare un servizio memorizzato all'interno del catalogo attraverso la classe `WMS_ServiceClass`;
- `ExecuteSFW`: esegue un'interrogazione, espressa attraverso il costrutto SQL "Select-From-Where", sul catalogo. Restituisce un dataset implementato attraverso la classe `DataSet` del namespace `System.Data` di .NET;
- `GetSingleValue`: come il precedente metodo, tranne che è ottimizzato per restituire un solo valore.

4.2.4.4. Il Servizio Web

Il servizio di catalogo utilizza l'interfaccia `CoreCatalogueInterface` per implementare le operazioni di manipolazione remota del catalogo MIR.

Come precedentemente esposto nel capitolo 2, in generale un servizio è identificato attraverso la sua URL ed il tipo di servizio OGC. L'invocazione di un metodo del servizio "Local Catalogue Service" (che riguarda un inserimento od una sincronizzazione) scaturisce una richiesta HTTP-GET all'istanza del servizio OGC, per ottenere il rispettivo documento delle capacità. Se la richiesta si conclude con esito positivo, viene istanziata una delle classi derivate da `ServiceClass` (del namespace `OWSServiceCrawler`). L'oggetto così creato, è fornito alla classe `CoreManager` invocando il metodo necessario ad implementare l'operazione del servizio. In figura 4.21 è illustrata l'architettura del servizio.

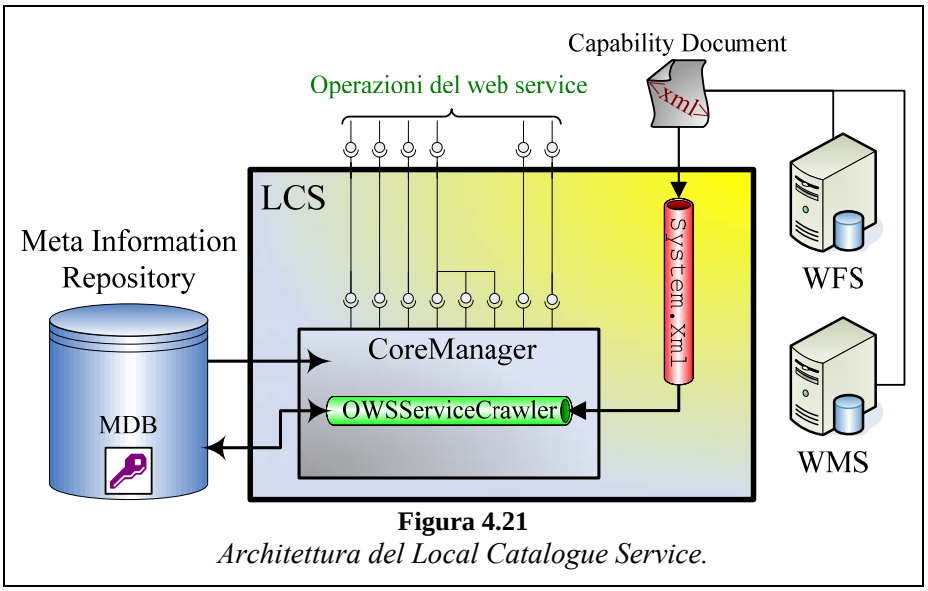


Figura 4.21
Architettura del Local Catalogue Service.

I metodi web esposti sono i seguenti:

- **`Insert`**: inserisce le meta informazioni di un'istanza di servizio, data la sua URL e il tipo di servizio;
- **`DeleteByURL`**: elimina dal catalogo le meta informazioni relative all'istanza di servizio identificata tramite URL e tipo di servizio;
- **`DeleteByID`**: come il metodo precedente, solo che l'istanza di servizio è identificata all'interno del catalogo in base alla sua chiave;
- **`UpdateByURL`**: sincronizza le informazioni di un'istanza di servizio, contenute all'interno del catalogo, tramite la URL e la tipologia del servizio;
- **`UpdateByID`**: la sincronizzazione avviene identificando l'istanza di servizio attraverso la sua chiave primaria relativa al catalogo MIR;
- **`GetServiceList`**: restituisce una lista di triple *<nome_servizio, identificatore_servizio, tipo_istanza_servizio>* ;
- **`GetWFSServiceList`**: come il precedente metodo, tranne che la lista contiene triple relative ad istanze di servizio WFS;
- **`GetWMSServiceList`**: la lista delle triple restituita è relativa ad istanze di servizio WMS;
- **`GetDataFromCatalog`**: questo metodo ha la stessa semantica di `ExecuteSFW`;
- **`Get_a_Value`**: come `GetOneValue`, restituisce un valore in base ad un'interrogazione espressa in SQL.

Per maggiori dettagli sulla classe, vedere appendice F.

Capitolo 5

Applicazioni D'Esempio

Per illustrare un possibile scenario d'uso dell'infrastruttura realizzata, sono state sviluppate due applicazioni web:

- ***Local Catalogue Manager,*** permette di amministrare il contenuto del catalogo MIR interfacciandosi con il servizio "Local Catalogue Service";
- ***GIS Application Service Provider*** (GISASP) un'applicazione WebGIS in grado di offrire un ambiente multi utente e multi progetto per effettuare analisi geospaziali su insiemi di layer provenienti sia da sorgenti dati OGC, sia proprietarie.

5.1. Web Catalogue Manager

L'applicazione web "Web Catalogue Manager" (WCM) è uno strumento scritto in C# e basato sulla tecnologia web form di ASP.NET, che consente a più utenti, mediante il web service LCS, di manipolare (inserire, aggiornare e rimuovere) le meta informazioni relative alle istanze dei servizi OGC WFS e WMS contenute nel catalogo MIR.

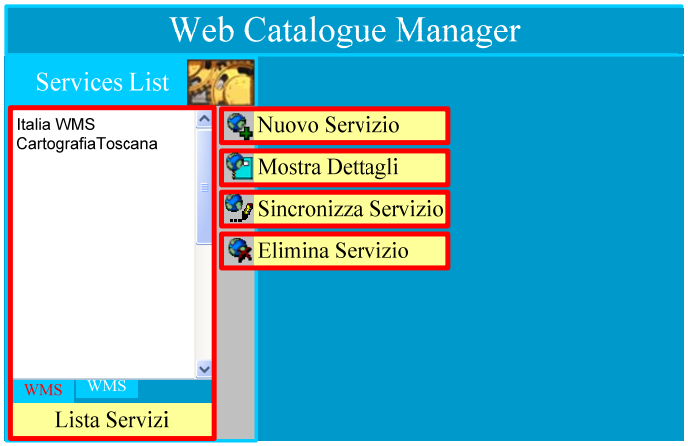


Figura 5.1
Pagina principale dell'applicazione Web Catalogue Manager.

L'applicazione (figura 5.1) è composta da:

- la finestra "`Service List`" in cui è visualizzata la lista delle istanze dei servizi WFS e WMS memorizzati all'interno del catalogo;
- quattro operazioni accessibili tramite bottoni, per:
  - aggiungere un nuovo servizio all'interno del catalogo, "`Nuovo Servizio`";
  - mostrare le meta informazioni associate al servizio selezionato, "`Mostra Dettagli`";

- allinea le meta informazioni di un servizio contenuto nel catalogo, "`Sincronizza Servizio`". Significa sincronizzare le meta informazioni di un servizio, contenute nel catalogo, con quelle dello stesso servizio, ma disponibile in rete;
- eliminare le meta informazioni di un servizio, memorizzate all'interno del catalogo, "`Elimina Servizio`".

L'operazione per aggiungere un nuovo servizio nel catalogo, accessibile mediante il bottone "`Nuovo Servizio`", permette all'utente (amministratore del catalogo) di accedere alla pagina mostrata in figura 5.2, dove è possibile inserire la URL del servizio e selezionare il tipo di istanza; Web Map Service o Web Feature Service.

Il bottone `Insert` determina, lato server, l'invocazione dell'operazione `Insert` offerta dal servizio "Local Catalogue Service"; se l'inserimento è andato a buon fine, il nome del servizio apparirà nella lista dei servizi.

Figura 5.2
Interfaccia per l'inserimento della URL di una nuova istanza di servizio all'interno del catalogo.

Selezionando un servizio nella lista ed attivando il bottone "`Mostra dettagli`", verrà presentata la pagina contenente le meta informazioni del servizio richiesto (figura 5.3).

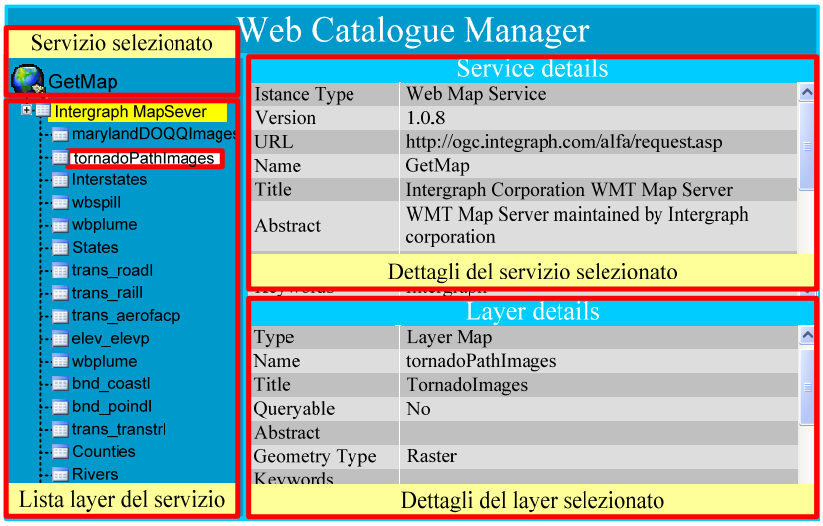


Figura 5.3
Dettagli per il servizio WMS GetMap; `Intergraph MapServer` costituisce il nome della categoria, mentre "`tornadoPathImages`" è il layer selezionato.

Sul lato sinistro della pagina è presente una lista che riporta il nome del servizio ed i relativi layer. Nella parte in alto a destra sono riportate tutte le meta informazioni relative al servizio, mentre in basso a destra quelle inerenti al layer selezionato (`tornadoPathImages`).

Per impaginare le meta informazioni lato server, dato che il catalogo MIR contiene solo una quantità limitata di meta informazioni, vengono sfruttate le classi del namespace `OWSServiceCrawler` per recuperare tutte le meta informazioni relative al servizio. In questo caso il servizio di catalogo è utilizzato solo per ottenere la URL del servizio OWS selezionato.

Le funzionalità di sincronizzazione ed eliminazione delle meta informazioni di un servizio ("`Sincronizza Servizio`", "`Elimina Servizio`"), sono ottenute invocando le rispettive operazioni del servizio LCS.

5.2. GIS Application Service Provider

Per illustrare un possibile impiego dei servizi GO-WS, è stata realizzata una semplice applicazione WebGIS GIS-Online (figura 5.4a) per consentire ad uno o più utenti di operare su insiemi di dati geografici organizzati in **progetti** (figura 5.4b).

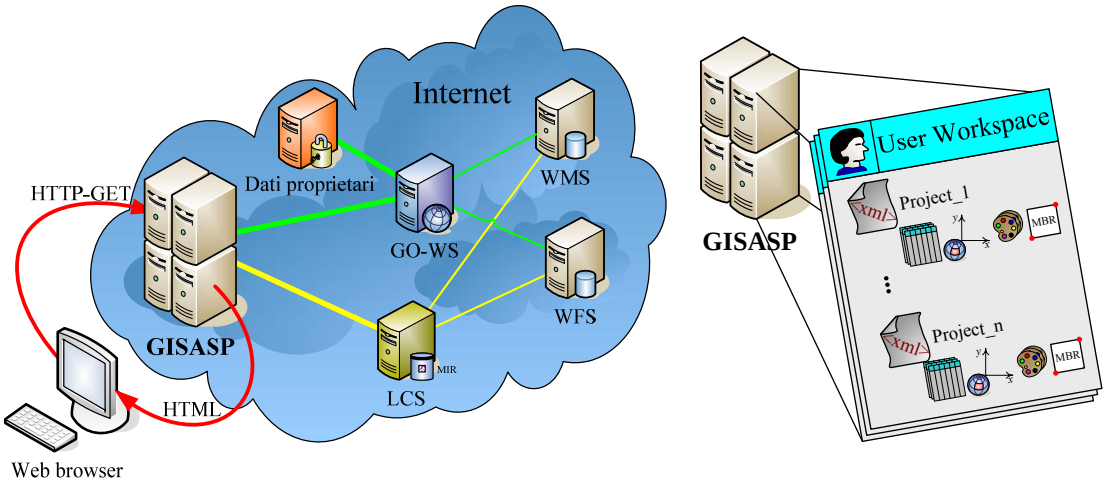


Figura 5.4a **Figura 5.4b**

Architettura dell'applicazione GIS Application Service Provider.

Un progetto è contenuto in un'area dedicata all'utente (*User Workspace*), posta all'interno del server che ospita l'applicazione WebGIS. Il progetto è costituito dalle seguenti informazioni:

- insieme di dataset geografici su cui operare;
- sistema di riferimento spaziale della mappa;
- dimensione della finestra di mappa (espressa in pixel);
- vestizione dei temi;

- estensione della mappa (area di interesse) espressa in coordinate reali.

All'interno del progetto, gli strati informativi sono identificati tramite i quattro parametri già illustrati nel capitolo precedente.

L'applicazione è sviluppata interamente utilizzando la tecnologia web form di ASP.NET ed è scritta in C# con tecnica code-behind.

Si compone di due sezioni: la prima ha il compito di guidare l'utente nella creazione del progetto, mentre la seconda offre una semplice interfaccia per la visualizzazione e la manipolazione dei dati geografici mediante le operazioni GIS implementate.

5.2.1. Project Manager

Permette di comporre un progetto sfruttando il servizio LCS per ottenere una lista ordinata di servizi OWS ed i relativi layer ed un servizio ausiliario per ottenere informazioni da sorgenti dati proprietarie.

La pagina iniziale mostra, sulla destra, i progetti creati dall'utente e sulla sinistra le operazioni effettuabili su di essi; selezionando un progetto, verranno mostrati tutti gli elementi principali che lo compongono, figura 5.5.

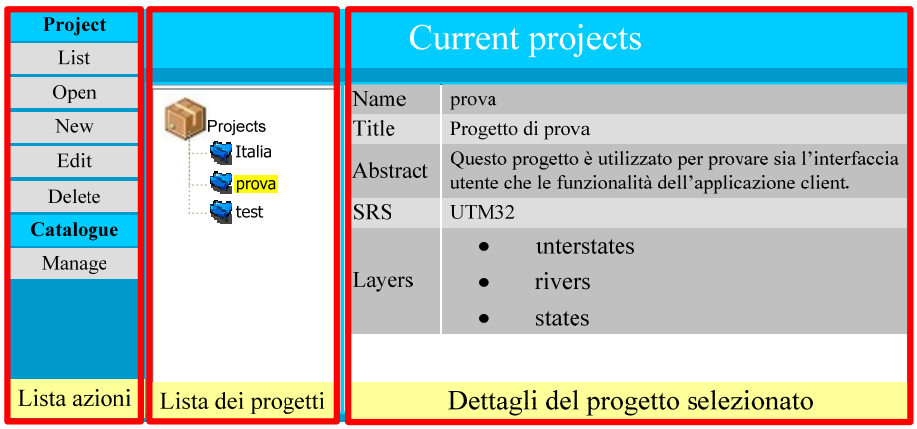


Figura 5.5

Schermata principale dell'applicazione Project Manager; nella lista progetti è stato selezionato il progetto "`prova`" ed a fianco sono mostrati gli elementi che lo compongono.

La lista delle azioni è divisa in due: le prime cinque voci si occupano della gestione del progetto, mentre l'ultima permette di accedere all'applicazione `Web Catalogue Manager` per amministrare il catalogo.

L'azione "`List`" consente in qualsiasi momento di accedere alla lista dei progetti, perdendo però le modifiche in atto sul progetto attivo.

Le azioni previste per operare sui progetti, sono:

- **`Open`**: selezionando un progetto ed attivando l'azione "`Open`", verrà aperta l'interfaccia dedicata alla manipolazione dei dati discussa nella prossima sezione;
- **`New`**: permette di creare un nuovo progetto attraverso quattro passi consecutivi (wizard). Il passo iniziale prevede l'inserimento del nome da associare al progetto (figura 5.6) mentre i successivi tre passi permettono di definire gli elementi costituenti il progetto.

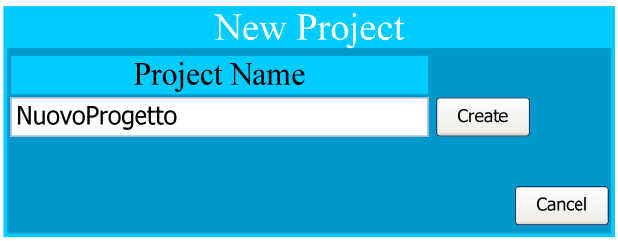


Figura 5.6
Interfaccia per la creazione nuovo progetto.

Passo 1) la pagina in figura 5.7 permette di inserire le seguenti informazioni:

- titolo del progetto: una breve descrizione testuale del progetto;
- sommario del progetto, in cui riportare la descrizione dettagliata del progetto (*abstract*);
- sistema di riferimento spaziale della mappa. La lista dei sistemi di riferimento disponibili è ottenuta mediante servizio GO-WS `SystemCoordTransform`.

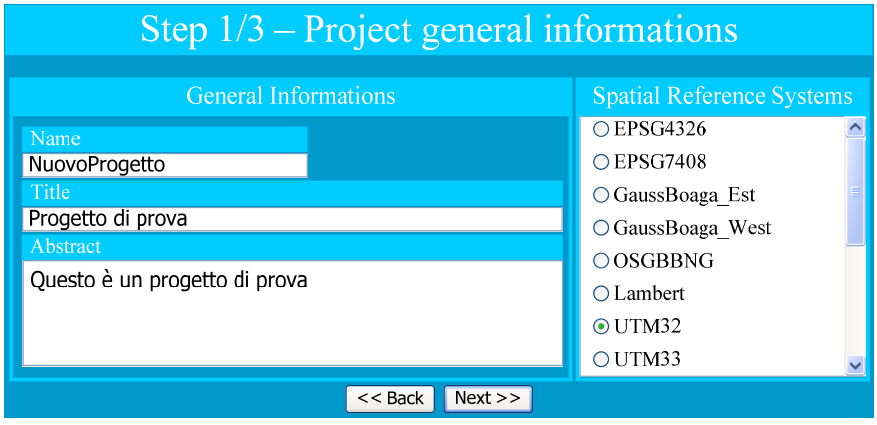


Figura 5.7
Interfaccia per l'inserimento delle informazioni descrittive sul progetto.

Passo 2) Il secondo passaggio (figura 5.8) consente di ricercare gli strati informativi in base alle informazioni contenute all'interno del catalogo MIR tramite il servizio LCS. La ricerca può essere eseguita esaminando i layer esposti da un servizio memorizzato, oppure in base

al valore di alcune meta informazioni quali: termini chiave, nome del layer o box di contenimento. La pagina offre una porzione di interfaccia per assegnare un alias allo strato al fine, ad esempio, di distinguerlo da eventuali omonimi. Lo stesso è possibile per aggiungere layer provenienti da sorgenti non OGC. Al momento di procedere al passo successivo, l'applicazione memorizza gli strati scelti e per ognuno di essi riporta le coordinate dello MBR nelle coordinate relative al sistema di riferimento di mappa tramite un'invocazione al GO-WS `SystemCoordTransform`.

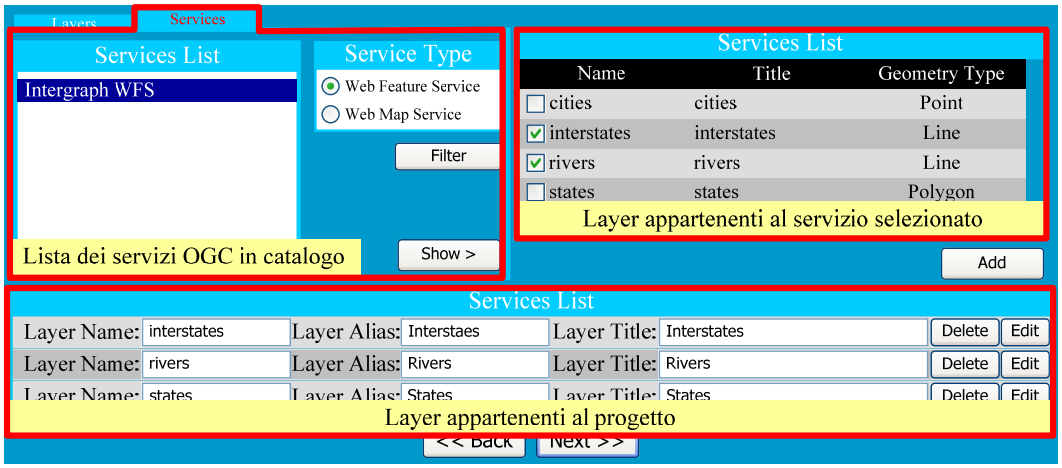


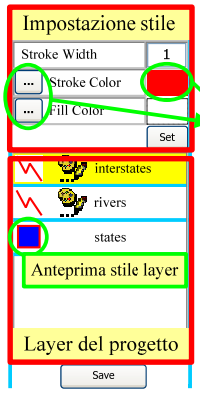
Figura 5.8
Interfaccia per l'inserimento dei layer all'interno del progetto

Passo 3) L'ultimo passo, da' la possibilità di impostare:

- lo stile di visualizzazione per gli strati informativi vettoriali. Per fornire un modo semplice per scegliere i colori da assegnare, viene impiegata la finestra di dialogo di sistema relativa alla palette dei colori (figura 5.9a);
- l'area di interesse sui cui operare che può essere definita manualmente, oppure in base ad uno strato informativo o scegliere quella con estensione maggiore calcolata automaticamente (figura 5.9b);
- il livello degli strati all'interno della gerarchia della mappa, (figura 5.9c).

Quando queste informazioni sono state definite, mediante i bottoni di salvataggio delle impostazioni, il progetto è pronto per essere aperto per la manipolazione dei dati geografici.

- **`Edit`**: permette, per il progetto selezionato, di accedere al primo dei tre passaggi sopra illustrati (figura 5.7);
- **`Delete`**: consente di eliminare il progetto selezionato.



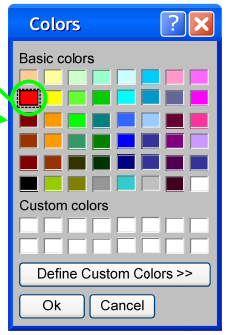


Figura 5.9a

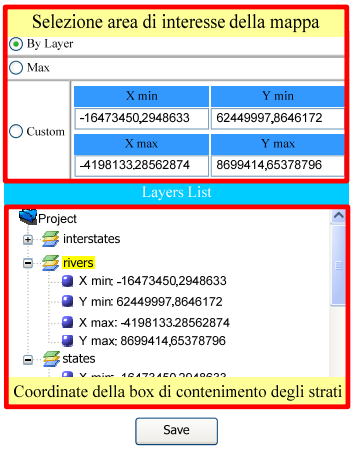


Figura 5.9b

Figura 5.9c

Da sinistra: la modifica dello stile di visualizzazione di un layer selezionato avviene mediante i due tasti [...] che aprono la finestra di dialogo per la selezione dei colori; il bottone "`Set`" è utilizzato per apportare le modifiche all'anteprima, mentre il tasto "`Save`" per riportare le modifiche nel progetto. L'impostazione dell'area di interesse è stata eseguita selezionando la bounding box del layer "`rivers`". La posizione del layer nella gerarchia è modificata selezionando il layer e spostandolo mediante i tasti freccia.

La creazione e modifica del progetto è assistita dalla libreria `UserProject`, scritta in C#, che consente di serializzare le informazioni inserite in un file XML dotato di uno schema definito in XSD.

5.2.2. GO-WS Client

La composizione del progetto costituisce solo un procedimento strutturato per raccogliere le meta informazioni legate agli strati informativi e le peculiarità della mappa, di seguito verrà presentata la parte dell'applicazione WebGIS relativa alla manipolazione dei dati geografici.

5.2.2.1. Descrizione dell'applicazione

L'applicazione WebGIS vera e propria è costituita da un insieme di web form ASP.NET, scritti in C# con tecnica code-behind, utilizzati lato server per:

- processare il file di progetto;
- interagire con i servizi GO-WS.

Lato client, l'applicazione è composta da tre aree funzionali: la legenda, il display grafico ed il panello per invocare le operazioni GIS sugli strati del progetto, figura 5.10.

L'area della legenda è composta dall'anteprima d'ogni tema assieme all'alias assegnatoli; per interagire sul client con ogni strato, la legenda offre un menù contestuale, (figura 5.11) con cui accedere alle funzionalità[48] di:

- mostra/nascondi tema;
- modifica livello del tema nella gerarchia della mappa.

Figura 5.10

Interfaccia utente dell'applicazione WebGIS. In alto a sinistra il pannello della legenda, al centro la scheda `Map` con il display grafico.

L'area centrale dell'interfaccia utente prevede due schede, `Map` e `Wizard`, per accedere rispettivamente al display grafico e all'interfaccia relativa alle operazioni GIS.

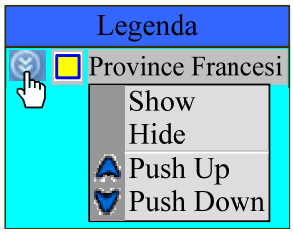


Figura 5.11

Interazione con il menù contestuale relativo ad un tema, tramite il pannello della legenda.

Il display grafico è pensato per visualizzare documenti codificati in Scalable Vector Graphic (SVG) in modo da usufruire delle operazioni di zoom e pan offerte dal browser web[49].

La dimensione della rappresentazione grafica della mappa è definita all'interno del progetto mediante i

[48] Dato lo scopo puramente dimostrativo di questa applicazione, è stato implementato solo un insieme minimo di funzionalità.

[49] La maggior parte dei web browser, come Microsoft Internet Explorer e Mozilla, supportano in modo nativo il formato SVG, nel caso di altri web browser è necessario il plug-ing di Adobe, fornito gratuitamente.

parametri di altezza e larghezza, espressi in pixel. L'applicazione permette di visualizzare mappe con risoluzioni superiori a quelle di schermo, mediante una semplice finestra provvista di barre di scorrimento orizzontali e verticali.

L'interfaccia utente relativa alle operazioni GIS è accessibile tramite la scheda `Wizard` che prevede tre categorie di operazioni:

- selezione per tema;
- geoprocessing;
- buffering.

5.2.2.2. Le Sorgenti Dati

Per introdurre le operazioni GIS ed i relativi esempi è opportuno descrivere, seppur brevemente, i dati geografici impiegati. In totale, i layer (di cui è riportato l'alias utilizzato all'interno dei progetti di test e fra parentesi il nome, se non coincidono) appartengono a cinque sorgenti dati fra loro eterogenee:

- *WMS Italia* [34]:servizio WMS realizzato appositamente per fornire layer relativi ad alcuni aspetti del territorio Italiano da utilizzare nei test; di questo servizio sono stati utilizzati i seguenti layer:
  - *Laghi Italiani* (*laghi*): descrive i maggiori laghi italiani;
  - *Province Italiane* (*province*): limiti amministrativi delle province italiane.

  Attualmente il servizio è in grado di offrire solo layer raster georeferenziati in EPSG:4326.
- *World Map* [35]: servizio WMS in grado di offrire numerosi strati relativi all'intero globo terrestre; di questi ne sono stati impiegati due:
  - *Topografia* (*topography*): topografia dei continenti emersi;
  - *Batimetria* (*bathymetry*): batimetria dei mari e degli oceani.

  Il servizio, conforme alle specifiche OGC, è in grado di offrire layer in formati raster (JPG, PNG, BMP e GIF) e vettoriali (Shockwave SWF) georeferenziati solo in EPSG:4326.
- *Intergraph WFS* [36]: servizio WFS in grado di offrire dataset geografici codificati in GML v2.1.2 georeferenziati in EPSG4326, riguardanti alcuni tematismi degli Stati Uniti d'America. Per i nostri scopi è stato utilizzato solamente:
  - *Hospitals*: modella gli ospedali delle contee di Loudoun, Fairfax, Falls Church City, Alexandria City, Arlington nello stato della Virgina ed il distretto di Columbia nello stato del Maryland; il layer è di tipo puntuale.

Sono state impiegate due sorgenti dati memorizzate in un archivio *MS Access* secondo la codifica proprietaria di GeoMedia:

- *USSample*: contiene gli strati relativi agli Stati Uniti d'America nel sistema di riferimento di riferimento Robinson. Sono stati impiegati i seguenti layer:
  - *States*: confini amministrativi degli stati USA, layer poligonale;
  - *Interstates*: autostrade, layer lineare;
  - *Rivers*: i fiumi più importanti, layer lineare;
  - *Cities*: le maggiori città degli stati, layer puntuale.
- *Italia*: definisce alcuni strati inerenti il territorio italiano come l'idrografia e le linee ferroviarie. Il sistema di coordinate è proiettato secondo UTM (Universal Traverse Mercator) zona 32. Gli strati informativi utilizzati sono:
  - *Regioni*: confini amministrativi delle regioni italiane definiti mediante poligoni;
  - *Fiumi Italiani* (*fiumi*): layer lineare rappresentante i maggiori fiumi italiani;
  - *Laghi Italiani* (*laghi*): si tratta di un layer poligonale con la stessa semantica di "*Laghi Italiani*" proveniente dalla sorgente dati OGC *WMS Italia*.

Le seguenti sorgenti dati utilizzano un formato di codifica proprietario, non di GeoMedia[50]:

- *France*: codificata secondo il formato *shape-file* di ESRI ArcView. Riporta un insieme di strati informativi inerenti lo stato francese, definiti in coordinate geografiche. È stato utilizzato solamente:
  - *Province Francesi* (*fra_prov*): limiti amministrativi delle province francesi rappresentati mediante poligoni.
- *Toscana*: utilizza la codifica DXF CAD e gli strati sono definiti in coordinate proiettate secondo Gauss-Boaga. Il layer utilizzato è:
  - *Città Toscane* (*sedicom*): layer puntuale che descrive le maggiori città della regione Toscana.

Ai fini di test, sono stati creati tre progetti:

- *Italia8*: è composto dai seguenti strati: Province Italiane, Laghi Italiani (dalla sorgente dati *WMS Italia*), Province Francesi, Città Toscane, Regioni Italiane, Topografia, Batimetria. Il layer "Province Italiane" si trova al livello più alto della gerarchia dei temi appartenenti alla mappa. La mappa definita dal progetto,

illustrata in figura5.10, è prodotta nel sistema di riferimento spaziale EPSG:4326 sia nel formato vettoriale SVG che PNG.

- *Italia6*: è composto dai seguenti strati: Città Toscane, Laghi Italiani (dalla sorgente dati *Italia*), Province Francesi, Fiumi Italiani, Province Italiane, Regioni Italiane. Il tema "Città Toscane" si trova al livello più alto della gerarchia dei temi della mappa. La mappa è prodotta nel sistema di riferimento UTM32 in SVG.
- *VirginiaHospitals*: sono stati integrati i seguenti strati degli Stati Uniti d'America: Hospitals, Cities, Interstates, Rivers, States, Topography, Bathymetry; con Hospital al livello più alto della gerarchia dei temi. Questo progetto illustra l'integrazione di dati provenienti da un servizio WFS, WMS e da una sorgente dati proprietaria. La mappa è prodotta nel sistema di riferimento EPSG:4326 in SVG.

5.2.2.3. Esecuzione delle Operazioni Geografiche

La scheda `Wizard` consente di accedere all'interfaccia utente relativa alle operazioni geografiche. Essa prevede una lista di tipi di operazioni: selezione per tema, geoprocessing e buffering. Selezionando un tipo di operazione, l'interfaccia mostra i controlli necessari per invocare le operazioni geografiche sui dati del progetto.

Per ogni tipologia di operazioni GIS verrà presentata una sola operazione sui dati geografici, definiti all'interno dei progetti sopra illustrati.

5.2.2.3.1. Italia8

Il progetto "Italia8", riassunto in figura 5.12, mostra la capacità dell'infrastruttura di accedere ad un insieme di dati geografici eterogenei e comporli in una mappa. Come mostrato in figura 5.10, i temi "Province Italiane" e "Laghi Italiani" (layer raster) seppur trovandosi ai livelli più alti della gerarchia, non coprono gli strati sottostanti in quanto il motore GIS è in grado di manipolare il canale alpha di trasparenza del formato PNG in cui i due temi sono inviati.

[50] Perché questi tipi di sorgenti dati possano essere utilizzati da GeoMedia WebMap, devono essere corredati di un file che ne descriva il sistema di riferimento.

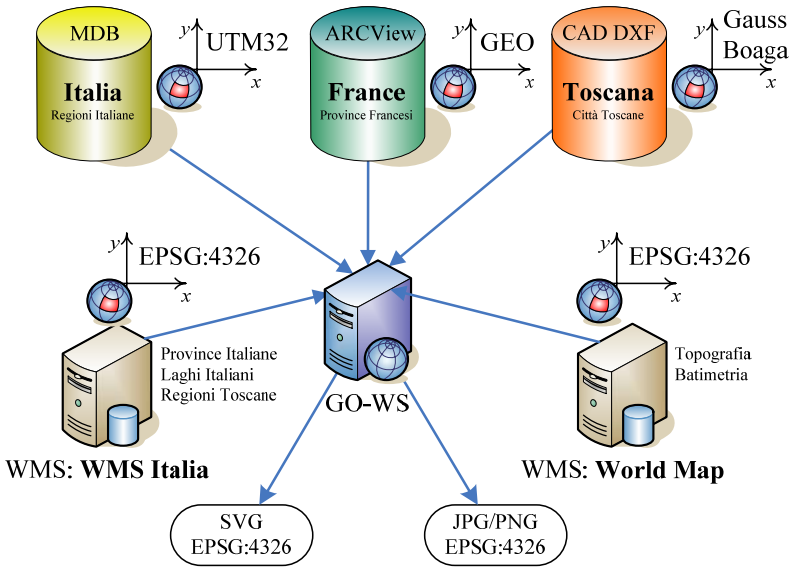


Figura 5.12

Configurazione dei dati geografici del progetto "Italia8" e del tipo di documento grafico utilizzato per creare la mappa.

L'operazione GIS realizzata consiste nella selezione per tema utilizzando come layer attivo lo strato informativo "Città Toscane" e "Province Francesi" come layer di selezione, mentre il criterio di selezione si basa sulla distanza fra le feature (figura 5.13).

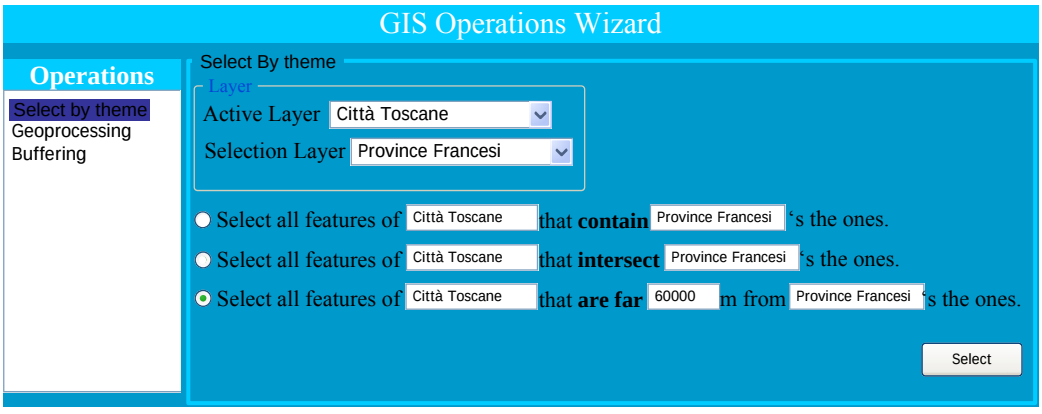


Figura 5.13

Interfaccia Wizard per le operazioni GIS di selezione per tema

Nel test effettuato sono selezionate tutte le feature di "Città Toscane" che si trovano entro 60Km da quelle di "Province Francesi" (figura 5.14). Il client riceve dal servizio i quattro identificatori relativi alle feature che soddisfano il criterio di selezione e rappresenta il risultato evidenziando in rosso, sul display grafico, le feature interessate. In questo caso l'uso del documento SVG risulta indispensabile in quanto l'oggetto DHTML che lo rappresenta permette di modificarne il contenuto direttamente sul client.

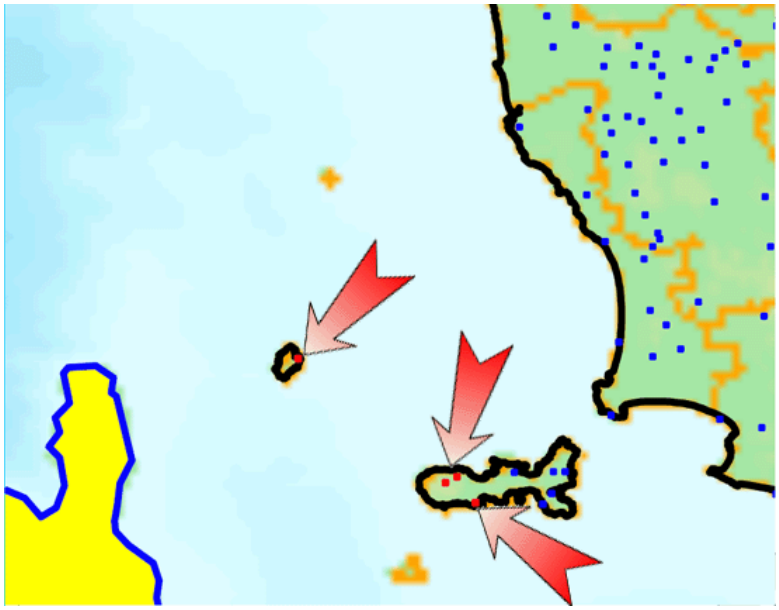


Figura 5.14

Risultato dell'operazione di selezione per tema in base alla distanza. Sono state selezionate quattro città, dall'alto: Isola Capraia, Marciana, Marciana Marina e Marina di Campo. La figura fa riferimento all'ingrandimento eseguito sulla mappa mostrata in figura 5.10.

5.2.2.3.2. Italia6

L'operazione di geoprocessing è stata eseguita sugli strati informativi definiti all'interno del progetto "Italia6" (figura 5.15) che prevede la restituzione della mappa nel sistema di riferimento proiettato UTM32.

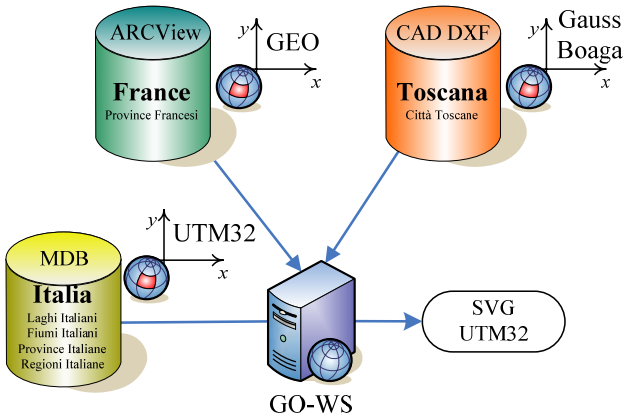


Figura 5.15

Configurazione dei dati geografici del progetto "Italia6" e del tipo di documento grafico utilizzato per restituire la mappa.

Selezionando la voce "`Geoprocessing`" dalla lista delle operazioni della scheda "`Wizard`" è possibile accedere all'interfaccia relativa alle operazioni di geoprocessing. Da questo insieme è stata scelta l'operazione di intersezione, utilizzando "Laghi Italiani" come layer di input e "Province Italiane" come layer di overlay (figura 5.16).

Figura 5.16

Interfaccia Wizard per le operazioni di geoprocessing.

Il risultato dell'operazione è composto da nuove feature che individuano le porzioni di lago contenute all'interno di una provincia (figura 5.17). Questo perché le feature che rappresentano i laghi italiani si sovrappongono a quelle delle province italiane.

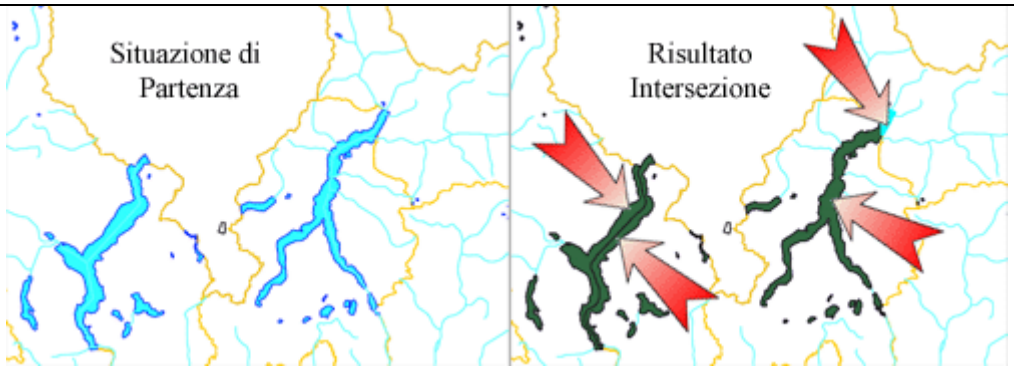


Figura 5.17

Risultato dell'operazione di intersezione fra "Laghi Italiani" e "Province Italiane". Come evidenziato nella figura di destra, il lago di Como ed il lago Maggiore sono stati divisi in quattro feature distinte.

L'operazione di buffering eseguita, utilizza il layer "Città Toscane" per creare un'area di rispetto di 5 Km di raggio intorno a ciascuna città della regione Toscana, (figura 5.18).

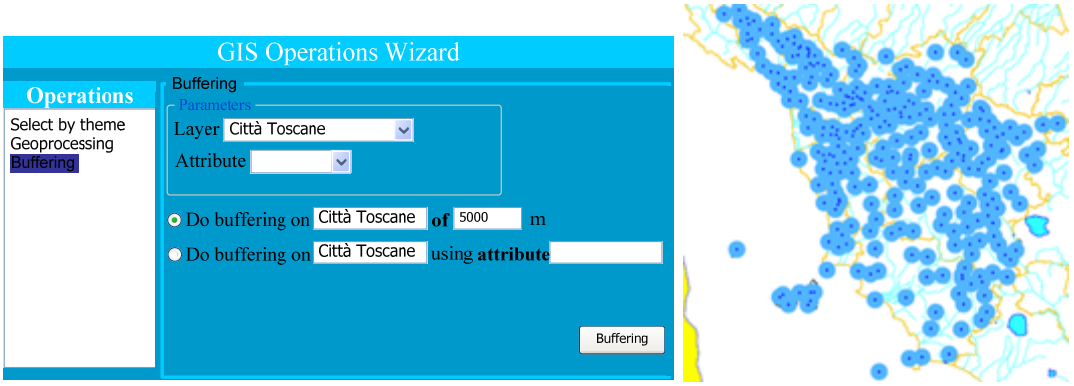


Figura 5.18

Da sinistra: interfaccia Wizard per le operazioni di buffering; risultato dell'operazione di buffering richiesta all'infrastruttura GO-WS.

5.2.2.3.3. VirginiaHospitals

Questo progetto (figura 5.19) è impiegato per dimostrare la capacità dell'infrastruttura di integrare layer raster provenienti da un servizio OGC WMS e layer vettoriali provenienti da una sorgente dati proprietaria ed una OGC WFS.

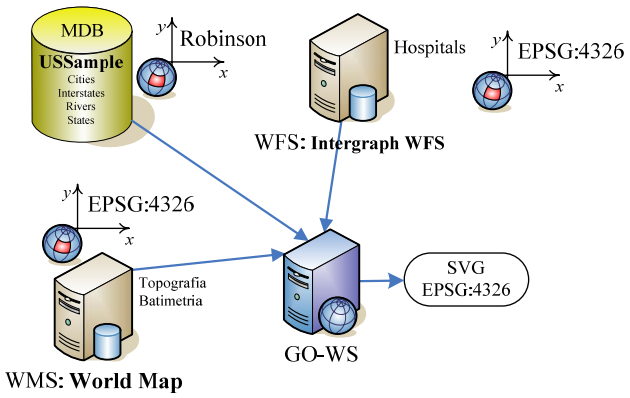


Figura 5.19
Configurazione dei dati geografici del progetto "VirginiaHospitals" e del tipo di documento grafico utilizzato per restituire la mappa.

Sulla mappa prodotta (figura 5.20) sono state eseguite due operazioni:

- *selezione per tema*: sono state selezionate tutte le feature del layer "Cities" che si trovano entro 3,5 Km da quelle di "Hospitals" (figura 5.21);
- *buffering*: creazione di un'area di interesse di 5 Km attorno alle feature di "Hospitals" (figura 5.22).

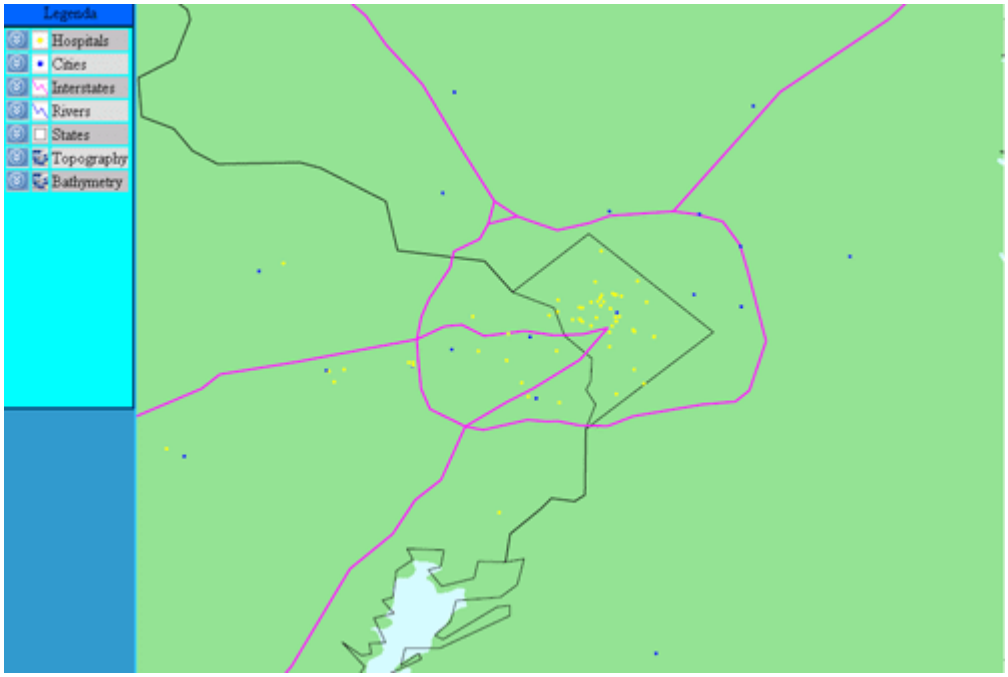


Figura 5.20
Mappa prodotta integrando i layer definiti all'interno del progetto "VirginiaHospitals". Le fature di "Hospital" sono rappresentate in giallo, quelle di "Cities" in blu e quelle di "Interstates" in fucsia.

Figura 5. 21

Selezione di tutte le feature appartenenti a "Cities" che si trovano entro 3.5 Km di distanza da quelle di "Hospitals".

Figura 5. 22

Operazione di buffering eseguita sul layer "Hospitals", dove l'area di rispetto creata è pari a 5 Km di raggio. L'aspetto oblungo dei buffer è dovuto al sistema di riferimento spaziale. Da notare che le aree di rispetto possiedono un loro identificatore e possono essere selezionate graficamente (il buffer celeste).

5.3. Riferimenti Alle Applicazioni

Le applicazioni sopra illustrate sono disponibili presso il web server del Centro Tecnologico Sistemi Informativi, appartenente all'Istituto di Scienza e Tecnologia dell'Informazione "A. Faedo": http://webgisserver.isti.cnr.it. Nella tabella di seguito, sono riportati gli indirizzi delle applicazioni.

Applicazione	URL
Web Catalogue Manager	http://webgisserver.isti.cnr.it/webcatalogmanager/
GIS Application Service Provider	http://webgisserver.isti.cnr.it/GisAsp/
Project Manager	http://webgisserver.isti.cnr.it/projectmanagerr/

I servizi sviluppati (GO-WS) si trovano agli indirizzi riportati nella tabella sottostante.

Servizio	URL
Buffering Service	http://webgisserver.isti.cnr.it/GisOperationsDeliverer/BufferingService.asmx/
Geoprocessing Service	http://webgisserver.isti.cnr.it/GisOperationsDeliverer/GeoProcessingService.asmx
GeoSelection Service	http://webgisserver.isti.cnr.it/GisOperationsDeliverer/GeoSelectionService.asmx
MapRender Service	http://webgisserver.isti.cnr.it/GisOperationsDeliverer/MapRenderService.asmx
Coordinate Translation Service	http://webgisserver.isti.cnr.it/GisOperationsDeliverer/SystemCoordTransform.asmx
Local Catalogue Service	http://webgisserver.isti.cnr.it/LocalCatalogueDriver/CatalogDriver.asmx

All'indirizzo http://webgisserver.isti.cnr.it/codicetesi/ è riportata l'intera documentazione relativa alle librerie sviluppate.

Capitolo 6

Conclusioni

L'infrastruttura realizzata permette di accedere ed integrare dati geografici eterogenei, provenienti da sorgenti remote e locali.

La possibilità di integrare dati eterogenei implica poter considerare allo stesso modo dati caratterizzati da sistema di riferimento spaziale, formato di codifica e tipologia fra loro diversi.

Sui dati geografici integrati, l'infrastruttura è in grado di offrire un insieme di operazioni geografiche orientate all'analisi spaziale. Tali operazioni, accessibili mediante la rete Internet, sono organizzate modularmente in modo tale da permetterne l'aggiunta di nuove, sfruttando la scalabilità dell'infrastruttura.

L'elemento portante sui cui poggia l'infrastruttura, è costituito dal motore GIS di GeoMedia WebMap che mette a disposizione le seguenti caratteristiche:

- **accessibilità**, per accedere ai dati geografici posti in una locazione di memoria locale o remota;
- **integrabilità**, cioè la capacità di trattare i dati geografici indipendentemente dalle loro caratteristiche, avendo inoltre la possibilità di uniformarli secondo uno specifico sistema di riferimento;
- **interoperabilità**, consiste in un insieme di meccanismi e funzionalità per realizzare ed eseguire operazioni geografiche sui dati integrati.

Per l'accesso ai dati remoti non proprietari sono state tenute in considerazione le raccomandazioni date da Intesa GIS riguardo alle specifiche dei servizi per la diffusione di dati geografici via Internet definite dal consorzio OpenGIS: Web Feature Service (WFS), Web Map Service (WMS) ed il linguaggio Geographic Markup Language (GML).

Per ridurre la complessità d'uso del motore GIS, è stata implementata una libreria di classi (GWM-Engine Interface) che costituisce un'interfaccia in grado di fornire un insieme di operazioni geografiche facilmente utilizzabili. Tale interfaccia è composta da una sezione principale, che si occupa di trattare i meccanismi legati alla gestione del motore GIS, e da una subalterna in cui sono implementate le operazioni geografiche.

Per la realizzazione della libreria GWM-Engine Interface sono state combinate le classi offerte da GeoMedia WebMap con quelle dal framework .NET; il risultato è una libreria

.NET, vincolata all'architettura di GWM, utilizzabile in qualsiasi contesto in cui sono richieste delle operazioni GIS.

Sullo strato realizzato dall'interfaccia, sono stati implementati quattro servizi remoti (GIS Operations – Web Service, GO-WS) per offrire operazioni geografiche su insiemi di dati eterogenei (locali e remoti). Tali operazioni, sono rese accessibili attraverso la rete Internet impiegando la tecnologia dei web service.

Le caratteristiche di questa tecnologia, hanno permesso di risolvere le problematiche legate all'interoperabilità ed all'integrazione funzionale fra piattaforme diverse. Ciò si traduce nella possibilità di utilizzare le operazioni GIS implementate indipendentemente da: sistema operativo, linguaggio, ambiente di sviluppo e tipologia di dispositivo in cui l'applicazione si trova.

Punti Di Forza

I punti di forza che caratterizzano l'infrastruttura sono:

- **accessibilità** a dati geografici, distribuiti e locali;
- **integrabilità** di dati geografici eterogenei;
- **integrabilità funzionale**: poter utilizzare le operazioni geografiche sviluppate in vari ambienti di sviluppo e piattaforme;
- **interoperabilità**: poter eseguire operazioni geografiche su dati geografici eterogenei;
- **interoperabilità funzionale**: combinare le operazioni geografiche sviluppate con i costrutti ed i meccanismi di vari ambienti di sviluppo;
- supporto degli standard **OpenGIS**: WMS, WFS e GML;
- **modularità** dell'architettura e strutturazione a strati.

Punti Deboli

Non è stato possibile valutare a pieno le prestazioni delle operazioni offerte dell'infrastruttura, data la scarsa quantità di feature geografiche attualmente ottenibili mediante servizi WFS.

L'infrastruttura inoltre, si basa su di un prodotto commerciale pensato esclusivamente per piattaforme Window Server e non esistono applicazioni simili per sistemi OpenSource.

Il supporto agli standard OpenGIS, offerto da GWM mediante espansioni esterne, non consente l'uso delle ultime versioni delle specifiche[51] e quindi accedere a funzionalità avanzate come, ad esempio, la rappresentazione di dati geografici tridimensionali via WFS.

Questo non accade invece per le nuove versioni di GWM (v5.2 e v6.0), che supportano nativamente l'accesso e la pubblicazione di dati geografici mediante le specifiche OWS.

Considerazioni

Avendo affrontato gli aspetti e le problematiche legate alla diffusione di servizi su dati geografici distribuiti, l'infrastruttura costituisce uno strumento informatico per raggiungere gli obiettivi dei progetti Comunitari legati alla realizzazione di infrastrutture SDI secondo le direttive di INSPIRE.

GeoMedia WebMap si è rivelato uno strumento potente e flessibile in quanto ha permesso la gestione dei dati geografici e la creazione di operazioni GIS in modo semplice e rapido. Inoltre, le caratteristiche di web mapping hanno permesso di poter distribuire documenti grafici codificati nel formato SVG. Tale formato consente di inserire, all'interno del documento grafico, script eseguibili lato client. Inoltre, sempre lato client, il documento può essere modificato mediante DOM e trasformazioni definite tramite XSL, dato che si tratta di una grammatica di XML.

Il supporto ai servizi WFS e WMS in GeoMedia WebMap, unito alla possibilità di importare ed esportare dataset geografici codificati in GML, è reso possibile mediante l'uso di quattro kit di espansione che vanno ad aggiungere nuovi driver per l'accesso a questi tipi di sorgenti dati. Nonostante i kit siano rilasciati senza un supporto ufficiale, si sono rivelati versatili ed affidabili (tranne in alcuni casi), consentendo di usufruire di dati geografici remoti e multi disciplinari, esposti da vari enti internazionali.

Dall'esame e dall'uso dei due servizi e del linguaggio definiti dal consorzio OpenGIS, è emerso che le rispettive specifiche sono soggette ad un continuo lavoro di revisione (mirato a consolidare, aggiornare ed espandere le specifiche) che porta alla necessità di disporre di driver capaci di supportare il maggior numero di versioni possibile (cosa che fanno di kit per GWM). Questo, perché non è detto le istanze dei servizi implementino più versioni della stessa specifica.

Una nota negativa sulla codifica dei dati dei servizi WFS, riguarda il fatto che si basa su XML. In caso di trasferimenti di grandi quantità di dati, si possono avere lunghi tempi di

[51] L'ultima versione di GML è la 3.1.1, per WMS è la 1.3 e per WFS è la 1.0.0, mentre sono supportate uguali e precedenti a WMS v1.1.1, WFS v1.0.0 ed a GML v2.1.2.

attesa. Alcuni fornitori di dati, prevedono l'uso dei servizi WFS per offrire un campionario di dati geografici. L'utilizzatore dei dati li valuterà per poi scaricarli completamente in un secondo momento (in vari formati, sia proprietari sia codificati in GML), così che da poterli utilizzare localmente.

L'impiego del framework .NET come ambiente di sviluppo ed esecuzione si è rivelato insostituibile perché, utilizzando le classi ed i meccanismi offerti (come code-behind, serializzazione, classi proxy), ha semplificato notevolmente l'implementazione sia delle librerie sia dei web service GO-WS.

L'uso del framework .NET come ambiente di sviluppo ed esecuzione, costituisce un approccio innovato nella realizzazione di applicazioni basate sulle classi di GWM. Questa scelta ha portato ad inevitabili problemi legati alla mancanza di supporto da parte della documentazione destinata allo sviluppatore. Tipicamente, GWM, è utilizzato in ambienti debolmente tipizzati (essenzialmente Visual Basic e ASP) dove non è necessario conosce il tipo di un oggetto per poterlo utilizzare, mentre il framework .NET è fortemente tipizzato. Se all'assenza di una dettagliata documentazione del modello delle classi di GWM, si aggiunge che (ad esempio) un oggetto `Query` è implementato in più librerie (e quindi definito in più namespace) con caratteristiche dissimili, si possono immaginare le difficoltà incontrate.

L'uso dei servizi sviluppati, ha permesso di realizzare in breve tempo e con poco sforzo, un'applicazione WebGIS (GISASP) in grado di fornire operazioni geografiche orientate all'analisi spaziale su dati eterogenei e distribuiti. Un responsabile della semplicità di realizzazione è nuovamente il framework .NET e le tecnologie ad esso legate, in quanto ha permesso di considerare i web service realizzati come normali classi e di realizzare velocemente pagine web dinamiche.

Futuri Sviluppi

L'infrastruttura si presta per future espansioni, ad esempio:

- definire le operazioni geografiche su insiemi di feature;
- nuove operazioni geografiche;
- uso della specifica WMS SLD.

Un'interessate espansione prevede lo sviluppo di servizi GIS legati ai dispositivi mobili quali Personal Digital Assistant (PDA) e telefoni cellulari. Ad esempio, perché un cellulare possa visualizzare mappe in formato SVG, è necessario che il documento sia convertito nel

formato SVG Tiny; sarà sviluppata un'estensione apposita per creare documenti vettoriali codificati in SVG Tiny.

Appendice A

OWS Capabilities

A.1. Frammenti di Capabilities di un Web Map Services

Questo frammento si riferisce al documento delle capacità, reperibile all'indirizzo http://www.digitalearth.gov/wmt/xml/capabilities_1_1_1.xml

```
<Layer queryable = "0" opaque = "0" noSubsets = "0">
   <Title>Acme Corp. Map Server</Title>
   <SRS>EPSG:4326</SRS>

   <Layer queryable = "0" opaque = "0" noSubsets = "0">
      <Name>ROADS_RIVERS</Name>
      <Title>Roads and Rivers</Title>
      <SRS>EPSG:26986</SRS>
      <LatLonBoundingBox minx = "-71.63" miny = "41.75"
                         maxx = "-70.78" maxy = "42.90" />
      <BoundingBox SRS = "EPSG:4326" minx = "-71.63" miny = "41.75"
                                     maxx = "-70.78" maxy = "42.90"  />
      <BoundingBox SRS = "EPSG:26986" minx = "189000" miny = "834000"
                                      maxx = "285000" maxy = "962000"/>
      <FeatureListURL>
         <Format>application/vnd.ogc.se_xml"</Format>
         <OnlineResource xmlns:xlink = "http://www.w3.org/1999/xlink"
                         xlink:type = "simple"
                        xlink:href = "http://.../roads_rivers.gml"/>
      </FeatureListURL>
      <Style>
         <Name>USGS</Name>
         <Title>USGS Topo Map Style</Title>
         <LegendURL width = "72" height = "72">
            <Format>image/gif</Format>
            <OnlineResource xmlns:xlink = "http://www.w3.org/1999/xlink"
                            xlink:type = "simple"
                            xlink:href = "http://.../legends/usgs.gif"/>
          </LegendURL>
         <StyleSheetURL>
            <Format>text/xsl</Format>
            <OnlineResource xmlns:xlink = "http://www.w3.org/1999/xlink"
                            xlink:type = "simple"
                            xlink:href = "http://... /usgs.xsl"/>
          </StyleSheetURL>
      </Style>
      <ScaleHint min = "4000" max = "35000" />
      <Layer queryable = "1" opaque = "0" noSubsets = "0">
        <Name>ROADS_1M</Name>
        <Title>Roads at 1:1M scale</Title>
        <Abstract>Roads at a scale of 1 to 1 million.</Abstract>
        <KeywordList>
           <Keyword>road</Keyword>
```

```
            <Keyword>transportation</Keyword>
            <Keyword>atlas</Keyword>
          </KeywordList>
          <MetadataURL type = “FGDC">
            <Format>text/plain</Format>
            <OnlineResource xmlns:xlink = “http://www.w3.org/1999/xlink”
                            xlink:type = “simple”
                            xlink:href = “http://.../metadata/roads.txt"/>
          </MetadataURL>
          <MetadataURL type = “FGDC">
            <Format>text/xml</Format>
            <OnlineResource xmlns:xlink = “http://www.w3.org/1999/xlink”
                            xlink:type = “simple”
                            xlink:href = “http://.../metadata/roads.xml"/>
          </MetadataURL>
          <FeatureListURL>
            <Format>application/vnd.ogc.se_xml"</Format>
            <OnlineResource xmlns:xlink="http://www.w3.org/1999/xlink"
                            xlink:type = "simple"
                            xlink:href = "http://.../data/roads_rivers.gml"/>
          </FeatureListURL>
          <Style>
            <Name>ATLAS</Name>
            <Title>Road atlas style</Title>
            <LegendURL width="72" height="72">
            <Format>image/gif</Format>
              <OnlineResource xmlns:xlink="http://www.w3.org/1999/xlink"
                              xlink:type = "simple"
                              xlink:href="http://...legends/atlas.gif"/>
            </LegendURL>
          </Style>
        </Layer>
        <Layer queryable="1" opaque="0" noSubsets="0">
          <Name>RIVERS_1M</Name>
          <Title>Rivers at 1:1M scale</Title>
          <KeywordList>
            <Keyword>river</Keyword>
            <Keyword>canal</Keyword>
          </KeywordList>
        </Layer>
      </Layer>
    </Layer>
```

A.2. Frammenti di Capabilities di un Web Feature Service

Il seguente frammento mostra la struttura della sezione FeatureTypeList: l’elemento `Operations` in testa alla lista indica che tutte le *feature type* possono essere interrogate, mentre solo la *feature* `myns:BUILTUPA_1M` può essere oggetto di transazioni.

```
<FeatureTypeList>
      <Operations>
            <Query/>
      </Operations>
      <FeatureType>
            <Name>myns:BUILTUPA_1M</Name>
            <SRS>EPSG:4326</SRS>
            <Operations>
                  <Insert/>
                  <Update/>
                  <Delete/>
            </Operations>
            <LatLongBoundingBox minx = "-179.1296081543"
                                miny = "-53.167423248291"
                                maxx = "178.44325256348"
                                maxy = "70.992721557617" />
      </FeatureType>
```

Appendice **B**

GenericLayerIdentifier

B.1. Modelli UML delle Classi

Il namespace `GenericLayeridentifier`, contenuto all'interno di una DLL scritta in C#, è definita la classe `LayerIdentifier` per identificare un layer in base ai seguenti quattro parametri:

1. `dataSource`: posizione della sorgente dati, URL o percorso di filesystem;
2. `dataSourceType`: formato sorgente dati;
3. `layerName`: nome del dataset relativo al layer ;
4. `additionalInfo`: informazioni generali sul dataset.

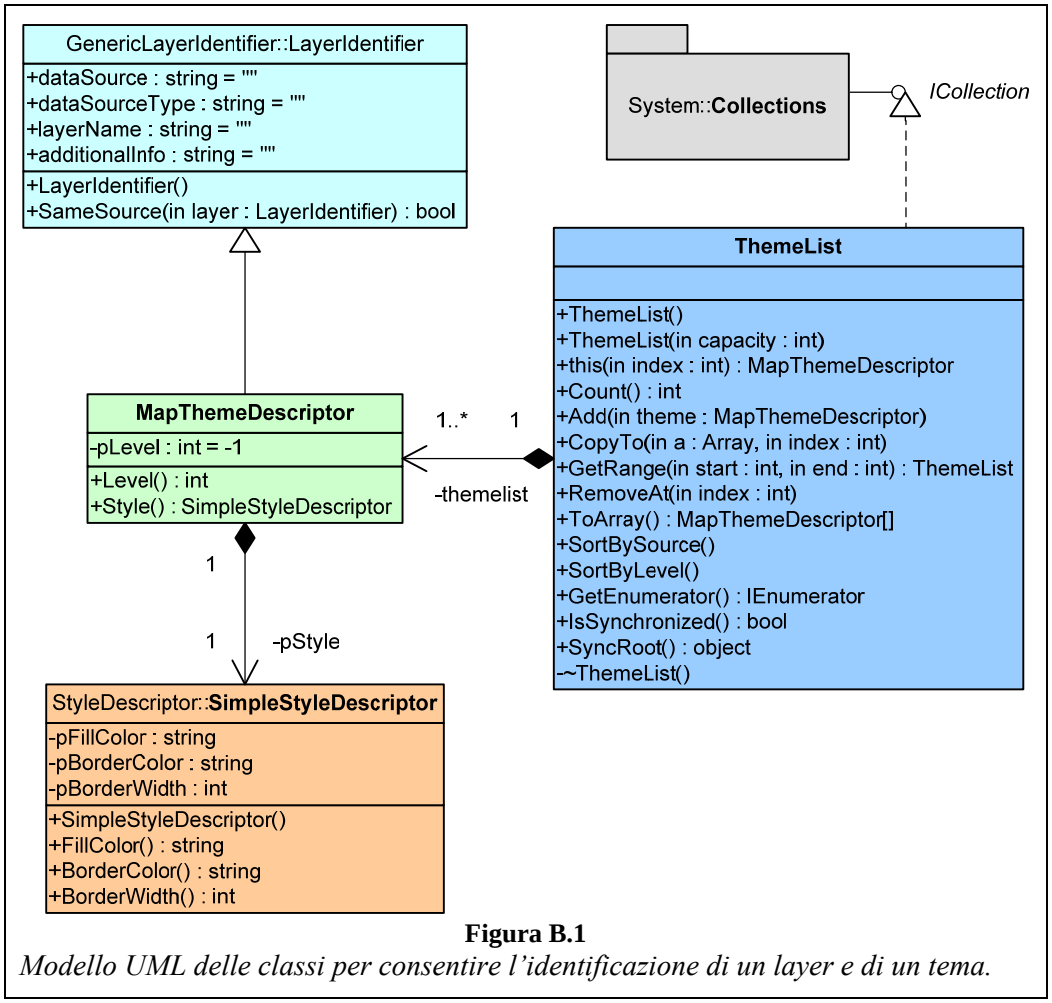


Figura B.1

Modello UML delle classi per consentire l'identificazione di un layer e di un tema.

Per quanto riguarda la classe `MapThemeDescriptor` (in figura B.1) si hanno le seguenti proprietà:

- `Style`: descrive lo stile di visualizzazione tramite la classe `SimpleStyleDescriptor`;

- `Level`: indica il livello del tema all'interno della gerarchia della mappa;

La classe `ThemeList` modella l'insieme dei temi di una mappa; implementa l'interfaccia `ICollection` del framework .NET.

Oltre ai metodi standard definiti dall'interfaccia, ne sono stati aggiunti tre specifici:

- `GetRange`: restituisce una porzione di lista definita tramite due indici;
- `SortBySource`: effettua un raggruppamento di temi ordinando la lista in base alla sorgente dati, secondo un criterio di ordinamento alfanumerico (figura B.2);
- `SortByLevel`: ordinamento in ordine crescente basato sul livello dei temi.

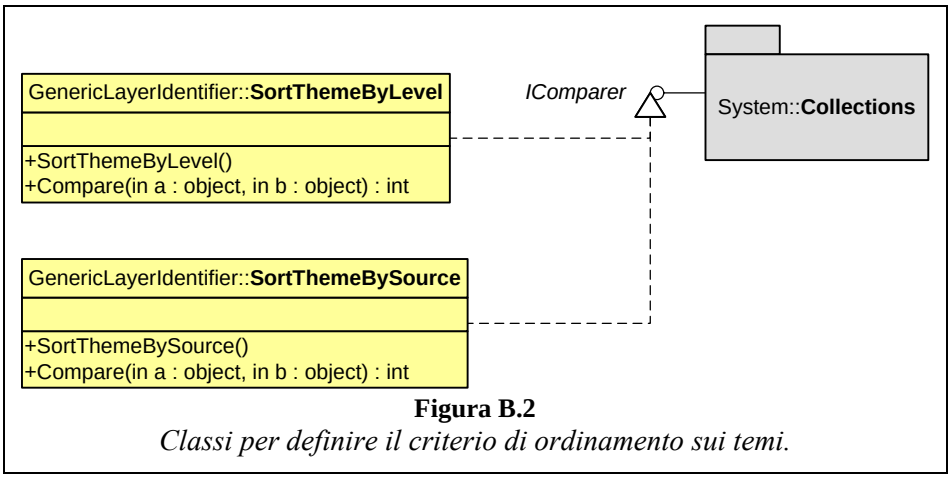


Figura B.2
Classi per definire il criterio di ordinamento sui temi.

I metodi di ordinamento operano su oggetti di tipo `MapThemeDescriptor` e si basano su due criteri di confronto definiti dalle classi:

- `SortThemeByLevel`: criterio di confronto per i livelli;
- `SortThemeBySource`: criterio di confronto per il percorso delle sorgenti dati.

Come mostrato in figura B.2, entrambe le classi di confronto implementano l'interfaccia `IComparer` del *framework* .NET che espone i metodi per effettuare il confronto fra due oggetti. Dati due oggetti *a* e *b* ed una relazione di confronto $\Re$, il metodo `Compare()` deve restituire i seguenti valori:

- -1: se $a \,\Re_{\leq}\, b$;
- 0: se $a \,\Re_{\equiv}\, b$;
- 1: se $a \,\Re_{\geq}\, b$.

All'interno del namespace vi è definita un'ulteriore classe utilizzata per modellare un *Minimum Bounding Box* (figura B.3). Si tratta del minimo rettangolo contenente una *feature*. È modellato mediante la classe `BoundingBox` (figura B.4) che espone proprietà per restituirne l'area ed il perimetro.

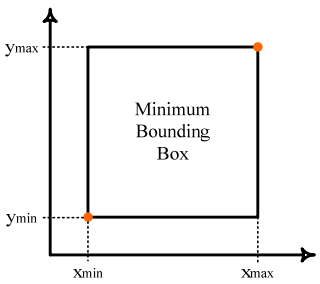


Figura B.3
Esempio di Minimum Bounding Box

La classe espone inoltre tre proprietà per restituire le coordinate dei vertici della box di contenimento come *array* di stringe, *double* e *object* (il supertipo da cui ereditano tutti i tipi all'interno del *framework .NET*).

BoundingBox
+xmin : double = 0 +ymin : double = 0 +xmax : double = 0 +ymax : double = 0
+Area() : double +Perimeter() : double +isZero() : bool +strSquare() : string[] +dblSquare() : double[] +objSquare() : object[] +BoundingBox() +BoundingBox(in xmin : double, in ymin : double, in xmax : double, in ymax : double) +operator >(in a : BoundingBox, in b : BoundingBox) : bool +operator <(in a : BoundingBox, in b : BoundingBox) : bool +operator ==(in a : BoundingBox, in b : BoundingBox) : bool +operator !=(in a : BoundingBox, in b : BoundingBox) : bool +Equals(in obj : object) : bool +GetHashCode() : int +ToString() : string

Figura B.4
Classe che implementa il Minimum Bounding Box.

Queste proprietà servono per mantenere compatibilità con i tipi COM utilizzati dalle classi di GWM.

B.2. Codice Namespace GenericLayerIdentifier

```
using System;
using System.IO;
using System.Collections;
//****** Namespace interni *******
using StyleDescriptor;

///<Author>Giulio Massei</Author>
///<LastUpdate>19/10/2005</LastUpdate>

namespace GenericLayerIdentifier
{
    /// <summary>
    /// Utilizzata per identificare un layer. Identifica la locazione e tipo del datasource ed il nome del layer.
    /// </summary>
    /// <remarks>La classe è serializzabile. </remarks>
    [Serializable()]
    public class LayerIdentifier
    {
        #region Attributi pubblici
        /// <summary>URL del datasource. </summary>
        public string dataSource = "";
        /// <summary>Tipo di datasource: WFS;WMS;GML;SHP;... </summary>
        public string dataSourceType= "";
        /// <summary>Nome del layer all'interno del datasource.</summary>
        public string layerName = "";
        /// <summary>Informazioni addizionali relative alla sorgente dati. </summary>
        public string additionalInfo = "";
        #endregion

        #region Costruttori
        /// <summary>Costruttore predefinito. </summary>

        public LayerIdentifier(){}
        /// <summary> Costruttore paramentrico</summary>
        /// <param name="strDataSource">Sorgente dati.</param>
        /// <param name="strDataSourceType">Tipo di sorgente dati.</param>
        /// <param name="strLayerName">Nome del layer.</param>
        /// <param name="strAdditionalInfo">
I       /// nformazioni addizionali relative alla sorgente dati.</param>
        public LayerIdentifier(string strDataSource, string strDataSourceType,
                                    string strLayerName, string strAdditionalInfo)
        {
            this.dataSource = strDataSource;this.dataSourceType= strDataSourceType;
            this.layerName = strLayerName;  this.additionalInfo = strAdditionalInfo;
        }
        #endregion

        #region Metodi pubblici
        /// <summary> Indica se il layer corrente ha la stessa origine dati di un altro layer. </summary>
        /// <param name="layer">Layer con cui effettuare il confronto.</param>
        /// <returns>"true" se i due layer hanno la stessa origine dati.</returns>
        public bool SameSource(LayerIdentifier layer)
        {
            if(null != layer)
                return ((this.dataSource == layer.dataSource) && (this.dataSourceType == layer.dataSourceType) &&
                        (this.additionalInfo == layer.additionalInfo));
            else
                return false;
        }
        #endregion
    }

    /// <summary>
    /// Descrive un tema. Un tema (qui) è inteso come un layer destinato alla rappresentazione cartografica digitale.
    /// </summary>
    /// <remarks> Eredita da °LayerIdentifier°. La classe è serializzabile. </remarks>
    [Serializable()]
    public class MapThemeDescriptor : LayerIdentifier
    {
        #region Attributi privati
        int pLevel = -1;
        SimpleStyleDescriptor pStyle = null;
        #endregion

        #region Proprietà pubbliche
        /// <summary>
        /// Indica il livello del tema nella mappa. Più il valore del
        /// livello è piccolo, più in alto verrà posto il tema nella gerarchia dei temi.
        /// </summary>
        public int Level
        {
            get{return this.pLevel;}
            set{this.pLevel = value;}
        }
        public SimpleStyleDescriptor Style
        {
            set{if(null!=value) this.pStyle = value;}
            get{return this.pStyle;}
        }
        #endregion
    }

    /// <summary>
    /// Deffinisce un "BoundingBox" o "Box di contenimento". Si tratta di un quadrilatero definito tramite le coordinate
    /// del punto in basso a sinistra e del punto in allto a destra.
    /// </summary>
    /// <remarks>La classe è serializzabile. </remarks>
    [Serializable()]
    public class BoundingBox
```

```
{
    #region Campi pubblici
        /// <summary> Coordinata x del punto in basso a sinistra. </summary>
        public double xmin = 0;
        /// <summary> Coordinata y del punto in basso a sinistra. </summary>
        public double ymin = 0;
        /// <summary> Coordinata x del punto in alto a destra. </summary>
        public double xmax = 0;
        /// <summary> Coordinata y del punto in alto a destra. </summary>
        public double ymax = 0;
    #endregion

    #region Proprietà pubbliche

    /// <summary> Restituisce l'area della box di contenimento. </summary>
    public double Area
        {get{double dx = this.xmax - this.xmin;double dy = this.ymax - this.ymin; return dx*dy;}}

    /// <summary> Restituisce il perimetro della box di contenimento. </summary
    public double Perimeter
        {get{double dx = this.xmax - this.xmin; double dy = this.ymax - this.ymin;return (2*dx) + (2*dy);}}

    /// <summary> Indica se la box è nulla (indica un punto e non un area). </summary>
    public bool isZero
    {
        get
        {
            if(null==this) return true;
            else return (xmin==0 && ymin==0 && xmax==0 && ymax==0);
        }
    }

    /// <summary>
    /// Restituisce un array contenente le otto coordinate (tipizzate °string°) che definiscono i quattro punti della
    /// box. (xmin,ymin) (xmax,ymin) (xmax,ymax) (xmin,ymax).
    /// </summary
    public string[] strSquare
    {
        get
        {
            string[] square = null;
            if(null!=this)
            {
                square = new string[8];
                square.Initialize();
                square.SetValue(this.xmin,0);square.SetValue(this.ymin,1);
                square.SetValue(this.xmax,2);square.SetValue(this.ymin,3);
                square.SetValue(this.xmax,4);square.SetValue(this.ymax,5);
                square.SetValue(this.xmin,6);square.SetValue(this.ymax,7);
                return square;
            }
            else
                return null;
        }
    }

    /// <summary>
    /// Restituisce un array contenente le otto coordinate coordinate che definiscono i quattro punti della box.
    /// (xmin,ymin) (xmax,ymin) (xmax,ymax) (xmin,ymax).
    /// </summary
    public double[] dblSquare
    {
        get
        {
            double[] square = null;
            if(null!=this)
            {
                square = new double[8];
                square[0] = this.xmin;square[1] = this.ymin;
                square[2] = this.xmax;square[3] = this.ymin;
                square[4] = this.xmax;square[5] = this.ymax;
                square[6] = this.xmin;square[7] = this.ymax;
                return square;
            }
            else
                return null;
        }
    }

    /// <summary>
    /// Restituisce un array contenente le otto coordinate (tipizzate °object°)coordinate che definiscono i quattro
    /// punti della box.(xmin,ymin) (xmax,ymin) (xmax,ymax) (xmin,ymax).
    /// </summary
    /// <remarks>Da utilizzare in caso sia necessario un array di elementi °variant°.</remarks>
    public object[] objSquare
    {
        get
        {
            object[] square = null;
            if(null!=this)
            {
                square = new object[8];
                square[0] = this.xmin.ToString().Replace(",",".");
                square[1] = this.ymin.ToString().Replace(",",".");
                square[2] = this.xmax.ToString().Replace(",",".");
                square[3] = this.ymin.ToString().Replace(",",".");
                square[4] = this.xmax.ToString().Replace(",",".");
                square[5] = this.ymax.ToString().Replace(",",".");
                square[6] = this.xmin.ToString().Replace(",",".");
                square[7] = this.ymax.ToString().Replace(",",".");
                return square;
            }
            else
                return null;
        }
```

```
        }
        #endregion

        #region Costruttori
        /// <summary> Costruttore predefinito. </summary>
        public BoundingBox(){}

        /// <summary>Costruttore parametrizzato. </summary>
        /// <param name="xmin">Coordinata x del punto in basso a sinistra.</param>
        /// <param name="ymin">Coordinata y del punto in basso a sinistra.</param>
        /// <param name="xmax">Coordinata x del punto in alto a destra.</param>
        /// <param name="ymax">Coordinata y del punto in alto a destra.</param>
        public BoundingBox(double xmin, double ymin, double xmax, double ymax)
            {this.xmin = xmin;  this.ymin = ymin; this.xmax = xmax; this.ymax = ymax;}
        #endregion

        #region Override operatori di confronto ">" "<" "==" "!="

            /// <summary> Relazione di maggioranza fra due box. </summary>
            /// <param name="a">Box-1.</param>
            /// <param name="b">Box-2.</param>
            /// <returns>true se box-1 è "maggiore in relazione" a box-2.</returns>
            /// <remarks>
            /// Il confronto di maggioranza viene fatto tramite il confronto fra
            /// le aree delle due box; se l'area di box-1 è maggiore di quella di box-2.
            /// </remarks>
            public static bool operator > (BoundingBox a, BoundingBox b)
            {
                if(null==((object)a) && null==((object)b) ) return false;
                if(null!=((object)a) && null==((object)b) ) return true;
                if(null==((object)a) && null!=((object)b) ) return false;
                return (a.Area>b.Area);
            }
            /// <summary> Relazione di minoranza fra due box. </summary>
            /// <param name="a">Box-1.</param>
            /// <param name="b">Box-2.</param>
            /// <returns>true se box-1 è "minore in relazione" a box-2.</returns>
            /// <remarks>
            /// Il confronto di minoranza viene fatto tramite il confronto fra
            /// le aree delle due box; se l'area di box-1 è minore di quella di box-2.
            /// </remarks>
            public static bool operator < (BoundingBox a, BoundingBox b)
            {
                if(null==((object)a) && null==((object)b) ) return false;
                if(null!=((object)a) && null==((object)b) ) return false;
                if(null==((object)a) && null!=((object)b) ) return true;
                return (a.Area<b.Area);
            }

            /// <summary> Relazione di uguaglianta fra due box. </summary>
            /// <param name="a">Box-1.</param>
            /// <param name="b">Box-2.</param>
            /// <returns>true se box-1 è "uguale" a box-2</returns>
            /// <remarks>
            /// La relazione di uguaglianza viene stabilita confrontando i due punti
            /// della box. Le quattro coordinate sono uguali allora le due box sono "uguali".
            /// </remarks>
            public static bool operator == (BoundingBox a, BoundingBox b)
            {
                if(null==((object)a) && null==((object)b) ) return true;
                if(null!=((object)a) && null==((object)b) ) return false;
                if(null==((object)a) && null!=((object)b) ) return false;
                return (a.xmin==b.xmin && a.ymin==b.ymin && a.xmax==b.xmax && a.ymax==b.ymax);
            }

            /// <summary> Relazione di diversità fra due box. </summary>
            /// <param name="a">Box-1.</param>
            /// <param name="b">Box-2.</param>
            /// <returns>true se box-1 è "uguale" a box-2</returns>
            /// <remarks>
            /// La relazione di "diversità" viene stabilita confrontando i due punti
            /// della box. Basta che una sola coordinata sia diversa dalla
            /// rispettiva dell'altra box perché le due box siano "diverse".
            /// </remarks>
            public static bool operator != (BoundingBox a, BoundingBox b)
            {
                if(null==((object)a) && null==((object)b) ) return false;
                if(null!=((object)a) && null==((object)b) ) return true;
                if(null==((object)a) && null!=((object)b) ) return true;
                return (a.xmin!=b.xmin || a.ymin!=b.ymin || a.xmax!=b.xmax || a.ymax!=b.ymax);
            }
        #endregion

        #region Override generici
            public override bool Equals(object obj){return base.Equals (obj);}

            public override int GetHashCode(){return base.GetHashCode ();}

            /// <summary> Restituisce la lista di coordinate separate dal carattere "|". </summary>
            /// <returns>Lista di coordinate separate dal carattere "|".</returns>
            /// <remarks>Override del metodo °ToString°.</remarks>
            public override string ToString()
            {
                string t="";
                int i=0;
                for (i=0;i<this.strSquare.Length;i++)
                    t+= this.strSquare.GetValue(i) + "|";
                return t.Substring(0,t.Length-1);
            }
        #endregion
    }

    #region Collezioni
    /// <summary>
    /// Definisce una lista di temi. Ogni elemento è appartiene alla classe °MapThemeDescriptor°.
```

```
        /// </summary>
        /// <remarks>Implementa l'interfaccia °ICollection°.La classe è serializzabile.</remarks>
        [Serializable()]
        public class ThemeList : ICollection
        {
            /// <summary> Utilizzata solo a fini di serializzazione. </summary>
            /// <remarks>Uso deprecato. </remarks>
            public ArrayList themelist;

            #region Costruttori

            /// <summary> Costruttore predefinito </summary>
            public ThemeList()
            {this.themelist = new ArrayList();}

            /// <summary>Costruttore parametrizzato.</summary>
            /// <param name="capacity">Dimensione della lista.</param>
            public ThemeList(int capacity){this.themelist = new ArrayList(capacity);}
            #endregion

            #region Distruttore
            /// <summary> Distruttore.</summary>
            ~ThemeList()
            {
                int l = this.themelist.Count-1;
                if(l>=0)this.themelist.RemoveRange(0,l); //Svuota la lista.
                this.themelist = null;
            }
            #endregion
            #region Properties

            /// <summary> Accessore agli elementi della lista di temi. </summary>
            public MapThemeDescriptor this[int index]{get{return (MapThemeDescriptor) this.themelist[index];}}

            /// <summary>Restituisce il numero di elementi della lista di temi. </summary>
            public int Count{get{return this.themelist.Count;}}

            public object SyncRoot{ get{return this;}}

            public bool IsSynchronized{ get{return false;}}
            #endregion

            #region Metodi
            /// <summary> Aggiunge un tema alla lista di temi. </summary>
            /// <param name="theme">Oggetto di tipo °MapThemeDescriptor°.</param>
            public void Add(MapThemeDescriptor theme){this.themelist.Add(theme);}

            public void CopyTo(Array a, int index){this.themelist.CopyTo(a,index);}

            /// <summary> Permette di estrarre un segmento dalla lista di temi. </summary>
            /// <param name="start"> Indice del primo elemento appartenente al segmento da estrarre. </param>
            /// <param name="end"> Indice dell'ultimo elemento appartenente al segmento da estrarre. </param>
            /// <returns>
            /// Sottolista di temi di tipo °ThemeList° se l'operazione ha avuto esito positivo, altrimenti "null".
            /// </returns>
            public ThemeList GetRange(int start, int end)
            {
                if(end<=this.themelist.Count && start <this.themelist.Count && start>=0)
                {
                    ThemeList t = null;
                    if(start==end)
                        t = new ThemeList(1);
                    else
                        t = new ThemeList(end-start+1);
                    try
                    {
                        if(start==end)
                            t.Add( this[start]);
                        else
                            t.themelist.InsertRange(0,this.themelist.GetRange(start,(end-start+1)));
                    }
                    catch{t = null;}
                    return t;
                }
                else
                    return null;
            }

            public IEnumerator GetEnumerator(){return this.themelist.GetEnumerator();}

            public void RemoveAt(int index){themelist.RemoveAt(index);}

            /// <summary> Converte la lita di temi in un array di temi. </summary>
            /// <returns>Oggetto di tipo °MapThemeDescriptor[]°.</returns>
            public MapThemeDescriptor[] ToArray()
            {return (MapThemeDescriptor[]) this.themelist.ToArray(typeof(MapThemeDescriptor));}

            #region Ordinamenti
            /// <summary>
            /// Ordinamento crescente dei livelli in base alla loro sorgente dati. L'elemento prioritario è costituito
            /// dall'attributo °dataSource°, successivamente da °dataSourceType° ed infine °additionalInfo°.
            /// </summary>
            /// <remarks>
            /// Per quanto riguarda il confronto, siano due temi t1 e t2 e la relazione di confrondo [R] per cui : t1 [R] t2:
            /// 1) se t1 e t2 nulli, allora t1 [UGAILE-a] t2.
            /// 2) se t1 è non nullo e t2 è nullo, allora il t1 [MAGGIORE-di] t2.
            /// 2) se t1 è nullo e t2 è non nullo, allora il t1 [MINORE-di] t2.
            /// </remarks>
            public void SortBySource(){SortThemeBySource sc = new SortThemeBySource();this.themelist.Sort(sc);sc = null;}

            /// <summary> Ordinamento crescente dei temi in base al loro livello. </summary>
            public void SortByLevel(){SortThemeByLevel sc = new SortThemeByLevel();this.themelist.Sort(sc);sc = null;}
            #endregion
            #endregion
        }
```

```
        #endregion

        #region Criteri di ordinamento
        /// <summary>
        /// Criterio di confronto fra due temi in base al loro livello.
        /// </summary>
        class SortThemeByLevel : IComparer
        {
            /// <summary>Costruttore predefinito. </summary>
            public SortThemeByLevel(){}

            /// <summary>Metodo di confronto basato sul livello del tema.</summary>
            /// <param name="a">Oggetto di tipo MapThemeDescriptor.</param>
            /// <param name="b">Oggetto di tipo MapThemeDescriptor.</param>
            /// <returns>Valore intero ottenuto dal confronto dei valori di livello.</returns>
            public int Compare(object a, object b)
            {
                MapThemeDescriptor aa =(MapThemeDescriptor)a;
                MapThemeDescriptor bb =(MapThemeDescriptor)b;
                if(null==aa && null==bb) return 0;
                if(null==aa && null!=bb) return -1;
                if(null!=aa && null==bb) return 1;
                if(aa.Level > bb.Level) return 1;
                if(aa.Level == bb.Level) return 0;
                else return -1; // (aa.Level < bb.Level)
            }
        }

        /// <summary>Criterio di confronto fra due temi in base alla loro origine dati. </summary>
        /// <remarks>
        /// L'elemento prioritario è costituito dall'attributo °dataSource°, successivamente da °dataSourceType° ed
        /// infine °additionalInfo°.Per quanto riguarda il confronto, siano due temi t1 e t2 e la relazione di confrondo [R]
        /// per cui : t1 [R] t2:
        /// 1) se t1 e t2 nulli, allora t1 [UGAILE-a] t2.
        /// 2) se t1 è non nullo e t2 è nullo, allora il t1 [MAGGIORE-di] t2.
        /// 2) se t1 è nullo e t2 è non nullo, allora il t1 [MINORE-di] t2.
        /// </remarks>
        class SortThemeBySource : IComparer
        {
            /// <summary> Costruttore predefinito.</summary>
            public SortThemeBySource(){}

            /// <summary> Metodo di confronto </summary>
            /// <param name="a">Oggetto di tipo MapThemeDescriptor.</param>
            /// <param name="b">Oggetto di tipo MapThemeDescriptor.</param>
            /// <returns>valore intero ottenuto dal confronto i descrittori della sorgente dati.</returns>
            /// <remarks>L'ordinamento segue questa priorità "dataSource->dataSourceType->additionalInfo"</remarks>
            public int Compare(object a, object b)
            {
                try
                {
                    GenericLayerIdentifier.MapThemeDescriptor aa =(GenericLayerIdentifier.MapThemeDescriptor)a;
                    GenericLayerIdentifier.MapThemeDescriptor bb =(GenericLayerIdentifier.MapThemeDescriptor)b;
                    if(null==aa && null==bb) return 0;
                    if(null==aa && null!=bb) return -1;
                    if(null!=aa && null==bb) return 1;
                    int t = String.Compare(aa.dataSource,bb.dataSource);
                    if (t!=0)
                        return t;
                    else
                    {
                        t = String.Compare(aa.dataSourceType,bb.dataSourceType);
                        if (t!=0)
                            return t;
                        else
                            return String.Compare(aa.additionalInfo,bb.additionalInfo);
                    }
                }
                catch(Exception e){ return 0;}
            }
        }
        #endregion
    }
```

Appendice C

GWM-Engine Interface

C.1. Modelli UML delle Classi

L'interfaccia verso il motore GIS di GeoMedia WebMap (`GWM-Engine Interface`) è composta da quattro classi definite all'interno del namespace principale `GWMEngineInterface`, dell'omonima libreria (figura C.1).

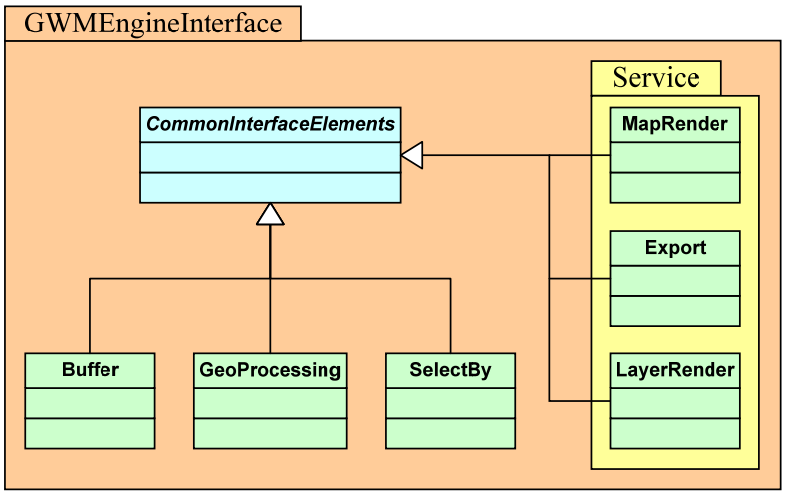


Figura C.1
Modello UML delle classi di GWM-Engine Interface.

C.1.1. CommonInterfaceElements

`CommonInterfaceElements` è una classe astratta di base, figura C.2, in cui è definito una parte della procedura SDES e le proprietà per riportare i messaggi di errore scaturiti dal sistema e dal motore GIS.

GWMEngineInterface::***CommonInterfaceElements***
#pMapServer : GWMMapServerHelper = null #pMapSrvMngr : MapServerManagerClass = null #setup : bool #pGeoEngineErrorMsg : string = “” #pSystemErrorMsg : string = “” #psrsPathName : string = “”
+CommonInterfaceElements() +CommonInterfaceElements(in Server : HttpServerUtility) +CommonInterfaceElements(in mapSrv : GWMMapServerHelp) +srsPathName() : string +GeoEngineErrorMsg() : string +SystemErrorMsg() : string +IsGeoEngineOK() : bool +MapServer() : GWMMapServerHelp +ShutDownGeoEngine() : bool +DataAccess(in layerA : LayerIdentifier, inout A : GRecordset, in layerB : LayerIdentifier, inout B : Grecordset, in InterestWindow : BoundingBox, inout msg : string) : bool +StandardizeConnectionParameters(in layerA : LayerIdentifier, in layerB : LayerIdentifier) -DisposeConnection() : int

Figura C.2
Classe astratta `CommonInterfaceElements`. Sono evidenziati, in giallo, i metodi pubblici relativi alla procedura SDES.

Della procedura SDES sono implementati i seguenti passi:

- **Start-Up**: tramite due costruttori parametrici. Ad uno viene passato il puntatore all'oggetto che consente di accedere al servizio di sistema `MapServer.exe` (mediante la classe `MapServerManagerClass`), mentre all'altro viene passato il puntatore all'oggetto che si occupa di processare le richieste web del web server (`HTTPServerUtility`);
- **Data Access**: il metodo `DataAccess` crea una o due connessioni distinte verso le sorgenti dati, specificate dagli identificatori di layer, modellati mediante la classe `LayerIdentifier` (vedere appendice B). L'esecuzione del metodo produce una sola connessione se la sorgente dati dei due layer coincide oppure se uno dei due layer è nullo. Eseguita la connessione, vengono estratti i dati relativi ai layer indicati. È possibile limitare la quantità di dati estratta specificando una box di contenimento tramite il parametro `InterestWindow`;
- **Shut-Down**: il metodo `ShutDownGeoEngine` esegue la disattivazione diretta del motore GIS, rilasciando le risorse allocate.

La fase "Execution" della procedura SDES, è implementata all'interno delle classi derivate da questa, mediante i metodi destinati all'esecuzione delle operazioni GIS.

Dato che le operazioni GIS possono essere eseguite su dataset caratterizzati da sistemi di riferimento spaziale eterogeneo, è necessario impostare un sistema di riferimento spaziale con cui georeferenziare il datatset prodotto da un'operazione di buffering o di overlay. La proprietà `SRSpath` permette di indicare il file (proprietario di GWM) che definisce il sistema di riferimento spaziale per georeferenziare il risultato.

Gli errori di sistema sono riportati mediante la proprietà `SystemErrMsg` mentre, la proprietà `GeoEngineErrMsg` riporta i malfunzionamenti relativi al motore GIS.

C.1.2. Buffer

La classe `Buffer` (figura C.3) espone i metodi necessari per eseguire due operazioni di buffering su un insieme di dati geografici definiti mediante un oggetto `GRecordset`, proprio delle librerie di GWM.

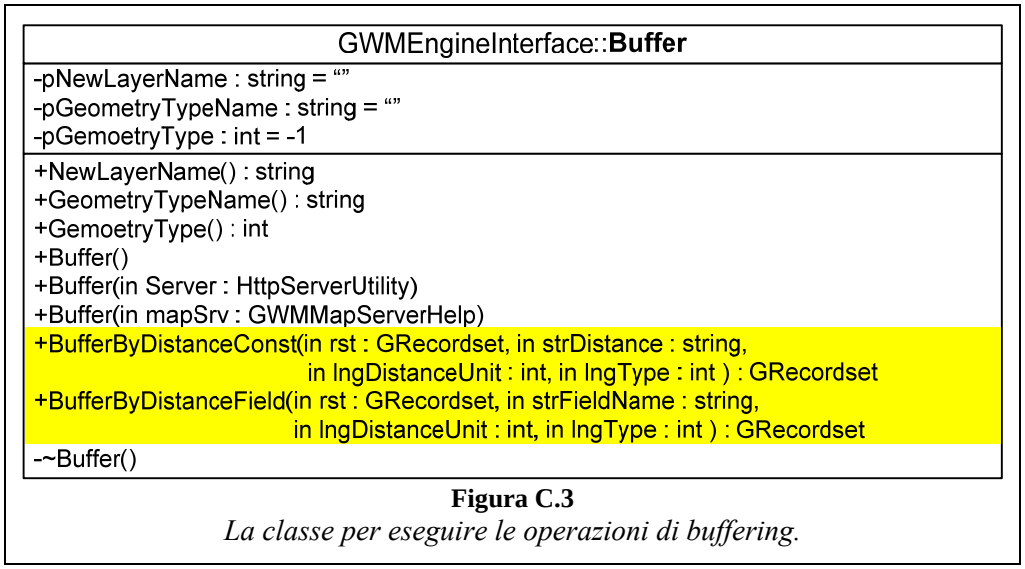


Figura C.3
La classe per eseguire le operazioni di buffering.

Il metodo `BufferByDistanceConst` permette di creare aree di rispetto di dimensione costante indicando una sequenza di intervalli codificati in una stringa. Il metodo `BufferByDistanceFiled` crea aree di rispetto in base al valore di un attributo numerico appartenente al dataset, indicato tramite il parametro `strFieldName`.

Il risultato dell'esecuzione dei metodi è un recordset corredato da un nome, passato tramite la proprietà `NewLayerName`, ed il tipo di geometria del recordset (che sarà sempre un poligono) mediante le seguenti proprietà (di sola lettura):

- `GeometryTypeName`: descrizione testuale del tipo di geometria (poligono, linea, punto);
- `GeometryType`: codice numerico relativo al tipo di geometria (1, 2, 10).

C.1.3. GeoProcessing

Questa classe espone i metodi per eseguire le operazioni di geoprocessing (figura C.4). Tutti i metodi ricevono in input due recordset contenenti le feature di input e le feature di overlay. Il risultato è modellato dell'esecuzione dei metodi, è rappresentato mediante un recordset.

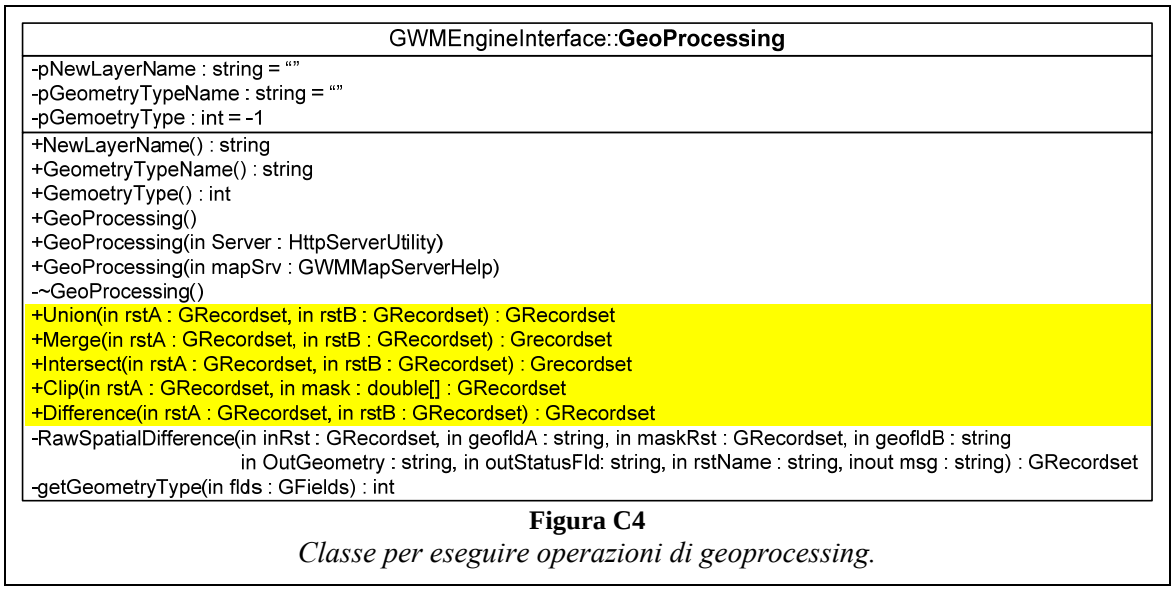


Figura C4

Classe per eseguire operazioni di geoprocessing.

La classe contiene il metodo privato `RawSpatialDifference` utilizzato per implementare l'operazione di differenza, `Difference`.

A differenza della classe `Buffer`, per determinare il tipo di geometria del risultato è utilizzato il metodo privato `getGeoemtryType` che analizza gli attributi del recordset prodotto: identifica l'attributo geometrico ed in base a questo determina se si tratta di una feature class poligonale puntuale o lineare.

C.1.4. SelectBy

La classe `SelectBy` (figura C.5) espone solo due metodi pubblici, uno per effettuare la selezione per tema in base al criterio di distanza, mentre il secondo esegue la selezione in base a come le feature del layer attivo e di selezione si sovrappongono (overlay).

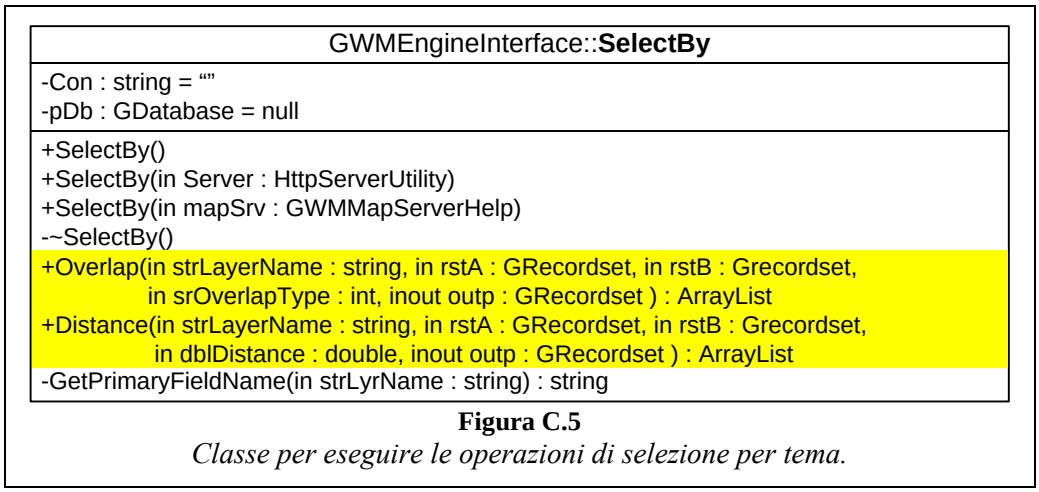


Figura C.5

Classe per eseguire le operazioni di selezione per tema.

Entrambi i metodi pubblici ricevono in ingresso due recordset contenenti rispettivamente il layer attivo e quello di selezione.

Il metodo `Overlap` si occupa della selezione in base alla sovrapposizione, il criterio di confronto spaziale di sovrapposizione è specificato mediante il parametro `srOverlapType`, i cui valori sono definiti all'interno del tipo enumerato `SpatialRelation` (figura C.6).

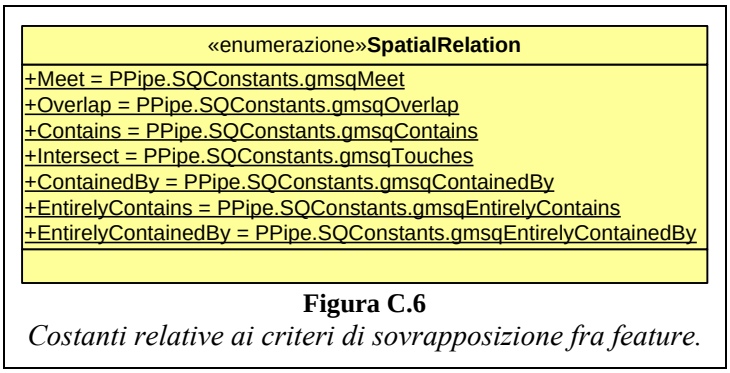


Figura C.6

Costanti relative ai criteri di sovrapposizione fra feature.

Il metodo `Distance` riceve come parametro un valore numerico che identifica la distanza da utilizzare nella relazione di confronto fra le feature dei due layer dati in ingresso.

Entrambi i metodi restituiscono una lista di identificatori ,rappresentata mediante la classe `ArrayList` del namespace `Collections` di .NET.

C.1.5. Il Namespace Service

Il namespace `Service`, definito all'interno del namespace principale, contiene tre classi dedicate alla rappresentazione dei recordset in formato grafico ed alla materializzazione su disco.

La classe **`MapRender`**, figura C.7, è pensata per creare una mappa "*on the fly*", mediante l'invocazione l'invocazione del metodo pubblico `RenderMap`. Come risulta dalla figura C.1, la classe utilizza la procedura SDES, ma internamente in modo che non sia accessibile all'utente.

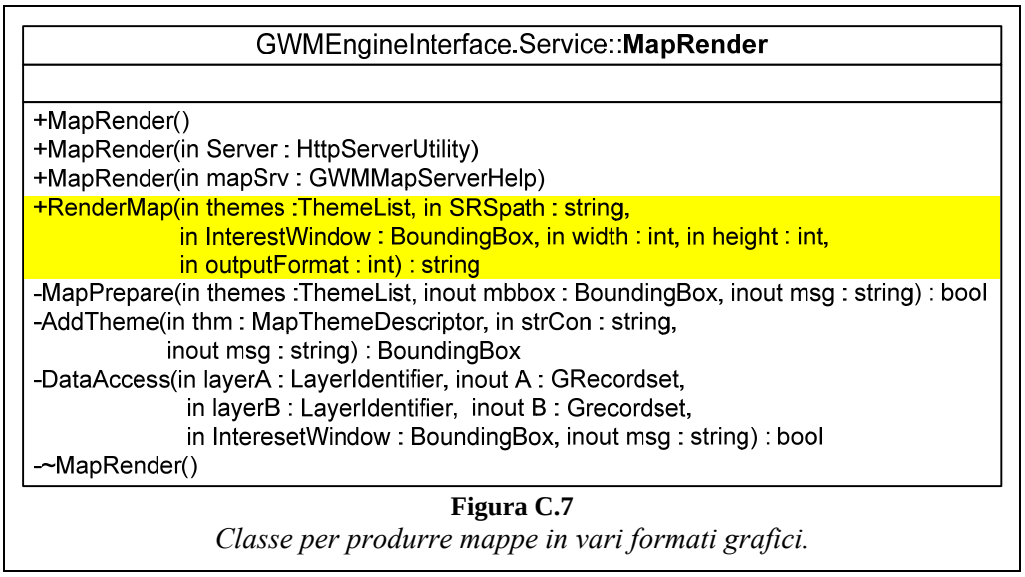


Figura C.7
Classe per produrre mappe in vari formati grafici.

Perché il metodo `RenderMap` possa creare una mappa, necessita dei seguenti parametri:

- `themes`: lista di temi descritti mediante la classe `ThemeList` descritta nell'appendice B;
- `SRSpath`: file che descrive il sistema di riferimento di mappa da utilizzare, il file deve essere completamente qualificato mediante il percorso nel file system;
- `width`, `height`: rispettivamente larghezza ed altezza, espressi in pixel, della mappa;
- `outputFormat`: costante intera che identifica il formato grafico dell'output (SVG, SVGZ, PNG, JPG, ACGM).

Il metodo restituisce la URL relativa al documento grafico prodotto.

La classe **`Export`** (figura C.8) espone un insieme di metodi per materializzare un dataset in alcuni formati di codifica. Attualmente sono stati previsti i tipi di codifica shape file e GML, ma la versatilità di GWM consente di esportare in altri formati come ad esempio DXF e Map di MapInfo.

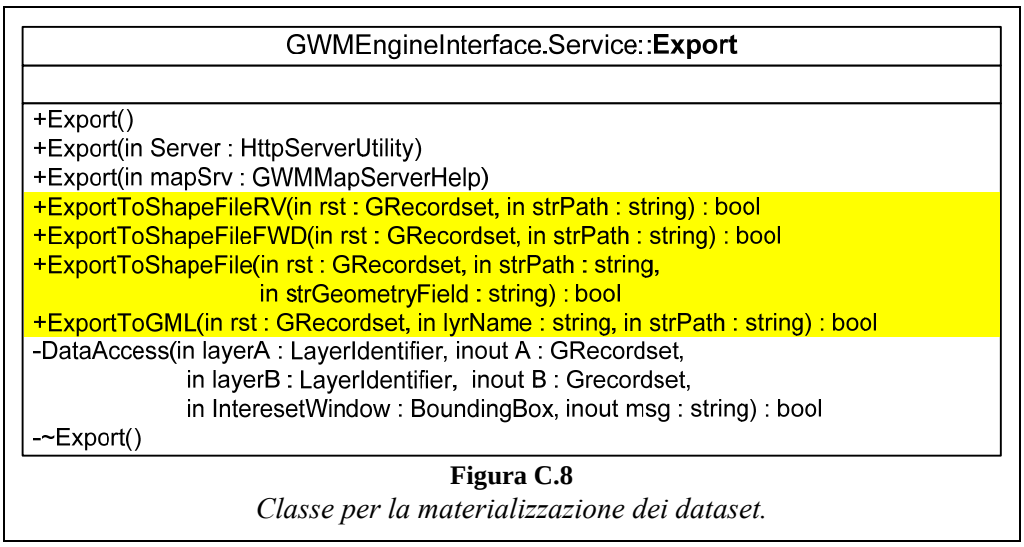


Figura C.8

Classe per la materializzazione dei dataset.

La classe è pensata per essere integrata con quelle del namespace principale, infatti, è possibile fornire l'ambiente del motore GIS, passando al suo costruttore il puntatore all'oggetto di tipo `MapServerManagerClass` istanziato all'interno di una delle istanze di `Buffer`, `GeoProcessing` o `SelectBy`.

I metodi che si occupano della materializzazione sono:

- `ExportToShapeFileRV`: crea il recordset in formato shape file nella posizione specificata dal parametro `strPath`. La particolarità di questo metodo consiste nel modo in cui effettua la ricerca dell'attributo geometrico. Il metodo esegue la scansione andando da destra a sinistra all'interno della lista degli attributi del recordset;
- `ExportToShapeFileFWD`: possiede la stessa semantica del precedente, tranne che l'attributo geometrico è individuato scandendo la lista degli attributi del recordset da sinistra verso destra.
- `ExportToShapeFile`: come i precedenti tranne che l'attributo geometrico è specificato dal parametro d'ingresso `strGeometryField`;
- `ExportToGML`: crea un documento codificato in GML v2.1.2. È obbligatorio assegnare il nome del layer da produrre, mediante il parametro d'ingresso `lyrName`.

Tutti i metodi accettano in ingresso un recordset ed il percorso dove materializzarlo, in caso di fallimento, restituiscono il valore logico "falso" e le proprietà relative alla gestione degli errori riportano la descrizione correlata.

La classe **`LayerRender`** (figura C.9) permette di restituire un solo recordset in un formato grafico quale ad esempio JPG, PNG, SVG, SVGZ, ACGM.

Come la precedente, è stata creata per poter essere utilizzata assieme alle classi che eseguono operazioni GIS.

Perché il layer possa essere restituito graficamente, deve essere corredato da uno stile di visualizzazione mediante la proprietà `Style`, così da poter essere considerato come tema.

Devono inoltre essere specificati le seguenti proprietà:

- `InputRecordset`: recordset di input di tipo `GRecordset`;
- `InterestArea`: opzionale, indica un'area d'interesse con cui limitare la porzione dei dati;
- `Width`, `Height`: larghezza ed altezza, espressi in pixel, del documento grafico;
- `GraphicThemeFormat`: formato grafico da utilizzare per codificare il tema.

Il risultato dell'operazione consiste in un documento grafico, riferito mediante una URL indicata tramite la proprietà `OutputFile`. In caso di fallimenti il metodo restituisce il valore logico "falso".

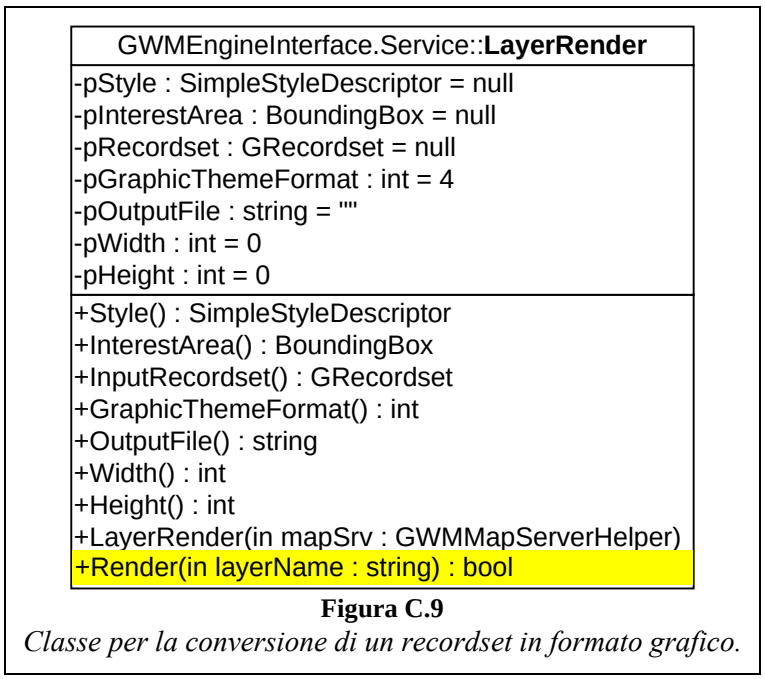


Figura C.9

Classe per la conversione di un recordset in formato grafico.

C.2. Codice Namespace GWMEngineInterface

```
using System;
using MapSrvLib = MAPSRVLib;
//********* GWM Namespaces **********
using MapSrvMngrLib = MAPSVRMNGRLib;
//******** Librearie interne ********
using GenericLayerIdentifier;

///<Author>Giulio Massei</Author>
///<LastUpdate>16/10/2005</LastUpdate>
namespace GWMEngineInterface
{
    /// <summary>Definisce gli elementi comuni a tutte le interfacce per GWM che supportano lo schema SDES:
    /// Start-up; Data access; Shut-down</summary>
    public abstract class CommonInterfaceElements
    {
        #region Costruttori
        /// <summary>Costruttore predefinto. </summary>
        public CommonInterfaceElements(){this.setup = false;}
        /// <summary> Costruttore ad inizializzazione interna.Il servizio GWM viene creato internamente alla classa.
        /// </summary>
        /// <param name="Server">Oggetto °Server° capace di istanziare il servizio GWM.</param>
        public CommonInterfaceElements(System.Web.HttpServerUtility Server)
        {
            try
            {
                this.pMapSrvMngr = (MapSrvMngrLib.MapServerManagerClass) Server.CreateObject("GMWebMap.MapServerManager");
                this.pMapServer = (MapSrvLib.GWMMapServerHelper) this.pMapSrvMngr.MapServer("");
                this.pMapServer.Clear();this.setup = true;
            }
            catch(Exception e)
            {
                this.pSystemErrorMsg = e.ToString();this.pGeoEngineErrorMsg = "Unable to turn on GWM-GIS engine.";
                this.setup = false;
            }
        }

        /// <summary>Costruttore basato su MapServer. </summary>
        /// <param name="mapSrv">Oggetto MapServer legato ai dati.</param>
        public CommonInterfaceElements(MapSrvLib.GWMMapServerHelper mapSrv)
            {this.setup = (null!=mapSrv);this.pMapServer = mapSrv;}
        #endregion

        #region Attributi Privati
        /// <summary>Ogetto MapServer sui cui si basa l'esecuzione delle operazioni.</summary>
        public MapSrvLib.GWMMapServerHelper pMapServer = null;
        internal MapSrvMngrLib.MapServerManagerClass pMapSrvMngr = null;
        /// <summary> Indica se il mapserver è stato impostato ed è valido (non nullo). </summary>
        internal bool setup = false; //Indica se è stato impostato il mapserver
        #endregion;

        #region Proprietà pubbliche
        internal string pSystemErrorMsg = "";
        /// <summary>Contiene i messaggi di errore del motore GIS (restituiti dall'esecuzione deglioggetti pipe da parte
        /// delMapServer). </summary>
        internal string pGeoEngineErrorMsg = "";
        internal string psrsPathName = "";
        public string srsPathName
        {
            set{this.psrsPathName = value;}
            get{return this.psrsPathName;}
        }
        /// <summary>
        /// Restituisce un messaggio di errore relativo all'operazione spaziale  che si è tentato di eseguire. Tale errore
        ///  è restitituito dal motore GIS. </summary>
        public string GeoEngineErrorMsg {get{return this.pGeoEngineErrorMsg;}}
        public bool IsGeoEngineOK{get{return this.setup;}}
        /// <summary>
        /// Restituisce un errore di sistema relativo all'operazione spaziale che si è tentato di eseguire.
        /// </summary>
        public string SystemErrorMsg{get{return this.pSystemErrorMsg;}}
        /// <summary>
        /// Parametro necessario, imposta un riferimentoal Map Server.
        /// Costituisce l'unico legame fra la classe ed l'oggetto MapServer.
        /// </summary>
        /// <remarks> Si può operare sul riferimento solo in modalità "write only". </remarks>
        public MapSrvLib.GWMMapServerHelper MapServer
        {
            set{his.setup = (null!=value);this.pMapServer = value;}
            get {return this.pMapServer;}
        }
        #endregion

        #region Metodi pubblici
        /// <summary>Forza lo shutdown del motore di GWM portando al rilascio il servizio. </summary>
        /// <returns>"True" se l'operazione ha avuto successo.</returns>
        public bool ShutDownGeoEngine()
        {
            bool t = false;
            try
            {
                if(this.pMapServer!=null)
                {
                    this.DisposeConnections();this.pMapServer.Clear();
                    System.Runtime.InteropServices.Marshal.ReleaseComObject(this.pMapServer);
                    this.pMapServer = null;
                }
                if(this.pMapSrvMngr!=null)
                    {System.Runtime.InteropServices.Marshal.ReleaseComObject(this.pMapSrvMngr);}
                t = true;
            }
```

```
            Catch{t = false;}
            return t;
        }
        /// <summary>Permette di accedere a uno o due layer e restituirne i rispettivi recordset.</summary>
        /// <param name="layerA">Identificatore del primo layer.</param>
        /// <param name="A">Oggetto che ospita il recordset del primo layer.</param>
        /// <param name="layerB">Identificatore del second layer.</param>
        /// <param name="B">Oggetto che ospita il recordset del secondo layer.</param>
        /// <param name="InterestWindow">Area di interesse.</param>
        /// <param name="msg">Messaggio di errore di sistema.</param>
        /// <returns>
        /// "true" se l'operazione ha avuto esito positivo. Altrimenti "false" ed il
        /// parametro °msg° viene inizializzato.</returns>
        /// <remarks> Questo metodo è replicato nelle clasis che implementano le "Interfacce-GWM".
        /// Può restituire anche un solo layer.</remarks>
        public bool DataAccess(LayerIdentifier layerA,ref PClient.GRecordset A, LayerIdentifier layerB, ref
                                        PClient.GRecordset B, BoundingBox InterestWindow, ref string msg)
        {
            bool t = false;string ConB = "ConA";string csfile = "";
            try
            {
                //Impostazione del sistema di riferimento.
                csfile = this.srsPathName;
                this.pMapServer.SetCoordinateSystem(csfile);
                this.StandardizeConnectionParameters(layerA,layerB);
                //Prima Connessione:
                this.pMapServer.Connect(layerA.dataSourceType,layerA.dataSource,layerA.additionalInfo,"ConA");
                if(null!=layerB)//Nel caso di buffering viene creato un solo record
                {
                    if(!layerA.SameSource(layerB))
                    {//Seconda Connessione:
                        this.pMapServer.Connect(layerB.dataSourceType,layerB.dataSource,layerB.additionalInfo,"ConB");
                        ConB = "ConB";
                    }
                }
                else
                {
                    #region RawQuery
                    MapSrvLib.IDGWMQuery qA = null;
                    qA = (MapSrvLib.IDGWMQuery) this.pMapServer.AddQuery("A","ConA",layerA.layerName);
                    qA.RequiredFieldNames = "*";
                    A = (PClient.GRecordset) qA.Recordset; qA = null;
                    if(null != layerB)//Nel caso di buffering viene creato un solo record
                    {
                        MapSrvLib.IDGWMQuery qB = null; msg += "_B_";
                        qB = (MapSrvLib.IDGWMQuery) this.pMapServer.AddQuery("B",ConB,layerB.layerName);
                        qB.RequiredFieldNames = "*";B = (PClient.GRecordset) qB.Recordset;qB = null;
                    }
                }
                t = true;
            }
                #region Catch & finally
            catch(System.Runtime.InteropServices.COMException Cex){t = false;msg = Cex.ToString();}
            catch(Exception ex){t = false;msg =ex.ToString();}
            #endregion
            return t;
        }

        internal void StandardizeConnectionParameters(LayerIdentifier layerA, LayerIdentifier layerB)
        {
            bool e= false;
            if(layerA.dataSourceType.ToLower().IndexOf(".gdatabase")<0)
                layerA.dataSourceType +=".GDatabase";
            if (null!=layerB)
                if(layerB.dataSourceType.ToLower().IndexOf(".gdatabase")<0)
                    layerB.dataSourceType +=".GDatabase";
            e=(layerA.SameSource(layerB));
            string t="";
            if(layerA.dataSourceType.ToUpper().IndexOf("WFS")>-1 ||
                layerA.dataSourceType.ToUpper().IndexOf("WMS")>-1)
            {
                t = @"CSFFOLDERPATH=C:\Program Files\GeoMedia WebMap Professional\CoordSystems;";
                t += "SWAPGEOCOORDS=FALSE;SWAPPROJCOORDS=FALSE;";
                t += "URI=" + layerA.dataSource + ";";layerA.dataSource = "dummy";
                layerA.additionalInfo = t;
            }
            if(e && null!=layerB)
                {layerB.additionalInfo = layerA.additionalInfo;layerB.dataSource = layerA.dataSource;}
            else
            if (null!=layerB)
                if(layerB.dataSourceType.ToUpper().IndexOf("WFS")>-1 ||
                    layerA.dataSourceType.ToUpper().IndexOf("WFMS")>-1)
                {
                    t = @"CSFFOLDERPATH=C:\Program Files\GeoMedia WebMap Professional\CoordSystems;";
                    t += "SWAPGEOCOORDS=FALSE;SWAPPROJCOORDS=FALSE;"; t += "URI=" + layerB.dataSource + ";";
                    layerB.dataSource = "dummyB";layerB.additionalInfo = t;
                }
        }
        private int DisposeConnections()
        {
            int i = 0;PClient.Connections cons = null;
            try
            {
                if (null!=this.pMapServer)
                {
                    cons = (PClient.Connections) this.pMapServer.DataSources;for(i=0;i<cons.Count;i++) cons.Delete(i);
                    if(null!=cons) System.Runtime.InteropServices.Marshal.ReleaseComObject(cons);
                }
            }
            catch(Exception e){ this.pSystemErrorMsg = e.ToString();i=-1;}
            cons = null;return i;
        }
        #endregion
    }
}
```

```
using System;
using System.Web;
using System.ComponentModel;
//********* GWM Namespaces **********
using PPipe;
using PDBPipe;
using PClient;
using Gdo = GDO;
using MapSrvLib = MAPSRVLib;
using MapSrvMngrLib = MAPSVRMNGRLib;
//******** Librearie interne ********
using GenericLayerIdentifier;


///<Author>Giulio Massei</Author>
///<LastUpdate>16/10/2005</LastUpdate>

namespace GWMEngineInterface
{
    /// <summary>Indica il tipo di buffer, come si presentano i bordi. </summary>
    public enum BufferEdgeType
    {
        /// <summary> Bordi arrotondati. </summary>
        SquareEdges = PPipe.GMBufferZoneTypeConstants.gmbztLinearRoundEnd,
        /// <summary>Bordi squadrati. </summary>
        RoundEdges = PPipe.GMBufferZoneTypeConstants.gmbztLinearRoundEnd
    }
    /// <summary>
    /// Permette di creare un buffer attornao alle feature di un layer. Definice la "Interfaccia-GWM" destinata al buffering.
    /// </summary>
    /// <remarks>
    /// Tutti i metodi restituiscono un recordset contenente la geometria del buffer,
    /// la distanza dalla feature e gli attributi originali del layer su cui si esegue il buffering.
    /// </remarks>
    public class Buffer:CommonInterfaceElements
    {
        #region Proprietà pubbliche
        private string pNewLayerName = "";private string pGeometryTypeName = "";private int pGeometryType = -1;
        public string NewLayerName
        {set{this.pNewLayerName = value;}get{return this.pNewLayerName;}}
        public string GeometryTypeName{get{return this.pGeometryTypeName;}}
        public int GeometryType{get{return this.pGeometryType;}}
        #endregion

        #region Costruttori
            /// <summary> Costruttore predefinto. </summary>
            public Buffer(){this.setup = false; }
            /// <summary>
            /// Costruttore ad inizializzazione interna. Il servizio GWM viene creato internamente alla classa.
            /// </summary>
            /// <param name="Server">Oggetto °Server° capace di istanziare il servizio GWM.</param>
            public Buffer(System.Web.HttpServerUtility Server)
            {
                try
                {   this.pMapSrvMngr = (MapSrvMngrLib.MapServerManagerClass) Server.CreateObject("GMWebMap.MapServerManager");
                    this.pMapServer = (MapSrvLib.GWMMapServerHelper) this.pMapSrvMngr.MapServer("");
                    this.pMapServer.Clear();this.setup = true;
                }
                catch(Exception e){ this.pSystemErrorMsg = e.ToString();this.pGeoEngineErrorMsg = "Unable to turn on GWM-GIS engine.";
                    this.setup = false;}
            }
            /// <summary>
            /// Costruttore basato su MapServer.
            /// </summary>
            /// <param name="mapSrv">Oggetto MapServer legato ai dati.</param>
            public Buffer(MapSrvLib.GWMMapServerHelper mapSrv) {this.setup = (null!=mapSrv);this.pMapServer = mapSrv;}
        #endregion

        #region Distruttore
        /// <summary> Distruttore </summary>
        /// <remarks> Forza lo shutdown del motore di GWM portando al rilascio il servizio. </remarks>
        ~Buffer()
        {
            if(this.pMapServer!=null)
            {this.pMapServer.Clear();System.Runtime.InteropServices.Marshal.ReleaseComObject(this.pMapServer);
                this.pMapServer = null;
            }
            if(this.pMapSrvMngr!=null){System.Runtime.InteropServices.Marshal.ReleaseComObject(this.pMapSrvMngr);}
        }
        #endregion

        #region Operatori
        /// <summary>Crea un buffer attorno alle feature di un layer secondo una distanza costante.</summary>
        /// <param name="rst">Recordset relativo al layer su cui effettuare l'operazione di buffering.</param>
        /// <param name="strDistance">  Raggio del buffer dal bordo della feature
        /// <remark>
        /// Il formato della distanza deve essere "generico", ie: "100.2".Per poter creare un buffer a più fasce,
        /// la distanza deve essere specificata in questo modo: "0:100;100:200;etc.". Il carattere ":" è il
        /// delimitatore di range, mentre ";" delimita le fasce.
        /// </remark>
        /// </param>
        /// <param name="lngDistanceUnit">
        /// Unità di misura lineare, nel sistema di riferimento del Layer. la distanza metrica decimale è 59</param>
        /// <param name="lngType">Tipo del buffer risultante: bordi arrotondati (RoundEdges) oppure squadrati (SquareEdges).</param>
        /// <returns>recordset contenente il buffer prodotto.</returns>
        public PClient.GRecordset BufferByDistanceConst(PClient.GRecordset rst, string strDistance, int lngDistanceUnit, int lngType)
        {
            this.pGeoEngineErrorMsg = "";this.pSystemErrorMsg = "";string rstName = this.pNewLayerName; string geofld = "";
            if(this.setup)
            {
                PPipe.BufferPipe bp = null; PClient.GRecordset t = null;Gdo.GRecordset tmp = null;
                if(rst.EOF)
                    {this.pGeoEngineErrorMsg = "Layer [" + rst.Name + "] is empty"; return null;}
                try
                {
```

```
                        geofld = Utils.GetGeometryFieldName(rst);
                        bp = (BufferPipe) this.pMapServer.CreateObject("GeoMedia.BufferPipe",bp);
                        //Peculiarità del Buffer: angoli rotondi o squadrati.
                        bp.BufferType = (PPipe.GMBufferZoneTypeConstants) lngType;
                        //Parametri di input legati al recordset.
                        tmp = (Gdo.GRecordset) rst; bp.set_InputRecordset(ref tmp);
                        bp.InputGeometryFieldName = geofld;
                        //Parametri di input legati alla distanza.
                        bp.InputDistance = strDistance; bp.DistanceType = PPipe.GMBufferZoneDistanceConstants.gmbzConstantDistance;
                        bp.InputDistanceUnit = lngDistanceUnit;
                        //Parametri di output.
                        bp.OutputDistanceFieldName = "DistanceFld"; bp.OutputGeometryFieldName = "OutGeometry";
                        bp.OutputStatusFieldName  = "OutStatusFld";
                    //Generazione recordset
                    tmp = bp.OutputRecordset;
                    if(null!=tmp)
                        if(!tmp.EOF)
                        {
                            if("" != tmp.GFields["OutStatusFld"].Value.ToString())
                                {this.pGeoEngineErrorMsg = tmp.GFields["OutStatusFld"].Value.ToString();t = null;}
                            else
                            {   MapSrvLib.GWMMarkerSystem mrkr = (MapSrvLib.GWMMarkerSystem)
                                        this.pMapServer.AddRecordsetMarker(rstName, (MapSrvLib.GRecordset) tmp,"OutGeometry");
                                t = (PClient.GRecordset) mrkr.Recordset;
                                this.pMapServer.DeleteMarker(rstName);this.pGeometryType = 2;  this.pGeometryTypeName ="Polygon";
                            }
                        }
                        else{this.pGeoEngineErrorMsg = "Empty result"; t = null;}
                }
                catch(Exception e){this.pSystemErrorMsg = e.ToString();t = null;}
                finally{if(null!=bp) System.Runtime.InteropServices.Marshal.ReleaseComObject(bp);}
                return t;
            }
            Else{this.pGeoEngineErrorMsg = "MapServer not set";return null;}
        }
        /// <summary>Crea un buffer attorno alle feature di un layer in base al valore di un attributo del
        /// layer.</summary>
        /// <param name="rst"> Recordset relativo al layer su cui effettuare l'operazione di buffering.</param>
        /// <param name="strFieldName">Nome dell'attributo su cui basere l'operazione di buffering.
        /// <remark>Sono ammessi solo valori numerici: double, single, long, integer, Byte e Currency. </remark> </param>
        /// <param name="lngDistanceUnit"> Unità di misura lineare, nel sistema di riferimento del Layer.</param>
        /// <param name="lngType">Tipo del buffer risultante: bordi arrotondati (RoundEdges) oppure squadrati (SquareEdges).
        /// </param>
        /// <returns>recordset contenente il buffer prodotto.</returns>
        public PClient.GRecordset BufferByDistanceField(PClient.GRecordset rst, string strFieldName, int lngDistanceUnit, int
                                                                                    lngType)
        {
            this.pGeoEngineErrorMsg = "";this.pSystemErrorMsg = "";string rstName = this.pNewLayerName;string geofld = "";
            if(this.setup)
            {
                PPipe.BufferPipe bp = null; PClient.GRecordset t = null;Gdo.GRecordset tmp = null;
                if(rst.EOF)
                    {this.pGeoEngineErrorMsg = "Layer [" + rst.Name + "] is empty"; return null;}
                try
                {
                    PClient.GField f = Utils.GetField(rst,strFieldName);
                    if(null==f)
                        {this.pGeoEngineErrorMsg = "Field [" + f.Name + "] not exists"; return null;}
                    if(f.Type!=PClient.GConstants.gdbDouble && f.Type!=PClient.GConstants.gdbSingle &&
                        f.Type!=PClient.GConstants.gdbLong && f.Type!=PClient.GConstants.gdbInteger &&
                        f.Type!=PClient.GConstants.gdbByte && f.Type!=PClient.GConstants.gdbCurrency)
                        {this.pGeoEngineErrorMsg = "Field [" + f.Name + "X" + strFieldName + "] has not a numeric type::" + f.Type;
                            return null;}f = null;
                    geofld = Utils.GetGeometryFieldName(rst);
                    bp = (BufferPipe) this.pMapServer.CreateObject("GeoMedia.BufferPipe",bp);
                    //Peculiarità del Buffer: angoli rotondi o squadrati.
                    bp.BufferType = (PPipe.GMBufferZoneTypeConstants) lngType;
                    //Parametri di input legati al recordset.
                    tmp = (Gdo.GRecordset) rst; bp.set_InputRecordset(ref tmp);
                    bp.InputGeometryFieldName = geofld;
                    //Parametri di input legati alla distanza.
                    bp.InputDistanceFieldName = strFieldName;
                    bp.DistanceType = PPipe.GMBufferZoneDistanceConstants.gmbzVariableDistance;
                    bp.InputDistanceUnit = lngDistanceUnit;
                    //Parametri di output.
                    bp.OutputDistanceFieldName = "DistanceFld"; bp.OutputGeometryFieldName = "OutGeometry";
                    bp.OutputStatusFieldName  = "OutStatusFld";
                    #endregion
                    //Generazione recordset
                    tmp = bp.OutputRecordset;
                    if(null!=tmp)
                        if(!tmp.EOF)
                        {
                            if("" != tmp.GFields["OutStatusFld"].Value.ToString())
                                {this.pGeoEngineErrorMsg = tmp.GFields["OutStatusFld"].Value.ToString();t = null;}
                            else
                            {
                                MapSrvLib.GWMMarkerSystem mrkr = (MapSrvLib.GWMMarkerSystem)
                                        this.pMapServer.AddRecordsetMarker(rstName,(MapSrvLib.GRecordset) tmp,"OutGeometry");
                                t = (PClient.GRecordset) mrkr.Recordset;
                                this.pMapServer.DeleteMarker(rstName);  this.pGeometryType = 2; this.pGeometryTypeName ="Polygon";
                            }
                        }
                        else{this.pGeoEngineErrorMsg = "Empty result"; t = null;}
                }
                catch(Exception e){ this.pSystemErrorMsg = e.ToString();t = null;}
                finally{if(null!=bp) System.Runtime.InteropServices.Marshal.ReleaseComObject(bp);}
                return t;
            }
            else{this.pGeoEngineErrorMsg = "MapServer not set";return null;}
        }
    }
}
```

```
using System;
using System.Web;
using System.ComponentModel;
//********* GWM Namespaces **********
using PPipe;
using PDBPipe;
using PClient;
using Gdo = GDO;
using MapSrvLib = MAPSRVLib;
using MapSrvMngrLib = MAPSVRMNGRLib;
//******** Librearie interne ********
using GenericLayerIdentifier;

///<Author>Giulio Massei</Author>
///<LastUpdate>11/11/2005</LastUpdate>

namespace GWMEngineInterface
{
    /// <summary>
    /// Espone un insieme di metodi per effettuare operazioni geospaziali su due strati informativi.
    /// Definice la "Interfaccia-GWM" destinata alle operazioni geospaziali.
    /// </summary>
    public class GeoProcessing:CommonInterfaceElements
    {
        #region Proprietà pubbliche
        private string pNewLayerName = "";private string pGeometryTypeName = "";
        private int pGeometryType = -1;
        /// <summary> Nome che l'operazione assegnerà al dataset </summary>
        public string NewLayerName
        {set{this.pNewLayerName = value;}get{return this.pNewLayerName;}}

        /// <summary> Descrizione della tipologia della geometria (Point, Line,..) </summary>
        public string GeometryTypeName{get{return this.pGeometryTypeName;}}

        /// <summary> Codice numerico relativo alla tipologia di geometria 1:line; 2 Polygon, 10 Point </summary>
        public int GeometryType{get{return this.pGeometryType;}}
        #endregion

        #region Costruttori
            /// <summary> Costruttore predefinito.</summary>
            /// <remarks>Se si utilizza questo costruttore è obbligatorio impostare la proprietà </remarks>
            public GeoProcessing(){}

            /// <summary> Costruttore ad inizializzazione interna. Il servizio GWM è creato internamente alla classa.
            /// </summary>
            /// <param name="Server">Oggetto °Server° capace di istanziare il servizio GWM.</param>
            public GeoProcessing(System.Web.HttpServerUtility Server)
            {
                try
                {
                    this.pMapSrvMngr = (MapSrvMngrLib.MapServerManagerClass) Server.CreateObject("GMWebMap.MapServerManager");
                    this.pMapServer = (MapSrvLib.GWMMapServerHelper) this.pMapSrvMngr.MapServer("");
                    this.pMapServer.Clear();this.setup = true;
                }
                catch(Exception e)
                {   this.pSystemErrorMsg = e.ToString();this.pGeoEngineErrorMsg = "Unable to turn on GWM-GIS engine.";
                    this.setup = false; }
            }

            /// <summary>
            /// Costruttore basato su MapServer.
            /// </summary>
            /// <param name="mapSrv">Oggetto MapServer legato ai dati.</param>
            public GeoProcessing(MAPSRVLib.GWMMapServerHelper mapSrv)
            {
                this.setup = (null!=mapSrv);
                this.pMapServer = mapSrv;
            }
        #endregion

        #region Distruttore

        /// <summary> Distruttore </summary>
        /// <remarks>Forza lo shutdown del motore di GWM portando al rilascio il servizio.</remarks>
        ~GeoProcessing()
        {
            try
            {
                if(this.pMapServer!=null)
                {
                    this.pMapServer.Clear();System.Runtime.InteropServices.Marshal.ReleaseComObject(this.pMapServer);
                    this.pMapServer = null;
                }
                if(this.pMapSrvMngr!=null)
                    System.Runtime.InteropServices.Marshal.ReleaseComObject(this.pMapSrvMngr);
            }
            catch{}
        }

        #endregion

        #region Operatori
        /// <summary>Effettua l'unione di due layer in uno. </summary>
        /// <remarks> L'unione è fatta considerando tutti gli attributi dei due layer. </remarks>
        /// <param name="rstA">Recordet che rappresenta il primo layer.</param>
        /// <param name="rstB">Recordet che rappresenta il secondo layer.</param>
        /// <returns>Recordset contenente l'unione.</returns>
        public PClient.GRecordset Union(PClient.GRecordset rstA, PClient.GRecordset rstB)
        {
            this.pGeoEngineErrorMsg = "";
            this.pSystemErrorMsg = "";
            if(this.setup)
            {
                PDBPipe.UnionPipe up = null;PClient.GRecordset t = null;
                try
                {//Creazione della pipe UnionPipe
                    up = (PDBPipe.UnionPipe) this.pMapServer.CreateObject("GeoMedia.UnionPipe",up);
```

```
                    up.InputRecordsets.Append((PDBPipe.GRecordset)rstA);up.InputRecordsets.Append((PDBPipe.GRecordset)rstB);
                    int gmfldA = this.getGeometryType(rstA.GFields);//Acquisizione attributo geometrico
                    int gmfldB = this.getGeometryType(rstB.GFields);
                    //Esecuzione Unione
                    up.OutputSchemaMode = PDBPipe.OutputSchemaModeConstants.gmosmUnionAllRecordsets;
                    t = (PClient.GRecordset) up.OutputRecordset;t.MoveFirst();
                    //Acquisizione attributo geometrico del recordsetprodotto
                    string geofld = Utils.GetGeometryFieldName(t);
                    string rstName = this.pNewLayerName;//Nome del recordset
                    //Associazione del nome al recordset tramite marker
                    this.pMapServer.AddRecordsetMarker( rstName,(MapSrvLib.GRecordset) t, geofld);
                    MapSrvLib.GWMMarkerSystem mrkr = (MapSrvLib.GWMMarkerSystem)
                                                    ((MapSrvLib.GWMDispatchCollection)this.pMapServer.Markers)[rstName];
                    t = (PClient.GRecordset) mrkr.Recordset;
                    if(gmfldA != gmfldB) this.getGeometryType((PClient.GFields) t.GFields);
                    this.pMapServer.DeleteMarker(rstName); mrkr = null;
                    t.MoveFirst();
                }
                #region Catch & finally
                catch(Exception e){ this.pSystemErrorMsg = e.Message;  t = null;}
                finally {if(null!=up) System.Runtime.InteropServices.Marshal.ReleaseComObject(up);}
                #endregion
                return t;
            }
            else{this.pGeoEngineErrorMsg = "MapServer not set";return null;}
        }

        /// <summary> Effetua la fusione (merge) fra due layer. </summary>
        /// <remarks>La fusione è concettualmente simile all'unione; gli attributi risultatnti dall'operazione
        /// sono quelli del primo layer. </remarks>
        /// <param name="rstA">Recordet che rappresenta il primo layer.</param>
        /// <param name="rstB">Recordet che rappresenta il secondo layer.</param>
        /// <returns>Recordset contenente la fusione.</returns>
        public PClient.GRecordset Merge(PClient.GRecordset rstA, PClient.GRecordset rstB)
        {
            this.pGeoEngineErrorMsg = "";this.pSystemErrorMsg = "";
            if(this.setup)
            {
                PDBPipe.UnionPipe up = null;PClient.GRecordset t = null;
                try
                {
                    up = (UnionPipe) this.pMapServer.CreateObject("GeoMedia.UnionPipe",up);
                    up.InputRecordsets.Append((PDBPipe.GRecordset)rstA);up.InputRecordsets.Append((PDBPipe.GRecordset)rstB);
                    up.OutputSchemaMode = PDBPipe.OutputSchemaModeConstants.gmosmFirstRecordset;
                    t = (PClient.GRecordset) up.OutputRecordset;
                    int gmfldA = this.getGeometryType(rstA.GFields);int gmfldB = this.getGeometryType(rstB.GFields);
                    string geofld = Utils.GetGeometryFieldName(t);string rstName = this.pNewLayerName;
                    this.pMapServer.AddRecordsetMarker( rstName,(MapSrvLib.GRecordset) t, geofld);
                    MapSrvLib.GWMMarkerSystem mrkr = (MapSrvLib.GWMMarkerSystem)
                                                    ((MapSrvLib.GWMDispatchCollection)this.pMapServer.Markers)[rstName];
                    t = (PClient.GRecordset) mrkr.Recordset;
                    this.pMapServer.DeleteMarker(rstName); mrkr = null;t.MoveFirst();
                    if(gmfldA != gmfldB) this.getGeometryType((PClient.GFields) t.GFields);
                }
                #region Catch & finally
                catch(Exception e)
                {this.pSystemErrorMsg = e.Message.ToString();t = null;}
                finally {if(null!=up) System.Runtime.InteropServices.Marshal.ReleaseComObject(up);}
                #endregion
                return t;
            }
            else
            {this.pGeoEngineErrorMsg = "MapServer not set";return null;}
        }

        /// <summary>Effettua l'intersezione fra due layer.</summary>
        /// <param name="rstA">Recordet che rappresenta il primo layer.</param>
        /// <param name="rstB">Recordet che rappresenta il secondo layer.</param>
        /// <returns>Recordset contenente l'intersezione.</returns>
        public PClient.GRecordset Intersect(PClient.GRecordset rstA, PClient.GRecordset rstB)
        {
            this.pGeoEngineErrorMsg = "";this.pSystemErrorMsg = "";
            if(this.setup)
            {
                PPipe.SpatialIntersectionPipe sip = null;
                PClient.GRecordset t = null;
                string geofldA = Utils.GetGeometryFieldName(rstA);string geofldB = Utils.GetGeometryFieldName(rstB);
                if(""!=geofldA.Trim())
                    if(""!=geofldB.Trim())
                    {
                        try
                        {
                            sip = (SpatialIntersectionPipe)this.pMapServer.CreateObject("GeoMedia.SpatialIntersectionPipe",sip);
                            sip.LeftGeometryFieldName = geofldA;sip.LeftRecordset = (Gdo.GRecordset) rstA;
                            sip.RightGeometryFieldName = geofldB;sip.RightRecordset = (Gdo.GRecordset) rstB;
                            sip.OutputGeometryFieldName = "OutGeometry";sip.OutputStatusFieldName = "OutStatusFld";
                            sip.SpatialOperator = PPipe.SQConstants.gmsqTouches;//gmsqContains
                            t = (PClient.GRecordset) sip.OutputRecordset;t.MoveLast();
                            int gmfldA = this.getGeometryType(rstA.GFields);int gmfldB = this.getGeometryType(rstB.GFields);
                            if(!t.EOF)
                            {
                                if(""!=t.GFields["OutStatusFld"].Value.ToString())
                                    {this.pGeoEngineErrorMsg = t.GFields["OutStatusFld"].Value.ToString();t = null;           }
                                else
                                {
                                    string geofld = "OutGeometry";string rstName = this.pNewLayerName;
                                    this.pMapServer.AddRecordsetMarker( rstName,(MapSrvLib.GRecordset) t, geofld);
                                    MapSrvLib.GWMMarkerSystem mrkr = (MapSrvLib.GWMMarkerSystem)
((MapSrvLib.GWMDispatchCollection)this.pMapServer.Markers)[rstName];
                                    t = (PClient.GRecordset) mrkr.Recordset;this.pMapServer.DeleteMarker(rstName); mrkr = null;
                                    if(gmfldA != gmfldB) this.getGeometryType((PClient.GFields) t.GFields);
                                }
                            }
                            else{this.pGeoEngineErrorMsg = "Empty result";t = null;}

                        }
                        #region Catch & finally
```

```
                    catch(Exception e){ this.pSystemErrorMsg = e.ToString();t = null;}
                    finally{if(null!=sip) System.Runtime.InteropServices.Marshal.ReleaseComObject(sip);}
                    #endregion
                }
                else
                    {t = null; this.pGeoEngineErrorMsg = "Layer [" + rstB.Name + "] has no geometry field";}
            else
                {t = null;this.pGeoEngineErrorMsg = "Layer [" + rstA.Name + "] has no geometry field";}
            return t;
        }
        else
            {this.pGeoEngineErrorMsg = "MapServer not set.";return null;}
    }

    /// <summary> Restituisce una porzione del primo layer identificata dalle feature del secondo. </summary>
    /// <remarks> Si pensi al secondo layer come uno stampino applicato al primo layer. </remarks>
    /// <param name="rstA">Recordet che rappresenta il primo layer.</param>
    /// <param name="rstB">Recordet che rappresenta il secondo layer.</param>
    /// <returns>Recordset contenente l'intersezione.</returns>
    public PClient.GRecordset Clip(PClient.GRecordset rstA, PClient.GRecordset rstB)
    {
        this.pGeoEngineErrorMsg = ""; this.pSystemErrorMsg = "";
        if(this.setup)
        {
            #region Dichiarazioni
            PPipe.SpatialDifferencePipe sdp = null; Gdo.GRecordset t = null;
            string geofldA = Utils.GetGeometryFieldName(rstA);
            string geofldB = Utils.GetGeometryFieldName(rstB);
            string msg ="";
            #endregion
            if(""!=geofldA.Trim())
                if(""!=geofldB.Trim())
                {
                    PClient.GField fA = Utils.GetGeometryField(rstA);PClient.GField fB = Utils.GetGeometryField(rstB);
                    if(
                        !(fA.SubType == PClient.GConstants.gdbAreal && fB.SubType == PClient.GConstants.gdbLinear) &&
                        !(fA.SubType == PClient.GConstants.gdbAreal && fB.SubType == PClient.GConstants.gdbPoint) &&
                        !(fA.SubType == PClient.GConstants.gdbLinear && fB.SubType == PClient.GConstants.gdbPoint))
                    {
                        #region Clipping
                        try
                        {//prima differenza (A\B)=C
                            t = this.RawSpatialDifference((Gdo.GRecordset) rstA, geofldA,(Gdo.GRecordset) rstB, geofldB,
                                                          "OutGeometryT","outStatusFldT","tmp", ref msg);
                            if(null!=t)
                            {
                                msg = "";   t.MoveLast();t.MoveFirst();
                                if(t.RecordCount>0)
                                {
                                    if(""!=t.GFields["outStatusFldT"].Value.ToString())
                                    {this.pGeoEngineErrorMsg = t.GFields["outStatusFldT"].Value.ToString(); t = null;}
                                    #region seconda diferenza
                                    else//Inizio seconda differenza: A\C
                                    {
                                        t = this.RawSpatialDifference((Gdo.GRecordset) rstA, geofldA,t, "OutGeometryT",
                                                                      "OutGeometry","OutStatusFld",this.NewLayerName,ref msg);
                                        if(null!=t)
                                        {
                                            msg = "";t.MoveLast();
                                            if(t.RecordCount>0)
                                                if(""!=t.GFields["OutStatusFld"].Value.ToString())
                                                {this.pGeoEngineErrorMsg = t.GFields["OutStatusFld"].Value.ToString();t = null}
                                        }
                                        else
                                        {this.pSystemErrorMsg = msg;t = null;}//Fine else secondo controllo null!=t
                                    }//Fine seconda differenza
                                    #endregion
                                }//Primo test sulla cardinalità di "t".
                            }//Fine primo controllo null!=t
                            else{this.pSystemErrorMsg = msg; t = null; }
                        }
                        #region Catch & finally
                        catch(Exception e) {this.pSystemErrorMsg = e.ToString();t = null; }
                        finally{if(null!=sdp) System.Runtime.InteropServices.Marshal.ReleaseComObject(sdp); }
                        #endregion
                        #endregion
                    }
                    else{t = null; this.pGeoEngineErrorMsg = "Unable to make clipping between layers";}
                }
                else{t = null; this.pGeoEngineErrorMsg = "Layer [" + rstB.Name + "] has no geometry field";}
            else
            {t = null; this.pGeoEngineErrorMsg = "Layer [" + rstA.Name + "] has no geometry fiel";}
            return (PClient.GRecordset) t;
        }
        else{this.pGeoEngineErrorMsg = "MapServer not set";return null; }
    }

    /// <summary> Effettua la differenza (booleana) fra due layer. </summary>
    /// <param name="rstA">Recordet che rappresenta il primo layer.</param>
    /// <param name="rstB">Recordet che rappresenta il secondo layer.</param>
    /// <returns>Recordset contenente l'intersezione.</returns>
    public PClient.GRecordset Difference(PClient.GRecordset rstA, PClient.GRecordset rstB)
    {
        this.pGeoEngineErrorMsg = "";this.pSystemErrorMsg = "";
        if(this.setup)
        {
            #region Dichiarazioni
            PPipe.SpatialDifferencePipe sdp = null;
            Gdo.GRecordset t = null; string msg ="";
            string geofldA = Utils.GetGeometryFieldName(rstA); string geofldB = Utils.GetGeometryFieldName(rstB);
            #endregion
            if(rstA.EOF) {this.pGeoEngineErrorMsg="Layer [" + rstA.Name + "] is empty"; return null;}
            if(rstB.EOF) {this.pGeoEngineErrorMsg="Layer [" + rstB.Name + "] is empty"; return null;}
            if(""!=geofldA.Trim())
                if(""!=geofldB.Trim())
                {PClient.GField fA = Utils.GetGeometryField(rstA);PClient.GField fB = Utils.GetGeometryField(rstB);
                    int gmfldA = this.getGeometryType(rstA.GFields); int gmfldB = this.getGeometryType(rstB.GFields);
```

```
                    if(
                        !(fA.SubType == PClient.GConstants.gdbAreal && fB.SubType == PClient.GConstants.gdbLinear) &&
                        !(fA.SubType == PClient.GConstants.gdbAreal && fB.SubType == PClient.GConstants.gdbPoint) &&
                        !(fA.SubType == PClient.GConstants.gdbLinear && fB.SubType == PClient.GConstants.gdbPoint))
                    {
                        #region Difference
                        try
                        {
                            t = this.RawSpatialDifference((Gdo.GRecordset) rstA, geofldA, (Gdo.GRecordset) rstB, geofldB,
                                                "OutGeometry","outStatusFld",this.NewLayerName, ref msg);
                            if(null!=t)
                                if(!t.EOF)
                                {
                                    if("" != t.GFields["OutStatusFld"].Value.ToString())
                                        {this.pGeoEngineErrorMsg = t.GFields["OutStatusFld"].Value.ToString();t = null;}
                                    else
                                    {
                                        string rstName = this.pNewLayerName;MapSrvLib.GWMMarkerSystem mrkr =
                                                    (MapSrvLib.GWMMarkerSystem)this.pMapServer.AddRecordsetMarker(rstName,
                                            (MapSrvLib.GRecordset) t,"OutGeometry");
                                        t = (Gdo.GRecordset) mrkr.Recordset;
                                        if(gmfldA != gmfldB) this.getGeometryType((PClient.GFields) t.GFields);
                                        this.pMapServer.DeleteMarker(rstName);
                                    }
                                }
                                else
                                {   if(""==msg) msg = "Empty result";this.pGeoEngineErrorMsg = msg;
                                    msg = ""; t = null;
                                }
                            else{this.pSystemErrorMsg = msg; t = null;}
                        }
                        #region Catch & finally
                        catch(Exception e) {this.pSystemErrorMsg = e.ToString();t = null; }
                        finally{if(null!=sdp) System.Runtime.InteropServices.Marshal.ReleaseComObject(sdp);}
                        #endregion
                    }
                    #endregion
                    else{t = null; this.pGeoEngineErrorMsg = "Unable to make difference between layers";}
                }
                else
                    {t = null; this.pGeoEngineErrorMsg = "Layer [" + rstB.Name + "] has no geometry field";}
            else
            {t = null; this.pGeoEngineErrorMsg = "Layer [" + rstA.Name + "] has no geometry fiel";}
            return (PClient.GRecordset) t;
        }
        else{this.pGeoEngineErrorMsg = "MapServer not set";return null; }
    }
    #endregion

    #region Metodi Privati
    /// <summary> effettua la differenza fra due recordset. </summary>
    /// <remarks>Viene impegato sia per l'operazione "pubblica" di differenza, che per implementare
    /// l'operazione di clipping in quanto: A c B = A /(A/B) </remarks>
    /// <param name="inRst">Recordset sinistro.</param>
    /// <param name="geofldA">Nome del campo che definisce la geometria del recordset sinistro.</param>
    /// <param name="maskRst">Recordset destro (recordset maschera).</param>
    /// <param name="geofldB">Nome del campo che definisce la geometria del recordse destro.</param>
    /// <param name="OutGeometry">Nome del campo che desinirà la geometria del recordset prodotto.</param>
    /// <param name="outStatusFld"> Nome del campo di status del recordset prodotto.
    /// Se vi sono degli errori il campo ne contiene la descrizione. </param>
    /// <param name="rstName">Nome del recordset da produrre</param>
    /// <param name="msg">Messaggio di errore.</param>
    /// <returns> Recordset contenente la differsenza fra i due. Altrimenti, in caso di errori, "null". </returns>
    private Gdo.GRecordset RawSpatialDifference(Gdo.GRecordset inRst, string geofldA, Gdo.GRecordset maskRst, string geofldB,
string OutGeometry, string outStatusFld, string rstName, ref string msg)
    {
        PPipe.SpatialDifferencePipe sdp = null;
        Gdo.GRecordset t = null;
        try
        {
            sdp = (SpatialDifferencePipe) this.pMapServer.CreateObject("GeoMedia.SpatialDifferencePipe",sdp);
            sdp.InputGeometryFieldName = geofldA; sdp.InputRecordset = inRst;
            sdp.MaskGeometryFieldName = geofldB; sdp.MaskRecordset = maskRst;
            sdp.OutputGeometryFieldName = OutGeometry; sdp.OutputStatusFieldName = outStatusFld;
            this.pMapServer.AddRecordsetMarker(rstName, (MapSrvLib.GRecordset) sdp.OutputRecordset, OutGeometry);
            MapSrvLib.GWMMarkerSystem mrkr = (MapSrvLib.GWMMarkerSystem)
                                            ((MapSrvLib.GWMDispatchCollection)this.pMapServer.Markers)[rstName];
            t = (Gdo.GRecordset) mrkr.Recordset;this.pMapServer.DeleteMarker(rstName);
        }
        catch(Exception e) {msg += e.ToString();t = null; }
        finally{if(null!=sdp) System.Runtime.InteropServices.Marshal.ReleaseComObject(sdp); }
        return t;
    }
    /// <summary>Individua il tipo di geometria della feature class.</summary>
    /// <param name="flds">Lista degli attributi del recordset</param>
    /// <returns>restituisce il codice numerico relativo al tipo di geometria.</returns>
    private int getGeometryType(PClient.GFields flds)
    {
        int i = flds.Count-1;short k = 0;string n = "";string nn = "H";
        while(i>=0 && flds[i].Type!=PClient.GConstants.gdbSpatial)
            {n = flds[i].Name.Trim();k = flds[i].Type;i--;}
        this.GpeometryType = flds[i].SubType;
        switch (this.pGeometryType)
        {
            case 1:  this.pGeometryTypeName = "Line";break;
            case 2:  this.pGeometryTypeName = "Polygon";break;
            case 10: this.pGeometryTypeName = "Point";break;
            default: this.pGeometryTypeName = "NA";this.GeometryType = -1;break;
        }
        return this.GeometryType;
    }
    #endregion
  }
}
```

```
using System.Web;
using System.Collections;
using System.ComponentModel;
//********* GWM Namespaces **********
using PPipe;
using PDBPipe;
using PClient;
using Gdo = GDO;
using MapSrvLib = MAPSRVLib;
using MapSrvMngrLib = MAPSVRMNGRLib;
//******** Librearie interne ********
using GenericLayerIdentifier;
///<Author>Giulio Massei</Author>
///<LastUpdate>16/10/2005</LastUpdate>
namespace GWMEngineInterface
{
    public enum SpatialRelation
    {
        Meet                        = PPipe.SQConstants.gmsqMeet,
        /// <summary> Sovrapposizione.</summary>
        Overlap                     = PPipe.SQConstants.gmsqOverlap,
        /// <summary> Contenimento: A contiene B. </summary>
        Contains                    = PPipe.SQConstants.gmsqContains,
        /// <summary>Intersezione: A interseca B. </summary>
        Intersect                   = PPipe.SQConstants.gmsqTouches,
        /// <summary> Contenimento inverso: A contenuto in B. </summary>
        ContainedBy                 = PPipe.SQConstants.gmsqContainedBy,
        /// <summary>Contenimento completo: A contiene completamente B</summary>
        EntirelyContains        = PPipe.SQConstants.gmsqEntirelyContains,
        /// <summary>Contenimento completo inverso: A è contenuto completamente in B</summary>
        EntirelyContainedBy = PPipe.SQConstants.gmsqEntirelyContainedBy
    }
    /// <summary>Questa classe permette di operare selezioni di feature secondo criteri spaziali.
    /// Definice la "Interfaccia-GWM" destinata alla selezione.</summary>
    public class SelectBy:CommonInterfaceElements
    {
        #region Attributi Privati (specifici di questa classe)
        /// <summary>Mantiene il nome della connesione ai dati. </summary>
        private string Con = "";
        /// <summary>Puntatore al database (il datasource in memoria, dove si trova il layer).</summary>
        private Gdo.GDatabase pDb = null;
        #endregion
        #region Costruttori
        /// <summary>Costruttore predefinito.</summary>
        public SelectBy(){this.setup = false;}
        /// <summary>Costruttore ad inizializzazione interna.Il servizio GWM viene creato internamente alla classa. </summary>
        /// <param name="Server">Oggetto °Server° capace di istanziare il servizio GWM.</param>
        public SelectBy(System.Web.HttpServerUtility Server)
        {
            try
            {
                this.pMapSrvMngr = (MapSrvMngrLib.MapServerManagerClass) Server.CreateObject("GMWebMap.MapServerManager");
                this.pMapServer = (MapSrvLib.GWMMapServerHelper) this.pMapSrvMngr.MapServer("");
                this.pMapServer.Clear();this.setup = true;
            }
            catch(Exception e)
            {this.pSystemErrorMsg = e.ToString();this.pGeoEngineErrorMsg = "Unable to turn on GWM-GIS engine."; this.setup = false;}
        }
        /// <summary> Costruttore basato su MapServer. </summary>
        /// <param name="mapSrv">Oggetto MapServer legato ai dati.</param>
        public SelectBy(MapSrvLib.GWMMapServerHelper mapSrv)
        {
            PClient.Connection con = null;MapSrvLib.GWMDataSource ds = null;
            MapSrvLib.GWMDispatchCollection tmp = null; this.setup = (null!=mapSrv);this.pMapServer = mapSrv;
            try
            {
                tmp = (MapSrvLib.GWMDispatchCollection) this.pMapServer.DataSources;
                ds = (MapSrvLib.GWMDataSource) tmp[1];  con =(PClient.Connection) ds.Connection;this.Con = con.ConnectionName; }
            catch(Exception e)
            {   this.Con = "";this.setup =false;this.pMapServer = null;this.pSystemErrorMsg = e.ToString();
                this.pGeoEngineErrorMsg = "";}
            finally{con = null; ds = null;  tmp = null; }
        }
        #endregion

        #region Distruttore
        /// <summary> Distruttore </summary>
        /// <remarks>Forza lo shutdown del motore di GWM portando al rilascio il servizio. </remarks>
        ~SelectBy()
        {
            if(this.pMapServer!=null)
            {   this.pMapServer.Clear();System.Runtime.InteropServices.Marshal.ReleaseComObject(this.pMapServer);
                this.pMapServer = null; }
            if(this.pMapSrvMngr!=null){System.Runtime.InteropServices.Marshal.ReleaseComObject(this.pMapSrvMngr);}
        }
        #endregion

        #region Operatori
        /// <summary> Selezezione di feature in base ad una relazione di sovrapposizione con una layer
        /// di confronto (filtro).</summary>
        /// <param name="strLayerName">Nome del layer che subisce la selezione.</param>
        /// <param name="rstA">Recordset del layer sui cui eseguire la selezione.</param>
        /// <param name="rstB">recordset del layer da utilizzare nel confronto (layer filtro).</param>
        /// <param name="srOverlapType">Tipo di relazione spaziale.</param>
        /// <param name="outp">recordset prodotto dalla selezione [opzionale].</param>
        /// <returns> Array contenente gli identificatori univoci (le chiavi primare nella relazione) delle feature. </returns>
        public ArrayList Overlap(string strLayerName,PClient.GRecordset rstA, PClient.GRecordset rstB, int srOverlapType, ref
PClient.GRecordset outp)
        {
            this.pSystemErrorMsg = "";this.pGeoEngineErrorMsg = "";
            if(this.setup)
            {
                PPipe.SpatialSubsetPipe ssp = null; PClient.GRecordset t = null;ArrayList temp = null;
                string geofldA = Utils.GetGeometryFieldName(rstA); string geofldB = Utils.GetGeometryFieldName(rstB);
                if(""!=geofldA.Trim())
                    if(""!=geofldB.Trim())
                    {
```

```
                        try
                        {
                            ssp = (SpatialSubsetPipe) this.pMapServer.CreateObject("GeoMedia.SpatialSubsetPipe",ssp);
                            ssp.InputRecordset = (Gdo.GRecordset) rstA;ssp.InputGeometryFieldName = geofldA;
                            ssp.FilterRecordset = (Gdo.GRecordset) rstB;ssp.FilterGeometryFieldName = geofldB;
                            ssp.OutputStatusFieldName = "OutStatusFld";ssp.NotOperator = false;
                            ssp.SpatialOperator =srOverlapType; t = (PClient.GRecordset) ssp.OutputRecordset;
                            if(!t.EOF)
                            {
                                t.MoveLast();t.MoveFirst();temp = new ArrayList(t.RecordCount);
                                string pfield = GetPrimaryFieldName(strLayerName); this.pGeoEngineErrorMsg = pfield;
                                while (!t.EOF)//creazione della lista
                                    {temp.Add(t.GFields[pfield].Value); t.MoveNext();}
                                t.MoveFirst();
                            }
                            outp = t;
                        }
                        catch(Exception e) {this.pSystemErrorMsg = e.ToString();temp = null; }
                        finally{if(null!=ssp) System.Runtime.InteropServices.Marshal.ReleaseComObject(ssp); }
                    }
                    else{temp = null; this.pGeoEngineErrorMsg = "Layer [" + rstB.Name + "] has no geometry field";}
                else temp = null; this.pGeoEngineErrorMsg = "Layer [" + rstA.Name + "] has no geometry field";}
                return temp;
            }
            else{this.pGeoEngineErrorMsg = "MapServer not set."; return null;}
        }
        /// <summary> Selezezione di feature in base alla distanza dalle feature di un layer di confronto (filtro).</summary>
        /// <param name="strLayerName">Nome del layer che subisce la selezione.</param>
        /// <param name="rstA">Recordset del layer sui cui eseguire la selezione.</param>
        /// <param name="rstB">recordset del layer da utilizzare nel confronto (layer filtro).</param>
        /// <param name="dblDistance">Distanza delle feature del layer da quelle del layer filtro.</param>
        /// <param name="outp">recordset prodotto dalla selezione [opzionale].</param>
        /// <returns>Array contenente gli identificatori univoci(le chiavi primare nella relazione) delle feature. </returns>
        public ArrayList Distance(string strLayerName,PClient.GRecordset rstA, PClient.GRecordset rstB, double dblDistance, ref
PClient.GRecordset outp)
        {
            this.pSystemErrorMsg = "";this.pGeoEngineErrorMsg = "";
            if(this.setup)
            {
                PPipe.SpatialSubsetPipe ssp = null; PClient.GRecordset t = null;
                ArrayList temp = null;
                string geofldA = Utils.GetGeometryFieldName(rstA); string geofldB = Utils.GetGeometryFieldName(rstB);
                if(""!=geofldA.Trim())
                    if(""!=geofldB.Trim())
                    {
                        try
                        {
                            ssp = (SpatialSubsetPipe) this.pMapServer.CreateObject("GeoMedia.SpatialSubsetPipe",ssp);
                            ssp.InputRecordset = (Gdo.GRecordset) rstA; ssp.InputGeometryFieldName = geofldA;
                            ssp.FilterRecordset = (Gdo.GRecordset) rstB; ssp.FilterGeometryFieldName = geofldB;
                            ssp.OutputStatusFieldName = "OutStatusFld";ssp.NotOperator = false;
                            PCSS.CoordSystemsMgr csm = (PCSS.CoordSystemsMgr) this.pMapServer.CoordSysMgr;
                            ssp.Distance = dblDistance / csm.CoordSystem.StorageToProjectionScale; csm = null;
                            ssp.SpatialOperator = PPipe.SQConstants.gmsqWithinDistance;t = (PClient.GRecordset) ssp.OutputRecordset;
                            if(!t.EOF)
                            {
                                this.pGeoEngineErrorMsg = t.GFields["OutStatusFld"].Value.ToString();
                                temp = new ArrayList(t.RecordCount);string pfield = GetPrimaryFieldName(strLayerName);
                                t.MoveFirst();
                                while (!t.EOF) {temp.Add(t.GFields[pfield].Value); t.MoveNext();}
                                t.MoveFirst();  outp = t;
                            }
                            Else this.pGeoEngineErrorMsg = "Empty selection.";
                            t = null;
                        }
                        catch(Exception e) {this.pSystemErrorMsg = e.ToString();temp = null; }
                        finally{if(null!=ssp) System.Runtime.InteropServices.Marshal.ReleaseComObject(ssp); }
                    }
                    else{temp = null; this.pGeoEngineErrorMsg = "Layer [" + rstB.Name + "] has no geometry field";}
                else
                {temp = null; this.pGeoEngineErrorMsg = "Layer [" + rstA.Name + "] has no geometry field";}
                return temp;
            }
            else{this.pGeoEngineErrorMsg = "MapServer not set."; return null;}
        }
        #endregion

        /// <summary>Dato il nome di un layer restituisce il campo che costituisce la chiave primaria della relazione. </summary>
        /// <param name="strLyrName">Nome del layer nel datasource.</param>
        /// <returns>Nome del campo; se non esiste, "".</returns>
        private string GetPrimaryFieldName(string strLyrName)
        {
            PClient.Connection con = null; MapSrvLib.GWMDataSource ds = null; MapSrvLib.GWMDispatchCollection tmp = null;
            Gdo.GTableDef td = null; string t = "";
            if (this.setup)
            {
                if(null==this.pDb)
                {
                    tmp = (MapSrvLib.GWMDispatchCollection) this.pMapServer.DataSources;
                    ds = (MapSrvLib.GWMDataSource) tmp[1]; con =(PClient.Connection) ds.Connection;
                    this.pDb = (Gdo.GDatabase) ds.Database; con = null; ds = null; tmp = null;
                }
                if(null!=this.pDb.GTableDefs[strLyrName])
                {
                    td = this.pDb.GTableDefs[strLyrName];
                    if(null!=td.GIndexes)
                    {   int i=0;
                        while (i<td.GIndexes.Count && (!td.GIndexes[i].Primary)) i++;
                        if(i<td.GIndexes.Count) t= td.GIndexes[i].GFields[0].Name;
                        td = null;}
                }
            }
            return t;
        }
    }
}
```

C.3. Codice Namespace GWMEngineInterface.Service

```
using System;
using MapSrvLib = MAPSRVLib;
using MapSrvMngrLib = MAPSVRMNGRLib;
using MapSrvComponentsLib = MAPSRVCOMPONENTSLib;
using GenericLayerIdentifier;
using Gdo = GDO;
using myRGBclass;

///<Author>Giulio Massei</Author>
///<LastUpdate>23/11/2005</LastUpdate>

namespace GWMEngineInterface.Service
{
    /// <summary>Data una lista di temi, permette di reastituire una mappa in vari formati grafici.</summary>
    /// <remarks>I tipi utilizzati afferiscono alle classi ReasuetResponseFormalizer eSimpleLayerIdentifier. </remarks>
    public class MapRender: GWMEngineInterface.CommonInterfaceElements
    {
        /// <summary>Nasconde il metodo ereditato da CommonInterfaceElements
</summary>
        new private bool DataAccess(LayerIdentifier layerA,ref PClient.GRecordset A, LayerIdentifier layerB, ref PClient.GRecordset
B, BoundingBox InterestWindow, ref string msg){return false;}

        #region Costruttori
        /// <summary>Costruttore predefinito. </summary>public MapRender(){this.setup = false; }

        /// <summary>Costruttore parametrico, abilita il motore di GWM tramite il puntatore all'oggetto server. </summary>
        /// <param name="Server">Oggetto server di ASP.NET.</param>
        /// <remarks>In caso di errori inizializza le proprietà contenenti i messaggi di errore. </remarks>
        public MapRender(System.Web.HttpServerUtility Server)
        {
            try
            {
                this.pMapSrvMngr = (MapSrvMngrLib.MapServerManagerClass) Server.CreateObject("GMWebMap.MapServerManager");
                this.pMapServer=(MapSrvLib.GWMMapServerHelper)this.pMapSrvMngr.MapServer("");this.pMapServer.Clear();this.setup=true;
            }
            catch(Exception e)
                {this.pSystemErrorMsg=e.ToString();this.pGeoEngineErrorMsg="Unable to turn on GWM-GIS engine.";this.setup=false;}
        }

        /// <summary>Costruttore basato su MapServer. </summary>
        /// <param name="mapSrv">Oggetto MapServer legato ai dati.</param>
        public MapRender(MapSrvLib.GWMMapServerHelper mapSrv) {this.setup = (null!=mapSrv); this.pMapServer = mapSrv;
}

        /// <summary> Distruttore </summary>
        /// <remarks> Forza lo shutdown del motore di GWM portando al rilascio il servizio. </remarks>
        ~MapRender()
        {
            if(this.pMapServer!=null)
            {this.pMapServer.Clear();System.Runtime.InteropServices.Marshal.ReleaseComObject(this.pMapServer);
                this.pMapServer = null; }
            if(this.pMapSrvMngr!=null){ System.Runtime.InteropServices.Marshal.ReleaseComObject(this.pMapSrvMngr);}
        }

        #region Metodi Pubblici
        /// <summary>Data una lista di temi effettua la restituzione della mappa secondo und dato formato. </summary>
        /// <param name="themes">Lista di temi.</param>
        /// <param name="SRSpath">File completamente qualificato inerente al sistema di riferimento.</param>
        /// <param name="InterestWindow">Area di interesse dei dati.</param>
        /// <param name="height">Altezza in pixel della mappa.</param>
        /// <param name="width">Larghezza in pixel della mappa.</param>
        /// <param name="outputFormat">Formato grafico della mappa, SVG, JPG, PNG</param>
        /// <returns>URL parziale del documento prodotto.</returns>
        public string RenderMap(ThemeList themes, string SRSpath, BoundingBox InterestWindow,int width, int height, int outputFormat)
        {
            int i=0,f=0,j=0; bool ext = false, err = false; string t = "";string msg ="";BoundingBox MBR = null;
            ThemeList frgmt = null; MapSrvLib.CGWMRangeScale oAtScl = null;
            if (null!=this.pMapServer)
            {
                try
                {
                    string csfile = SRSpath;
                    //nel caso non vi sia l'estensione del file
                    if(csfile.ToLower().LastIndexOf(".csf")<0) csfile += ".csf";
                    if(!System.IO.File.Exists(csfile))//controllo esistenza file del sistema di riferimento
                    {this.pGeoEngineErrorMsg = SRSpath + " not recognized";t = "";}
                    else
                    {
                        this.pMapServer.SetCoordinateSystem(csfile);themes.SortBySource();
                        #region Definizione AutoScale
                        oAtScl = (MapSrvLib.CGWMRangeScale) this.pMapServer.AddRangeAutoScale("AutoScale");
                        oAtScl.Multiplier = 1;
                        #endregion
                        this.pMapServer.PrimaryKeyAsElementID = 1;
                        //Se non è definta un'area di interesse si utilizza la massima
                        if(null==InterestWindow) MBR = new BoundingBox();
                        else MBR = InterestWindow;
                        j=0;
                        while (j<themes.Count && !err)
                        {
                            //ricerca frammento di lista contenenti temi con stessa sorgente dati.
                            while(j<themes.Count && !ext)
                                if(!themes[j].SameSource(themes[i]))  ext = true;
                                else j++;
                            f = j-1;//se j-- non esco dal while primario
                            frgmt = themes.GetRange(i,f);i = j;
                            if(!this.MapPrepare(frgmt,ref MBR, ref msg))
                                {this.pGeoEngineErrorMsg = "Map rendering error: " + msg;err = true;}
                            ext=false;
                        }
                        if(!err)// non si è verificato alcun errore di pre-rendering
```

```
                        {
                            //Se è definta un'area di interesse la si utilizza.
                            if(null!=InterestWindow) MBR = InterestWindow;
                            this.pMapServer.SVGFileSaveMode = 1; this.pMapServer.PersistenceMode = 0;
                            this.pMapServer.BackgroundAlpha = 255; this.pMapServer.OutputType= outputFormat;
                            this.pMapServer.CreateMapByRange(MBR.xmin,MBR.ymin,MBR.xmax,MBR.ymax,width,height,"AutoScale");
                            t = this.pMapServer.OutputFile;
                        }
                    }
                }
                #region Catch & Finally
                catch(Exception e) {this.pSystemErrorMsg = e.ToString();}
                finally{this.ShutDownGeoEngine();}
                #endregion
            }
            Else this.pGeoEngineErrorMsg ="Unable to startup GWM-GeoEngine";
            return t;
        }
        #endregion

        #region Metodi Privati
        /// <summary>Prepara la porzione di mappa composta da temi appartenenti alla stessasorgente dati. </summary>
        /// <param name="themes">Lista di temi appartenenti alla stessa sorgente dati.</param>
        /// <param name="Mbbox">MBR massimo fra quelli del frammento.</param>
        /// <remarks>Se il parametro °Mbbox° è nullo, non viene calcolato lo MBR massimo.</remarks>
        /// <param name="msg">Messaggio di errore di sistema.</param>
        /// <returns>"True" se l'operazione ha avuto successo.</returns>
        private bool MapPrepare(ThemeList themes, ref BoundingBox Mbbox, ref string msg)
        {
            int i = 0; bool err = false; BoundingBox t = null; string dataSource = "";string additionalInfo = "";
            string dataSourceType = themes[0].dataSourceType;
            if(dataSourceType.ToUpper().IndexOf("WFS")>-1)
            {
                additionalInfo = @"CSFFOLDERPATH=C:\Program Files\GeoMedia WebMap Professional\CoordSystems;";
                additionalInfo += "SWAPGEOCOORDS=FALSE;SWAPPROJCOORDS=FALSE;";
                additionalInfo += "URI=" + themes[0].dataSource + ";";dataSource = "dummyWFS" + System.DateTime.Now.Ticks.ToString();
            }
            if(dataSourceType.ToUpper().IndexOf("WMS")>-1)
            {   additionalInfo = @"NOCSFOUND=FAIL;";additionalInfo += "URI=" + themes[0].dataSource + ";";
                dataSource = "dummyWMS" + System.DateTime.Now.Ticks.ToString();}
            else{dataSource   = themes[0].dataSource;additionalInfo = themes[0].additionalInfo;}
            string currConn = dataSource.GetHashCode().ToString() ;
            try
            {
                string pdataSourceType = "";
                pdataSourceType = dataSourceType + (dataSourceType.IndexOf(".GDatabase")<0? ".GDatabase" : "");
                this.pMapServer.Connect(pdataSourceType,dataSource,additionalInfo,currConn);
                for(i=0;i<themes.Count && (!err);i++)
                {
                    t = AddTheme(themes[i],currConn, ref msg);
                    if(null!=t)//Se non ci sono stati errori, si calcola lo MBR massimo.
                    {if(null!=Mbbox)//Se non esiste un'area di interesse, viene calcolato lo MBR massimo.
                        {if(t>Mbbox) Mbbox = t;}
                    Else err = true;
                }
                return !err;
            }
            catch(Exception e){msg += "MapPrepare::" + e.ToString();return false;}
        }
        /// <summary>Crea il tema all'interno del mapserver. </summary>
        /// <param name="thm">Descrittore del tema</param>
        /// <param name="strCon">nome delle cannoessione a cui appartiene il tema.</param>
        /// <param name="msg">Messaggio di errore di sistema.</param>
        /// <returns>Lo MBR del thema in caso positivo, altrimenti "null" ed i parametro "msg" viene inizializzato.</returns>
        private BoundingBox AddTheme(MapThemeDescriptor thm, string strCon, ref string msg)
        {
            MapSrvComponentsLib.GWMFeatureSymbology oFSymb = null;
            MapSrvComponentsLib.GWMFeatureSymbology oss = null;
            BoundingBox t = null;
            try
            {
                MapSrvLib.GWMQuery oQry=(MapSrvLib.GWMQuery)this.pMapServer.AddQuery(thm.layerName.Trim(),strCon,
thm.layerName.Trim());
                GDO.GRecordset tmp = (GDO.GRecordset) oQry.Recordset;tmp.MoveFirst();int i = tmp.RecordCount;
                MapSrvLib.GWMDisplayRule oDr = (MapSrvLib.GWMDisplayRule) oQry.NewDisplayRule();
                oDr.Priority = thm.Level;//Livello del tema nella mappa. Più la priorità è piccola più il tema è di alto live
                oDr.SheetName = thm.layerName;//Nome del tema nella mappa
                oFSymb = (MapSrvComponentsLib.GWMFeatureSymbology)this.pMapServer.CreateObject("GMWebMap.GWMFeatureSymbology",oFSymb);
                if(thm.dataSourceType.ToUpper().IndexOf("WMS")==-1)
                {
                    myRGBclass.RGBUtils color = new myRGBclass.RGBUtils();
                    Random rnd = new Random((int) System.DateTime.Now.Ticks);int tmpFc,tmpBc,tmpBw;
                    tmpFc = color.FromRGB((byte)rnd.Next(255),(byte)rnd.Next(255),(byte)rnd.Next(255));
                    tmpBc = color.FromRGB((byte)rnd.Next(255),(byte)rnd.Next(255),(byte)rnd.Next(255));tmpBw = 1;
                    if(null!=thm.Style)
                    {
                        if(""!=thm.Style.FillColor)  tmpFc = color.FromHEX(thm.Style.FillColor);
                        if(""!=thm.Style.BorderColor) tmpBc = color.FromHEX(thm.Style.BorderColor);
                        if(0!=thm.Style.BorderWidth) tmpBw = thm.Style.BorderWidth;
                    }//aspetto del tratto. 0= continuo.
                    oFSymb.FillColor = tmpFc; oFSymb.Color = tmpBc; oFSymb.Weight = tmpBw; oFSymb.Style = 0;
                    rnd = null; color = null;
                }
                    oss = (MapSrvComponentsLib.GWMFeatureSymbology) this.pMapServer.CreateObject("GMWebMap.GWMFeatureSymbology",oss);
                    oss = Utils.DoHotSpotSymbology(oss);
                #endregion
                oDr.DisplaySymbology = oFSymb; oDr.HotspotSymbology = oss;
                t = new BoundingBox();Array v = (Array) oQry.Range;//"v" è una struttura di 'comodo'.
                t.xmin = (double)v.GetValue(0); t.ymin = (double)v.GetValue(1); t.xmax = (double)v.GetValue(2);
                t.ymax = (double)v.GetValue(3); v = null;
            }
            catch(Exception e) {msg += "AddTheme::" + e.ToString();t = null; }
            finally{if(null!=oFSymb) {System.Runtime.InteropServices.Marshal.ReleaseComObject(oFSymb);oFSymb = null;}}
            return t;
        }
}
```

```
using System;
//******** Namespace di GWM ********
using MapSrvLib = MAPSRVLib;
using MapSrvMngrLib = MAPSVRMNGRLib;
using Gdo = GDO;
using ExportToShapefile;
using ExportToGML;
using GenericLayerIdentifier;

///<Author>Giulio Massei</Author>
///<LastUpdate>20/10/2005</LastUpdate>

namespace GWMEngineInterface.Service
{
    /// <summary>Dato un recordset, permette di materializzarlo in vari formati. Definice la "Interfaccia-GWM" destinata alla
    ///materializzazione dei recordset. </summary>
    /// <remarks> Nel caso di esportazione in formato "shape file", vengono creati i tre file: geometria (.shp), relazione tabellare
    /// (.dbf) e indice (.shx). </remarks>
    public class Export: CommonInterfaceElements
    {
        /// <summary>Nasconde il metodo ereditato da CommonInterfaceElements</summary>
        new private bool DataAccess(LayerIdentifier layerA,ref PClient.GRecordset A, LayerIdentifier layerB, ref PClient.GRecordset
B, BoundingBox InterestWindow, ref string msg) {return false; }

        #region Costruttori
        /// <summary>Costruttore predefinito. </summary>
        public Export(){this.setup = false; }

        /// <summary>Costruttore parametrico, abilita il motore di GWM tramite il puntatore all'oggetto server.</summary>
        /// <param name="Server">Oggetto server di ASP.NET.</param>
        /// <remarks> In caso di errori inizializza le proprietà contenenti i messaggi di errore.</remarks>
        public Export(System.Web.HttpServerUtility Server)
        {
            try
            {
                this.pMapSrvMngr = (MapSrvMngrLib.MapServerManagerClass) Server.CreateObject("GMWebMap.MapServerManager");
                this.pMapServer = (MapSrvLib.GWMMapServerHelper) this.pMapSrvMngr.MapServer("");
                this.pMapServer.Clear();this.setup = true;
            }
            catch(Exception e)
            {
                this.pSystemErrorMsg = e.ToString();this.pGeoEngineErrorMsg = "Unable to turn on GWM-GIS engine."; this.setup = false;
            }
        }

        /// <summary>Costruttore basato su MapServer. </summary>
        /// <param name="mapSrv">Oggetto MapServer legato ai dati.</param>
        public Export(MapSrvLib.GWMMapServerHelper mapSrv)
        {this.setup = (null!=mapSrv); this.pMapServer = mapSrv; }
        #endregion

        #region Distruttore
        /// <summary> Distruttore</summary>
        /// <remarks>Forza lo shutdown del motore di GWM portando al rilascio il servizio. </remarks>
        ~Export()
        {
            if(this.pMapServer!=null)
            {
                this.pMapServer.Clear();System.Runtime.InteropServices.Marshal.ReleaseComObject(this.pMapServer);
                this.pMapServer = null;
            }
            if(this.pMapSrvMngr!=null){System.Runtime.InteropServices.Marshal.ReleaseComObject(this.pMapSrvMngr);}
        }
        #endregion

        #region Materializzazione in Shape File
        /// <summary>ExportShapeFileReVerse: esportazione di un recordset in Shape File.</summary>
        /// <remarks>Il nome della campo geometria è cercato da destra verso sinistra nella collezione dei campi della relazione.
        /// </remarks>
        /// <param name="rst">Recordset di input.</param>
        /// <param name="shpName">Nome dei tre file.</param>
        /// <param name="strPath">Percorso dove materializzare i tre file.</param>
        /// <returns>"true" se la materializzazione è avvenuta con successo.</returns>
        public bool ExportToShapeFileRV(Gdo.GRecordset rst, string shpName, string strPath)
        {
            this.pSystemErrorMsg = "";this.pGeoEngineErrorMsg = "";ExportToShapeFileService exp = null;bool t = false;
            try
            {
                exp = (ExportToShapeFileService) this.pMapServer.CreateObject("GeoMedia.ExportToShapeFileService",exp);
                exp.InputRecordset = rst;GDO.GFields flds = (Gdo.GFields) rst.GFields;
                int i=flds.Count-1;
                while(i>=0 && flds[i].Type!=PClient.GConstants.gdbSpatial)i--;
                exp.InputGeometryFieldName = flds[i].Name;string tmp = flds[i].Name;int c = flds[i].SubType;
                exp.OutputGeometrySubtype = ((3==flds[i].SubType)? 2 : flds[i].SubType);
                exp.OutputFileName= strPath + shpName;exp.Execute();t = true;
            }
            catch(Exception exc){this.pSystemErrorMsg = exc.ToString();t = false;}
            finally{if(exp!=null)System.Runtime.InteropServices.Marshal.ReleaseComObject(exp);}
            return t;
        }

        /// <summary>ExportShapeFileForWarD: esportazione di un recordset in Shape File.</summary>
        /// <remarks>
        /// Il nome della campo geometria è cercato da sinistra verso destra nella collezione dei campi della relazione.</remarks>
        /// <param name="rst">Recordset di input.</param>
        /// <param name="shpName">Nome dei tre file.</param>
        /// <param name="strPath">Percorso dove materializzare i tre file.</param>
        /// <returns>"true" se la materializzazione è avvenuta con successo.</returns>
        public bool ExportToShapeFileFWD(Gdo.GRecordset rst, string shpName, string strPath)
        {
            this.pSystemErrorMsg = "";this.pGeoEngineErrorMsg = "";bool t = false;
            ExportToShapeFileService exp = null;
            try
            {
                exp = (ExportToShapeFileService) this.pMapServer.CreateObject("GeoMedia.ExportToShapeFileService",exp);
                exp.InputRecordset = rst;GDO.GFields flds = (Gdo.GFields) rst.GFields;int i=0;
                while(i<flds.Count && flds[i].Type!=PClient.GConstants.gdbSpatial)i++;
```

```
                exp.InputGeometryFieldName = flds[i].Name;exp.OutputGeometrySubtype = flds[i].SubType;
                exp.OutputFileName= strPath + shpName;exp.Execute();t = true;
            }
            catch(Exception exc){this.pSystemErrorMsg = exc.ToString();t = false;}
            finally{if(exp!=null)System.Runtime.InteropServices.Marshal.ReleaseComObject(exp);}
            return t;
        }

        /// <summary>esportazione di un recordset in Shape File.</summary>
        /// <param name="rst">Recordset di input.</param>
        /// <param name="strGeometryField">Nome del campo che rappresenta la geomtria.</param>
        /// <param name="shpName">Nome dei tre file.</param>
        /// <param name="strPath">Percorso dove materializzare i tre file.</param>
        /// <returns>"true" se la materializzazione è avvenuta con successo.</returns>
        public bool ExportToShapeFile(Gdo.GRecordset rst, string strGeometryField, string shpName, string strPath)
        {
            this.pSystemErrorMsg = "";this.pGeoEngineErrorMsg = "";ExportToShapeFileService exp = null;bool t = false;
            try
            {
                exp = (ExportToShapeFileService) this.pMapServer.CreateObject("GeoMedia.ExportToShapeFileService",exp);
                exp.InputRecordset = rst;GDO.GFields flds = (Gdo.GFields) rst.GFields;
                int i=0;
                while(i<flds.Count && flds[i].Name.ToLower()!=strGeometryField.ToLower())i++;
                if(i>flds.Count)
                {this.pGeoEngineErrorMsg = "Field [" + strGeometryField + "] not exists.";t = false;}
                else
                {
                    exp.InputGeometryFieldName = flds[i].Name;exp.OutputGeometrySubtype = flds[i].SubType;
                    exp.OutputFileName= strPath + shpName;exp.Execute();t = true;
                }
            }
            catch(Exception exc){this.pSystemErrorMsg = exc.ToString();}
            finally{if(exp!=null)System.Runtime.InteropServices.Marshal.ReleaseComObject(exp);}
            return t;
        }
        #endregion

        #region Materializzazione in GML
        /// <summary>Esporta un recordset in GML con relativo GML application schema</summary>
        /// <param name="rst">Recordset da materializzare</param>
        /// <param name="lyrName">nome del recordset</param>
        /// <param name="strPath">percorso dove materializzare i dati</param>
        /// <returns>
        /// True se operazione eseguita con successo.il file è accessibile componendo strPath con lyrName aggiungendo ".gml"
        /// e ".xsd" per lo application schema.</returns>
        public bool ExportToGML(Gdo.GRecordset rst, string lyrName, string strPath)
        {
            ExportToGML.ExportToGMLService exp = null;
            PDBPipe.GRecordset trst = null;
            PDBPipe.Query tmpQry = null;
            bool t = false;
            try
            {
                exp = (ExportToGMLService) this.pMapServer.CreateObject("GeoMedia.ExportToGMLService",exp);
                PDBPipe.SchemaProjectionPipe spp = null;
                spp = (PDBPipe.SchemaProjectionPipe) this.pMapServer.CreateObject("GeoMedia.SchemaProjectionPipe",spp);
                spp.InputRecordset = (PDBPipe.GRecordset) rst;
                spp.FieldList = null;

                PDBPipe.FieldDefinition fieldDef = spp.FieldDefinitions.Add("ID");
                PDBPipe.ExtendedPropertySet fieldExt = (PDBPipe.ExtendedPropertySet) fieldDef.GetExtension("ExtendedPropertySet");
                fieldExt.SetValue("Key",true);
                fieldExt.SetValue("Displayable",true);

                trst = spp.OutputRecordset;
                tmpQry = (PDBPipe.Query) trst.GetExtension("Name");
                tmpQry.Name = lyrName;

                exp.InputRecordsets = trst;
                exp.SRSNameAttribute = "EPSG:4326";
                exp.SwapCoordOrder = false;
                exp.OutputDataTarget = strPath + lyrName + ".gml";
                exp.OutputSchemaTarget = strPath + lyrName + ".xsd";
                exp.SchemaLocationAttribute = "http://webgisserver.isti.cnr.it";
                exp.Execute();
                t = true;
            }
            catch(Exception exc)
            {
                t = false;
                this.pSystemErrorMsg = exc.ToString();
            }
            finally
            {
                if(exp!=null)System.Runtime.InteropServices.Marshal.ReleaseComObject(exp);
            }
            return t;
        }
        #endregion
    }
}
```

```
using System;
//************ Namespace di GWM **************
using PClient;
using Gdo = GDO;
using MapSrvLib = MAPSRVLib;
using MapSrvMngrLib = MAPSVRMNGRLib;
using MapSrvComponentsLib = MAPSRVCOMPONENTSLib;
//******* Namespace di librerie interne ********
using StyleDescriptor;
using GenericLayerIdentifier;

///<Author>Giulio Massei</Author>
///<LastUpdate>11/11/2005</LastUpdate>

namespace GWMEngineInterface.Service
{
    /// <summary>Permette di restituire un recordet, contenentedati geografici, in veste grafica.</summary>
    public class LayerRender : CommonInterfaceElements
    {
        #region Proprietà pubbliche
        private SimpleStyleDescriptor pStyle = null;private BoundingBox pInterestArea = null;
        private MapSrvLib.GRecordset pRecordset = null;private int pGraphicThemeFormat = 4;// Svg valore predefinito
        private string pOutputFile = "";int pWidth = 0, pHeight = 0;

        /// <summary>Stile di visualizzazione del layer</summary>
        public SimpleStyleDescriptor Style
        {
            set{if(null!= value) this.pStyle = value;}
            get{return this.pStyle;}
        }
        /// <summary>Area di interesse in cui restituire i dati</summary>
        public BoundingBox InterestArea
        {
            set
            {
                BoundingBox t = value;
                if (null==t) this.InterestArea = null;
                else if(!t.isZero) {this.pInterestArea = t; t = null;}
                     else this.pInterestArea = null;
            }
            get{return this.pInterestArea;}
        }
        /// <summary>Recordset di input.</summary>
        public PClient.GRecordset InputRecordset
        {
            set{if(null!=value){this.pRecordset = (MapSrvLib.GRecordset) value;}}
            get
            {
                if(null!=this.pRecordset)
                    return (PClient.GRecordset) this.pRecordset;
                else
                    return null;
            }
        }
        /// <summary>Formato grafico in cui restituire il layer.</summary>
        /// <remarks>Sono le costanti intere definite per GWM</remarks>
        public int GraphicThemeFormat
        {
            set{if ((value>0 && value<5) || (value==32)) this.pGraphicThemeFormat = value;}
            get{return this.pGraphicThemeFormat;}
        }
        /// <summary>Percorso del file grafico prodotto.</summary>
        public string OutputFile{get{return this.pOutputFile;}}
        /// <summary>Alrghezza in pixel della mappa.</summary>
        public int Width
        {
            set{this.pWidth = value;}
            get{return this.pWidth;}
        }
        /// <summary>Altezza della mappa in pixel.</summary>
        public int Height
        {
            set{this.pHeight = value;}
            get{return this.pHeight;}
        }
        #endregion

        #region Costruttore
        /// <summary>Costruttore basato su MapServer.</summary>
        /// <param name="mapSrv">Oggetto MapServer legato ai dati.</param>
        public LayerRender(MAPSRVLib.GWMMapServerHelper mapSrv){this.setup = (null!=mapSrv);this.pMapServer = mapSrv;}
        #endregion

        #region Metodi pubblici
        public bool Render(string layerName)
        {
            MapSrvComponentsLib.GWMFeatureSymbology oFSymb = null;MapSrvComponentsLib.GWMFeatureSymbology oss = null;
            bool res = false;
            if(this.setup)
            {
                try
                {
                    MapSrvLib.CGWMRangeScale oAtScl = null;
                    if(!System.IO.File.Exists(this.srsPathName))//constrollo esistenza file del sistema di riferimento
                    {this.pGeoEngineErrorMsg = this.srsPathName + " not recognized";res = false;}
                    else
                    {
                        this.pMapServer.SetCoordinateSystem(this.srsPathName);
                        string geometryname =
Utils.GetGeometryFieldNameFromRight((PClient.GFields)this.pRecordset.GFields);
                        if("" != geometryname)
                        {
                            #region Definizione AutoScale
                            oAtScl = (MapSrvLib.CGWMRangeScale) this.pMapServer.AddRangeAutoScale("AutoScale");
                            oAtScl.Multiplier = 1;
```

```
                            MapSrvLib.GWMMarkerSystem mrkr = (MapSrvLib.GWMMarkerSystem)
this.pMapServer.AddRecordsetMarker(layerName,this.pRecordset,geometryname);
                            MapSrvLib.GWMDisplayRule oDr = (MapSrvLib.GWMDisplayRule) mrkr.NewDisplayRule();
                            oDr.SheetName = layerName;
                            oFSymb = (MapSrvComponentsLib.GWMFeatureSymbology)
this.pMapServer.CreateObject("GMWebMap.GWMFeatureSymbology",oFSymb);
                                oss = (MapSrvComponentsLib.GWMFeatureSymbology)
this.pMapServer.CreateObject("GMWebMap.GWMFeatureSymbology",oss);
                                oss = Utils.DoHotSpotSymbology(oss);
                            myRGBclass.RGBUtils rgb = new myRGBclass.RGBUtils();
                            if(null != this.pStyle)
                            {
                                oFSymb.FillColor = rgb.FromHEX(this.pStyle.FillColor);
                                oFSymb.Color = rgb.FromHEX(this.pStyle.BorderColor);oFSymb.Weight =
this.pStyle.BorderWidth;
                            }
                            else
                            {
                                this.pStyle = new SimpleStyleDescriptor();
                                rgb.RandomRGB();
                                oFSymb.FillColor = rgb.FromRGB();this.pStyle.FillColor = rgb.RGB2Hex();
                                do
                                {
                                    rgb.RandomRGB();rgb.RandomRGB();oFSymb.Color = rgb.FromRGB();
                                    this.pStyle.BorderColor = rgb.RGB2Hex();
                                }
                                while(oFSymb.Color==oFSymb.FillColor);
                                oFSymb.Weight = 1;this.pStyle.BorderWidth = 1;
                            }
                            rgb = null;oDr.DisplaySymbology = oFSymb;oDr.HotspotSymbology = oss;
                            if(null == this.pInterestArea)//Se l'area di interesse non esiste
                            {
                                GDO.GRecordset gg= (GDO.GRecordset)mrkr.Recordset;int jj= gg.RecordCount;
                                this.pInterestArea = new BoundingBox();object[] rng = new object[4];
                                rng = (object[]) mrkr.Range;rng = null;
                            }
                            this.pMapServer.SVGFileSaveMode = 1;this.pMapServer.PersistenceMode = 0;
                            this.pMapServer.OutputType  = this.pGraphicThemeFormat;//formato della mappa in output
secondole costanti
                            this.pMapServer.CreateMapByRange(this.pInterestArea.xmin,this.pInterestArea.ymin,
                                                             this.pInterestArea.xmax,this.pInterestArea.ymax,
                                                              this.pWidth,this.pHeight,
                                                              "AutoScale");
                            this.pOutputFile =  this.pMapServer.OutputFile;
                            if(this.pOutputFile.ToLower().IndexOf("isti.cnr.it")==-1)
                                    this.pOutputFile =
this.pOutputFile.ToLower().Replace("webgisserver","webgisserver.isti.cnr.it");
                            res = true;
                        }
                        Else this.pGeoEngineErrorMsg = "Unable to locate the geomtry attribute";
                    }
                }
                catch(Exception e){res = false; this.pSystemErrorMsg = e.ToString();}
                finally{if(null!=oFSymb) {System.Runtime.InteropServices.Marshal.ReleaseComObject(oFSymb);oFSymb = null;}}
            }
            return res;
        }
        #endregion

        /// <summary>Nasconde il metodo ereditato da CommonInterfaceElements</summary>
        new private bool DataAccess(LayerIdentifier layerA,ref PClient.GRecordset A, LayerIdentifier layerB, ref
PClient.GRecordset B, BoundingBox InterestWindow, ref string msg){return false;}
    }
}
```

Appendice **D**

GIS Operation – Web Service

D.1. Modelli UML delle Classi

Tutti i servizi di seguito presentati, sono accomunati da due metodi che forniscono lo schema XML, codificato in XSD, relativo ai messaggi di richiesta. I metodi sono:

- `GetResponseSchemaURL`: restituisce il riferimento allo schema XML del messaggio di richiesta;
- `GetRequestSchemaURL`: restituisce il riferimento allo schema XML del messaggio di risposta.

Il servizio `BufferingService` è implementato mediante l'omonima classe `BufferingService` (figura D.1) e permette di accedere ad operazioni per creare un'area di rispetto a partire da un layer identificato mediante la classe `LayerIdentifier` (vedere appendice B).

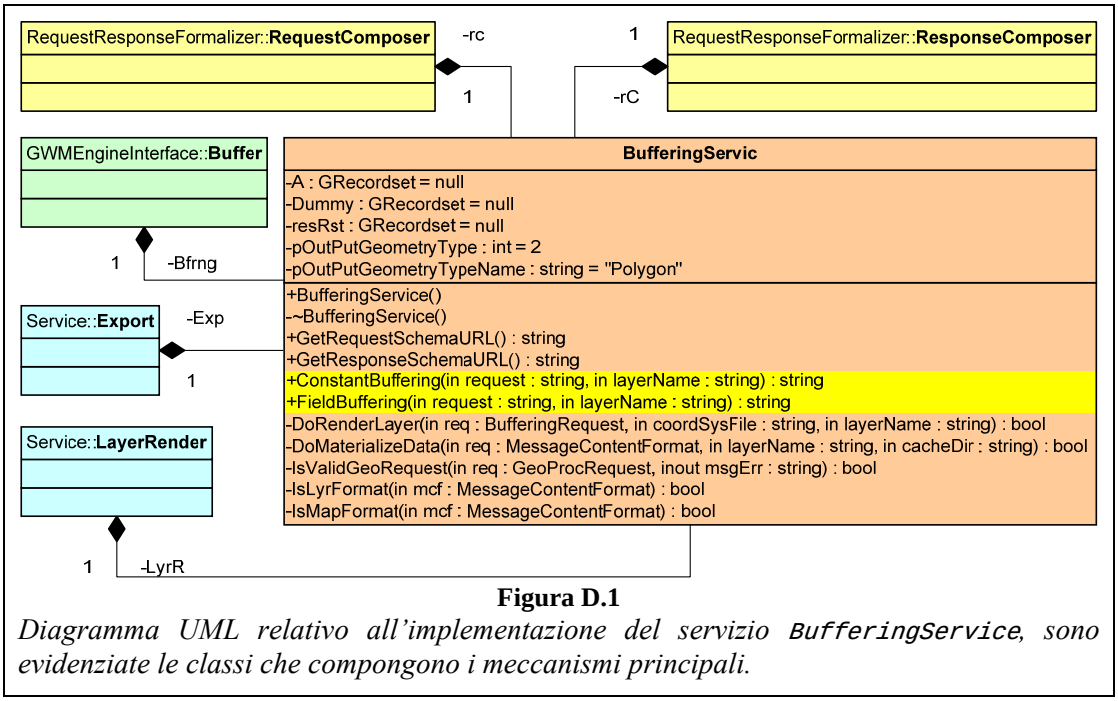


Figura D.1

Diagramma UML relativo all'implementazione del servizio `BufferingService`, sono evidenziate le classi che compongono i meccanismi principali.

I metodi che eseguono le operazioni di buffering, sono:

- `ConstantBuffering`: crea aree di rispetto in base ad un valore costante oppure ad una sequenza d'intervalli costanti e consecutivi, ad esempio "0..100;101..200";
- `FieldBuffering`: crea aree di rispetto in base al valore di un attributo numerico appartenente al layer identificato nella richiesta.

Entrambi i metodi creano un nuovo layer (od un tema in base al formato del contenuto richiesto) a cui viene associato un nome mediante il parametro di input `layerName`; il riferimento (URL) al layer creato, è restituito come output del metodo.

L'implementazione del servizio `GeoProcessingService`, implementato mediante l'omonima classe `GeoProcessingService`, prevede l'impiego della classe `GeoProcessing` (figura D.2).

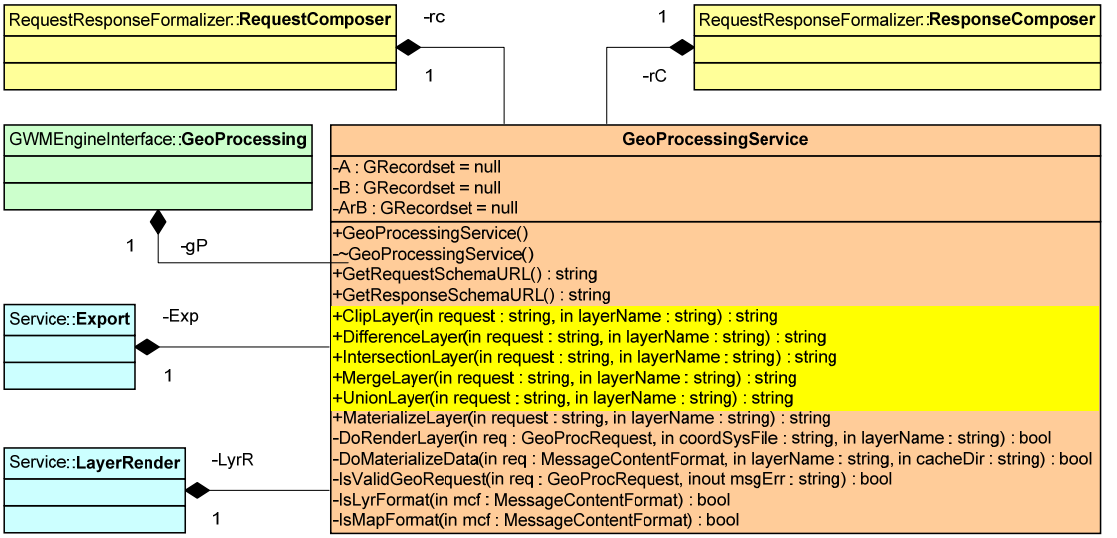


Figura D.2

Diagramma UML delle classi che compongono il servizio `GeoProcessingService`.

La classe `GeoProcessingService` espone i metodi che vanno ad implementare le operazioni di overlay fornite dal servizio; ogni metodo riceve in ingresso la richiesta mediante il parametro `request` ed il nome del layer risultante dall'esecuzione dell'operazione, mediante il parametro `layerName`. Il risultato delle operazioni è costituito dal riferimento (nel caso di shape file sono tre) al documento prodotto mediante URL.

I metodi che implementano le operazioni di overlay operano su due layer (identificati mediante i quattro parametri discussi nel capitolo 4.2.1 e nell'appendice B) e sono:

- `ClipLayer`: esegue l'operazione di Clip;
- `DifferenceLayer`: effettua l'operazione di Differenzae;
- `IntersectLayer`: esegue l'operazione di Intersezione;
- `MergeLayer`: effettua l'operazione di Merge;
- `UnionLayer`: esegue l'operazione di Unione.

Come per il servizio `BufferingService`, è possibile ottenere il risultato dell'operazione materializzato in un formato specifico (attualmente shape file e GML)

tramite l'ausilio della classe `Export`, oppure rappresentato con un documento grafico (ad esempio, informato SVG o PNG) mediante la classe `LayerRender`.

La classe `GeoSelectionService` (figura D.3) implementa il servizio `GeoselectionService` creato per fornire le operazioni di selezione per tema.

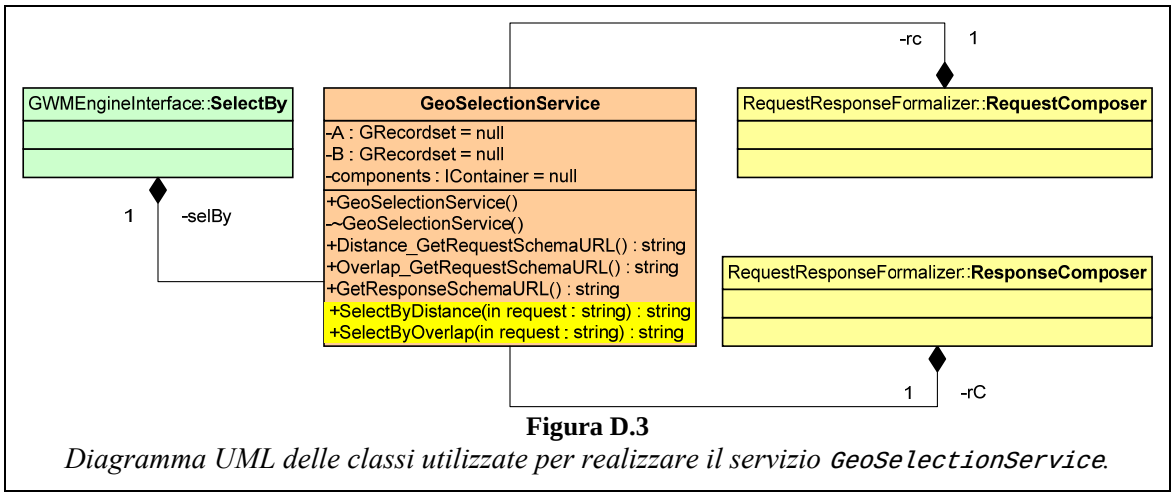


Figura D.3

Diagramma UML delle classi utilizzate per realizzare il servizio `GeoSelectionService`.

I metodi relativi alle operazioni di selezione per tema sono i seguenti:

- `SelectByDistance`: effettua la selezione per tema utilizzando la distanza lineare fra feature, come criterio di confronto spaziale;
- `SelectByOverlap`: selezione per tema in base alla relazione di sovrapposizione fra le feature.

Per come sono state diversificate le due operazioni, sono stati creati due messaggi di richiesta e conseguentemente due schemi XML distinti anche se simili.

Per ottenere i due schemi sono stati creati i seguenti metodi pubblici:

- `Distance_GetRequestSchemaURL`: restituisce lo schema per i messaggi destinati al metodo di selezione per tema basato sulla distanza;
- `Overlap_GetRequestSchemaURL`: restituisce lo schema per i messaggi destinati al metodo di selezione per tema basato su come le feature si sovrappongono.

Il servizio `MapRenderService` permette di creare una mappa da un insieme di temi. La classe `MapRenderService` (figura D.4) implementa il servizio ed espone il metodo `CreatreMap` per effettuare l'operazione di WebMapping.

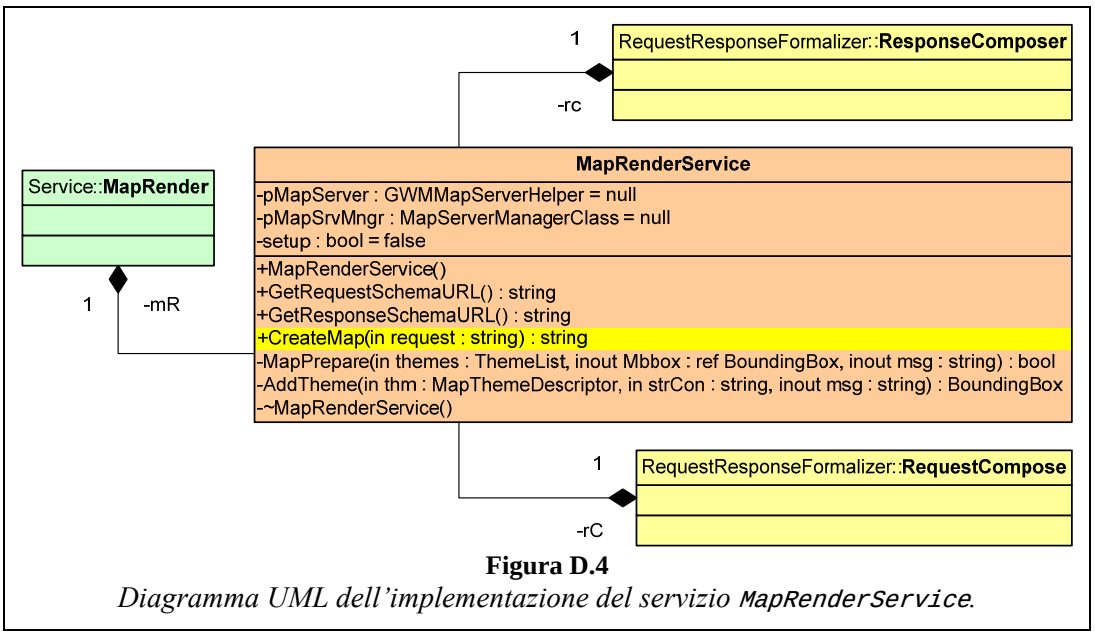


Figura D.4

Diagramma UML dell'implementazione del servizio `MapRenderService`.

Il servizio secondario `SystemCoordTransform`, permette di trasformare una o più coordinate bidimensionali da un sistema di riferimento ad un altro.

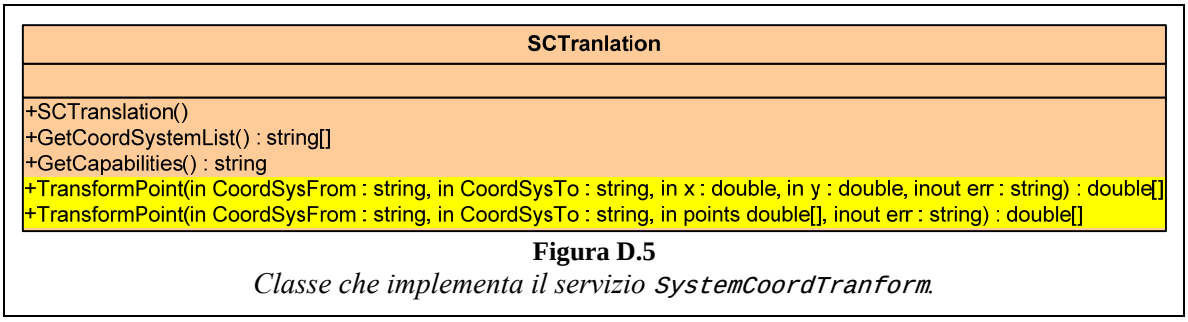


Figura D.5

Classe che implementa il servizio `SystemCoordTranform`.

La classe `SCTranslation` (figura D.5) implementa il servizio ed espone i seguenti metodi:

- `GetCoordSystemList`: restituisce la lista dei sistemi di riferimento gestiti. La lista è codificata mediante un documento XML;
- `GetCapabilities`: restituisce un documento XML in cui sono descritti i sistemi di riferimento utilizzati;
- `TransformPoint`: trasforma un punto bidimensionale, specificato dai parametri `x` e `y` (coordinate del punto), da un sistema di riferimento (indicato dal parametro `coordSysFrom`) ad un altro (indicato dal parametro `coordSysTo`). Il risultato è restituito come lista di coordinate (x,y). In caso di errori la lista sarà nulla ed il parametro `err` conterrà la descrizione dell'errore;
- `TransformPoints`: come il precedente, tranne che accetta una lista di coordinate bidimensionali (del tipo x,y) fornita mediante il parametro `points`.

Il servizio non necessita di messaggi di richiesta, al contrario dei servizi principali, in quanto ha un numero prefissato di parametri ed il loro tipo è definito all'interno dello *XML Schema Part II*.

D.2. Codice Namespace GisOperationDeliverer

```
using System;
using System.Web;
using System.Web.Services;
using System.Data;
using System.Diagnostics;
using System.Collections;
using System.ComponentModel;
//********* Spatial Operations Interface *********
using GWMEngineInterface;
//** Request & Response Formalization Namespace **
using RequestResponseFormalizer;
//********** Geomedia Web Map Namespace ***********
using MapSrvMngrLib = MAPSVRMNGRLib;
using MapSrvLib = MAPSRVLib;

///<Author>Giulio Massei</Author>
///<LastUpdate>15/11/2005</LastUpdate>

namespace GisOperationsDeliverer
{
/// <summary>Data una richiesta XML, permette di creare un buffer attorno alle feture di un layer.</summary>
[WebService(Namespace="http://webgisserver.isti.cnr.it/GisWebService/BufferingService/")]
public class BufferingService : System.Web.Services.WebService
    {
    // Una "Interfaccia-GWM" è una classe che permette di accedere
    //ai meccanismi del motore GIS di GWM.
    #region Attributi privati
    /// <summary>Interfaccia-GWM per il buffering.</summary>
    private GWMEngineInterface.Buffer Bfrng = null;

    private GWMEngineInterface.Service.LayerRender LyrR = null;

    /// <summary>Primo recordset.</summary>
    private PClient.GRecordset A = null;

    /// <summary>Serve solo per correttezza sintattica dei parametri formali.</summary>
    private PClient.GRecordset Dummy = null;

    /// <summary>Recordset contenete il risultato.</summary>
    private PClient.GRecordset resRst = null;

    /// <summary>Interfaccia-GWM di materializzazione dati.</summary>
    private GWMEngineInterface.Service.Export Exp = null;

    private RequestComposer rc = null;

    private ResponseComposer rC = null;

    private int pOutPutGeometryType = 2;

    private string pOutPutGeometryTypeName = "Polygon";
    #endregion

    #region Costruttore
    /// <summary>Costruttore predefinito.</summary>
    public BufferingService(){InitializeComponent();}
    #endregion

    #region Distruttore
    /// <summary>Distruttore</summary>
    /// <remarks>Forza lo shutdown del motore di GWM portando al rilascio il servizio.</remarks>
    ~BufferingService(){if (null!=this.Bfrng){this.Bfrng.ShutDownGeoEngine();this.Bfrng = null;}}
    #endregion

    #region Web Method
    /// <summary>Restituisce lo URL dello schema XSD della richiesta.</summary>
    /// <returns>URL schema XSD.</returns>
    [WebMethod(Description="Returns The XSD request schema document.")]
    public string GetRequestSchemaURL()
    {return Application["XSD_URL"].ToString() + Application["BufferXSD"].ToString();}

    /// <summary>Restituisce lo URL dello schema XSD della risposta.</summary>
    /// <returns>URL schema XSD.</returns>
    [WebMethod(Description="Returns the URL of XSD document of selection by overlap request.")]
    public string GetResponseSchemaURL(){return Application["XSD_URL"].ToString() + Application["SimpleRespXSD"].ToString();}

    /// <summary>Crea un buffer attorno alle feature secondo uno o più valori costanti.Infatti è possibile ottenere buffer a più
    /// fasce.</summary>
    /// <remarks> Per maggiori dettagli vedere la releativa rischiesta XML di buffering.</remarks>
    /// <param name="request">Richiesta XML.</param>
    /// <param name="layerName">Nome dellayer risultante.</param>
    /// <returns>Restituisce un messaggio di risposta.</returns>
    [WebMethod(Description="Performs a Buffering by a costanstant distance value.")]
    public string CostantBuffering(string request, string layerName)
        {
        string res = "";SimpleResponse sr = null;string msg = "";string cachedir = Application["CachePath"].ToString();
           try
            {
            this.Bfrng = new GWMEngineInterface.Buffer(Server);RequestComposer rc = new RequestComposer();
            BufferingRequest req = (BufferingRequest) rc.FormalizeRequestFromXml(request,RequestType.Buffering);
            string csfile = Application["CoordSystemDir"].ToString(); csfile += req.SRSName + ".csf";
            this.Bfrng.srsPathName = csfile;this.Bfrng.NewLayerName = layerName;
            if(this.Bfrng.DataAccess(req.LayerA, ref this.A, null, ref this.Dummy, req.InterestWindow, ref msg))
                {
                this.Dummy = null;
                this.resRst = this.Bfrng.BufferByDistanceConst(this.A,req.distanceCriteria, req.measureUnit, req.edgeType);
                this.pOutPutGeometryType = this.Bfrng.GeometryType;   this.pOutPutGeometryTypeName = this.Bfrng.GeometryTypeName;
                if(null!=this.resRst)
                    {
                    if(this.IsLyrFormat(req.RequestContentFormat))
```

```
                    this.DoMaterializeData(req.RequestContentFormat, layerName,cachedir);
                if(this.IsMapFormat(req.RequestContentFormat))
                    this.DoRenderLayer(req, csfile,layerName);
                }
               else
                {
                 this.rC = new ResponseComposer();sr = (SimpleResponse) this.rC.Response;
                 sr.Body = new ArrayList(3);sr.Body.Insert(0, "Materialization error");
                 sr.Body.Insert(1, this.Exp.GeoEngineErrorMsg);    sr.Body.Insert(2, this.Exp.SystemErrorMsg);
                }
              this.Bfrng.ShutDownGeoEngine();//Rilascio del servizio GWM.
              }
            else
              {
               this.rC = new ResponseComposer(MessageContent.Error,MessageContentFormat.NA_Raw);
               sr = (SimpleResponse) this.rC.Response;
               sr.Body = new ArrayList(3); sr.Body.Insert(0, "Buffering error");
               sr.Body.Insert(1, "");sr.Body.Insert(2,"ConstantBuffering::DataAccess::" + msg);
              }
          }
        catch(Exception ex)
          {
           this.rC = new ResponseComposer(MessageContent.Error,MessageContentFormat.NA_Raw);
           sr = (SimpleResponse) this.rC.Response;sr.Body = new ArrayList(3);
           sr.Body.Insert(0, "Buffering error");sr.Body.Insert(1, "");sr.Body.Insert(2,"ConstantBuffering::" + ex.ToString());
          }
         finally
          {if(this.Bfrng!=null){this.Bfrng.ShutDownGeoEngine();this.Bfrng = null;}
           if(this.Exp!=null){this.Exp.ShutDownGeoEngine();this.Exp = null;}}
        res = this.rC.FormalizeResponseToXml();this.rC = null;
        return res;
        }

    /// <summary>Crea un buffer attorno alle feature secondo il valore di un campo (attributo) numerico del layer.</summary>
    /// <param name="request">Richiesta XML.</param>
    /// <param name="layerName">Nome dellayer risultante.</param>
    /// <returns>Restituisce un messaggio di risposta.</returns>
    [WebMethod(Description="Performs a Buffering by the value of a numeric field (attribute) of the layer.")]
    public string FieldBuffering(string request, string layerName)
        {
        string res = "";SimpleResponse sr = null;       string msg = "";string cachedir = Application["CachePath"].ToString();
           try
            {
             this.Bfrng = new GWMEngineInterface.Buffer(Server);RequestComposer rc = new RequestComposer();
             BufferingRequest req = (BufferingRequest) rc.FormalizeRequestFromXml(request,RequestType.Buffering);
             this.pOutPutGeometryType = this.Bfrng.GeometryType;this.pOutPutGeometryTypeName = this.Bfrng.GeometryTypeName;
             string csfile = Application["CoordSystemDir"].ToString();
             csfile += req.SRSName + ".csf";this.Bfrng.srsPathName = csfile;         this.Bfrng.NewLayerName = layerName;
             if(this.Bfrng.DataAccess(req.LayerA, ref this.A,null, ref this.Dummy,req.InterestWindow, ref msg))
                 {
                 this.Dummy = null;
                 this.resRst = this.Bfrng.BufferByDistanceField(this.A,req.distanceCriteria, req.measureUnit,(int) req.edgeType);
                 if(this.resRst!=null)
                     {
                     if(this.IsLyrFormat(req.RequestContentFormat))
                         this.DoMaterializeData(req.RequestContentFormat, layerName,cachedir);
                     if(this.IsMapFormat(req.RequestContentFormat))
                         this.DoRenderLayer(req, csfile,layerName);
                     }
                   else
                     {
                     this.rC = new ResponseComposer(MessageContent.Error,MessageContentFormat.NA_Raw);
                     sr = (SimpleResponse) this.rC.Response;
                     sr.Body = new ArrayList(3);sr.Body.Insert(0, "Materialization error");
                     sr.Body.Insert(1, this.Exp.GeoEngineErrorMsg);    sr.Body.Insert(2, this.Exp.SystemErrorMsg);
                     }
                 this.Bfrng.ShutDownGeoEngine();
                 }
               else
                 {
                 this.rC = new ResponseComposer(MessageContent.Error,MessageContentFormat.NA_Raw);
                 sr = (SimpleResponse) this.rC.Response;sr.Body = new ArrayList(3);  sr.Body.Insert(0, "Buffering error");
                 sr.Body.Insert(1, "");sr.Body.Insert(2,"FieldBuffering::DataAccess::" + msg);
                 }
            }
        catch(Exception ex)
            {
            this.rC = new ResponseComposer(MessageContent.Error,MessageContentFormat.NA_Raw);
            sr = (SimpleResponse) this.rC.Response;     sr.Body = new ArrayList(3); sr.Body.Insert(0, "Buffering error");
            sr.Body.Insert(1, "");sr.Body.Insert(2,"FieldBuffering::" + ex.ToString());
            }
         finally
            {
            if(this.Bfrng!=null){this.Bfrng.ShutDownGeoEngine();this.Bfrng = null;}
            if(this.Exp!=null){this.Exp.ShutDownGeoEngine();this.Exp = null;}
            }
        #endregion
        res = this.rC.FormalizeResponseToXml();this.rC = null;return res;
        }
    #endregion

    #region Metodi privati
    private bool DoRenderLayer(BufferingRequest req, string coordSysFile, string layerName)
        {
        bool b = true;SimpleResponse sr = null;
           try
            {
             this.LyrR = new GWMEngineInterface.Service.LayerRender(this.Bfrng.MapServer);this.LyrR.Style = req.Style;
             this.LyrR.Width = req.Width;this.LyrR.Height = req.Height;this.LyrR.srsPathName = coordSysFile;
             this.LyrR.GraphicThemeFormat = req.MapFormat();
             if(null != req.InterestWindow) this.LyrR.InterestArea = req.InterestWindow;
             this.LyrR.InputRecordset = this.resRst;
             if(this.LyrR.Render(layerName))
                 {
                 this.rC = new ResponseComposer(MessageContent.Layer,req.RequestContentFormat);
```

```
                sr = (SimpleResponse) this.rC.Response; sr.Body = new ArrayList(2); sr.Body.Insert(0, this.LyrR.OutputFile);
                sr.Body.Insert(1, layerName);
                sr.Body.Insert(2, this.pOutPutGeometryType + "|" +this.LyrR.Style.FillColor  + "|" +
                    this.LyrR.Style.BorderColor + "|" + this.LyrR.Style.BorderWidth );
                }
            else//Gestione fallimento
                {
                this.rC = new ResponseComposer(MessageContent.Error,MessageContentFormat.NA_Raw);
                sr = (SimpleResponse) this.rC.Response; sr.Body = new ArrayList(3); sr.Body.Insert(0, "Materialization error");
                sr.Body.Insert(1, this.Exp.GeoEngineErrorMsg);        sr.Body.Insert(2, this.Exp.SystemErrorMsg);
                }
            }
        catch(Exception e)
            {
            this.rC = new ResponseComposer(MessageContent.Error,MessageContentFormat.NA_Raw);
            sr = (SimpleResponse) this.rC.Response;sr.Body = new ArrayList(3);sr.Body.Insert(0, "Layer Rendering error");
            sr.Body.Insert(1, "");sr.Body.Insert(2, "Layer Rendering::" + e.ToString());
            }
         finally{if(null != this.LyrR)  {this.LyrR.ShutDownGeoEngine();this.LyrR = null;}}
        return b;
        }

    /// <summary>Indica se la richiesta è valida per questo web service.</summary>
    /// <param name="req">oggetto per richiesta geoprocessing</param>
    /// <param name="msgErr">In caso di fallimento, contiene un messaggio di errore descrittivo.</param>
    /// <returns>True se la richiesta è valida per questo web service.</returns>
    private bool IsValidGeoRequest(GeoProcRequest req, ref string msgErr)
        {
        bool validrequest;bool ValidContent = true;
        ValidContent = (req.RequestContent == MessageContent.Layer);  bool ValidContentRequestFrmtLyr = true;
        ValidContentRequestFrmtLyr = (req.RequestContentFormat == MessageContentFormat.Layer_Gml || req.RequestContentFormat ==
MessageContentFormat.Layer_Shp);
        //2.2: Il formato è grafico: richiede esistenda del sistema di riferimento
        bool ValidContentRequestFrmtMap = true;
        ValidContentRequestFrmtMap = (req.RequestContentFormat >= MessageContentFormat.Map_A16 && req.RequestContentFormat <=
MessageContentFormat.Map_A32);
        if(ValidContent)
            {
            if(ValidContentRequestFrmtLyr)  validrequest = true;
            else
                if(ValidContentRequestFrmtMap)
                    if(""!=req.SRSName) validrequest = true;
                    else{validrequest = false;msgErr ="Spatial Reference Sistyem needed.";}
                else{validrequest = false;msgErr ="Invalid request content format.";}
          else{validrequest = false;msgErr ="Invalid request content.";}
        if(""!=msgErr) msgErr = "Invalid Request:" + msgErr;
        return validrequest;
        }

    /// <summary>Indica se il formato del contenuto di una richiesta è un layer damaterializzare.</summary>
    /// <param name="mcf">Formato del contenuto del mesaggio</param>
    /// <returns>True se è un formato per materializzazione.</returns>
    private bool IsLyrFormat(MessageContentFormat mcf)
    {return (mcf == MessageContentFormat.Layer_Gml || mcf == MessageContentFormat.Layer_Shp);}

    /// <summary>Indica se il formato del contenuto di una richiesta è un formato grafico.</summary>
    /// <param name="mcf">Formato del contenuto del mesaggio</param>
    /// <returns>True se è un formato grafico.</returns>
    private bool IsMapFormat(MessageContentFormat mcf)
    {return (mcf >= MessageContentFormat.Map_A16 && mcf <= MessageContentFormat.Map_A32);}
    #endregion
    }
}
```

```
using System;
using System.Web;
using System.Web.Services;
using System.Data;
using System.Diagnostics;
using System.Collections;
using System.ComponentModel;
//********* Geomedia Web Map Namespace ***********
using MapSrvMngrLib = MAPSVRMNGRLib;
using MapSrvLib = MAPSRVLib;
//************* GWM-Engine Intercafe *************
using GWMEngineInterface;
using GWMEngineInterface.Service;
//************** Namespace Interni ***************
using RequestResponseFormalizer;
using GenericLayerIdentifier;

///<Author>Giulio Massei</Author>
///<LastUpdate>23/11/2005</LastUpdate>

namespace GisOperationsDeliverer
{
/// <summary>Data una richiesta XML, permette di effettuare operazioni geospaziali su due layer
// definiti tramite un "Identificatore di Layer."</summary>
// <remarks>Le richiesta e le risposta XML sono definite tramite uno schema XSD ben definito.</remarks>
[WebService(Namespace="http://webgisserver.isti.cnr.it/GisWebService/GeoSpatialOperationsService/")]
public class GeoProcessingService : System.Web.Services.WebService
    {
     #region Attributi privati
     /// <summary>Riferimento a GWM-Engine Interface per geoprocessing.</summary>
     private GWMEngineInterface.GeoProcessing gP = null;

     /// <summary>Primo recordset.</summary>
     private PClient.GRecordset A = null;

     /// <summary>Secondo recordset.</summary>
     private PClient.GRecordset B = null;

     /// <summary>Recordset contenete il risultato.</summary>
     private PClient.GRecordset ArB = null;

     /// <summary>Riferimento a GWM-Engine Interface per la materializzazione dati.</summary>
     private GWMEngineInterface.Service.Export Exp = null;

     /// <summary>Riferimento a GWM-Engine Interface per il rendering di un layer.</summary>
     private GWMEngineInterface.Service.LayerRender LyrR = null;

     private RequestComposer rc = null;

     private ResponseComposer rC = null;

     /// Indicano il tipo di primitiva contenuta nel layer prodotto da un'operazione.
     private int OutPutGeometryType = -1;

     private string OutPutGeometryTypeName = "";
     #endregion

     #region Costruttore
     /// <summary>Costruttore predefinito.</summary>
     public GeoProcessingService()  {InitializeComponent();}
     #endregion

     #region Distruttore
     /// <summary>Distruttore</summary>
     /// <remarks>Forza lo shutdown del motore di GWM portando al rilascio il servizio.</remarks>
     ~GeoProcessingService()
         {
          if (null!=this.gP){this.gP.ShutDownGeoEngine(); this.gP = null;}
          if(null!=this.Exp){this.Exp.ShutDownGeoEngine();this.Exp = null;}
          if(null!=this.LyrR){this.LyrR.ShutDownGeoEngine();this.LyrR = null;}
         }
     #endregion

     #region Web Method
     /// <summary>Restituisce lo URL dello schema XSD della richiesta.</summary>
     /// <returns>URL schema XSD.</returns>
     [WebMethod(Description="Returns the URL of XSD request schema document.")]
     public string GetRequestSchemaURL(){return Application["XSD_URL"].ToString() + Application["GeoSpatialXSD"].ToString();}

     /// <summary>Restituisce lo URL dello schema XSD della risposta.</summary>
     /// <returns>URL schema XSD.</returns>
     [WebMethod(Description="Returns the URL of XSD document of selection by overlap request.")]
     public string GetResponseSchema(){return Application["XSD_URL"].ToString() + Application["SimpleRespXSD"].ToString();}

     /// <summary>Effettua l'operazione di clipping fra due layer.</summary>
     /// <param name="request">Richiesta XML.</param>
     /// <param name="layerName">Nome del layer risultante.</param>
     /// <returns>Restituisce un messaggio di risposta.</returns>
     [WebMethod(Description="Performs the clipping between two layers.")]
     public string ClipLayer(string request, string layerName)
         {
          string res = "";
          SimpleResponse sr = null;
          string msg = "";
          string cachedir = Application["CachePath"].ToString();
            try
             {
              this.gP = new GWMEngineInterface.GeoProcessing(Server);
              string csfile = Application["CoordSystemDir"].ToString();
              RequestComposer rc = new RequestComposer();
              GeoProcRequest req = (GeoProcRequest) rc.FormalizeRequestFromXml(request,RequestType.GeoSpatial); rc = null;
              csfile += req.SRSName + ".csf";
              this.gP.srsPathName = csfile;
              if(gP.DataAccess(req.LayerA,ref this.A, req.LayerB,ref this.B, req.InterestWindow, ref msg))
                  {
                   this.gP.NewLayerName = layerName;//nome del layer risultante.
                   this.ArB = this.gP.Clip(this.A, this.B);
```

```
                if(this.ArB!=null)
                    {
                     if(this.IsLyrFormat(req.RequestContentFormat))
                         this.DoMaterializeData(req.RequestContentFormat, layerName,cachedir);
                     if(this.IsMapFormat(req.RequestContentFormat))
                         this.DoRenderLayer(req, csfile,layerName);
                    }
                #endregion
                  else
                    {
                     rC = new ResponseComposer(MessageContent.Error,MessageContentFormat.NA_Raw);
                     sr = (SimpleResponse) rC.Response;  sr.Body = new ArrayList(3); sr.Body.Insert(0, "Clip error");
                     sr.Body.Insert(1, this.gP.GeoEngineErrorMsg);sr.Body.Insert(2, this.gP.SystemErrorMsg);
                    }
                this.gP = null;
                }
              else
                {
                #region Gestione errore "DataAccess"
                rC = new ResponseComposer(MessageContent.Error,MessageContentFormat.NA_Raw);
                sr = (SimpleResponse) rC.Response;sr.Body = new ArrayList(3);sr.Body.Insert(0, "Clip error");sr.Body.Insert(1, "");
                sr.Body.Insert(2, "ClipLayer::DataAccess::" + msg);
                }
            this.gP.ShutDownGeoEngine();
            }
        #region Catch & finally
        catch(Exception ex)
            {
            rC = new ResponseComposer(MessageContent.Error,MessageContentFormat.NA_Raw);
            sr = (SimpleResponse) rC.Response;sr.Body = new ArrayList(3);sr.Body.Insert(0, "Clip error");sr.Body.Insert(1, "");
            sr.Body.Insert(2, "ClipLayer::" + ex.ToString());
            }
         finally
            {
            if(this.gP!=null) {this.gP.ShutDownGeoEngine(); this.gP = null;}
            if(this.Exp!=null) {this.Exp.ShutDownGeoEngine(); this.Exp = null;}
            }
        #endregion
        res = this.rC.FormalizeResponseToXml();
        this.rC = null;
        return res;
        }

    /// <summary>Effettua la differenza booleana fra due layer.</summary>
    /// <param name="request">Richiesta XML.</param>
    /// <param name="layerName">Nome del layer risultant.</param>
    /// <returns>Restituisce un messaggio di risposta.</returns>
    [WebMethod(Description="Performs the difference between two layers.")]
    public string DifferenceLayer(string request, string layerName)
        {
        string res = "";SimpleResponse sr = null;string msg = "";
        string cachedir = Application["CachePath"].ToString();
           try
            {
            this.gP = new GWMEngineInterface.GeoProcessing(Server);
            string csfile = Application["CoordSystemDir"].ToString();
            RequestComposer rc = new RequestComposer();
            GeoProcRequest req = (GeoProcRequest) rc.FormalizeRequestFromXml(request,RequestType.GeoSpatial); rc = null;
            csfile += req.SRSName + ".csf";
            this.gP.srsPathName = csfile;
            if(gP.DataAccess(req.LayerA,ref this.A, req.LayerB,ref this.B, req.InterestWindow, ref msg))
                {
                this.gP.NewLayerName = layerName;//nome del layer risultante.this.ArB = this.gP.Difference(this.A, this.B);
                if(this.ArB!=null)
                    {
                     if(this.ArB!=null)
                         {
                          if(this.IsLyrFormat(req.RequestContentFormat))
                              this.DoMaterializeData(req.RequestContentFormat, layerName,cachedir);
                          if(this.IsMapFormat(req.RequestContentFormat))
                              this.DoRenderLayer(req, csfile,layerName);
                         }
                    }
                  else
                    {
                     rC = new ResponseComposer(MessageContent.Error,MessageContentFormat.NA_Raw);sr = (SimpleResponse) rC.Response;
                     sr.Body = new ArrayList(3);sr.Body.Insert(0, "Difference error");
                     sr.Body.Insert(1, this.gP.GeoEngineErrorMsg);     sr.Body.Insert(2, this.gP.SystemErrorMsg);
                    }
                this.gP = null;
                #endregion
                }
              else
                {
                rC = new ResponseComposer(MessageContent.Error,MessageContentFormat.NA_Raw);
                sr = (SimpleResponse) rC.Response;
                sr.Body = new ArrayList(3);sr.Body.Insert(0, "Difference error");
                sr.Body.Insert(1, "");sr.Body.Insert(2, "Difference::DataAccess::" + msg);
                }
            this.gP.ShutDownGeoEngine();
            }
        #region Catch & finally
        catch(Exception ex)
            {
            rC = new ResponseComposer(MessageContent.Error,MessageContentFormat.NA_Raw);
            sr = (SimpleResponse) rC.Response;sr.Body = new ArrayList(3);sr.Body.Insert(0, "Difference error");
            sr.Body.Insert(1, "");sr.Body.Insert(2, "Difference::" + ex.ToString());
            }
         finally
            {
            if(this.gP!=null) {this.gP.ShutDownGeoEngine(); this.gP = null;}
            if(this.Exp!=null) {this.Exp.ShutDownGeoEngine(); this.Exp = null;}
            }
        #endregion
        res = this.rC.FormalizeResponseToXml();
        this.rC = null;
        return res;
```

```
        }

    /// <summary>
    /// Effettua l'operazione di intersezione fra due layer.
    /// </summary>
    /// <param name="request">Richiesta XML.</param>
    /// <param name="layerName">Nome del layer risultante.</param>
    /// <returns>Restituisce un messaggio di risposta.</returns>
    [WebMethod(Description="Performs the Intersection between two layers.")]
    public string IntersectionLayer(string request, string layerName)
        {
        string res = "";
        SimpleResponse sr = null;
        string msg = "";
        string reqstErr = "";
        string cachedir = Application["CachePath"].ToString();
           try
            {
            string csfile = Application["CoordSystemDir"].ToString();
            this.gP = new GWMEngineInterface.GeoProcessing(Server);//Start-up GWM Engine
            #region Deserializzazione Richiesta
            this.rc = new RequestComposer();
            GeoProcRequest req = (GeoProcRequest) this.rc.FormalizeRequestFromXml(request,RequestType.GeoSpatial); rc = null;
            #endregion
            if(true)//this.IsValidGeoRequest(req,ref reqstErr))
                {
                csfile += req.SRSName + ".csf";
                this.gP.srsPathName = csfile;
                if(gP.DataAccess(req.LayerA,ref this.A,
                    req.LayerB,ref this.B, req.InterestWindow, ref msg))
                    {
                    #region Effettua Intersezione
                    this.gP.NewLayerName = layerName;//nome del layer risultante.
                    this.ArB = this.gP.Intersect(this.A, this.B);
                    this.OutPutGeometryType = this.gP.GeometryType;
                    this.OutPutGeometryTypeName = this.gP.GeometryTypeName;
                    if(this.ArB!=null)
                        {
                        if(this.IsLyrFormat(req.RequestContentFormat))
                            this.DoMaterializeData(req.RequestContentFormat, layerName,cachedir);
                        if(this.IsMapFormat(req.RequestContentFormat))
                            this.DoRenderLayer(req, csfile,layerName);
                        }
                      else
                        {
                        rC = new ResponseComposer(MessageContent.Error,MessageContentFormat.NA_Raw)sr = (SimpleResponse) rC.Response;
                        sr.Body = new ArrayList(3);sr.Body.Insert(0, "Intersection error");
                        sr.Body.Insert(1, this.gP.GeoEngineErrorMsg); sr.Body.Insert(2, this.gP.SystemErrorMsg);
                        }
                    this.gP.ShutDownGeoEngine();this.gP = null;
                    }
                  else
                    {
                    rC = new ResponseComposer(MessageContent.Error,MessageContentFormat.NA_Raw);
                    sr = (SimpleResponse) rC.Response;  sr.Body = new ArrayList(3);
                    sr.Body.Insert(0, "Intersection error");sr.Body.Insert(1, "");
                    sr.Body.Insert(2, "Intersection::DataAccess::" + msg);
                    }
                }
              else
                {
                rC = new ResponseComposer(MessageContent.Error,MessageContentFormat.NA_Raw);
                sr = (SimpleResponse) rC.Response;sr.Body = new ArrayList(3);
                sr.Body.Insert(0, "Intersection error");sr.Body.Insert(1, "");sr.Body.Insert(2, "Intersection::" + reqstErr);
                }
            }
        catch(Exception ex)
            {
            rC = new ResponseComposer(MessageContent.Error,MessageContentFormat.NA_Raw);
            sr = (SimpleResponse) rC.Response;sr.Body = new ArrayList(3);          sr.Body.Insert(0, "Intersection error");
            sr.Body.Insert(1, "");sr.Body.Insert(2, "Intersection::" + ex.ToString());
            }
         finally
            {
            if(this.gP!=null) {this.gP.ShutDownGeoEngine(); this.gP = null;}
            if(this.Exp!=null) {this.Exp.ShutDownGeoEngine(); this.Exp = null;}
            }
        #endregion
        res = this.rC.FormalizeResponseToXml();this.rC = null;return res;
        }

    /// <summary>
    /// Effettua l'operazione di Merge fra due layer.
    /// </summary>
    /// <param name="request">Richiesta XML.</param>
    /// <param name="layerName">Nome del layer risultante.</param>
    /// <returns>Restituisce un messaggio di risposta.</returns>
    [WebMethod(Description="Performs the merge between two layers.")]
    public string MergeLayer(string request, string layerName)
        {
        string res = "";SimpleResponse sr = null;string msg = "";     string cachedir = Application["CachePath"].ToString();
           try
            {
            this.gP = new GWMEngineInterface.GeoProcessing(Server);
            string csfile = Application["CoordSystemDir"].ToString();
            RequestComposer rc = new RequestComposer();
            GeoProcRequest req = (GeoProcRequest) rc.FormalizeRequestFromXml(request,RequestType.GeoSpatial); rc = null;
            csfile += req.SRSName + ".csf";this.gP.srsPathName = csfile;
            if(gP.DataAccess(req.LayerA,ref this.A,
                req.LayerB,ref this.B, req.InterestWindow, ref msg))
                {
                this.gP.NewLayerName = layerName; this.ArB = this.gP.Merge(this.A, this.B);
                if(this.ArB!=null)
                    {
                    if(this.ArB!=null)
                        {
                        if(this.IsLyrFormat(req.RequestContentFormat))
```

```
                        this.DoMaterializeData(req.RequestContentFormat, layerName,cachedir);
                    if(this.IsMapFormat(req.RequestContentFormat))
                        this.DoRenderLayer(req, csfile,layerName);
                    }
                  }
                 else
                  {
                   rC = new ResponseComposer(MessageContent.Error,MessageContentFormat.NA_Raw);
                   sr = (SimpleResponse) rC.Response;  sr.Body = new ArrayList(3); sr.Body.Insert(0, "Merge error");
                   sr.Body.Insert(1, this.gP.GeoEngineErrorMsg);sr.Body.Insert(2, this.gP.SystemErrorMsg);
                  }
               this.gP.ShutDownGeoEngine();this.gP = null;
               #endregion
              }
             else
              {
               rC = new ResponseComposer(MessageContent.Error,MessageContentFormat.NA_Raw);
               sr = (SimpleResponse) rC.Response;sr.Body = new ArrayList(3);       sr.Body.Insert(0, "Merge error");
               sr.Body.Insert(1, "");sr.Body.Insert(2, "Merge::DataAccess::" + msg);
              }
          }
        #region Catch & finally
        catch(Exception ex)
          {
           rC = new ResponseComposer(MessageContent.Error,MessageContentFormat.NA_Raw);
           sr = (SimpleResponse) rC.Response;sr.Body = new ArrayList(3);sr.Body.Insert(0, "Merge error");
           sr.Body.Insert(1, "");sr.Body.Insert(2, "Merge::" + ex.ToString());
          }
         finally
          {if(this.gP!=null) {this.gP.ShutDownGeoEngine(); this.gP = null;}
           if(this.Exp!=null) {this.Exp.ShutDownGeoEngine(); this.Exp = null;}
          }
        #endregion
        res = this.rC.FormalizeResponseToXml();this.rC = null;return res;
       }

    /// <summary>Effettua l'operazione di Unione fra due layer.</summary>
    /// <param name="request">Richiesta XML.</param>
    /// <param name="layerName">Nome del layer risultante.</param>
    /// <returns>Restituisce un messaggio di risposta.</returns>
    [WebMethod(Description="Performs the union between two layers.")]
    public string UnionLayer(string request, string layerName)
       {
        string res = "";SimpleResponse sr = null;string msg = "";string cachedir = Application["CachePath"].ToString();
          try
           {
            this.gP = new GWMEngineInterface.GeoProcessing(Server);string csfile = Application["CoordSystemDir"].ToString();
            RequestComposer rc = new RequestComposer();
            GeoProcRequest req = (GeoProcRequest) rc.FormalizeRequestFromXml(request,RequestType.GeoSpatial); rc = null;
            csfile += req.SRSName + ".csf";this.gP.srsPathName = csfile;
            if(gP.DataAccess(req.LayerA,ref this.A, req.LayerB,ref this.B, req.InterestWindow, ref msg))
               {
                this.gP.NewLayerName = layerNamethis.ArB = this.gP.Union(this.A,this.B);
                this.OutPutGeometryType = this.gP.GeometryType;this.OutPutGeometryTypeName = this.gP.GeometryTypeName;
                if(this.ArB!=null)
                    {
                     if(this.ArB!=null)
                         {
                          if(this.IsLyrFormat(req.RequestContentFormat))
                              this.DoMaterializeData(req.RequestContentFormat, layerName,cachedir);
                          if(this.IsMapFormat(req.RequestContentFormat))
                              this.DoRenderLayer(req, csfile,layerName);
                         }
                    }
                   else
                    {
                     #region Gestione errore causato dall'unione
                     rC = new ResponseComposer(MessageContent.Error,MessageContentFormat.NA_Raw);  sr = (SimpleResponse) rC.Response;
                     sr.Body = new ArrayList(3);sr.Body.Insert(0, "Union error");
                     sr.Body.Insert(1, this.gP.GeoEngineErrorMsg);     sr.Body.Insert(2, this.gP.SystemErrorMsg);
                     #endregion
                    }
                this.gP.ShutDownGeoEngine();this.gP = null;
               }
              else
               {
                rC = new ResponseComposer(MessageContent.Error,MessageContentFormat.NA_Raw);
                sr = (SimpleResponse) rC.Response;sr.Body = new ArrayList(3);       sr.Body.Insert(0, "Union error");
                sr.Body.Insert(1, "");sr.Body.Insert(2, "UnionLayer::DataAccess::" + msg);
               }
          }
        catch(Exception ex)
          {
           this.rC = new ResponseComposer(MessageContent.Error,MessageContentFormat.NA_Raw);sr = (SimpleResponse) rC.Response;
           sr.Body = new ArrayList(3);sr.Body.Insert(0, "Union error");
           sr.Body.Insert(1, "");sr.Body.Insert(2, "UnionLayer::" + ex.ToString());res = "";
          }
         Finally
          {
           if(this.gP!=null) {this.gP.ShutDownGeoEngine(); this.gP = null;}
           if(this.Exp!=null) {this.Exp.ShutDownGeoEngine(); this.Exp = null;}
          }
        #endregion
        res = this.rC.FormalizeResponseToXml();this.rC = null;return res;
       }
    #endregion

    #region Metodi privati
    /// <summary> Esegue il rendering del dataset e compone l'oggetto risposta. </summary>
    /// <param name="req">Oggetto richiesta</param>
    /// <param name="coordSysFile">Sistema di riferimento completamente identificato</param>
    /// <param name="layerName">Nome da attribuire al dataset</param>
    /// <returns>False in caso di errori.</returns>
    private bool DoRenderLayer(GeoProcRequest req, string coordSysFile, string layerName)
       {
        bool b = true;SimpleResponse sr = null;
          try
```

```
            {
             this.LyrR = new GWMEngineInterface.Service.LayerRender(this.gP.MapServer); this.LyrR.srsPathName = coordSysFile;
             this.LyrR.Style = req.Style;this.LyrR.Width = req.Width;this.LyrR.Height = req.Height;
             this.LyrR.GraphicThemeFormat = req.MapFormat();
             if(null != req.InterestWindow) this.LyrR.InterestArea = req.InterestWindow;
             this.LyrR.InputRecordset = this.ArB;
             if(this.LyrR.Render(layerName))
                 {
                  this.rC = new ResponseComposer(MessageContent.Layer,req.RequestContentFormat);
                  sr = (SimpleResponse) this.rC.Response;
                  sr.Body = new ArrayList(2);sr.Body.Insert(0, this.LyrR.OutputFile); sr.Body.Insert(1, layerName);
                  sr.Body.Insert(2, this.OutPutGeometryType  + "|" + this.LyrR.Style.FillColor + "|" +
                                 this.LyrR.Style.BorderColor + "|" + this.LyrR.Style.BorderWidth );
                 }
             else//Gestione fallimento
                 {
                  this.rC = new ResponseComposer(MessageContent.Error,MessageContentFormat.NA_Raw);
                  sr = (SimpleResponse) rC.Response;sr.Body = new ArrayList(3);sr.Body.Insert(0, "Materialization error");
                  sr.Body.Insert(1, this.Exp.GeoEngineErrorMsg);sr.Body.Insert(2, this.Exp.SystemErrorMsg);
                 }
            }
         catch(Exception e)
            {
             rC = new ResponseComposer(MessageContent.Error,MessageContentFormat.NA_Raw);
             sr = (SimpleResponse) rC.Response;sr.Body = new ArrayList(3);
             sr.Body.Insert(0, "Layer Rendering error"); sr.Body.Insert(1, "");
             sr.Body.Insert(2, "Layer Rendering::" + e.ToString());b = false;
            }
          finally{if(null != this.LyrR){this.LyrR.ShutDownGeoEngine();this.LyrR = null;}}
         return b;
        }

    /// <summary>Materializza un dataset</summary>
    /// <param name="req">Oggetto richiesta</param>
    /// <param name="layerName">Nome da attribuire al dataset</param>
    /// <param name="cacheDir">Percorso filesystem dove materializzare i dati.</param>
    /// <returns>False in caso di errori.</returns>
    private bool DoMaterializeData(MessageContentFormat req, string layerName, string cacheDir)
        {
         bool b = true;
         SimpleResponse sr = null;
           try
            {
             this.Exp = new GWMEngineInterface.Service.Export(this.gP.MapServer);
             if(req == MessageContentFormat.Layer_Shp)
                 #region Esportazione SHP
             if(this.Exp.ExportToShapeFileRV((GDO.GRecordset)ArB,layerName,cacheDir))
                 {
                  this.Exp.ShutDownGeoEngine();this.Exp = null;
                  this.rC = new ResponseComposer(MessageContent.Layer,MessageContentFormat.Layer_Shp);
                  sr = (SimpleResponse) this.rC.Response;sr.Body = new ArrayList(3);
                  sr.Body.Insert(0, Application["CacheURL"].ToString() + layerName + ".shp");
                  sr.Body.Insert(1, Application["CacheURL"].ToString() + layerName + ".dbf");
                  sr.Body.Insert(2, Application["CacheURL"].ToString() + layerName + ".shx");
                 }
               else
                 {
                  this.rC = new ResponseComposer(MessageContent.Error,MessageContentFormat.NA_Raw);
                  sr = (SimpleResponse) rC.Response;sr.Body = new ArrayList(3);sr.Body.Insert(0, "Materialization error");
                  sr.Body.Insert(1, this.Exp.GeoEngineErrorMsg);sr.Body.Insert(2, this.Exp.SystemErrorMsg);
                 }
             #endregion
               else
                  if(req == MessageContentFormat.Layer_Gml)
                  if(this.Exp.ExportToGML((GDO.GRecordset)ArB,layerName,cacheDir))
                      {
                       this.Exp.ShutDownGeoEngine();
                       this.Exp = null;
                       this.rC = new ResponseComposer(MessageContent.Layer,MessageContentFormat.Layer_Gml);
                       sr = (SimpleResponse) this.rC.Response;
                       sr.Body = new ArrayList(3);
                       sr.Body.Insert(0, Application["CacheURL"].ToString() + layerName + ".gml");
                       sr.Body.Insert(1, Application["CacheURL"].ToString() + layerName + ".shp");
                      }
                     else
                      {
                       this.rC = new ResponseComposer(MessageContent.Error,MessageContentFormat.NA_Raw);
                       sr = (SimpleResponse) rC.Response;sr.Body = new ArrayList(3);
                       sr.Body.Insert(0, "Materialization error");sr.Body.Insert(1, this.Exp.GeoEngineErrorMsg);
                       sr.Body.Insert(2, this.Exp.SystemErrorMsg);
                      }
             #endregion
            }
         catch(Exception e)
            {
             rC = new ResponseComposer(MessageContent.Error,MessageContentFormat.NA_Raw);
             sr = (SimpleResponse) rC.Response;sr.Body = new ArrayList(3);sr.Body.Insert(0, "Materialization error");
             sr.Body.Insert(1, "");sr.Body.Insert(2, "Materialization::" + e.ToString());b = false;
            }
        #endregion
         finally{if(null!=this.Exp) {this.Exp.ShutDownGeoEngine();this.Exp = null;}}
        return b;
        }

    /// <summary>Indica se la richiesta è valida per questo web service. </summary>
    /// <param name="req">oggetto per richiesta geoprocessing</param>
    /// <param name="msgErr">In caso di fallimento, contiene un messaggio di errore descrittivo.</param>
    /// <returns>True se la richiesta è valida per questo web service.</returns>
    private bool IsValidGeoRequest(GeoProcRequest req, ref string msgErr)
        {
         bool validrequest;
         //Controllo validità richiesta, 2 passi:
         //1.Controllo validità contenuto richiesta:
         bool ValidContent = true;
         ValidContent = (req.RequestContent == MessageContent.Layer);
         //2.Controllo validità Formato contenuto richiesta, 2 parti:
         //2.1:Il formato è un dataset materializzato
```

```
536             bool ValidContentRequestFrmtLyr = true;
537             ValidContentRequestFrmtLyr = (req.RequestContentFormat == MessageContentFormat.Layer_Gml || req.RequestContentFormat ==
538         MessageContentFormat.Layer_Shp);
539             //2.2: Il formato è grafico: richiede esistenda del sistema di riferimento
540             bool ValidContentRequestFrmtMap = true;
541             ValidContentRequestFrmtMap = (req.RequestContentFormat >= MessageContentFormat.Map_A16 && req.RequestContentFormat <=
542         MessageContentFormat.Map_A32);
543             if(ValidContent)
544                 {
545                  if(ValidContentRequestFrmtLyr)
546                      validrequest = true;
547                    else
548                     {
549                      if(ValidContentRequestFrmtMap)
550                          if(""!=req.SRSName) validrequest = true;
551                           else {validrequest = false;msgErr ="Spatial Reference Sistyem needed.";}
552                        else  {validrequest = false; msgErr ="Invalid request content format.";}
553                     }
554                 }
555                else
556                 {validrequest = false;msgErr ="Invalid request content.";}
557             if(""!=msgErr) msgErr = "Invalid Request:" + msgErr;
558             return validrequest;
559            }
560
561         /// <summary>
562         /// Indica se il formato del contenuto di una richiesta è un layer da
563         /// materializzare.
564         /// </summary>
565         /// <param name="mcf">Formato del contenuto del mesaggio</param>
566         /// <returns>True se è un formato per materializzazione.</returns>
567         private bool IsLyrFormat(MessageContentFormat mcf)
568         {return (mcf == MessageContentFormat.Layer_Gml || mcf == MessageContentFormat.Layer_Shp);}
569
570         /// <summary>
571         /// Indica se il formato del contenuto di una richiesta è un formato
572         /// grafico.
573         /// </summary>
574         /// <param name="mcf">Formato del contenuto del mesaggio</param>
575         /// <returns>True se è un formato grafico.</returns>
576         private bool IsMapFormat(MessageContentFormat mcf)
577         {return (mcf >= MessageContentFormat.Map_A16 && mcf <= MessageContentFormat.Map_A32);}
578         #endregion
579         }
580     }
```

```
using System;
using System.Web;
using System.Web.Services;
using System.Data;
using System.Diagnostics;
using System.Collections;
using System.ComponentModel;
//********* Spatial Operations Interface *********
using GWMEngineInterface;
//** Request & Response Formalization Namespace **
using RequestResponseFormalizer;
//********* Geomedia Web Map Namespace ***********
using MapSrvMngrLib = MAPSVRMNGRLib;
using MapSrvLib = MAPSRVLib;

///<Author>Giulio Massei</Author>
///<LastUpdate>09/08/2005</LastUpdate>

namespace GisOperationsDeliverer
{
/// <summary>Data una richiesta XML, permette di selezionare feature in base a criteri spaziali. </summary>
[WebService(Namespace="http://webgisserver.isti.cnr.it/GisOperations/GeographicSelectionService/")]
public class GeoSelectionService : System.Web.Services.WebService
    {
    // Una "Interfaccia-GWM" è una classe che permette di accedere
    //ai meccanismi GWM.
    #region Campi privati
    /// <summary> Interfaccia-GWM per il buffering. </summary>
    private GWMEngineInterface.SelectBy selBy = null;
    /// <summary>Primo recordset. </summary>
    private PClient.GRecordset A = null;
    /// <summary> Secondo recordset. </summary>
    private PClient.GRecordset B = null;

    private RequestComposer rc = null;

    private ResponseComposer rC = null;
    #endregion
    /// <summary>Costruttore predefinito. </summary>
    public GeoSelectionService(){InitializeComponent();}
    #region Distruttore
    /// <summary>Distruttore</summary>
    /// <remarks>Forza lo shutdown del motore di GWM portando al rilascio il servizio. </remarks>
    ~GeoSelectionService()
    {if (null!=this.selBy){this.selBy.ShutDownGeoEngine();this.selBy = null;}}
    #endregion

    #region Web Method
    /// <summary> Restituisce lo schema XSD della richiesta per selezione basata sulla distanza. </summary>
    /// <returns>URL schema XSD.</returns>
    [WebMethod(Description="Returns The XSD request schema document.")]
    public string Distance_GetRequestSchemaURL()
    {return Application["XSD_URL"].ToString() + Application["SelectByDistXSD"].ToString();}

    /// <summary>Restituisce lo schema XSD della richiesta per selezione tramite overlap. </summary>
    /// <returns>URL schema XSD.</returns>
    [WebMethod(Description="Returns the URL ofXSD document of selection by distance request .")]
    public string Overlap_GetRequestSchemaURL()
    {return Application["XSD_URL"].ToString() + Application["SelectByOverXSD"].ToString();}

/// <summary>Restituisce lo URL dello schema XSD della risposta. </summary>
    /// <returns>URL schema XSD.</returns>
    [WebMethod(Description="Returns the URL of XSD document of selection by overlap request.")]
    public string GetResponseSchema()
    {return Application["XSD_URL"].ToString() + Application["SimpleRespXSD"].ToString();}

    /// <summary>
    /// Permette di selezionare le feature di un layer che sono ad una data distanza dalle feature dell'altro layer. </summary>
    /// <param name="request">Richiesta XML.</param>
    /// <returns>Restituisce un messaggio di risposta.</returns>
    [WebMethod(Description="Performs the selection of features of a layer by the distance from the features of another layer.")]
    public string SelectByDistance(string request)
        {
        string res = "";SimpleResponse sr = null;string msg = "";ArrayList t = null;PClient.GRecordset dummy = null;
           try
            {
            this.selBy = new SelectBy(Server);this.rc = new RequestComposer();
            DistanceSelectionRequest req = (DistanceSelectionRequest)
            this.rc.FormalizeRequestFromXml(request,RequestType.SelectionByDistance); this.rc = null;
            string csfile = Application["CoordSystemDir"].ToString();
            csfile += req.SRSName + ".csf";this.selBy.srsPathName = csfile;
            if(this.selBy.DataAccess(req.LayerA, ref this.A, req.LayerB, ref this.B, req.InterestWindow, ref msg))
                {
                t =this.selBy.Distance(req.LayerA.layerName, this.A, this.B, req.Distance, ref dummy);
                this.selBy.ShutDownGeoEngine();
                if(null!=t)
                    {
                    this.rC = new ResponseComposer(MessageContent.Identifiers,MessageContentFormat.Identifiers_Set);
                    this.rC.Response.Body = t; t = null;
                    }
                  else
                    {
                    this.rC = new ResponseComposer(MessageContent.Error,MessageContentFormat.NA_Raw);
                    sr = (SimpleResponse) this.rC.Response;sr.Body = new ArrayList(3);
                    sr.Body.Insert(0, "Selection error");sr.Body.Insert(1, selBy.GeoEngineErrorMsg);
                    sr.Body.Insert(2, selBy.SystemErrorMsg);
                    }
                #endregion
                }
              else
                {
                #region Gestione errore di "DataAccess"
                this.rC = new ResponseComposer(MessageContent.Error,MessageContentFormat.NA_Raw);
                sr = (SimpleResponse) this.rC.Response;sr.Body = new ArrayList(3); sr.Body.Insert(0, "Selection error");
                sr.Body.Insert(1, "");sr.Body.Insert(2,"SelectByDistance::DataAccess::" + msg);
                #endregion
                }
            }
```

```
        #region Catch & finally
        catch(Exception ex)
            {
             this.rC = new ResponseComposer(MessageContent.Error,MessageContentFormat.NA_Raw);
             sr = (SimpleResponse) this.rC.Response;
             sr.Body = new ArrayList(3);
             sr.Body.Insert(0, "Selection error");
             sr.Body.Insert(1, "");
             sr.Body.Insert(2,"SelectByDistance::" + ex.ToString());
            }
        Finally {if(this.selBy!=null){this.selBy.ShutDownGeoEngine();this.selBy = null;}}
        #endregion
        res = this.rC.FormalizeResponseToXml();
        this.rC = null;
        return res;
       }

    /// <summary>
    /// Permette di selezionare le feature di un layer in base ad un dato criterio
    /// di confronto spaziale con un altro layer.
    /// </summary>
    /// <param name="request">Richiesta XML.</param>
    /// <returns>Restituisce un messaggio di risposta.</returns>
    [WebMethod(Description="Performs the selection of features by the overlappig relationship between the features of the two
layers.")]
    public string SelectByOverlap(string request)
        {
        string res = "";
        this.rC = null;
        SimpleResponse sr = null;
        string msg = "";
        ArrayList t = null;
        PClient.GRecordset dummy = null;
           try
            {
             this.selBy = new SelectBy(Server);
             this.rc = new RequestComposer();
             OverlapSelectionRequest req = (OverlapSelectionRequest)
this.rc.FormalizeRequestFromXml(request,RequestType.SelectionByOverlap);this.rc = null;
             string csfile = Application["CoordSystemDir"].ToString();
             csfile += req.SRSName + ".csf";
             this.selBy.srsPathName = csfile;
             if(this.selBy.DataAccess(req.LayerA, ref this.A, req.LayerB, ref this.B, req.InterestWindow, ref msg))
                {
                t =this.selBy.Overlap(req.LayerA.layerName, this.A, this.B, (int) req.overlapType, ref dummy);
                this.selBy.ShutDownGeoEngine();
                if(null!=t)
                    {
                     this.rC = new ResponseComposer(MessageContent.Identifiers,MessageContentFormat.Identifiers_Set);
                     this.rC.Response.Body = t; t = null;
                    }
                  else
                    {
                     #region Gestione errore di Selezione
                     this.rC = new ResponseComposer(MessageContent.Error,MessageContentFormat.NA_Raw);
                     sr = (SimpleResponse) this.rC.Response;
                     string hg = "";
                     if("" == selBy.GeoEngineErrorMsg)
                        hg = "Empty selection.";
                       else
                         hg = selBy.GeoEngineErrorMsg;
                     sr.Body = new ArrayList(3);
                     sr.Body.Insert(0, "Overlap selection error");
                     sr.Body.Insert(1, hg );
                     sr.Body.Insert(2, selBy.SystemErrorMsg);
                     #endregion
                    }
                }
               else
                {
                 #region Gestione errore di "DataAccess"
                 this.rC = new ResponseComposer(MessageContent.Error,MessageContentFormat.NA_Raw);
                 sr = (SimpleResponse) this.rC.Response;
                 sr.Body = new ArrayList(3);
                 sr.Body.Insert(0, "Overlap selection error");
                 sr.Body.Insert(1, "");
                 sr.Body.Insert(2,"SelectByOverlap::DataAccess::" + msg);
                 #endregion
                }
            }
        #region Catch & finally
        catch(Exception ex)
            {
             this.rC = new ResponseComposer(MessageContent.Error,MessageContentFormat.NA_Raw);
             sr = (SimpleResponse) this.rC.Response;
             sr.Body = new ArrayList(3);
             sr.Body.Insert(0, "Overlap selection error");
             sr.Body.Insert(1, "");
             sr.Body.Insert(2,"SelectByOverlap::" + ex.ToString());
            }
         finally
            {
             if(this.selBy!=null){this.selBy.ShutDownGeoEngine();this.selBy = null;}
            }
        #endregion
        res = this.rC.FormalizeResponseToXml();
        this.rC = null;
        return res;
       }
    #endregion
    }
}
```

```
using System;
using System.Web;
using System.Data;
using System.Collections;
using System.Diagnostics;
using System.Web.Services;
using System.ComponentModel;
//**************** GWM Namespaces ***************
using PPipe;
using PDBPipe;
using PClient;
using Gdo = GDO;
using MapSrvLib = MAPSRVLib;
using MapSrvMngrLib = MAPSVRMNGRLib;
using MapSrvComponentsLib = MAPSRVCOMPONENTSLib;
//************** Namespace Interni ***************
using GenericLayerIdentifier;
//Request & Response Formalization Namespace
using RequestResponseFormalizer;
//GWM-Engine Interface
using GWMEngineInterface;
//************************
using myRGBclass;

///<Author>Giulio Massei</Author>
///<LastUpdate>20/10/2005</LastUpdate>

namespace GisOperationsDeliverer
{
/// <summary>
/// Data una richiesta contenete un insieme di temi, il servizio crea una rappresentazione grafica della mappa.
/// </summary>
[WebService(Namespace="http://webgisserver.isti.cnr.it/GisOperations/MapRenderService/")]
public class MapRenderService : System.Web.Services.WebService
    {
    #region Attributi Privati
        private MapSrvLib.GWMMapServerHelper pMapServer = null;

        private MapSrvMngrLib.MapServerManagerClass pMapSrvMngr = null;

        private bool setup = false; //Indica se è stato impostato il mapserver

        private GWMEngineInterface.Service.MapRender mR = null;

        private RequestComposer rc = null;

        private ResponseComposer rC = null;
    #endregion

    #region Costruttori
    /// <summary>Costruttore predefinito.</summary>
    public MapRenderService(){InitializeComponent();}
    #endregion

    #region Distruttore
    /// <summary>Distruttore </summary>
    /// <remarks> Forza lo shutdown del motore di GWM portando al rilascio il servizio. </remarks>
    ~MapRenderService()
        {
        if(this.pMapServer!=null)
            {
            this.pMapServer.Clear();System.Runtime.InteropServices.Marshal.ReleaseComObject(this.pMapServer);
            this.pMapServer = null;
            }
        if(this.pMapSrvMngr!=null) System.Runtime.InteropServices.Marshal.ReleaseComObject(this.pMapSrvMngr);
        this.setup = false;
        }
    #endregion

    #region Web Method
    /// <summary>Restituisce lo URL dello schema XSD della richiesta. </summary>
    /// <returns>URL schema XSD.</returns>
    [WebMethod(Description="Returns the URL of XSD request schema document.")]
    public string GetRequestSchemaURL(){return Application["XSD_URL"].ToString() + Application["MapRenderXSD"].ToString();}

    /// <summary>Restituisce lo URL dello schema XSD della risposta. </summary>
    /// <returns>URL schema XSD.</returns>
    [WebMethod(Description="Returns the URL of XSD document of selection by overlap request.")]
    public string GetResponseSchema() {return Application["XSD_URL"].ToString() + Application["SimpleRespXSD"].ToString();}

    /// <summary>Data una richiesta produce una mappa. </summary>
    /// <param name="request">Richiesta espressa in XML</param>
    /// <param name="width">Larghezza, espressa in pixel, della mappa.</param>
    /// <param name="height">Altezza, espressa in pixel, della mappa.</param>
    /// <returns>Restituisce un messaggio di risposta.</returns>
    [WebMethod(Description="Given a set of themes, the service renders them in a graphical format.")]
    public string CreateMap(string request)
        {
        string res = "";MapRequest req = null;SimpleResponse sr = null;
        try
            {
            this.rc = new RequestComposer();//Classe per comporre la richiesta
            req = (MapRequest) this.rc.FormalizeRequestFromXml(request,RequestType.Mapping); this.rc = null;
            this.mR = new GWMEngineInterface.Service.MapRender(Server);
            if(null != this.mR)
                {
                string csfile = Application["CoordSystemDir"].ToString();csfile += req.SRSName + ".csf";
                res = this.mR.RenderMap(req.MapThemes,csfile,req.InterestWindow,req.Width, req.Height,req.MapFormat());
                if("" == res)
                    {
                    this.rC = new ResponseComposer(MessageContent.Error,MessageContentFormat.NA_Raw);
                    sr = (SimpleResponse) this.rC.Response;sr.Body = new ArrayList(3);
                    sr.Body.Insert(0, "Rendering error");sr.Body.Insert(1, this.mR.GeoEngineErrorMsg);
                    sr.Body.Insert(2, this.mR.SystemErrorMsg);
                    }
                else
                    {
                    this.rC = new ResponseComposer(MessageContent.Map,req.RequestContentFormat);
```

```
                    sr = (SimpleResponse) this.rC.Response;sr.Body = new ArrayList(1);sr.Body.Insert(0, res);
                   }
                this.mR.ShutDownGeoEngine(); this.mR = null;
               }
            }
        catch(Exception ex)
            {
            this.rC = new ResponseComposer(MessageContent.Error,MessageContentFormat.NA_Raw);
            sr = (SimpleResponse) this.rC.Response;sr.Body = new ArrayList(3);      sr.Body.Insert(0, "Rendering error");
            sr.Body.Insert(1, "");sr.Body.Insert(2, "MapRender::" + ex.ToString());
            }
         Finally    {if(this.mR!=null) {this.mR.ShutDownGeoEngine(); this.mR = null;}}
        res = rC.FormalizeResponseToXml();rC = null;    return res;
       }
    #endregion

    #region Metodi Privati
    /// <summary>Prepara la porzione di mappa composta da temi appartenenti alla stessasorgente dati. </summary>
    /// <param name="themes">Lista di temi appartenenti alla stessa sorgente dati.</param>
    /// <param name="Mbbox">MBR massimo fra quelli del frammento.</param>
    /// <remarks>Se il parametro °Mbbox° è nullo, non viene calcolato lo MBR massimo.</remarks>
    /// <param name="msg">Messaggio di errore di sistema.</param>
    /// <returns>"True" se l'operazione ha avuto successo.</returns>
    private bool MapPrepare(ThemeList themes, ref BoundingBox Mbbox, ref string msg)
        {
        int i = 0;bool err = false;BoundingBox t = null; string dataSource = "";string additionalInfo = "";
        string dataSourceType = themes[0].dataSourceType;
        if(dataSourceType.ToUpper().IndexOf("WFS")>-1)
            {
            additionalInfo = @"CSFFOLDERPATH=C:\Program Files\GeoMedia WebMap Professional\CoordSystems;";
            additionalInfo += "SWAPGEOCOORDS=FALSE;SWAPPROJCOORDS=TRUE;";
            additionalInfo += "URI=" + themes[0].dataSource + ";";dataSource = "dummy";
            }
          else
            if(dataSourceType.ToUpper().IndexOf("WMS")>-1)
                {additionalInfo = @"NOCSFOUND=FAIL;";additionalInfo += "URI=" + themes[0].dataSource + ";"dataSource = "dummy";}
            else{dataSource = themes[0].dataSource;additionalInfo = themes[0].additionalInfo;}
        // Esiste solo un'unica connessione con questo nome. Ciò è garantito
        //dall'ordinamento della lista dei Themi di mappa.
        string currConn = dataSourceType + dataSource + additionalInfo;
           try
            {
            string pdataSourceType = "";
            pdataSourceType = dataSourceType + (dataSourceType.IndexOf(".GDatabase")<0? ".GDatabase" : "");
            this.pMapServer.Connect(pdataSourceType,dataSource,additionalInfo,currConn);
            for(i=0;i<themes.Count && (!err);i++)
                {
                t = AddTheme(themes[i],currConn, ref msg);
                if(null!=t)//Se non ci sono stati errori, si calcola lo MBR massimo.
                    {if(null!=Mbbox){if(t>Mbbox) Mbbox = t;}}
                  else  err = true;
                }
            return !err;
            }
        catch(Exception e) {msg += "MapPrepare::" + e.ToString();return false;}
        }

    /// <summary>Crea il tema all'interno del mapserver. </summary>
    /// <param name="thm">Descrittore del tema</param>
    /// <param name="strCon">nome delle cannoessione a cui appartiene il tema.</param>
    /// <param name="msg">Messaggio di errore di sistema.</param>
    /// <returns>Lo MBR del thema in caso positivo, altrimenti "null" ed i parametro "msg" viene inizializzato.</returns>
    private BoundingBox AddTheme(MapThemeDescriptor thm, string strCon, ref string msg)
        {
        MapSrvComponentsLib.GWMFeatureSymbology oFSymb = null;
        BoundingBox t = null;
           try
            {
            MapSrvLib.GWMQuery oQry = (MapSrvLib.GWMQuery) this.pMapServer.AddQuery(thm.layerName,strCon,thm.layerName);
            MapSrvLib.GWMDisplayRule oDr = (MapSrvLib.GWMDisplayRule) oQry.NewDisplayRule();
            oDr.Priority = thm.Level;//Livello del tema nella mappa. Più la priorità è piccola più il tema è di alto livello.
            oDr.SheetName = thm.layerName;//Nome del tema nella mappa
            oFSymb = (MapSrvComponentsLib.GWMFeatureSymbology) this.pMapServer.CreateObject("GMWebMap.GWMFeatureSymbology",oFSymb);
            #region Layer Symbology
                myRGBclass.RGBUtils rgb = new myRGBclass.RGBUtils();
                Random rnd = new Random((int) System.DateTime.Now.Ticks);
                if(""==thm.Style.FillColor)
                    oFSymb.FillColor = rgb.FromRGB((byte)rnd.Next(255),(byte)rnd.Next(255),(byte)rnd.Next(255));
                  else
                    oFSymb.FillColor = rgb.FromHEX(thm.Style.FillColor);
                if(""==thm.Style.BorderColor)
                    oFSymb.Color = rgb.FromRGB((byte)rnd.Next(255),(byte)rnd.Next(255),(byte)rnd.Next(255));
                  else
                    oFSymb.Color = rgb.FromHEX(thm.Style.BorderColor);
                oFSymb.Weight = ((thm.Style.BorderWidth<=0)? 1: thm.Style.BorderWidth);
                oFSymb.Style = 0;//aspetto del tratto. 0= continuo.
                rnd = null;
                rgb = null;
            #endregion
            oDr.DisplaySymbology = oFSymb;
            t = new BoundingBox();
            Array v = (Array) oQry.Range;//"v" è una struttura di 'comodo'.         t.xmin = (double)v.GetValue(0);
            t.ymin = (double)v.GetValue(1);t.xmax = (double)v.GetValue(2);t.ymax = (double)v.GetValue(3);
            v = null;
            return t;
            }
        catch(Exception e)
            {msg += "AddTheme::" + e.ToString();return null;}
         finally
            {if(null!=oFSymb) {System.Runtime.InteropServices.Marshal.ReleaseComObject(oFSymb);oFSymb = null;}}
        }
    #endregion//************************
    }
}
```

```
using System;
using System.IO;
using System.Xml;
using System.Web;
using System.Web.Services;
using System.Data;
using System.Diagnostics;
using System.Collections;
using System.ComponentModel;
using PCSS;
using MapSrvLib = MAPSRVLib;
using MapSrvMngrLib = MAPSVRMNGRLib;

namespace GisOperationsDeliverer
{
/// <summary>
/// Descrizione di riepilogo per Service1.
/// </summary>
[WebService(Namespace="http://webgisserver.isti.cnr.it/GisOperations/SystemCoordTransf/")]
public class SCTranslation : System.Web.Services.WebService
    {
    /// <summary> Costruttore predefinito </summary>
    public SCTranslation(){InitializeComponent();}

    #region Metodi web
    /// <summary>Restituisce la lista dei codici inerentii sistemi di riferimento supportati. </summary>
    /// <returns>Array di stringhe</returns>
    [WebMethod]
    public string[] GetCoordSystemList()
        {
        FileInfo[] fi = null;DirectoryInfo di = null;
        di = new DirectoryInfo(Application["CoordSystemDir"].ToString());fi = di.GetFiles();
        string[] l = new string[fi.Length];int i=0;
        for(i=0;i<fi.Length;i++)l[i] = fi.GetValue(i).ToString().Substring(0,fi.GetValue(i).ToString().Length-4);
        Array.Sort(l);di = null; fi = null;return l;
        }

    /// <summary>restituisce il documento con la descrizione dei sistemi di riferimento </summary>
    /// <returns>Stringa XML</returns>
    [WebMethod]
    public string GetCapabilities()
        {
        string s; XmlDocument xd = new XmlDocument();
        string p= Server.MapPath(@"capabilities\TransformCapabilities.xml");
        p = p.Trim();xd.Load(p);s = xd.OuterXml;xd = null;            return s;
        }

    /// <summary> Trasforma un punto da un sistema di riferimento all'altro. </summary>
    /// <param name="coordSysFrom">Sistema di riferimento di partenza</param>
    /// <param name="coordSysTo">Sistema di riferimento di arrivo</param>
    /// <param name="x">Coordinata x del punto</param>
    /// <param name="y">Coordinata y del punto</param>
    /// <param name="err">Eventuale messaggio d'errore</param>
    /// <returns>Array di double (x,y) </returns>
    [WebMethod]
    public double[] TransformPoint(string coordSysFrom, string coordSysTo,double x,double y, ref string err)
        {
        string csfFrom = "", csfTo = "";
        double Xt = 0, Yt = 0;
        Array tmp = null;//new double[2];
        string from = coordSysFrom.Trim(); from = from.Replace(":","");
        string to = coordSysTo.Trim(); to = to.Replace(":","");
        csfFrom = Application["CoordSystemDir"] + from + ".csf";//sistema di coordinate di partenza
        csfTo = Application["CoordSystemDir"] + to + ".csf";//sistema di coordinate di partenza
        if(File.Exists(csfFrom))
            {
            if (File.Exists(csfTo))
                {
                MapSrvLib.GWMMapServerHelper MapSvr = null;
                MapSrvMngrLib.MapServerManagerClass MapSrvMngr = null;
                tmp = new double[2];
                  try
                   {
                    MapSrvMngr =(MapSrvMngrLib.MapServerManagerClass)
                        Server.CreateObject("GMWebMap.MapServerManager");
                    MapSvr = (MapSrvLib.GWMMapServerHelper) MapSrvMngr.MapServer("");
                    MapSvr.Clear();
                    #region Passaggio da Coordinate iniziali a Coordinate geografiche
                    MapSvr.SetCoordinateSystem(csfFrom);
                    tmp =(Array) MapSvr.TransformPoint(12,4,x,y);
                    Xt = double.Parse(tmp.GetValue(0).ToString()); Yt = double.Parse(tmp.GetValue(1).ToString());
                    #endregion
                    #region Passaggio da Coordinate geografoche a Coordinate finali
                    MapSvr.SetCoordinateSystem(csfTo);
                    tmp =(Array) MapSvr.TransformPoint(4,12,Xt,Yt);
                    #endregion
                    }
                catch(Exception er){err += er;tmp = null;}
                if(null!=MapSvr)System.Runtime.InteropServices.Marshal.ReleaseComObject(MapSvr);
                if(null!=MapSrvMngr)System.Runtime.InteropServices.Marshal.ReleaseComObject(MapSrvMngr);
               }
            else err+= "SystemCoordTransform:Unable to locate [" +coordSysTo + "] as Coordinate System destination.";
            }
        else err+= "SystemCoordTransform:Unable to locate [" +coordSysFrom + "] as Coordinate System start.";
        double[] res = new double[2];
        if(tmp!=null)   Array.Copy(tmp,res,2);
          else  res = null;
        return res;
        }

    /// <summary>
    /// Trasforma un vettore di punti da un sistema di riferimento all'altro.
    /// </summary>
    /// <param name="coordSysFrom">Sistema di riferimento di partenza</param>
```

```
105         /// <param name="coordSysTo">Sistema di riferimento di arrivo</param>
106         /// <param name="points">Vettore di punti della forma x1,y1,x2,y2,..</param>
107         /// <param name="err">Eventuale messaggio d'errore</param>
108         /// <returns>Vettore di punti della forma x1,y1,x2,y2,..</returns>
109         [WebMethod]
110         public double[] TransformPoints(string coordSysFrom, string coordSysTo,double[] points, ref string err)
111            {
112             string csfFrom = "", csfTo = "";
113             double Xt = 0, Yt = 0;
114             Array tmp = new double[2];
115             double[] res = null;
116             string from = coordSysFrom.Trim(); from = from.Replace(":","");
117             string to = coordSysTo.Trim(); to = to.Replace(":","");
118             csfFrom = Application["CoordSystemDir"] + from + ".csf";//sistema di coordinate di partenza
119             csfTo = Application["CoordSystemDir"] + to + ".csf";//sistema di coordinate di partenza
120             if(File.Exists(csfFrom))
121                {
122                 if (File.Exists(csfTo))
123                    {
124                     MapSrvLib.GWMMapServerHelper MapSvr = null;
125                     MapSrvMngrLib.MapServerManagerClass MapSrvMngr = null;
126                     err = "\n";
127                     tmp = new double[2];
128                     int i,l;
129                        try
130                         {
131                         MapSrvMngr =(MapSrvMngrLib.MapServerManagerClass)
132                         Server.CreateObject("GMWebMap.MapServerManager");
133                         MapSvr = (MapSrvLib.GWMMapServerHelper) MapSrvMngr.MapServer("");
134                         MapSvr.Clear();
135                         l = points.Length/2;
136                         #region Passaggio da Coordinate iniziali a Coordinate geografiche
137                         MapSvr.SetCoordinateSystem(csfFrom);
138                         for(i=0;i<=l;i+=2)
139                             {
140                             Xt = points[i]; Yt = points[i+1];
141                             err += i +":" + Xt + ";" + (i+1) + ":" + points[i+1] + "\n";
142                             tmp =(Array) MapSvr.TransformPoint(12,4,Xt,Yt);
143                             points[i] = double.Parse(tmp.GetValue(0).ToString());
144                             points[i+1] = double.Parse(tmp.GetValue(1).ToString());
145                            }
146                         #endregion
147                         #region Passaggio da Coordinate geografiche a Coordinate finali
148                         MapSvr.SetCoordinateSystem(csfTo);
149                         for(i=0;i<=l;i+=2)
150                             {
151                             Xt = points[i]; Yt = points[i+1];
152                             tmp =(Array) MapSvr.TransformPoint(4,12,Xt,Yt);
153                             points[i] = double.Parse(tmp.GetValue(0).ToString());
154                             points[i+1] = double.Parse(tmp.GetValue(1).ToString());
155                            }
156                         #endregion
157                         res = points;
158                         tmp = null;
159                        }
160                     catch(Exception er)
161                         {
162                         err = er.ToString();
163                         res = null;
164                        }
165                     if(null!=MapSvr)System.Runtime.InteropServices.Marshal.ReleaseComObject(MapSvr);
166                     if(null!=MapSrvMngr)System.Runtime.InteropServices.Marshal.ReleaseComObject(MapSrvMngr);
167                    }
168                 else err+= "SystemCoordTransform:Unable to locate [" +coordSysTo + "] as Coordinate System destination.";
169                }
170             else err+= "SystemCoordTransform:Unable to locate [" +coordSysFrom + "] as Coordinate System start.";
171             return res;
172            }
173         #endregion
174        }
175     }
```

Appendice
E Messaggistica

E.1. Modelli UML delle Classi

All'interno del namespace `RequesteResponseFormalizer`, sono definite le classi che modellano il meccanismo di serializzazione e deserializzazione utilizzato dai servizi GO-WS per manipolare i messaggi di richiesta e risposta.

E.1.1. Gestione Messaggi Di Richiesta

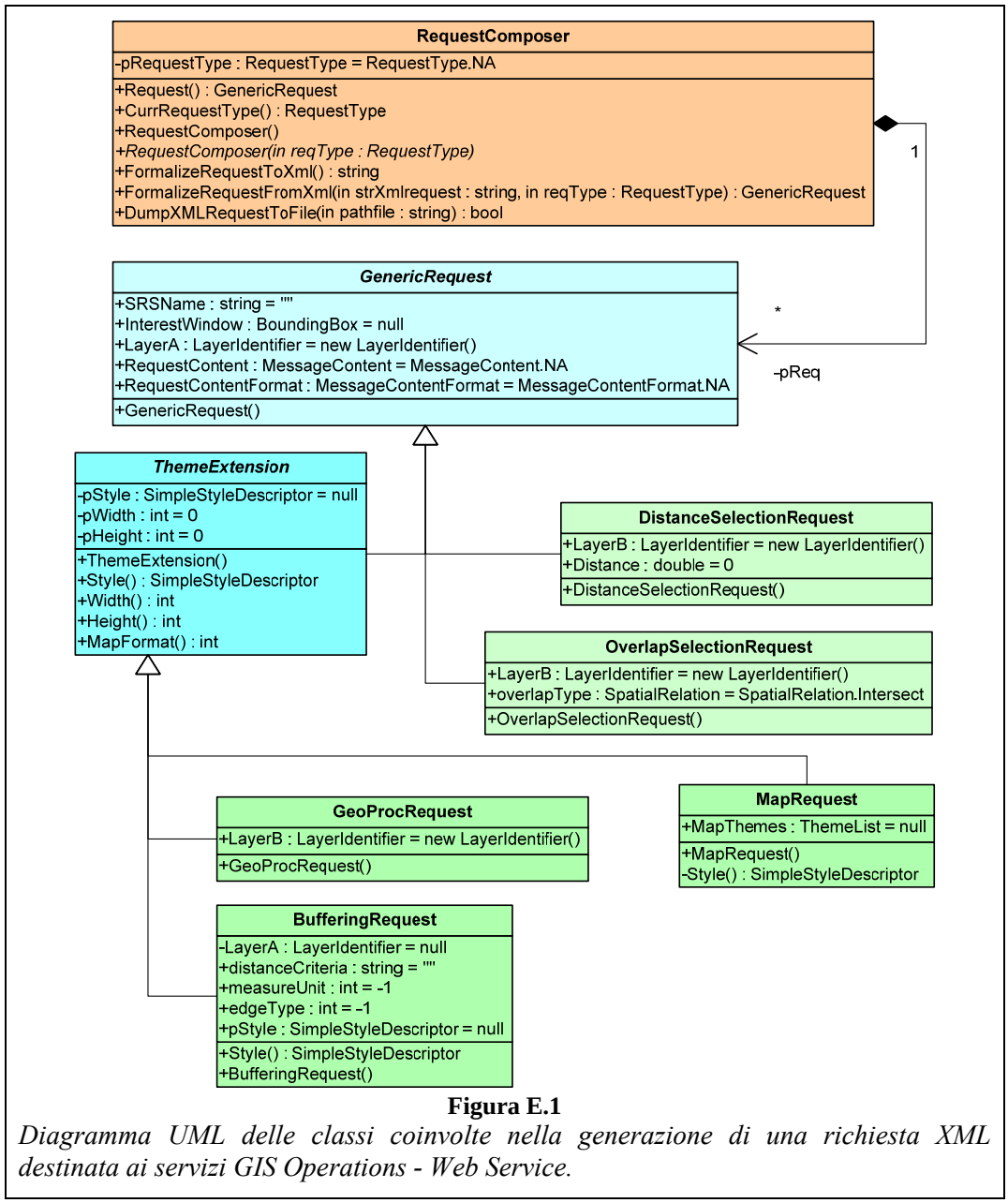


Figura E.1

Diagramma UML delle classi coinvolte nella generazione di una richiesta XML destinata ai servizi GIS Operations - Web Service.

Utilizzando esclusivamente la classe `RequestComposer` è possibile creare una richiesta sfruttando fra l'altro il polimorfismo dei tipi; la proprietà `Request`, infatti, è un puntatore alla classe astratta `GenericRequest`, figura E.2.

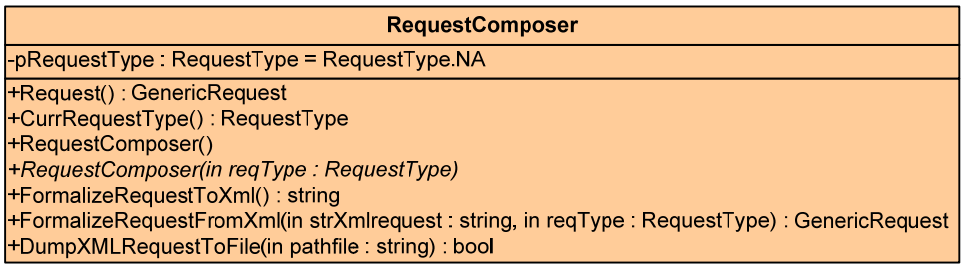


Figura E.2

Classe `RequestConposer`.

Il costruttore parametrico riceve il parametro `reqType` che indica il tipo di messaggio di richiesta, in base al suo valore viene istanziata la classe corrispondente. All'interno di questa il costruttore imposta il valore predefinito per il contenuto del messaggio e del formato. Tali valori sono descritti all'interno dei tipi enumerati `MessageContent` e `MessageContentFormat`, figura E.3.

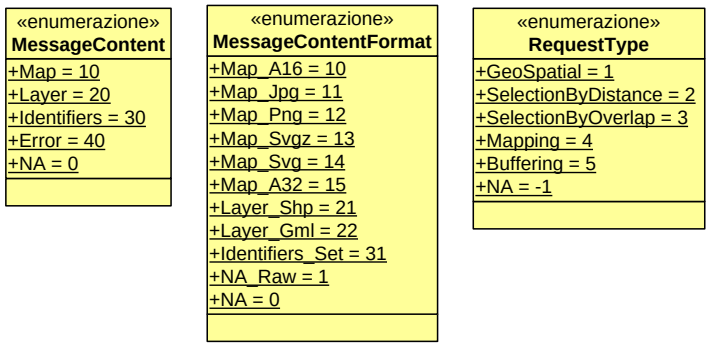


Figura E.3

Tipi enumerati, da sinistra: valori per il contenuto del messaggio; valori per i formati del contenuto; valori relativi al tipo di richiesta.

I valori del tipo enumerato `RequestType` corrispondono a determinate classi:

- `Buffering`: `BufferingRequest`, richiede l'operazione di *buffering*. L'attributo `edgeType` indica l'aspetto degli spigoli dell'area di interesse, i valori possibili sono definiti all'interno del tipo enumerato `EdgeType`, figura E.4;
- `GeoSpatial`: `GeoProcRequest`, richieste per operazioni di *geoprocessing*;
- `Mapping`: `MapRequest`, richiede la restituzione grafica (mappa) di un insieme di temi, descritti tramite la classe `ThemeList` (vedere appendice B);
- `SelectionByDistance`: `DistanceSelectionRequest`, selezione per tema in base alla distanza fra feature;
- `SelectionByOverlap`: `DistanceSelectionRequest`, selezione per tema in base ad un criterio di confronto spaziale non basato sulla distanza. L'attributo `overlapType` (figura E.1) indica il tipo di operatore spaziale. I valori di questo attributo sono definiti dal tipo enumerato `SpatialRelation`, figura E.4.

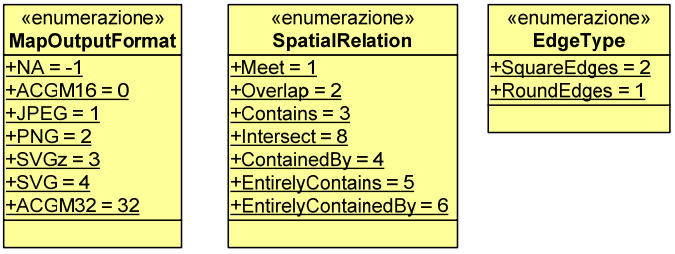


Figura E.4

Tipi enumerati, da sinistra;costanti per i formati grafici; costanti per gli operatori spaziali; costanti per l'aspetto dei bordi.

La classe astratta `ThemeExtesion` (figura E.1) estende una richiesta con attributi destinati alla restituzione grafica di dataset. In particolare, arricchisce la classe con il metodo `MapFormat`. Questo, in base al valore del formato del contenuto, restituisce una costante di GWM per identificare il formato grafico con cui restituire un dataset; tali costanti sono definite all'interno del tipo enumerato `MapOutputFormat` (figura E.4).

I valori definiti tramite i tipi enumerati `MapOutputFormat`, `SpatialRelation`, e `EdgeType` sono specifici di GWM.

I metodi per la creazione della richiesta, in formato XML o come oggetto in memoria, esposti dalla classe `RequestComposer` sono:

- `FormalizeRequestToXml`: una volta che la richiesta è stata compilata, questo metodo restituisce una stringa contenente il codice XML relativo alla richiesta;
- `FormalizeRequestFromXml`: deserializza la stringa XML passata come parametro e crea la classe relativa alla richiesta. Il metodo accetta il tipo di richiesta che ci si aspetta come valore definito in `RequestType`.

Esempio E.1	Creazione di una richiesta tramite `RequestComposer`

Creazione di una richiesta per trasformare il risultato di un'operazione di *geoprocessing* in un documento codificato in SVGZ:

```
//Namespace necessari
using RequestResponseFormalizer;
using GenericLayerIdentifier; //Per istanziare un nuovo identificatore di layer.
using StyleDescriptor; //Per lo stile di visualizzazione.

//Creazione messaggio di richiesta per operazione geospaziale.
RequestComposer reqComp = new RequestComposer(RequestType.GeoSpatial);

//Impostazione sistema di riferimento spaziale ed area di interesse:
reqComp.Request.SRSName = "UTM32";
reqComp.Request.InterestWindow = new BoundingBox(minx, miny, maxx, maxy);
//Primo operando da servizio WFS:

reqComp.Request.LayerA.dataSource = "http://.../WFSService.asp";
reqComp.Request.LayerA.dataSourceType = "WFS";
```

```
reqComp.Request.LayerA.additionalInfo = "";
reqComp.Request.LayerA.layerName = "Layer01";

//Secondo operando da servizio WMS:
((GeoProcRequest)reqComp.Request).LayerB.dataSource = "http://.../WMSservice.asp";
((GeoProcRequest)reqComp.Request).LayerB.dataSourceType = "WMS";
((GeoProcRequest)reqComp.Request).LayerB.layerName = "Category02";

//Creazione dello stile di visualizzazione del risultato:
((GeoProcRequest)reqComp.Request).Style = new SimpleStyleDescriptor();
((GeoProcRequest)reqComp.Request).Style.FillColor = "FFFF00";
((GeoProcRequest)reqComp.Request).Style.BorderColor = "#000000";
((GeoProcRequest)reqComp.Request).Style.BorderWidth = 2;

//Impostazione dimensione del documento grafico da produrre come risultato:
((GeoProcRequest)reqComp.Request).Height = 600;
((GeoProcRequest)reqComp.Request).Width = 800;

//Formato del contenuto (rappresentazione)
((GeoProcRequest)reqComp.Request).RequestContentFormat = MessageContentFormat.Map_Svgz;

//Generazione della richiesta in XML:
string richiesta = reqComp.FormalizeRequestToXml();
```

E.1.2. Gestione Messaggi Di Risposta

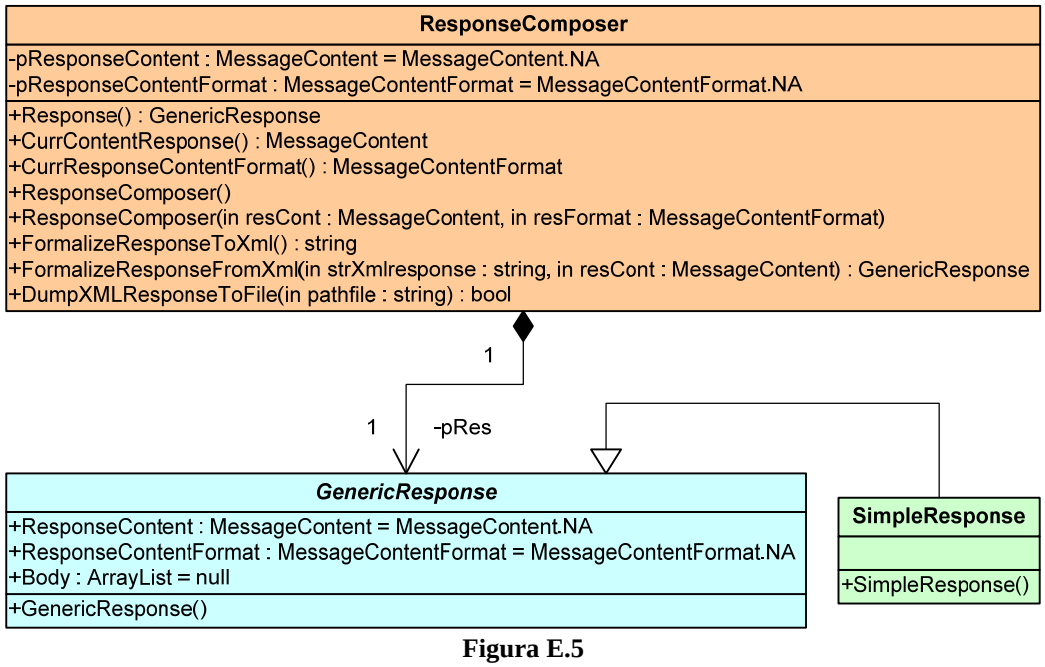


Figura E.5

Modello UML delle classi coinvolte nella generazione di una risposta XML destinata al client di uni servizio GIS Web Service.

Come accade per le richieste, anche le risposte sono create attraverso una classe apposita: `ResponseComposer`.

Il meccanismo di funzionamento è esattamente identico a `RequestComposer`, i metodi `FormalizeResponseToXml` e `FormalizeResponseFromXml` eseguono rispettivamente la serializzazione e deserializzazione delle risposte in XML.

La classe `GenericResponse` (figura E.5) modella il contenuto della risposta come una lista (di tipo `ArrayList`) di elementi che possono contenere URL, stringhe di testo per indicare errori oppure gli identificatori delle feature selezionate attraverso un'operazione di selezione per tema.

La classe `SimpleResponse` eredita dalla sua classe base senza estenderla; questa scelta è stata fatta per mantenere simmetria con il modello delle richieste e consentire future espansioni.

E.2. Schemi XML Dei Messaggi

Quelli che seguono sono gli schemi XML, codificati in XSD ,dei messaggi di richiesta.

I nomi degli schemi sono identici a quelli delle classi sopra illustrate.

BufferingRequest

```
<?xml version="1.0"?>
<xs:schema id="BufferingRequest"
targetNamespace="http://webgisserver.isti.cnr.it/GO-WS/messages/buffer_In.xsd"
xmlns:mstns="http://webgisserver.isti.cnr.it/GO-WS/messages/buffer_In.xsd"
xmlns="http://webgisserver.isti.cnr.it/GO-WS/messages/buffer_In.xsd"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:msdata="urn:schemas-microsoft-com:xml-msdata" attributeFormDefault="qualified"
elementFormDefault="qualified">
   <xs:element name="BufferingRequest">
      <xs:complexType>
      <xs:sequence>
         <xs:element name="SRSName" type="xs:string" />
         <xs:element name="RequestContent">
            <xs:simpleType>
               <xs:restriction base="xsd:string">
                  <xs:enumeration value="Map" />
                  <xs:enumeration value="Layer" />
                  <xs:enumeration value="Identifiers" />
                  <xs:enumeration value="NA" />
               </xs:restriction>
            </xs:simpleType>
         </xs:element>
         <xs:element name="RequestContentFormat">
            <xs:simpleType>
               <xs:restriction base="xsd:string">
               <xs:enumeration value="Map_A16" />
                  <xs:enumeration value="Map_A32" />
                  <xs:enumeration value="Map_Svg" />
                  <xs:enumeration value="Map_Svgz" />
                  <xs:enumeration value="Map_Png" />
                  <xs:enumeration value="Map_Jpg" />
                  <xs:enumeration value="Layer_Gml" />
                  <xs:enumeration value="Layer_Shp" />
                  <xs:enumeration value="Identifiers_Set" />
                  <xs:enumeration value="NA" />
               </xs:restriction>
            </xs:simpleType>
         </xs:element>
         <xs:element name="Width" type="xs:unsignedInt" minOccurs="0" />
         <xs:element name="Height" type="xs:unsignedInt" minOccurs="0" />
         <xs:element name="distanceCriteria" type="xs:string" />
         <xs:element name="measureUnit" type="xs:string" />
         <xs:element name="edgeType" type="xs:string" />
         <xs:element name="InterestWindow" minOccurs="0">
            <xs:complexType>
               <xs:sequence>
                  <xs:element name="xmin" type="xs:string" />
                  <xs:element name="ymin" type="xs:string" />
                  <xs:element name="xmax" type="xs:string" />
                  <xs:element name="ymax" type="xs:string" />
               </xs:sequence>
```

```
            </xs:complexType>
         </xs:element>
         <xs:element name="LayerA">
            <xs:complexType>
               <xs:sequence>
                  <xs:element name="dataSource" type="xs:string" />
                  <xs:element name="dataSourceType" type="xs:string" />
                  <xs:element name="layerName" type="xs:string" />
                  <xs:element name="additionalInfo" type="xs:string" minOccurs="0" />
               </xs:sequence>
            </xs:complexType>
         </xs:element>
         <xs:element name="Style" minOccurs="0" maxOccurs="1">
            <xs:complexType>
               <xs:sequence>
                  <xs:element name="FillColor" type="xs:string" />
                  <xs:element name="BorderColor" type="xs:string" />
                  <xs:element name="BorderWidth" type="xs:string" />
               </xs:sequence>
            </xs:complexType>
         </xs:element>
      </xs:sequence>
      </xs:complexType>
   </xs:element>
</xs:schema>
```

GeoProcRequest

```
<?xml version="1.0"?>
<xs:schema id="GeoProcRequest" targetNamespace="http://webgisserver.isti.cnr.it/GO-
WS/messages/GeoProcessing_In.xsd" xmlns:mstns="http://webgisserver.isti.cnr.it/GO-
WS/messages/GeoProcessing_In.xsd" xmlns="http://webgisserver.isti.cnr.it/GO-
WS/messages/GeoProcessing_In.xsd" xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:msdata="urn:schemas-microsoft-com:xml-msdata" attributeFormDefault="qualified"
elementFormDefault="qualified">
<xs:element name="GeoProcRequest">
   <xs:complexType>
      <xs:sequence>
         <xs:element name="SRSName" type="xs:string" />
         <xs:element name="RequestContent">
            <xs:simpleType>
               <xs:restriction base="xsd:string">
                  <xs:enumeration value="Map" />
                  <xs:enumeration value="Layer" />
                  <xs:enumeration value="Identifiers" />
                  <xs:enumeration value="NA" />
               </xs:restriction>
            </xs:simpleType>
         </xs:element>
         <xs:element name="RequestContentFormat">
            <xs:simpleType>
               <xs:restriction base="xsd:string">
                  <xs:enumeration value="Map_A16" />
                  <xs:enumeration value="Map_A32" />
                  <xs:enumeration value="Map_Svg" />
                  <xs:enumeration value="Map_Svgz" />
                  <xs:enumeration value="Map_Png" />
                  <xs:enumeration value="Map_Jpg" />
                  <xs:enumeration value="Layer_Gml" />
                  <xs:enumeration value="Layer_Shp" />
                  <xs:enumeration value="Identifiers_Set" />
                  <xs:enumeration value="NA" />
               </xs:restriction>
            </xs:simpleType>
         </xs:element>
         <xs:element name="Width" type="xs:unsignedInt" minOccurs="0" />
         <xs:element name="Height" type="xs:unsignedInt" minOccurs="0" />
         <xs:element name="InterestWindow" minOccurs="0">
            <xs:complexType>
               <xs:sequence>
                  <xs:element name="xmin" type="xs:string" />
                  <xs:element name="ymin" type="xs:string" />
                  <xs:element name="xmax" type="xs:string" />
                  <xs:element name="ymax" type="xs:string" />
               </xs:sequence>
            </xs:complexType>
```

```
        </xs:element>
        <xs:element name="LayerA">
           <xs:complexType>
              <xs:sequence>
                 <xs:element name="dataSource" type="xs:string" />
                 <xs:element name="dataSourceType" type="xs:string" />
                 <xs:element name="layerName" type="xs:string" />
                 <xs:element name="additionalInfo" type="xs:string" minOccurs="0" />
              </xs:sequence>
           </xs:complexType>
        </xs:element>
        <xs:element name="Style" minOccurs="0" maxOccurs="1">
           <xs:complexType>
              <xs:sequence>
                 <xs:element name="FillColor" type="xs:string" />
                 <xs:element name="BorderColor" type="xs:string" />
                 <xs:element name="BorderWidth" type="xs:string" />
              </xs:sequence>
           </xs:complexType>
        </xs:element>
        <xs:element name="LayerB">
           <xs:complexType>
              <xs:sequence>
                 <xs:element name="dataSource" type="xs:string" />
                 <xs:element name="dataSourceType" type="xs:string" />
                 <xs:element name="layerName" type="xs:string" />
                 <xs:element name="additionalInfo" type="xs:string" />
              </xs:sequence>
           </xs:complexType>
        </xs:element>
      </xs:sequence>
      </xs:complexType>
   </xs:element>
</xs:schema>
```

MapRequest

```
<?xml version="1.0"?>
<xs:schema id ="MapRequest"
targetNamespace="http://webgisserver.isti.cnr.it/GO-WS/messages/RenderMap_In.xsd"
xmlns:mstns="http://webgisserver.isti.cnr.it/GO-WS/messages/RenderMap_In.xsd"
xmlns="http://webgisserver.isti.cnr.it/GO-WS/messages/RenderMap_In.xsd"
xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:msdata="urn:schemas-microsoft-com:xml-
msdata" attributeFormDefault="qualified" elementFormDefault="qualified">
   <xs:element name="MapRequest">
      <xs:complexType>
         <xs:sequence>
            <xs:element name="SRSName" type="xs:string" />
            <xs:element name="RequestContent">
               <xs:simpleType>
                  <xs:restriction base="xsd:string">
                     <xs:enumeration value="Map" />
                     <xs:enumeration value="Layer" />
                     <xs:enumeration value="Identifiers" />
                     <xs:enumeration value="NA" />
                  </xs:restriction>
               </xs:simpleType>
            </xs:element>
            <xs:element name="RequestContentFormat">
             <xs:simpleType>
                <xs:restriction base="xsd:string">
                   <xs:enumeration value="Map_A16" />
                   <xs:enumeration value="Map_A32" />
                   <xs:enumeration value="Map_Svg" />
                   <xs:enumeration value="Map_Svgz" />
                   <xs:enumeration value="Map_Png" />
                   <xs:enumeration value="Map_Jpg" />
                   <xs:enumeration value="Layer_Gml" />
                   <xs:enumeration value="Layer_Shp" />
                   <xs:enumeration value="Identifiers_Set" />
                   <xs:enumeration value="NA" />
                </xs:restriction>
             </xs:simpleType>
            </xs:element>
            <xs:element name="Width" type="xs:unsignedInt" />
            <xs:element name="Height" type="xs:unsignedInt" />
```

```
            <xs:element name="InterestWindow" minOccurs="0">
               <xs:complexType>
                  <xs:sequence>
                     <xs:element name="xmin" type="xs:string" />
                     <xs:element name="ymin" type="xs:string" />
                     <xs:element name="xmax" type="xs:string" />
                     <xs:element name="ymax" type="xs:string" />
                  </xs:sequence>
               </xs:complexType>
            </xs:element>
            <xs:element name="MapThemes">
               <xs:complexType>
                  <xs:sequence minOccurs="1" maxOccurs="unbounded">
                     <xs:element name="MapThemeDescriptor">
                        <xs:complexType>
                           <xs:sequence>
                              <xs:element name="dataSource" type="xs:string" />
                              <xs:element name="dataSourceType" type="xs:string" />
                              <xs:element name="layerName" type="xs:string" />
                              <xs:element name="additionalInfo" type="xs:string"
                              minOccurs="0"/>
                              <xs:element name="Level" type="xs:negativeInteger" />
                              <xs:element name="Style">
                                 <xs:complexType>
                                    <xs:sequence>
                                       <xs:element name="FillColor" type="xs:string" />
                                       <xs:element name="BorderColor" type="xs:string" />
                                       <xs:element name="BorderWidth" type="xs:string" />
                                    </xs:sequence>
                                 </xs:complexType>
                              </xs:element>
                           </xs:sequence>
                        </xs:complexType>
                     </xs:element>
                  </xs:sequence>
               </xs:complexType>
            </xs:element>
         </xs:sequence>
      </xs:complexType>
   </xs:element>
</xs:schema>
```

DistanceSelectionRequest

```
<?xml version="1.0"?>
<xs:schema id="DistanceSelectionRequest"
targetNamespace="http://webgisserver.isti.cnr.it/GO-
WS/messages/SelectionByDistance_In.xsd" xmlns:mstns="http://webgisserver.isti.cnr.it/GO-
WS/messages/SelectionByDistance_In.xsd" xmlns="http://webgisserver.isti.cnr.it/GO-
WS/messages/SelectionByDistance_In.xsd" xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:msdata="urn:schemas-microsoft-com:xml-msdata" attributeFormDefault="qualified"
elementFormDefault="qualified">
   <xs:element name="DistanceSelectionRequest">
      <xs:complexType>
         <xs:sequence>
            <xs:element name="RequestContent">
               <xs:simpleType>
                  <xs:restriction base="xsd:string">
                     <xs:enumeration value="Map" />
                     <xs:enumeration value="Layer" />
                     <xs:enumeration value="Identifiers" />
                     <xs:enumeration value="NA" />
                  </xs:restriction>
               </xs:simpleType>
            </xs:element>
            <xs:element name="RequestContentFormat">
               <xs:simpleType>
                  <xs:restriction base="xsd:string">
                     <xs:enumeration value="Map_A16" />
                     <xs:enumeration value="Map_A32" />
                     <xs:enumeration value="Map_Svg" />
                     <xs:enumeration value="Map_Svgz" />
                     <xs:enumeration value="Map_Png" />
                     <xs:enumeration value="Map_Jpg" />
                     <xs:enumeration value="Layer_Gml" />
                     <xs:enumeration value="Layer_Shp" />
```

```
                  <xs:enumeration value="Identifiers_Set" />
                  <xs:enumeration value="NA" />
               </xs:restriction>
            </xs:simpleType>
         </xs:element>
         <xs:element name="Distance" type="xs:double" />
         <xs:element name="InterestWindow" minOccurs="0">
            <xs:complexType>
               <xs:sequence>
                  <xs:element name="xmin" type="xs:string" />
                  <xs:element name="ymin" type="xs:string" />
                  <xs:element name="xmax" type="xs:string" />
                  <xs:element name="ymax" type="xs:string" />
               </xs:sequence>
            </xs:complexType>
         </xs:element>
         <xs:element name="LayerA">
            <xs:complexType>
               <xs:sequence>
                  <xs:element name="dataSource" type="xs:string" />
                  <xs:element name="dataSourceType" type="xs:string" />
                  <xs:element name="layerName" type="xs:string" />
                  <xs:element name="additionalInfo" type="xs:string" minOccurs="0" />
               </xs:sequence>
            </xs:complexType>
         </xs:element>
         <xs:element name="LayerB">
            <xs:complexType>
               <xs:sequence>
                  <xs:element name="dataSource" type="xs:string" />
                  <xs:element name="dataSourceType" type="xs:string" />
                  <xs:element name="layerName" type="xs:string" />
                  <xs:element name="additionalInfo" type="xs:string" minOccurs="0" />
               </xs:sequence>
            </xs:complexType>
         </xs:element>
      </xs:sequence>
   </xs:complexType>
</xs:element>
</xs:schema>
```

OverlapSelectionRequest

```
<?xml version="1.0" ?>
<xs:schema id ="OverlapSelectionRequest"
targetNamespace="http://webgisserver.isti.cnr.it/GO-
WS/messages/SelectionByOverlap_In.xsd"
xmlns:mstns="http://webgisserver.isti.cnr.it/GO-WS/messages/SelectionByOverlap_In.xsd"
xmlns="http://webgisserver.isti.cnr.it/GO-WS/messages/SelectionByOverlap_In.xsd"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:msdata="urn:schemas-microsoft-com:xml-msdata" attributeFormDefault="qualified"
elementFormDefault="qualified">
   <xs:element name="OverlapSelectionRequest">
      <xs:complexType>
         <xs:sequence>
            <xs:element name="SRSName" type="xs:string" />
            <xs:element name="RequestContent">
               <xs:simpleType>
                  <xs:restriction base="xsd:string">
                     <xs:enumeration value="Map" />
                     <xs:enumeration value="Layer" />
                     <xs:enumeration value="Identifiers" />
                     <xs:enumeration value="NA" />
                  </xs:restriction>
               </xs:simpleType>
            </xs:element>
            <xs:element name="RequestContentFormat">
               <xs:simpleType>
                  <xs:restriction base="xsd:string">
                     <xs:enumeration value="Map_A16" />
                     <xs:enumeration value="Map_A32" />
                     <xs:enumeration value="Map_Svg" />
                     <xs:enumeration value="Map_Svgz" />
                     <xs:enumeration value="Map_Png" />
                     <xs:enumeration value="Map_Jpg" />
                  <xs:enumeration value="Layer_Gml" />
```

```
                  <xs:enumeration value="Layer_Shp" />
               <xs:enumeration value="Identifiers_Set" />
            <xs:enumeration value="NA" />
               </xs:restriction>
            </xs:simpleType>
         </xs:element>
         <xs:element name="overlapType">
            <xs:simpleType>
               <xs:restriction base="xsd:string">
                  <xs:enumeration value="meet" />
                  <xs:enumeration value="overlap" />
                  <xs:enumeration value="contains" />
                  <xs:enumeration value="intersect" />
                  <xs:enumeration value="containedby" />
                  <xs:enumeration value="entirelycontainedby" />
                  <xs:enumeration value="entirelycontains" />
               </xs:restriction>
            </xs:simpleType>
         </xs:element>
         <xs:element name="InterestWindow" minOccurs="0">
            <xs:complexType>
               <xs:sequence>
                  <xs:element name="xmin" type="xs:string" />
                  <xs:element name="ymin" type="xs:string" />
                  <xs:element name="xmax" type="xs:string" />
                  <xs:element name="ymax" type="xs:string" />
               </xs:sequence>
            </xs:complexType>
         </xs:element>
         <xs:element name="LayerA">
            <xs:complexType>
               <xs:sequence>
                  <xs:element name="dataSource" type="xs:string" />
                  <xs:element name="dataSourceType" type="xs:string" />
                  <xs:element name="layerName" type="xs:string" />
                  <xs:element name="additionalInfo" type="xs:string" minOccurs="0" />
               </xs:sequence>
            </xs:complexType>
         </xs:element>
         <xs:element name="LayerB">
            <xs:complexType>
               <xs:sequence>
                  <xs:element name="dataSource" type="xs:string" />
                  <xs:element name="dataSourceType" type="xs:string" />
                  <xs:element name="layerName" type="xs:string" />
                  <xs:element name="additionalInfo" type="xs:string" minOccurs="0" />
               </xs:sequence>
            </xs:complexType>
         </xs:element>
      </xs:sequence>
   </xs:complexType>
</xs:element>
</xs:schema>
```

Di seguito è riportato lo schema dei messaggi di risposta prodotti dai servizi GO-WS; il nome dello schema, come per i precedenti, conincide con quello della classe impiegata per generare i messaggi di risposta.

SimpleResponse

```
<?xml version="1.0"?>
<xs:schema id="SimpleResponse"
targetNamespace="http://webgisserver.isti.cnr.it/GO-WS/messages/SimpleResponse.xsd"
xmlns:mstns="http://webgisserver.isti.cnr.it/GO-WS/messages/SimpleResponse.xsd"
xmlns="http://webgisserver.isti.cnr.it/GO-WS/messages/SimpleResponse.xsd"
xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:msdata="urn:schemas-microsoft-com:xml-
msdata" attributeFormDefault="qualified" elementFormDefault="qualified">
   <xs:element name="SimpleResponse">
      <xs:complexType>
         <xs:sequence>
            <xs:element name="RequestContent">
               <xs:simpleType>
                  <xs:restriction base="xsd:string">
                     <xs:enumeration value="Map" />
```

```
                    <xs:enumeration value="Layer" />
                    <xs:enumeration value="Identifiers" />
                    <xs:enumeration value="NA" />
                    <xs:enumeration value="Error" />
                </xs:restriction>
            </xs:simpleType>
        </xs:element>
        <xs:element name="RequestContentFormat">
            <xs:simpleType>
                <xs:restriction base="xsd:string">
                    <xs:enumeration value="Map_A16" />
                    <xs:enumeration value="Map_A32" />
                    <xs:enumeration value="Map_Svg" />
                    <xs:enumeration value="Map_Svgz" />
                    <xs:enumeration value="Map_Png" />
                    <xs:enumeration value="Map_Jpg" />
                    <xs:enumeration value="Layer_Gml" />
                    <xs:enumeration value="Layer_Shp" />
                    <xs:enumeration value="Identifiers_Set" />
                    <xs:enumeration value="NA" />
                    <xs:enumeration value="NA_Raw" />
                </xs:restriction>
            </xs:simpleType>
        </xs:element>
        <xs:element name="Body">
            <xs:complexType>
                <xs:sequence minOccurs="1" maxOccurs="unbounded">
                    <xs:element name="anyType" type="xs:anyType" nillable="true" />
                </xs:sequence>
            </xs:complexType>
        </xs:element>
    </xs:sequence>
  </xs:complexType>
 </xs:element>
</xs:schema>
```

In questo ultimo schema, sono stati evendenziati il contenuto ed il rispettivo formato di un messaggio di risposta che trasporta un messaggio di errore.

E.3. Codice Namespace RequestResponseFormalizer

```
using System;
using System.IO;
using System.Xml;
using System.Xml.Serialization;
using System.Collections;
//******Namespace Interni *********
using StyleDescriptor;
using GenericLayerIdentifier;

///<Author>Giulio Massei</Author>
///<LastUpdate>20/10/2005</LastUpdate>

namespace RequestResponseFormalizer
{
    #region Classe formalizzatore
    ///<summary>
    /// Gestisce la "Formalizzazione" di una richiesta.
    /// Per "Fromalizzazione" si intende la possibilità di tradurre una
    /// richiesta in un formalismo equivalente.
    ///</summary>
    ///<remarks>
    /// L'unico formalismo gestito è lo "XML".
    ///</remarks>
    public class RequestComposer
    {
        #region Campi privati
        ///<summary>
        /// Indica il tipo di richiesta che si sta costruendo.
        ///</summary>
        ///<remarks>
        /// Campo privato.
        ///</remarks>
        private RequestType pRequestType = RequestType.NA;

        ///<summary>
        /// Oggetto richiesta
        ///</summary>
        ///<remarks>
        /// Campo privato il suo tipo dipende dall'operazione che si vuole effettuare.
        ///</remarks>
        private GenericRequest pReq = null;
        #endregion

        #region Proprietà
        ///<summary>
        /// Restituisce la richiesta che si stà formalizzando.
        ///</summary>
        public GenericRequest Request
        {
            get
            {
                switch (pRequestType)
                {
                    case RequestType.GeoSpatial: return (GeoProcRequest) this.pReq;
                    case RequestType.SelectionByDistance: return (DistanceSelectionRequest) this.pReq;
                    case RequestType.SelectionByOverlap: return (OverlapSelectionRequest) this.pReq;
                    case RequestType.Mapping: return (MapRequest) this.pReq;
                    case RequestType.Buffering: return (BufferingRequest) this.pReq;
                    default: return null;
                }
            }
        }


        ///<summary>
        /// Proprietà in sola lettura; indica il tipo di richiesta che si sta formalizzando.
        ///</summary>
        ///
        public RequestType CurrRequestType
        {
            get{return this.pRequestType;}
        }

        #endregion

        #region Costruttori
        ///<summary>
        /// Costruttore predefinito
        ///</summary>
        public RequestComposer(){}

        ///<summary>
        /// Costruttore parametrizzato.
        ///</summary>
        ///<param name="reqType">Tipo di richiesta</param>
        public RequestComposer(RequestType reqType)
        {
            switch (reqType)
            {
                case RequestType.GeoSpatial: this.pReq = new GeoProcRequest(); break;
                case RequestType.SelectionByDistance: this.pReq = new DistanceSelectionRequest(); break;
                case RequestType.SelectionByOverlap: this.pReq = new OverlapSelectionRequest(); break;
                case RequestType.Mapping: this.pReq = new MapRequest(); break;
                case RequestType.Buffering: this.pReq = new BufferingRequest(); break;
                default: this.pReq = null;break;
            }
            this.pRequestType = reqType;
        }
        #endregion
```

```
        #region Formalizzatori
        ///<summary>
        /// Restituisce la richiesta in formato XML
        ///</summary>
        ///<returns>Stringa che contiene la richiesta formalizzata in XML.</returns>
        public string FormalizeRequestToXml()
        {
            System.IO.MemoryStream ms = new MemoryStream();
            XmlSerializer s = null;
            try
            {
                switch (this.pRequestType)
                {
                    case RequestType.GeoSpatial:
                        s = new XmlSerializer(typeof(GeoProcRequest));
                        s.Serialize(ms,(GeoProcRequest)this.pReq); break;
                    case RequestType.SelectionByDistance:
                        s = new XmlSerializer(typeof(DistanceSelectionRequest));
                        s.Serialize(ms,(DistanceSelectionRequest)this.pReq); break;
                    case RequestType.SelectionByOverlap:
                        s = new XmlSerializer(typeof(OverlapSelectionRequest));
                        s.Serialize(ms,(OverlapSelectionRequest)this.pReq); break;
                    case RequestType.Buffering:
                        s = new XmlSerializer(typeof(BufferingRequest));
                        s.Serialize(ms,(BufferingRequest)this.pReq); break;
                    case RequestType.Mapping:
                        s = new XmlSerializer(typeof(MapRequest));
                        s.Serialize(ms,(MapRequest)this.pReq); break;
                    default: ms = null; break;
                }
                s = null;
                byte[] t = null;
                t = ms.GetBuffer();
                int i=0;
                string tt = "";
                for(i=0;i<t.Length;i++) tt+= (char)t[i];
                ms.Close(); ms = null;
                System.Xml.XmlDocument p = new XmlDocument();
                p.LoadXml(tt);
                tt = p.OuterXml;p = null;
                return tt;
            }
            catch(Exception e)
            {return "";}
        }

        ///<summary>
        /// Converte una richiesta formalizzata in xml in una classe "Request"
        ///</summary>
        ///<param name="strXmlrequest">richiesta formalizzata in xml.</param>
        ///<param name="reqType">Tipo di richiesta.</param>
        ///<returns>Oggetto di tipo richiesta inerente al tipo selezionato.</returns>
        ///<remarks>
        /// La richiesta coinvertita diventa quella attuale.
        ///</remarks>
        public GenericRequest FormalizeRequestFromXml(string strXmlrequest, RequestType reqType)
        {
            XmlSerializer s = null;
            XmlTextReader xtr = new XmlTextReader(strXmlrequest,XmlNodeType.Document,null);
            GenericRequest gt = null;

            switch (reqType)
            {
                case RequestType.GeoSpatial:
                    s = new XmlSerializer(typeof(GeoProcRequest));
                    gt = (GeoProcRequest) s.Deserialize(xtr); break;
                case RequestType.SelectionByDistance:
                    s = new XmlSerializer(typeof(DistanceSelectionRequest));
                    gt = (DistanceSelectionRequest) s.Deserialize(xtr); break;
                case RequestType.SelectionByOverlap:
                    s = new XmlSerializer(typeof(OverlapSelectionRequest));
                    gt = (OverlapSelectionRequest) s.Deserialize(xtr); break;
                case RequestType.Mapping:
                    s = new XmlSerializer(typeof(MapRequest));
                    gt = (MapRequest) s.Deserialize(xtr); break;
                case RequestType.Buffering:
                    s = new XmlSerializer(typeof(BufferingRequest));
                    gt = (BufferingRequest) s.Deserialize(xtr); break;
                default: gt = null; break;
            }
            s = null;
            xtr.Close(); xtr = null;
            this.pReq = gt;
            return gt;
        }
        #endregion

        #region Metodi Pubblici
        ///<summary>
        /// Restituisce la richiesta in formato XML su di un file
        ///</summary>
        ///<param name="pathfile">Percorso e nome del file. L'estensione è opzionale.</param>
        ///<remarks>L'estensione del file prodotto è sempre ".xml".</remarks>
        ///<returns>Stringa che contiene la richiesta formalizzata in XML.</returns>
        public bool DumpXMLRequestToFile(string pathfile)
        {
            if(pathfile.ToLower().IndexOf(".xml")<0) pathfile+=".xml";
            System.IO.StreamWriter ms = new StreamWriter(pathfile);//
            XmlSerializer s = null;
            try
            {
                switch (this.pRequestType)
                {
                    case RequestType.GeoSpatial:
                        s = new XmlSerializer(typeof(GeoProcRequest));
                        s.Serialize(ms,(GeoProcRequest)this.pReq); break;
```

```
                    case RequestType.SelectionByDistance:
                        s = new XmlSerializer(typeof(DistanceSelectionRequest));
                        s.Serialize(ms,(DistanceSelectionRequest)this.pReq); break;
                    case RequestType.SelectionByOverlap:
                        s = new XmlSerializer(typeof(OverlapSelectionRequest));
                        s.Serialize(ms,(OverlapSelectionRequest)this.pReq); break;
                    case RequestType.Buffering:
                        s = new XmlSerializer(typeof(BufferingRequest));
                        s.Serialize(ms,(BufferingRequest)this.pReq); break;
                    case RequestType.Mapping:
                        s = new XmlSerializer(typeof(MapRequest));
                        s.Serialize(ms,(MapRequest)this.pReq); break;
                    default: ms = null;return false;
                }
                return true;
            }
            catch(Exception e)
            {return false;}
        }
        #endregion
    }
    #endregion

    #region Classi che definiscono una richiesta

    ///<summary>
    /// Classe astratta, definisce una generica richiesta
    ///</summary>
    [Serializable()]
    public abstract class GenericRequest
    {
        #region Attributi pubblici

        ///<summary>
        /// Identificatore del sistema di riferimento
        ///</summary>
        public string SRSName ="";

        ///<summary>
        /// Area di interesse, definita tramite on oggetto °BoundingBox°.
        ///</summary>
        public BoundingBox InterestWindow = null;

        ///<summary>
        /// Identificatore di layer.
        ///</summary>
        public LayerIdentifier LayerA = new LayerIdentifier();

        ///<summary>
        /// Definisce il contenuto della richiesta: Mappa, recordset, o layer
        ///</summary>
        public MessageContent RequestContent = MessageContent.NA;

        ///<summary>
        /// Indica il formato del contenuto della richiesta. Ad esempio per una Mappa: SVG, JPEG, ...
        ///</summary>
        public MessageContentFormat RequestContentFormat = MessageContentFormat.NA;

        #endregion

        #region Costruttori

        ///<summary>
        /// Costruttore predefinito.
        ///</summary>
        public GenericRequest(){}
        #endregion
    }

    ///<summary>
    /// Fornisce supporto alla generazione in formato grafico di un recordset.
    ///</summary>
    [Serializable()]
    public abstract class ThemeExtension : GenericRequest
    {
        public ThemeExtension()
        {
            if(this.pStyle!=null)
                if((""==this.pStyle.FillColor) && (""==this.pStyle.BorderColor) & (0==this.pStyle.BorderWidth))
                    this.pStyle = null;
        }

        #region Proprietà pubbliche
        private SimpleStyleDescriptor pStyle = null;
        private int pWidth = 0;
        private int pHeight = 0;

        ///<summary>
        /// Stile di visualizzazione del layer prodotto.
        ///</summary>
        public SimpleStyleDescriptor Style
        {
            set{if(null!=value) this.pStyle = value;}
            get{return this.pStyle;}
        }

        ///<summary>
        /// Larghezza in pixel dell'oggetto grafico.
        ///</summary>
        public int Width
        {
            set{this.pWidth = value;}
            get{return this.pWidth;}
        }

        ///<summary>
```

```
            /// Altezza in pixel dell'oggetto grafico.
            ///</summary>
            public int Height
            {
                set{this.pHeight = value;}
                get{return this.pHeight;}
            }
            #endregion

            #region Metodi pubblici
            ///<summary>
            /// restituisce la costante di GWM che rappresenta il
            /// formato di restituzione della mappa: SVG, JPG,...
            ///</summary>
            public int MapFormat()
            {
                return GWMMApConst.MessageContentFormat2GWMMapFormatI(this.RequestContentFormat);
            }
            #endregion
        }

        ///<summary>
        /// Definisce una richiesta per operazioni geospaziali.
        ///</summary>
        ///<remarks>
        /// Classe serializzabile.
        ///</remarks>
        [Serializable()]
        public class GeoProcRequest : ThemeExtension
        {
            #region Attributi pubblici
            ///<summary>
            /// Secondo layer per le operazioni.
            ///</summary>
            public LayerIdentifier LayerB = new LayerIdentifier();
            #endregion

            #region Costruttori
            ///<summary>
            /// Costruttore predefinito
            ///</summary>
            ///<remarks>
            /// Per default, un'operazione geospaziale ritorna un dataset
            /// codificato in GML.
            ///</remarks>
            public GeoProcRequest()
            {
                this.RequestContent = MessageContent.Layer;
                this.RequestContentFormat = MessageContentFormat.Layer_Gml;
            }
            #endregion
        }

        ///<summary>
        /// Definisce una richiesta per operazioni di selezione per distanza.
        ///</summary>
        ///<remarks>
        /// Classe serializzabile.
        ///</remarks>
        [Serializable()]
        public class DistanceSelectionRequest : GenericRequest
        {
            #region Attributi pubblici
            ///<summary>
            /// Secondo layer per le operazioni.
            ///</summary>
            public LayerIdentifier LayerB = new LayerIdentifier();

            ///<summary>
            /// Indica la distanza fra le feature dei due layer
            ///</summary>
            public double Distance = 0;
            #endregion

            #region Costruttori
            ///<summary>
            /// Costruttore predefinito
            ///</summary>
            public DistanceSelectionRequest()
            {
                this.RequestContent = MessageContent.Identifiers;
                this.RequestContentFormat = MessageContentFormat.Identifiers_Set;
            }
            #endregion
        }

        ///<summary>
        /// Definisce una richiesta per operazioni di selezione tramite sovrapposizione
        /// (overlap).</summary>
        ///<remarks>
        /// Classe serializzabile.
        ///</remarks>
        [Serializable()]
        public class OverlapSelectionRequest : GenericRequest
        {
            #region Attributi pubblici
            ///<summary>
            /// Secondo layer per le operazioni.
            ///</summary>
            public LayerIdentifier LayerB = new LayerIdentifier();

            ///<summary>
            /// Indica la relazione spaziale di sovrapposizione: intersezione, contenimento,...
            ///</summary>
            public SpatialRelation overlapType = SpatialRelation.Intersect;
            #endregion
```

```
419
420             #region Costruttori
421             ///<summary>
422             /// Costruttore predefinito
423             ///</summary>
424             public OverlapSelectionRequest()
425             {
426                 this.RequestContent = MessageContent.Identifiers;
427                 this.RequestContentFormat = MessageContentFormat.Identifiers_Set;
428             }
429             #endregion
430         }
431
432         ///<summary>
433         /// Definisce una richiesta per operazioni di buffering.
434         ///</summary>
435         ///<remarks>
436         /// Classe serializzabile.
437         ///</remarks>
438         [Serializable()]
439         public class BufferingRequest : ThemeExtension
440         {
441             new private LayerIdentifier LayerA = null;
442
443             #region Attributi pubblici
444             ///<summary>
445             /// Il criterio di distanza utilizzato per il buffering può essere sia un
446             /// insieme di distanze oppure il nome di un campo numerico usato per
447             /// definire la distanza fra le feature ed il bordo del buffer.
448             ///</summary>
449             ///<remarks>
450             /// Bisogna passare valori double con il carattere "." come separatore decimale .
451             /// In caso di distanze fisse, se ne possono avere più di una andando a
452             /// definire delle fasce passando alla prorpietà una stringa così formata:
453             /// "0:100;100:230" dove ":" separa il valore minimo e massimo della fascia e
454             /// ";" separa le fasce. Per un valore singolo, specificarlo normalmente: "1000.12".
455             ///</remarks>
456             public string distanceCriteria = "";
457
458             ///<summary>
459             /// Identificativo dell'unità di misura (secondo GWM) rispetto al sistema di riferimento.
460             ///</summary>
461             public int measureUnit = -1;
462
463             ///<summary>
464             /// Aspetto dei bordi, che possono essere arrotondati o squadrati.
465             ///</summary>
466             public int edgeType =-1;
467             #endregion
468
469             #region Proprietà pubbliche
470             public SimpleStyleDescriptor pStyle = null;
471
472             ///<summary>
473             /// Stile di visualizzazione del buffer
474             ///</summary>
475             public SimpleStyleDescriptor Style
476             {
477                 set{if(null!=value) this.pStyle = value;}
478                 get{return this.pStyle;}
479             }
480             #endregion
481
482             #region Costruttori
483             ///<summary>
484             /// Costruttore predefinito
485             ///</summary>
486             public BufferingRequest()
487             {
488                 this.LayerA = null;
489                 this.RequestContent = MessageContent.Layer;
490                 this.RequestContentFormat = MessageContentFormat.Layer_Gml;
491             }
492             #endregion
493         }
494
495         ///<summary>
496         /// Definisce una richiesta per restituzione cartografica di layer.
497         ///</summary>
498         ///<remarks>
499         /// Classe serializzabile.
500         ///</remarks>
501         [Serializable()]
502         public class MapRequest : ThemeExtension
503         {
504             #region Attributi pubblici
505
506             ///<summary>Lista di temi che costituiranno la mappa.</summary>
507             public GenericLayerIdentifier.ThemeList MapThemes = null;
508             #endregion
509
510             #region Costruttori
511             ///<summary>Costruttore predefinito</summary>
512             public MapRequest()
513             {this.RequestContent = MessageContent.Map;this.RequestContentFormat = MessageContentFormat.Map_Svg;}
514             #endregion
515
516             new private SimpleStyleDescriptor Style{get{return null;}}
517         }
518         #endregion
519     }
```

```
using System;
using System.Collections;
using System.IO;
using System.Xml;
using System.Xml.Serialization;

///<Author>Giulio Massei</Author>
///<LastUpdate>20/10/2005</LastUpdate>

namespace RequestResponseFormalizer
{

    #region Formalizzatore
    ///<summary>Descrizione di riepilogo per Response.</summary>
    public class ResponseComposer
    {
        #region Campi
        ///<summary>Indica il tipo della risposta.</summary>
        ///<remarks>Campo privato.</remarks>
        private MessageContent pResponseContent = MessageContent.NA;

        ///<summary>Indica il formato della risposta.</summary>
        ///<remarks>Campo privato.</remarks>
        private MessageContentFormat pResponseContentFormat = MessageContentFormat.NA;

        ///<summary>Oggetto richiesta</summary>
        ///<remarks>Campo privato il suo tipo dipende dall'operazione che si vuole effettuare.</remarks>
        private GenericResponse pRes = null;
        #endregion

        #region Proprietà

        ///<summary>Restituisce la risposta.</summary>
        public GenericResponse Response{get{return (SimpleResponse) this.pRes;}}

        ///<summary>Indica il contenuto della risposta.</summary>
        public MessageContent CurrContentResponse{get{return this.pResponseContent;}}

        ///<summary>Indica il formato della risposta. </summary>
        public MessageContentFormat CurrResponseContentFormat{get{return this.pResponseContentFormat;}}
        #endregion

        #region Costruttori
        public ResponseComposer(){}

        public ResponseComposer(MessageContent resCont, MessageContentFormat resFormat)
        {
            switch (resCont)
            {
                case MessageContent.NA: break;
                default:
                            this.pRes = new SimpleResponse();this.pRes.ResponseContent = resCont;
                            this.pRes.ResponseContentFormat = resFormat;break;
            }
            this.pResponseContent = resCont;this.pResponseContentFormat = resFormat;
        }
        #endregion

        #region Formalizzatori
        ///<summary>Restituisce la risposta in formato XML</summary>
        ///<returns>Stringa che contiene la richiesta formalizzata in XML.</returns>
        public string FormalizeResponseToXml()
        {
            int i=0;string tt = "";byte[] t = null;XmlSerializer s = null; System.IO.MemoryStream ms = new MemoryStream();
            try
            {
                switch (this.pResponseContent)
                {
                    case MessageContent.NA:ms = null; return null;
                    default:
                        s = new XmlSerializer(typeof(SimpleResponse));s.Serialize(ms,(SimpleResponse)this.pRes); break;
                }
                s = null;t = ms.GetBuffer();ms.Close(); ms = null;
                for(i=0;i<t.Length;i++) tt+= (char)t[i];
                System.Xml.XmlDocument p = new XmlDocument();p.LoadXml(tt);tt = p.OuterXml;p = null;return tt;
            }
            catch{return "";}
        }

        ///<summary>Converte una risposta formalizzata in xml in una classe "Response"</summary>
        ///<param name="strXmlresponse">Risposta formalizzata in xml.</param>
        ///<param name="reqType">Tipo di risposta.</param>
        ///<returns>Oggetto di tipo Response inerente al tipo selezionato.</returns>
        ///<remarks>La risposta coinvertita diventa quella attuale.</remarks>
        public GenericResponse FormalizeResponseFromXml(string strXmlresponse, MessageContent resCont)
        {
            XmlSerializer s = null;
            XmlTextReader xtr = new XmlTextReader(strXmlresponse,XmlNodeType.Document,null);SimpleResponse sr = null;
            switch (resCont)
            {
                case MessageContent.NA: sr = null; break;
                default:
                    s = new XmlSerializer(typeof(SimpleResponse));
                    sr = (SimpleResponse) s.Deserialize(xtr); this.pResponseContent = sr.ResponseContent;
                    this.pResponseContentFormat = sr.ResponseContentFormat; break;
            }
            s = null;xtr.Close(); xtr = null;this.pRes = sr;return sr;
        }
        #endregion

        #region Metodi pubblici
        ///<summary>Restituisce la risposta in formato XML su di un file. </summary>
        ///<param name="pathfile">Percorso e nome del file. L'estensione è opzionale.</param>
        ///<remarks>L'estensione del file prodotto è sempre ".xml".</remarks>
        ///<returns>Stringa che contiene la risposta formalizzata in XML.</returns>
        public bool DumpXMLResponseToFile(string pathfile)
```

```
107             {
108                 XmlSerializer s = null;
109                 if(pathfile.ToLower().IndexOf(".xml")<0) pathfile+=".xml";
110                 System.IO.StreamWriter ms = new StreamWriter(pathfile);
111                 try
112                 {
113                     switch (this.pResponseContent)
114                     {
115                         case MessageContent.NA:ms = null; return false;
116                         default:
117                             s = new XmlSerializer(typeof(SimpleResponse));
118                             s.Serialize(ms,(SimpleResponse)this.pRes); break;
119                     }
120                     s = null;ms = null;return true;
121                 }
122                 catch(Exception e){return false;}
123             }
124             #endregion
125         }
126         #endregion
127 
128         #region Classi che definiscono una risposta
129         ///<summary>Classe astratta che definisce una risposta generica. </summary>
130         public abstract class GenericResponse
131         {
132             #region Attributi pubblici
133             ///<summary> Definisce il contenuto della risposta: Mappa, recordset, o layer</summary>
134             public MessageContent ResponseContent = MessageContent.NA;
135 
136             ///<summary> Indica il formato del contenuto della risposta. Ad esempio per una Mappa: SVG, JPEG, </summary>
137             public MessageContentFormat ResponseContentFormat = MessageContentFormat.NA;
138 
139             ///<summary> Corpo della risposta, insieme di elementi</summary>
140             public ArrayList Body = null;
141             #endregion
142 
143             #region Costruttori
144             ///<summary>Costruttore predefinito</summary>
145             public GenericResponse(){}
146             #endregion
147         }
148 
149         ///<summary> Classe derivata da °GenericResponse°; definisce una risposta semplice.</summary>
150         public class SimpleResponse: GenericResponse
151         {public SimpleResponse(){}}
152         #endregion
153     }
```

Appendice **F**

Local Catalogue Service

F.1. Meta Information Repository

La trasformazione dello schema ad oggetti, figura F.1a, in quello entita-relazioni (ER), figura F.1b, introduce i seguenti attributi:

- `Sid`, chiave esterna in `Tbl_Layers`,
- `Lid`, chiave esterna in `Tbl_SRSBox`.

Il diagramma ER illustrato figura F.1, riporta solo le chiavi primarie, le chiavi esterne e la trasformazione dell'attributo strutturato MBR in quattro elementi semplici.

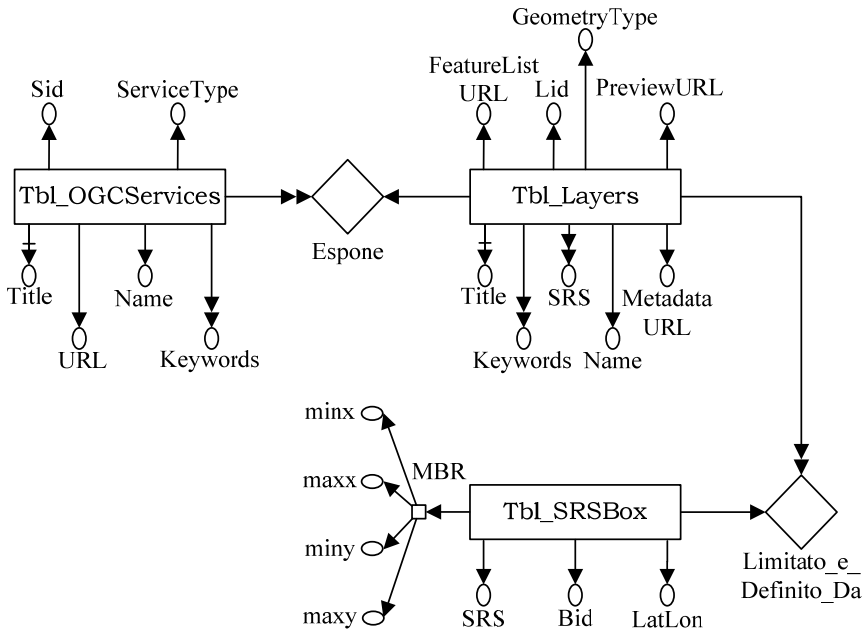


Figura F.1a

Schema ad oggetti del Meta Information Repository.

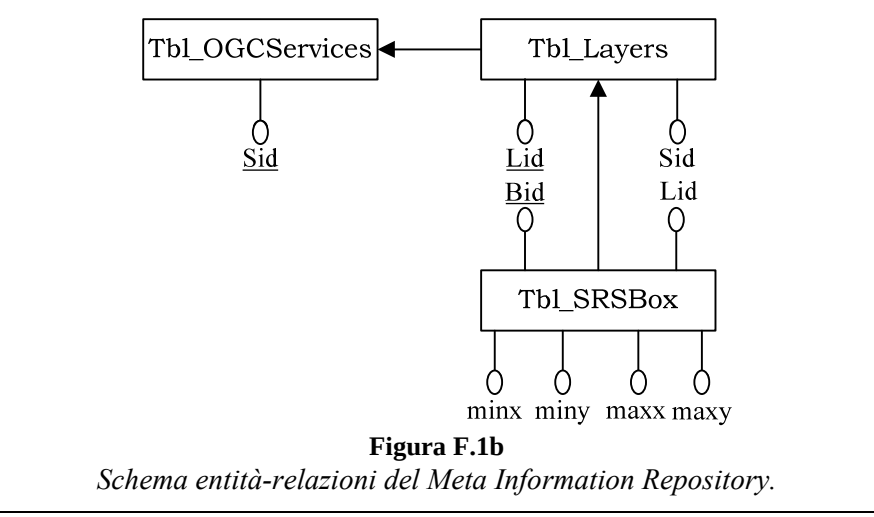


Figura F.1b

Schema entità-relazioni del Meta Information Repository.

F.2. Modello UML di ServiceCrawler

Il namespace `ServiceCrawler` può essere diviso in quattro sezioni contenenti:

- classi astratte necessarie per modellare la struttura generale dei servizi OGC;
- classi per modellare il documento delle capacità dei servizi WFS;
- classi per modellare il documento delle capacità dei servizi WMS;
- collezioni di classi.

Questa sezione illustra solamente metodi, proprietà ed attributi pubblici; per la descrizione degli elementi non accessibili pubblicamente.

F.2.1. Classi Astratte

La classe `ServiceClass` modella le proprietà generiche legate alle istanze di servizio, come: namespace XML del documento delle capacità, URL, tipo di istanza (`IstanceType`) e versione del servizio (figura F.2).

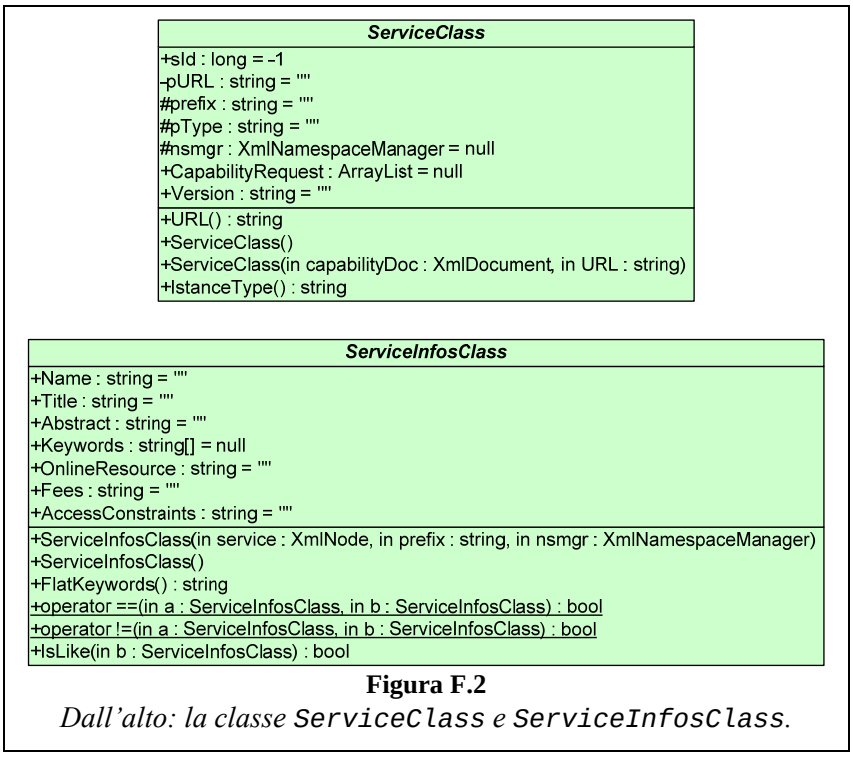


Figura F.2

Dall'alto: la classe `ServiceClass` e `ServiceInfosClass`.

La classe `ServiceInfosClass` descrive tutti gli attributi del documento delle capacità, relativi al servizio, ed include la proprietà di sola lettura `FlatKeywords` per restituire una stringa contenente la concatenazione dei termini chiave separati dal carattere ";".

I layer dei servizi WFS e WMS possiedono meta informazioni comuni che vengono modellati attraverso la classe `LayerClass` (figura F.3).

La classe `LayerClass` comprende:

- `MetadataURLClass`: un descrittore che consente di recuperare i metadati associati; i metadati sono indirizzati attraverso una URL e descritti secondo un tipo di codifica predefinito (Dublin Core, FDGC, CEN/TC, …);
- `MBRClass`: per descrivere il *minimum bounding box* associato al layer.

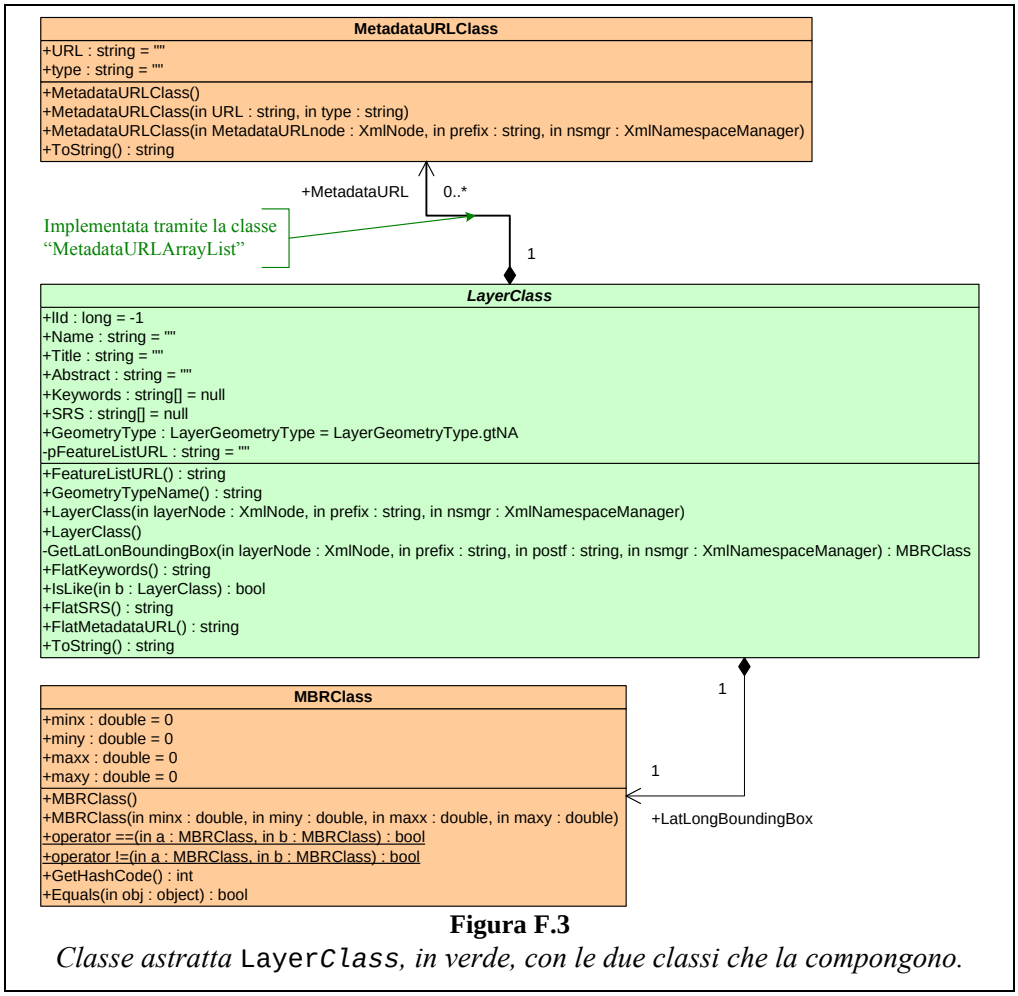


Figura F.3

Classe astratta `LayerClass`*, in verde, con le due classi che la compongono.*

La proprietà `GeometryTypeName`, associata all'attributo `GeometryType`, fornisce una descrizione testuale del tipo di geometria del layer; è utilizzata solo a fini di rappresentazione.

Il metodo `IsLike()` consente di stabilire se un dato layer "è come" un altro; tale criterio definisce un'uguaglianza basata sulle informazioni che vengono memorizzate nel catalogo MIR. Questo operatore di confronto è basilare per poter eseguire la sincronizzazione delle meta informazioni memorizzate all'interno del catalogo.

F.2.2. Classi per Web Map Service

Il diagramma sotto riportato (figura F.4) mostra le classi che permettono la manipolazione del documento delle capacità per le istanze del servizio WMS.

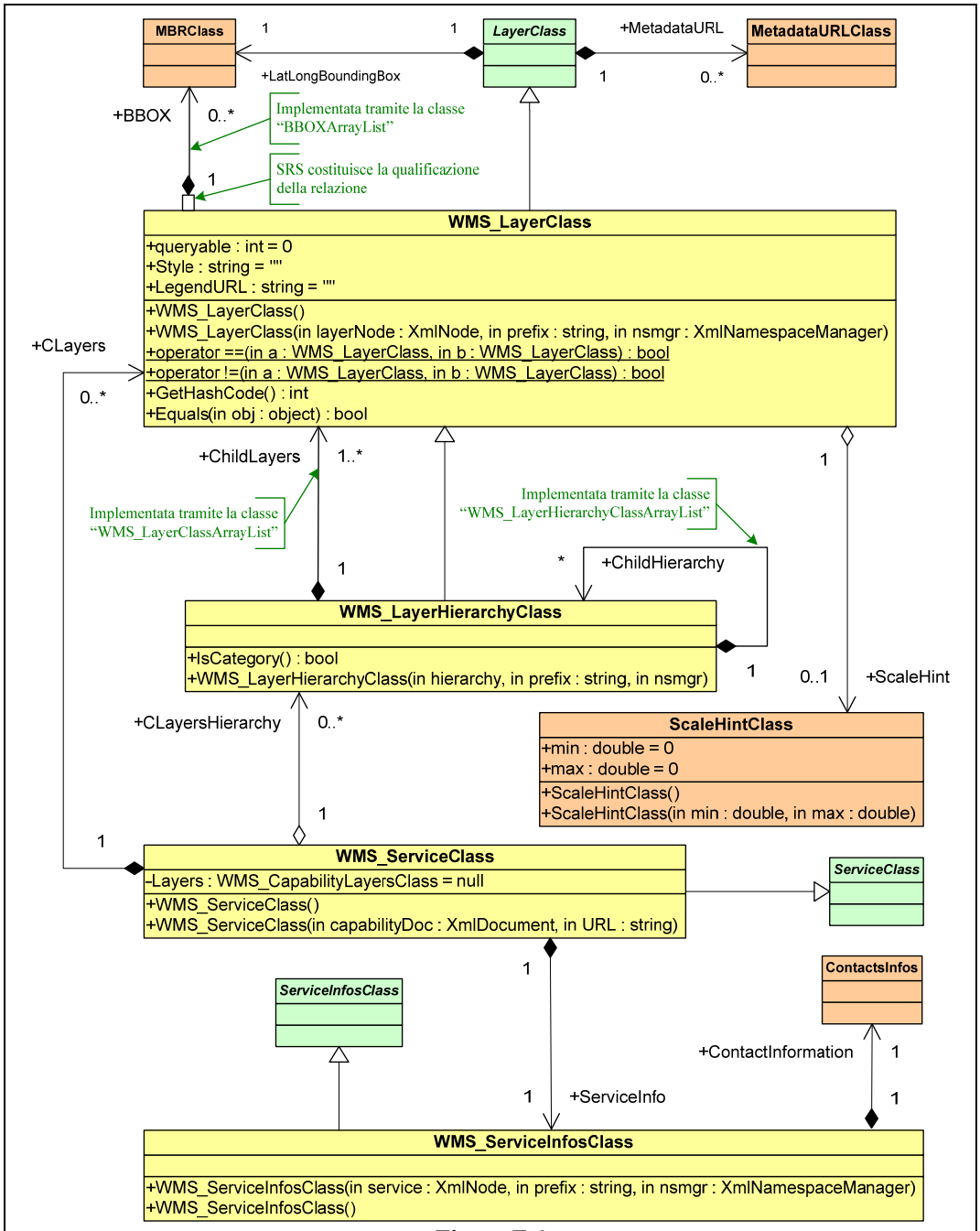


Figura F.4

Diagramma UML per le classi necessarie a rappresentare un documento delle capacità per un servizio WMS.

La classe `WMS_LayerClass` estende `LayerClass` aggiungendo i seguenti attributi:

- `LegendURL` che contiene la URL della legenda associata al layer;
- `Style`: stringa che descrive gli stili associati;
- `queryable`: indica che è possibile accedere alla feature GML tramite interrogazioni;

- `ScaleHint`: indica l'intervallo di scala in cui il layer è visibile.

Oltre a queste proprietà è stata aggiunta un'associazione per mettere in corrispondenza un layer con il relativo *box* di contenimento, modellata attraverso la classe `MBRClass`; come mostrato in figura F.4, l'associazione è qualificata dalla proprietà `SRS`, visto che ogni *box* di contenimento è definito in un sistema di riferimento spaziale.

Facendo riferimento a quanto detto nel capitolo 2.1, i layer di un WMS sono organizzati in categorie; per questo motivo è stata creata la classe `WMS_LayerHierarchyClass`. Come illustrato in figura F.4, essa eredita da `WMS_LayerClass` in quanto una categoria (con nome) può essere considerata a tutti gli effetti come un layer. Inoltre, all'interno di una categoria ve ne possono essere annidate altre. La collezione `ChildHierarchy` permette di strutturare i layer secondo una gerarchia ad albero. Per poter accedere a tale struttura, la classe `WMS_ServiceInfosClass` espone le seguenti proprietà:

- `CLayers`: riferimento alla collezione dei layer del primo livello;
- `CLayersHierarchy`: riferimento all'insieme delle categorie del primo livello.

F.2.3. Classi per Web Feature Service

Il diagramma sotto riportato (figura F.4) mostra le classi per la manipolazione del documento delle capacità per le istanze del servizio WFS.

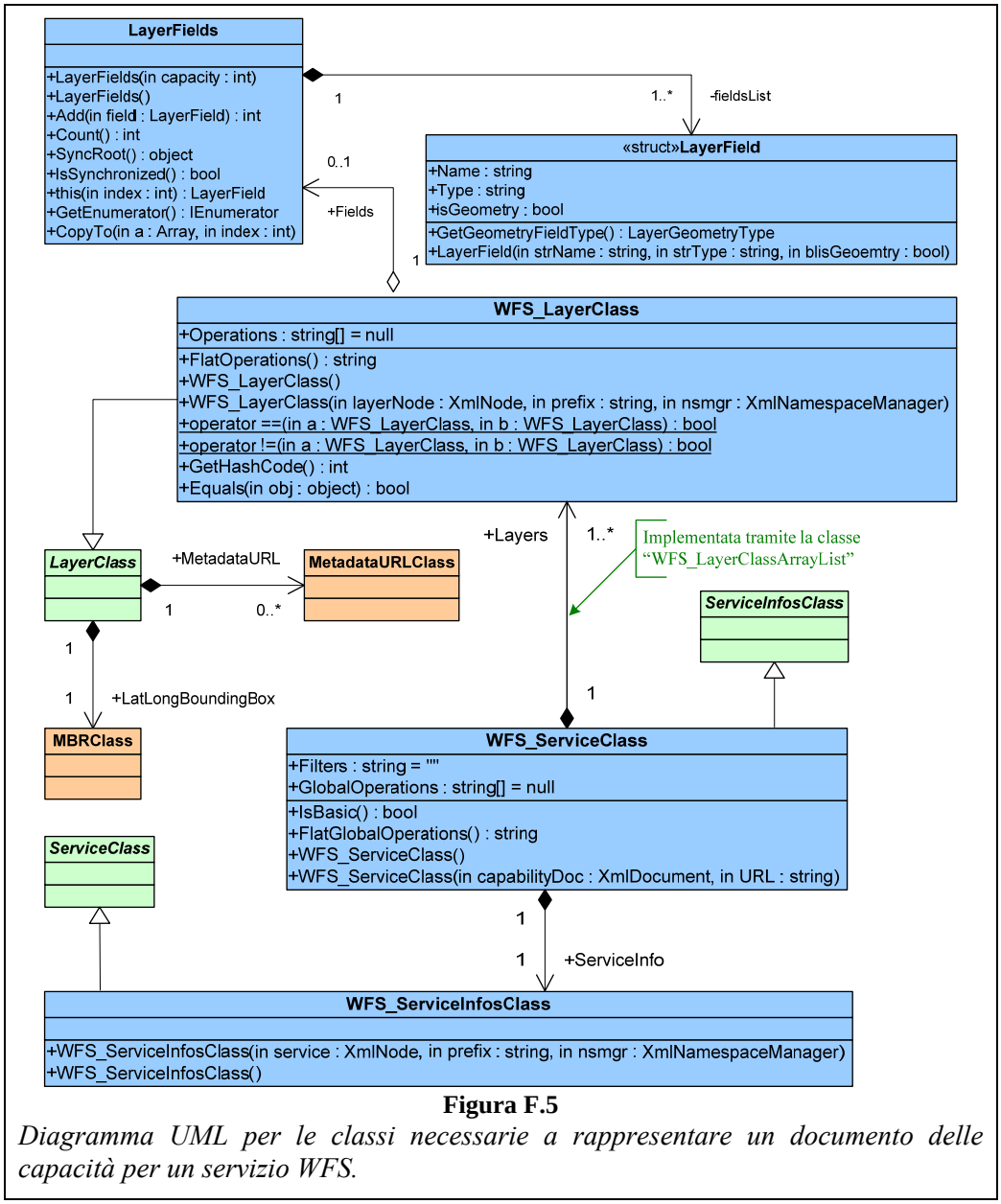


Figura F.5

Diagramma UML per le classi necessarie a rappresentare un documento delle capacità per un servizio WFS.

Rispetto alla classe base, `WFS_ServiceClass` descrive le operazioni comuni a tutti i layer oltre che ad indicare se il servizio si tratta di un *Transaction WFS* oppure un *Basic WFS*.

A differenza del servizio WMS, i layer sono strutturati linearmente senza annidamenti, inoltre, per ogni layer è mantenuta la lista degli attributi con associato il rispettivo tipo.

F.2.4. Collezioni

Come mostrato nei diagrammi UML precedenti, alcune associazioni sono state modellate mediante collezioni specifiche definite, all'interno del namespace `OWServiceCrawler.TypedArrayList`.

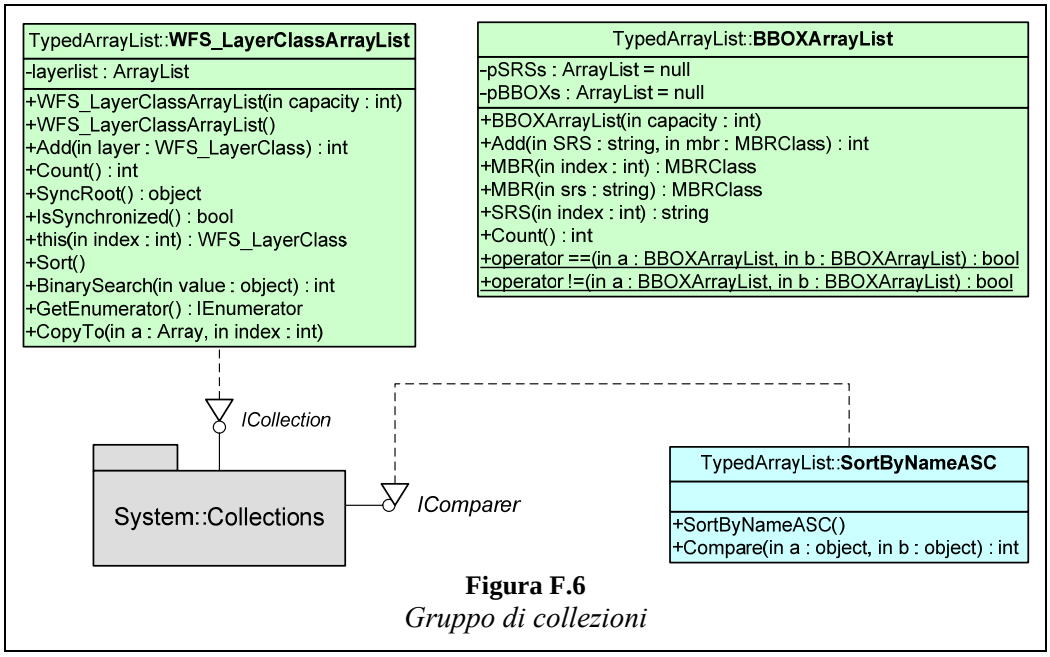


Figura F.6
Gruppo di collezioni

Per esigenze implementative, le collezioni:

- implementano l'interfaccia `ICollection` (figura F.6):
  - `WFS_LayerClassArrayList` modella le collezioni di layer appartenenti a servizi WFS. La classe `SortByNameASC` consente di creare un criterio di ordinamento ascendente, su oggetti di tipo `WMS_LayerClass`, basato sulla proprietà al loro `Name`;
- Classe `ArrayList` (igura F.7):
  - `WMS_LayerClassArrayList` è utilizzata per definire le collezione di layer WMS;
  - `WFS_LayerHierarchyClassArrayList`: definisce una categoria;
  - `MetadataURLArrayList`: Lista di riferimenti a metadati relativi al layer.

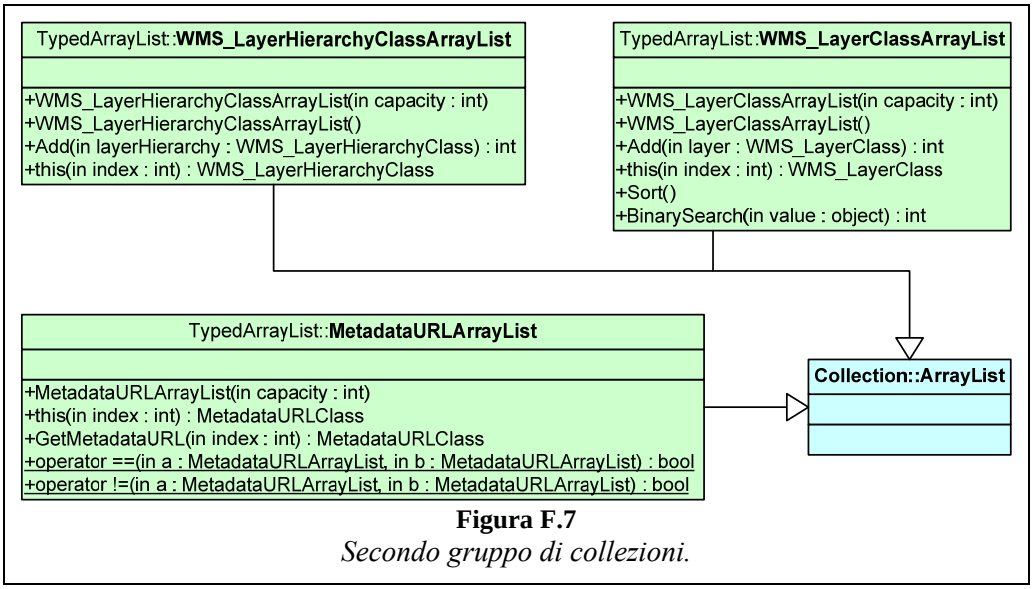


Figura F.7

Secondo gruppo di collezioni.

La classe `BBOXArrayList`, figura F.6, è utilizzata per modellare l'associazione che mette in relazione la *box* di contenimento di un layer con il sistema di riferimento spaziale.

F.3. Modello UML di CoreCatalogueInterface

Il namespace `CoreCatalogueInterface` definisce al suo interno le classi necessarie all'implementazione del gestore del catalogo MIR.

La classe primaria che gestisce il catalogo è chiamata `CoreManager` (figura F.8) ed espone tutti i metodi necessari alla diretta manipolazione del catalogo.

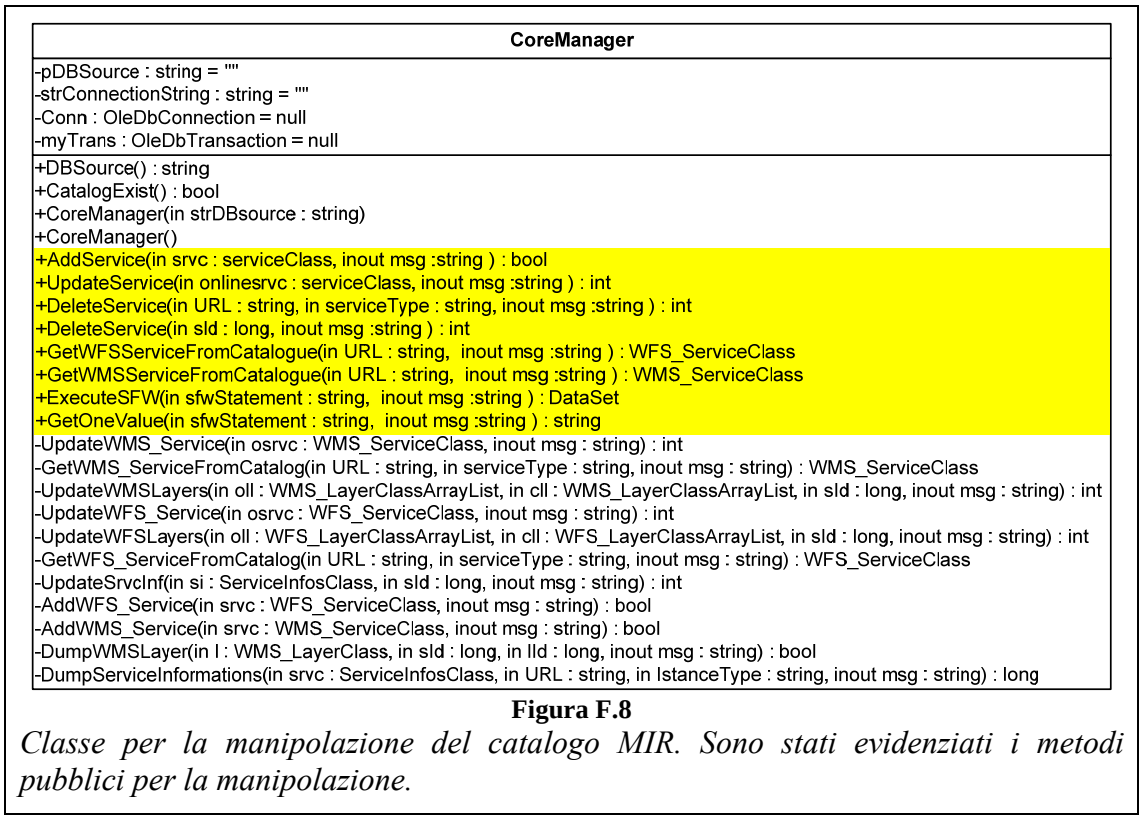


Figura F.8

Classe per la manipolazione del catalogo MIR. Sono stati evidenziati i metodi pubblici per la manipolazione.

Il costruttore parametrico consente di specificare il percorso del catalogo strutturato secondo lo schema MIR (figura F.1b).

Le operazioni di inserimento, aggiornamento (sincronizzazione) ed eliminazione delle meta informazioni relative ad un'istanza di servizio WFS o WMS, sono possibili mediante i metodi `AddService`, `UpdateService` e `DeleteService`. Quest'ultimo metodo permette l'eliminazione mediante la URL dell'istanza oppure l'identificatore univoco dell'istanza, all'interno del catalogo.

La classe prevede un metodo per poter estrarre informazioni dal catalogo mediante il costrutto SQL "SELECT-FROM-WHERE" eseguendo il metodo `ExecuteSFW`. Il risultato è rappresentato attraverso la classe `DataSet` del namespace `System.Data` del framework .NET. È presente inoltre, un metodo ottimizzato per restituire un solo valore, `GetOneValue`, che esegue la selezione analogamente al metodo `ExecuteSFW`.

I metodi `GetWFSServiceFromCatalog` e `GetWMSServiceFromCatalog` consentono di restituire, rispettivamente, la rappresentazione dei documenti delle capacità

delle istanze dei servizi WFS e WMS contenute nel catalogo, descritte mediante le rispettive classi `WFS_ServiceClass` e `WMS_ServiceClass`.

Pe la gestione dei documenti delle capacità per i servizi WMS è stata creata una classe apposita, `WMS_Stuffs` (figura F.9).

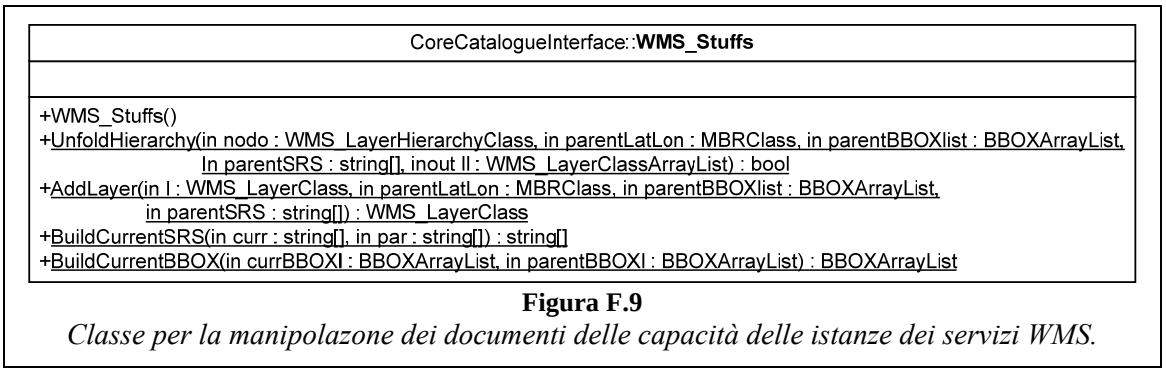


Figura F.9

Classe per la manipolazone dei documenti delle capacità delle istanze dei servizi WMS.

La classe espone il metodo `UnfoldHierarchy` che ha il compito di "srotolare" ricorsivamente una categoria (Category) od un Layer Map in modo tale da avere, anziché un insieme di layer strutturati gerarchicamente, una semplice lista di layer. Questo perché il catalogo non consente di rappresentare categorie o Layer Map. Il risultato sarà quindi una lista rappresentata mediante la collezione `WMS_LayerClassArrayList`.

Ogni layer della collezione è corredato da tutti i sistema di riferimento e box di contenimento erditati dai nodi padri. Questo grazie ai rispettivi metodi ausiliari `BuildCurrentSRS` e `BuildCurrentBBOX` che sono invocati ad ogni passo di ricorsione per costruire le liste temporanee di box di contenimento e sistemi di riferimento.

Una volta che le liste sono completate, l'invocazione del metodo `AddLayer` memorizza il layer con le informazioni "srotolate" all'interno della lista di tipo `WMS_LayerClassArrayList`, passata come parametro al metodo `UnfoldHierarchy`.

F.4. Local Catalogue Service

Per completezza, di seguito (figura F.10) è riportato il digramma UML delle classi che realizzano il servizio `LocalCatalogueService`.

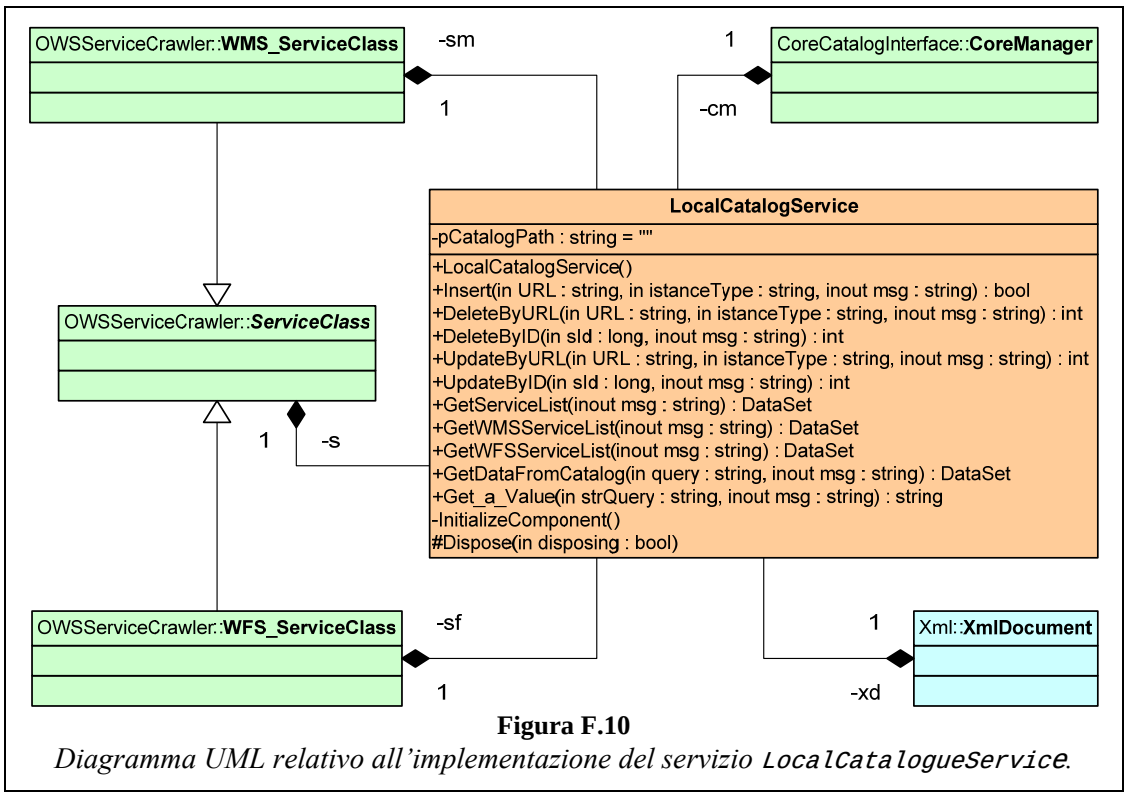


Figura F.10

Diagramma UML relativo all'implementazione del servizio `LocalCatalogueService`.

F.5. Codice Namespace OWSServiceCrawler

```
using System;
using System.Xml;

namespace OWSServiceCrawler
{
    ///<class>LayerClass</class>
    /// <summary>
    /// Rappresenta la definizione generale di un Layer.
    /// </summary>
    /// <remarks>
    /// Classe Astratta. Utilizzata per derivare le classi <see cref="WFS_LayerClass">WFS_LayerClass</see>
    /// e <see cref="WMS_LayerClass">WMS_LayerClass</see>.
    /// </remarks>
    public abstract class LayerClass
    {
        #region Proprietà & attributi pubblic
        public long lId = -1;

        /// <summary>Elemento <Name>.</summary>
        public string Name = "";

        /// <summary>Elemento <Title>.</summary>
        public string Title = "";

        /// <summary>Elemento <Abstract>.</summary>
        public string Abstract = "";

        /// <summary>Matrice di parole chiave.</summary>
        public string[] Keywords = null;

        /// <summary>MBR del layer. </summary>
        public OWSServiceCrawler.MBRClass LatLongBoundingBox = null;

        /// <summary>Lsita di elementi <SRS>.</summary>
        public string[] SRS = null;

        /// <summary>Indirizzo della parte "tabellare" dei dati geografici. Sitratta di un documento GML</summary>
        public string FeatureListURL
        {
            get{return this.pFeatureListURL;}
            set{this.pFeatureListURL = value.Trim();}
        }

        public LayerGeometryType GeometryType = LayerGeometryType.gtNA;

        /// <summary>
        /// Ritorna il nome del tipo di geometria
        /// </summary>
        public string GeometryTypeName
        {
            get
            {
                switch ((int)this.GeometryType)
                {
                    case 33: return "Raster";
                    case 10: return "Point";
                    case 5: return "Graphical text";
                    case 2: return "Areal";
                    case 1: return "Linear";
                    case -1: return "NA";
                    default: return "Other";
                }
            }
        }
        /// <summary>
        /// Elemento <Name>, lista di oggetti di tipo MetadataURLClass che
        /// descrivono i metadati secondo un tipo e li identificano con un URL. </summary>
        //      public System.Collections.ArrayList MetadataURL = null;
        public OWSServiceCrawler.TypedArrayList.MetadataURLArrayList MetadataURL = null;
        #endregion

        private string pFeatureListURL = "";
        #region Costruttori
        ///<summary>Costruttore.</summary>
        /// <param name="layerNode">Nodo XML che rappresenta un Layer. </param>
        /// <param name="prefix">Prefisso legato ai tag del documento "capabilities".</param>
        /// <example> <wms:ContactInformation> restituisce "wms" </example>
        /// <param name="nsmgr">Oggetto di tipo "XmlNamespaceManager" relativo al documento "capabilities".</param>
        public LayerClass(XmlNode layerNode, string prefix, XmlNamespaceManager nsmgr)
        {
            XmlNode tmp = null;XmlNodeList ltmp = null;int i=0;
            tmp = layerNode.SelectSingleNode(@"./" + prefix + "Name",nsmgr);
            if(null!=tmp) this.Name = tmp.InnerText;
            tmp = layerNode.SelectSingleNode(@"./" + prefix + "Title",nsmgr);
            if(null!=tmp) this.Title = tmp.InnerText;
            tmp = layerNode.SelectSingleNode(@"./" + prefix + "Abstract",nsmgr);
            if(null!=tmp) this.Abstract = tmp.InnerText;
            tmp = layerNode.SelectSingleNode(@"./" + prefix + "SRS",nsmgr);// /!\ No gestisce il caso <></> <></>
            if(null!=tmp) this.SRS = tmp.InnerText.Split(' ');
            this.Keywords = Utils.GetKeywordList(layerNode,prefix,nsmgr);
            this.LatLongBoundingBox = this.GetLatLonBoundingBox(layerNode,prefix,"LatLongBoundingBox",nsmgr);
            if (null==this.LatLongBoundingBox)
                this.LatLongBoundingBox = this.GetLatLonBoundingBox(layerNode,prefix,"LatLonBoundingBox",nsmgr);
            //Gestione dei MetadataURL
            ltmp = layerNode.SelectNodes(@"./" + prefix + "MetadataURL",nsmgr);
            if ( (null!=ltmp) && (ltmp.Count>0) )
            {
                this.MetadataURL = new OWSServiceCrawler.TypedArrayList.MetadataURLArrayList(ltmp.Count);
                for (i=0;i<ltmp.Count;i++)//MetadataURL ha cardinalità 0..*
                    this.MetadataURL.Add(new MetadataURLClass(ltmp.Item(i),prefix,nsmgr));
```

```
            }//If gestione dei MetadataURL
            tmp=null; ltmp=null;
        }

        /// <summary>Costruttore secco!</summary>
        public LayerClass(){}
        #endregion

        #region Metodi pubblici
        /// <summary>Estrae lo MBR del layer</summary>
        /// <param name="layerNode">
        /// Nodo padre che contiene l'elemento <LatLonBoundingBox> (per WMS 1.1.1 e 1.0)
        /// oppure <LatLongBoundingBox> (per WFS 1.0).</param>
        /// <param name="prefix">Prefisso legato ai tag del documento "capabilities".</param>
        /// <param name="postf">
        /// Deve valere "LatLongBoundingBox" nel caso di un WFS 1.0 e"LatLonBoundingBox" nel caso di un WMS 1.1.0 e 1.1.1
        /// </param>
        /// <param name="nsmgr">Oggetto di tipo "XmlNamespaceManager" relativo al documento "capabilities".</param>
        /// <returns>  Minimun Bounding Box: un oggetto di tipo MBRClassNull :nel caso non vi sia un MBR.</returns>
        private MBRClass GetLatLonBoundingBox(XmlNode layerNode, string prefix, string postf, XmlNamespaceManager nsmgr)
        {
            OWSServiceCrawler.MBRClass bbox = new MBRClass();
            int i=0;
            XmlNode tmp = layerNode.SelectSingleNode(@"./" + prefix + postf,nsmgr);
            if( (null!=tmp) && (tmp.Attributes.Count>0) )
            {
                string val = "";
                XmlAttributeCollection atts = tmp.Attributes;
                for (i=0;i<atts.Count;i++)
                {
                    val=atts.Item(i).InnerText.Replace(".",",");
                    switch (atts.Item(i).LocalName)
                    {
                        case "minx":bbox.minx = double.Parse(val); break;
                        case "miny":bbox.miny = double.Parse(val); break;
                        case "maxx":bbox.maxx = double.Parse(val); break;
                        case "maxy":bbox.maxy = double.Parse(val); break;
                    }
                }
                return bbox;
            }
            else return null;
        }

        /// <summary>Costruisce una stringa di keyword separate dal carattere ";".</summary>
        public string FlatKeywords{get{return Utils.FlatternArray(this.Keywords);}}

        /// <summary>
        /// Indica se il layer corrente è "come" il layer b secondo le informazioniu mantenute nel catalogo.</summary>
        /// <param name="b">Oggetto WFS_LayerClass b.</param>
        /// <returns>"true" se catalogo-equivalenti.</returns>
        public bool IsLike (LayerClass b )
        {
            if(null== (object)b) return false;
            else
            {
                return (this.Title == b.Title) &&
                    (this.LatLongBoundingBox == b.LatLongBoundingBox ) &&
                    (this.FlatKeywords == b.FlatKeywords) &&
                    (this.FlatMetadataURL == b.FlatMetadataURL) &&
                    (this.FlatSRS == b.FlatSRS);
            }
        }
        /// <summary>Costruisce una stringa di SRS separate dal carattere ";".</summary>
        public string FlatSRS{get{return Utils.FlatternArray(this.SRS);}}

        /// <summary>Cosruisce una stringa di coppie "URL:type" separate dal carattere ";".</summary>
        public string FlatMetadataURL
        {
            get
            {
                if (null==this.MetadataURL) return "";
                if (this.MetadataURL.Count>0)
                {
                    int i;string t = "";
                    for (i=0;i<this.MetadataURL.Count;i++)
                        t+=this.MetadataURL[i].URL + ":" + this.MetadataURL[i].type + ";";
                    return t;
                }
                else return "";
            }
        }

        /// <summary>Restituisce il nome del layer.</summary>
        /// <returns>Attributo "Name" del layer.</returns>
        public override string ToString(){return this.Name;}
        #endregion
    }// END CLASS DEFINITION LayerClass

    /// <summary>Costanti per identificare iltipo di geometria di un layer</summary>
    public enum LayerGeometryType
    {
        gtNA = -1,
        gtLinear = 1,
        gtAreal = 2,
        gtAny = 3,
        gtGraphicalText=5,
        gtPoint=10,
        gtRaster=33,
    };
} // OWSServiceCrawler
```

```cs
using System;
using System.Xml;
using System.Collections;
namespace OWSServiceCrawler
{
    /// <summary>Definisce la struttura generale di un servizio</summary>
    /// <remarks>
    /// Classe Astratta. Utilizzata per derivare le classi WFS_ServiceClass e WMS_ServiceClass.
    /// </remarks>
    public abstract class ServiceClass
    {
        /// <summary>Indice del servizio nel catalogo.</summary>
        public long sId=-1;

        private string pURL = "";

        ///<remarks>Questi due campi sono visilbili solo all'interno delle classi derivate.</remarks>
        /// <summary> Utilizzato internamente per mantenere il prefisso legato al documento.
        /// Si presenta nella forma "prefisso:". </summary>
        protected string prefix = "";
        /// <summary>
        /// Indica il tipo di sitanza del servizio.
        /// </summary>
        /// <value>"WFS", "WMS", ...</value>
        protected string pType = "";
        ///<summary>
        ///Mantiene il namespacemanager relativo al documento.
        ///</summary>
        protected XmlNamespaceManager nsmgr = null;
        /// <summary>
        /// Contiene la lista delle possibili richieste effettuabili al servizio.
        /// Tali richieste sono definite nell'elemento <Request> di <Capability>.
        /// </summary>
        public ArrayList CapabilityRequest = null;
        /// <summary>
        /// Specifica la versione del servizio.
        /// </summary>
        public string Version = "";
        /// <summary>
        /// Specifica la URL del servizio.
        /// </summary>
        /// <remarks>
        /// Proprietà in sola lettura.
        /// </remarks>
        public string URL
        {
            get{return this.pURL;}
            set{this.pURL = value;}
        }

        /// <summary>
        /// Costrutore predefinito.
        /// </summary>
        public ServiceClass(){}

        /// <summary>
        /// Costruttore.
        /// </summary>
        /// <param name="capabilityDoc">Documento ottenuto da una richiesta di tipo "GetCapability" al servizio
OGC</param>
        /// <param name="URL">URL del servizio OGC</param>
        public ServiceClass(XmlDocument capabilityDoc, string URL)
        {
            this.pURL=URL;
            this.nsmgr = new XmlNamespaceManager(capabilityDoc.NameTable);
            XmlNode root = capabilityDoc.DocumentElement;
            this.prefix = Utils.GetDocumentPrefix(root,this.nsmgr);
            if (""!=this.prefix) this.prefix += ":";
            this.Version = root.Attributes.GetNamedItem("version").Value;

            XmlNode tmp = root.SelectSingleNode(@"./" + this.prefix + "Capability/" + this.prefix +
"Request",this.nsmgr);
            if ( (null!=tmp) && (tmp.HasChildNodes) )
            {
                int i=0;
                this.CapabilityRequest = new ArrayList(tmp.ChildNodes.Count);
                for(i=0;i<tmp.ChildNodes.Count;i++)
                    this.CapabilityRequest.Add(tmp.ChildNodes.Item(i).LocalName);
            }
        }
        /// <summary>
        /// Indica il tirpo di servizio; se si tratta.
        /// </summary>
        /// <value>"WMS" per Web Map Service, oppure "WFS" per Web Feature Services.</value>
        public virtual string IstanceType
        {
            get{return this.pType;}
            set{this.pType = value;}
        }
    }
}
```

```
using System.Xml;
using System.Collections;
namespace OWSServiceCrawler
{
    /// <summary>Descrive la struttura generale del tag <Service>.</summary>
    /// <remarks> Classe astratta; si tratta della classe base per le classi "WMS_ServiceInfoClass" e
    /// WFS_ServiceInfoClass". Tutti i campi di questa classe rispecchiano gli elementi di <ContactInformation>.
    /// </remarks>
    public abstract class ServiceInfosClass
    {
        /// <summary>Elemento <Name>.</summary>
        public string Name = "";
        /// <summary>Elemento <Title>.</summary>
        public string Title = "";
        /// <summary>Elemento <Abstract>.</summary>
        public string Abstract = "";
        /// <summary>Contiene tutte le keyword definite sia per un servizio WFS che WMS.</summary>
        /// <remarks>
        /// Un servzio WFS definisce le parole chiave all'interno del tag <Keyword>:
        /// <Keywords>keyword_1 ,..., keyword_n</Keyword>.
        /// Un servizio WMS definisce le parole chiave secondo una lista di elementi <Keyword> identificata
        /// da <KeywordList> : <KeywordList> <Keyword>keyword_1<Keyword> ... <Keyword>keyword_n<Keyword> </KeywordList>.
        /// </remarks>
        public string[] Keywords = null;
        /// <summary>Elemento <OnlineResource>.</summary>
        public string OnlineResource = "";
        /// <summary>Elemento <Fees>.</summary>
        public string Fees = "";
        /// <summary>Elemento <AccessConstraints>.</summary>
        public string AccessConstraints = "";
        /// <summary>Costruttore.</summary>
        /// <param name="service">Nodo XML <Service> del documento "capability". </param>
        /// <param name="prefix">Prefisso legato ai tag del documento "capability".</param>
        /// <example> <wms:ContactInformation> restituisce "wms" </example>
        /// <param name="nsmgr">Oggetto di tipo "XmlNamespaceManager" relativo al documento "capability".</param>
        public ServiceInfosClass(XmlNode service,string prefix, XmlNamespaceManager nsmgr )
        {
            XmlNode tmp = null;
            tmp = service.SelectSingleNode(@"./" + prefix + "Name",nsmgr);
            if(null!=tmp) this.Name = tmp.InnerText;
            tmp = service.SelectSingleNode(@"./" + prefix + "Title",nsmgr);
            if(null!=tmp) this.Title = tmp.InnerText;
            tmp = service.SelectSingleNode(@"./" + prefix + "Abstract",nsmgr);
            if(null!=tmp) this.Abstract = tmp.InnerText;
            tmp = service.SelectSingleNode(@"./" + prefix + "Fees",nsmgr);
            if(null!=tmp) this.Fees = tmp.InnerText;
            tmp = service.SelectSingleNode(@"./" + prefix + "OnlineResource",nsmgr);
            if(null!=tmp) this.OnlineResource = tmp.InnerText;
            this.Keywords = Utils.GetKeywordList(service,prefix,nsmgr);tmp=null;
        }
        public ServiceInfosClass(){}
        public string FlatKeywords{get{return Utils.FlatternArray(this.Keywords);}}
        /// <summary>Permette di stabilire l'uguaglianza fra due Layer.</summary>
        /// <param name="a">Oggetto ServiceInfosClass a.</param>
        /// <param name="b">Oggetto ServiceInfosClass b.</param>
        /// <returns>"true" se sono layer identici.</returns>
        public static bool operator == (ServiceInfosClass a, ServiceInfosClass b)
        {
            if( ((object)a)==null || ((object)b)==null )
                if( ((object)a)==null && ((object)b)==null )
                    return true;
                else return false;
            if(a.Name==b.Name)
            {
                return ((a.Title == b.Title) &&(a.Abstract == b.Abstract) && (a.Fees == b.Fees) &&
                        (a.OnlineResource == a.OnlineResource) && (a.AccessConstraints == b.AccessConstraints) &&
                        (a.FlatKeywords == b.FlatKeywords));
            }
            else
                return false;
        }
        /// <summary>Permette di stabilire la non uguaglianza fra due Layer.</summary>
        /// <param name="a">Oggetto ServiceInfosClass a.</param>
        /// <param name="b">Oggetto ServiceInfosClass b.</param>
        /// <returns>"true" se non sono layer identici.</returns>
        public static bool operator != (ServiceInfosClass a, ServiceInfosClass b)
        {
            if( ((object)a)==null || ((object)b)==null )
                if( ((object)a)==null && ((object)b)==null )
                    return false;
                else return true;
            if(a.Name!=b.Name)
            {
                return ((a.Title != b.Title) && (a.Abstract != b.Abstract) && (a.Fees != b.Fees) &&
                        (a.OnlineResource != a.OnlineResource) &&(a.AccessConstraints != b.AccessConstraints) &&
                        (a.FlatKeywords != b.FlatKeywords));
            }
            else
                return true;
        }
        /// <summary>Indica se l'oggetto ServiceInfosClass del servizio corrente sono
        /// "come" l'oggetto ServiceInfosClass b secondo le informazioniu mantenute nel catalogo.</summary>
        /// <param name="b">Oggetto ServiceInfosClass b.</param>
        /// <returns>"true" se catalogo-equivalenti.</returns>
        public bool IsLike (ServiceInfosClass b )
        {
            if(null== (object)b) return false;
            if(this.Name==b.Name)
                return ((this.Title == b.Title) && (this.FlatKeywords == b.FlatKeywords) );
            else
                return false;
        }
    }// END CLASS DEFINITION ServiceInfosClass
} // OWSServiceCrawler
```

```
using System.Xml;

namespace OWSServiceCrawler
{
    /// <summary>Definisce un layer secondo la strutturazione di un WFS dove un layer è identificato come
    /// un elemento <FeatureType>.</summary>
    public class WFS_LayerClass : LayerClass
    {
        /// <summary>Lista delle possibili operazioni che è possibile effettuare sul layer.</summary>
        public string[] Operations = null;
        /// <summary>Costruisce una stringa di Operation separati dal carattere ";".</summary>
        public string FlatOperations{get{return Utils.FlatternArray(this.Operations);}}
        /// <summary>Lista di campi del layer.</summary>
        public OWSServiceCrawler.TypedArrayList.LayerFields Fields = null;
        #region Costruttori
        /// <summary>
        /// Costruttore
        /// </summary>
        public WFS_LayerClass() {this.Operations = null;}
        /// <summary>Costruttore</summary>
        /// <param name="layerNode">Nodo XML che rappresenta un Layer. </param>
        /// <param name="prefix">Prefisso legato ai tag del documento "capability".</param>
        /// <example> <wms:ContactInformation> restituisce "wms" </example>
        /// <param name="nsmgr">Oggetto di tipo "XmlNamespaceManager" relativo al documento "capability".</param>
        public WFS_LayerClass(XmlNode layerNode, string prefix, XmlNamespaceManager nsmgr) : base(
layerNode,prefix,nsmgr)
        {
            XmlNode tmp = null;XmlNodeList ltmp = null;int i = 0;
            tmp = layerNode.SelectSingleNode("./" + prefix + "Operations",nsmgr);
            if( (null!=tmp) && (tmp.HasChildNodes) )
            {
                ltmp = tmp.ChildNodes;this.Operations = new string[ltmp.Count];
                for (i=0;i<ltmp.Count; i++)
                    this.Operations[i] = ltmp[i].LocalName;
            }
            tmp = null; ltmp = null;
        }
        #endregion
        #region Operatori di confronto; == e !=
        /// <summary>Permette di stabilire l'uguaglianza fra due Layer.</summary>
        /// <param name="a">Oggetto WFS_LayerClass a.</param>
        /// <param name="b">Oggetto WFS_LayerClass b.</param>
        /// <returns>"true" se sono layer identici.</returns>
        /// <remarks>Overload di "=="</remarks>
        public static bool operator == (WFS_LayerClass a,WFS_LayerClass b )
        {
            if( ((object)a)==null || ((object)b)==null )
                if( ((object)a)==null && ((object)b)==null )
                    return true;
                else return false;
            bool r = true;
            if ((a.Name == b.Name))
            {
                r =( (a.Abstract == b.Abstract) && (a.Title == b.Title) );
                if(r)
                {
                    r =( (a.LatLongBoundingBox == b.LatLongBoundingBox) && r);
                    if(r) r =( (a.FlatKeywords == b.FlatKeywords) &&  (a.FlatOperations == b.FlatOperations) &&
                                (a.FlatMetadataURL == b.FlatMetadataURL) && (a.FlatSRS == a.FlatSRS));
                }
            }
            return r;
        }
        /// <summary>Permette di stabilire la non uguaglianza fra due Layer.</summary>
        /// <param name="a">Oggetto WFS_LayerClass a.</param>
        /// <param name="b">Oggetto WFS_LayerClass b.</param>
        /// <returns>"true" se non sono layer identici.</returns>
        /// <remarks>Overload di "!="</remarks>
        public static bool operator != (WFS_LayerClass a,WFS_LayerClass b )
        {
            if( ((object)a)==null || ((object)b)==null )
                if( ((object)a)==null && ((object)b)==null )
                    return false;
                else return true;
            bool r = true;
            if ((a.Name != b.Name))
            {
                r =( (a.Abstract != b.Abstract) && (a.Title != b.Title) );
                if(r)
                {
                    r =( (!a.LatLongBoundingBox.Equals(b.LatLongBoundingBox) && r));
                    if(r) r =( (a.FlatKeywords != b.FlatKeywords) && (a.FlatOperations != b.FlatOperations) &&
                                (a.FlatMetadataURL != b.FlatMetadataURL) &&(a.FlatSRS != a.FlatSRS) );
                }
            }
            return r;
        }
        #endregion
        #region Override di GetHashCode e Equals
        /// <summary>Override di GetHashCode</summary>
        public override int GetHashCode(){return base.GetHashCode();}
        /// <summary>Override di Equals</summary>
        public override bool Equals(object obj){return base.Equals (obj);}
        #endregion
    }// END CLASS DEFINITION WFS_LayerClass
} // OWSServiceCrawler
```

```
using System;
using System.Xml;
using System.Xml.XPath;
using System.Collections;

namespace OWSServiceCrawler
{
    /// <summary>
    /// Descrive un servizio WFS.
    /// </summary>
    public class WFS_ServiceClass : ServiceClass
    {
        #region Proprietà & Attributi
        /// <summary> Contiene i tag xml che definiscono i filtri operabili dal servizio.</summary>
        public string Filters = "";

        /// <summary> Insieme di layer offerti dal servizio. Rappresenta l'elemento <FeatureTypeList>.</summary>
        public OWSServiceCrawler.TypedArrayList.WFS_LayerClassArrayList Layers;

        /// <summary>Informazioni relative al servizio. Mappatura dell'elemento <Service>.</summary>
        public OWSServiceCrawler.WFS_ServiceInfosClass ServiceInfo;

        /// <summary>Operazioni che il servizio può effettuare su tutti i layers.</summary>
        public string[] GlobalOperations = null;
        /// <summary> Indica se si tratta di un servizio WFS Basic. </summary>
        /// <value>
        /// True: WFS Basic.
        /// False: WFS Transaction.
        /// </value>
        public bool IsBasic
        {
            get
            {
                if((null==this.CapabilityRequest) || (0==this.GlobalOperations.Length))
                    return true;
                else
                    return !(this.CapabilityRequest.Contains("Transaction"));
            }
        }

        /// <summary>Indica il tirpo di servizio; se si tratta. </summary>
        /// <value>"WMS" per Web Map Service, oppure "WFS" per Web Feature Services.</value>
        public override string IstanceType
        {get{return "WFS";} }

        /// <summary>Restituisce una stringa di operazioni, ognuna separata dal carattere ";".</summary>
        public string FlatGlobalOperations
        {get{return Utils.FlatternArray(this.GlobalOperations);}}

        #endregion

        #region Costruttori
        public WFS_ServiceClass(){}

        /// <summary> Costruttore. </summary>
        /// <param name="capabilityDoc">Documento ottenuto da una richiesta di tipo "GetCapability" al servizio
OGC</param>
        /// <param name="URL">URL del servizio OGC</param>
        public WFS_ServiceClass(XmlDocument capabilityDoc, string URL) : base(capabilityDoc, URL)
        {
            XmlNode root = capabilityDoc.DocumentElement;
            string tp = "";
            if(nsmgr.HasNamespace("ogc")) tp = "ogc:";
            XmlNode tmp = root.SelectSingleNode(@"./" + tp + "Filter_Capabilities",this.nsmgr);
            if(null!=tmp) this.Filters = tmp.InnerXml;
            tmp = root.SelectSingleNode(@"./" + this.prefix + "Service",this.nsmgr);
            if(null!=tmp) this.ServiceInfo = new WFS_ServiceInfosClass(tmp,this.prefix,this.nsmgr);

            XmlNodeList ltmp = root.SelectNodes(@"./" + this.prefix + "FeatureTypeList/" + this.prefix +
"FeatureType",this.nsmgr);
            if ( (null!=ltmp) && (ltmp.Count>0) )
            {
                int i=0 ,j=0;
                #region Recupero documento Schema tramite richiesta "DescribeFeatureType"
                    string msg = "";
                    XmlDocument xd = new XmlDocument();
                    xd.Load(URL + "?request=describeFeatureType");
                #endregion
                #region Recupero Layer
                    this.Layers = new TypedArrayList.WFS_LayerClassArrayList(ltmp.Count);
                    WFS_LayerClass l = null;
                    for(i=0;i<ltmp.Count;i++)
                    {
                        l = new WFS_LayerClass(ltmp.Item(i),this.prefix,this.nsmgr);
                        l.FeatureListURL = URL + "?request=getfeature&TYPENAME=" + l.Name;
                        l.Fields = Utils.GetFieldsFromDFT(l.Name,xd,ref msg);//Creazione lista Attributi LAyer
                        if(null!=l.Fields)//Ricerca Tipo Geometria
                        {
                            j = l.Fields.Count-1;
                            while(j>=0 && (!l.Fields[j].isGeometry)) j--;
                            if(j>=0) l.GeometryType = l.Fields[j].GetGeometryFieldType;
                        }
                        else
                            l.GeometryType = OWSServiceCrawler.LayerGeometryType.gtNA;
                        this.Layers.Add(l);
                    }
                    l = null;
                #endregion
                xd = null;
                #region Recupero delle operazioni effettuabili su tutti i layer
                tmp = root.SelectSingleNode(@"./" + this.prefix + "FeatureTypeList/" + this.prefix +
"Operations",this.nsmgr);
                if( (null!=tmp) && (tmp.HasChildNodes) )
                {
```

```
                    ltmp = tmp.ChildNodes;
                    this.GlobalOperations = new string[ltmp.Count];
                    for (i=0;i<ltmp.Count; i++)
                        this.GlobalOperations[i] = ltmp[i].LocalName;
                }
                #endregion
                ltmp = null;
            }
            tmp = null;
        }
        #endregion
    }// END CLASS DEFINITION WFS_ServicesClass
}// OWSServiceCrawler
```

WFS_ServiceInfosClass.cs

```
using System.Xml;
namespace OWSServiceCrawler
{
    /// <summary>
    /// Riporta le informazioni relative ad un servizio WFS contenute identificate dall'elemento <Service>.
    /// </summary>
    /// <remarks>
    /// Attualmente deriva direttamente dalla classe base
    ///<seealso cref="ServiceInfosClass">ServiceInfosClass</seealso>.
    /// </remarks>
    public class WFS_ServiceInfosClass : ServiceInfosClass
    {
        #region Costruttori
        /// <summary>
        /// Costruttore.
        /// </summary>
        /// <param name="service">
        ///Nodo XML <Service> del documento "capability".
        ///</param>
        /// <param name="prefix">
        ///Prefisso legato ai tag del documento "capability".
        ///</param>
        /// <example> <wms:ContactInformation> restituisce "wms" </example>
        /// <param name="nsmgr">
        ///Oggetto di tipo "XmlNamespaceManager" relativo al documento "capability".
        ///</param>
        public WFS_ServiceInfosClass(XmlNode service, string prefix, XmlNamespaceManager nsmgr):
            base (service,prefix,nsmgr){}

        /// <summary>
        /// Costruttore predefinito.
        /// </summary>
        /// <remarks>
        /// Uso deprecato.
        /// </remarks>
        public WFS_ServiceInfosClass(){}
        #endregion
    }// END CLASS DEFINITION WFS_ServiceInfosClass
} // OWSServiceCrawler
```

```
using System.Xml;

namespace OWSServiceCrawler
{
    /// <summary> Modella un layer per un'istanza di servizio WMS. </summary>
    public class WMS_LayerClass : LayerClass
    {
        #region Proprietà & Attributi
            /// <summary> Rappresenta l'elemento <ScaleHint> del documento di "capabilities". </summary>
            public ScaleHintClass ScaleHint = null;

            /// <summary>
            /// Indica se il layer è interrogabile. Significa che il layer ha un documento GMl associato.
            /// </summary>
            public int queryable = 0;

            /// <summary> Stile del layer. </summary>
            /// <remarks>
            /// Attualmente non gestito. Lo XML/CSS che descrive lo stiule è memorizzato come stringa.
            /// Utilizzato per futuri sviluppi
            /// </remarks>
            public string Style = "";

            /// <summary> URL dell'immagine della legenda associata al layer. </summary>
            public string LegendURL = "";

            /// <summary>
            /// Lista che rappresenta gli elementi <Boundingbox> del documento di "capabilities".
            /// </summary>
            public TypedArrayList.BBOXArrayList BBOX = null;
        #endregion

        #region Costruttori
        /// <summary> Costruttore Predefinito! </summary>
        public WMS_LayerClass()
        {
            ScaleHint = null;   BBOX = null;
            queryable = 0;
            Style = "";     LegendURL = "";FeatureListURL = "";
            this.GeometryType = LayerGeometryType.gtRaster;
        }

        /// <summary> Costruttore. </summary>
        /// <param name="layerNode">Nodo Xml del documento dell capabilities che descrive il layer. </param>
        /// <param name="prefix">
        /// Prefisso utilizzato per gli elelemnti nel documento XML delle capabilities. </param>
        /// <param name="nsmgr"> Risolutore di naspace per il documento di capabilities. </param>
        public WMS_LayerClass(XmlNode layerNode, string prefix, XmlNamespaceManager nsmgr) : base(
layerNode,prefix,nsmgr)
        {
            if (null!=layerNode.Attributes.GetNamedItem("queryable"))
            {
                string att = layerNode.Attributes.GetNamedItem("queryable").Value;
                if(null!=att) this.queryable =int.Parse(att);
            }
            XmlNode tmp = layerNode.SelectSingleNode(@"./" + prefix + "Style", nsmgr);
            if(null!=tmp) this.Style = tmp.InnerXml;

            #region Creazione di ScaleHint
            tmp = layerNode.SelectSingleNode(@"./" + prefix + "ScaleHint", nsmgr);
            if(null!=tmp)
            {
                string t;
                this.ScaleHint = new ScaleHintClass();
                t = tmp.Attributes.GetNamedItem("max").Value.ToString();
                this.ScaleHint.max = double.Parse(t);
                t = t = tmp.Attributes.GetNamedItem("min").Value.ToString();
                this.ScaleHint.min = double.Parse(t);
            }
            #endregion
            //Recupero del legend URL
            tmp = layerNode.SelectSingleNode(@"./" + prefix + "Style/" + prefix + "LegendURL/" + prefix +
"OnlineResource", nsmgr);
            if(null!=tmp)
                this.LegendURL = tmp.Attributes.GetNamedItem("xlink:href").Value.ToString();
            //Recupero del Feature list URL
            tmp = layerNode.SelectSingleNode(@"./" + prefix + "FeatureListURL/" + prefix + "OnlineResource", nsmgr );
            if(null!=tmp)
                this.FeatureListURL = tmp.Attributes.GetNamedItem("xlink:href").Value.ToString();

            #region Creazione dei BoundingBox
            XmlNodeList ltmp = layerNode.SelectNodes(@"./" + prefix + "BoundingBox", nsmgr);
            if ( (null!=ltmp) && (ltmp.Count>0))
            {
                string x = "";
                MBRClass t = new MBRClass();
                this.BBOX = new OWSServiceCrawler.TypedArrayList.BBOXArrayList(ltmp.Count);
                for (int i = 0; i<ltmp.Count;i++)
                {
                    x = ltmp.Item(i).Attributes.GetNamedItem("minx").Value.Replace(".",",");
                    t.minx = double.Parse(x);
                    x = ltmp.Item(i).Attributes.GetNamedItem("miny").Value.Replace(".",",");
                    t.miny = double.Parse(x);
                    x = ltmp.Item(i).Attributes.GetNamedItem("maxx").Value.Replace(".",",");
                    t.maxx = double.Parse(x);
                    x = ltmp.Item(i).Attributes.GetNamedItem("maxy").Value.Replace(".",",");
                    t.maxy = double.Parse(x);
                    x = ltmp.Item(i).Attributes.GetNamedItem("SRS").Value.ToString();
                    this.BBOX.Add(x,t);
                    t = new MBRClass();
                }
                t = null;
            }
            #endregion
```

```
105             this.GeometryType = LayerGeometryType.gtRaster;
106         }
107         #endregion
108 
109         #region Operatori di confronto; == e !=
110         /// <summary>
111         /// Permette di stabilire l'uguaglianza fra due Layer.
112         /// </summary>
113         /// <param name="a">Oggetto WMS_LayerClass a.</param>
114         /// <param name="b">Oggetto WMS_LayerClass b.</param>
115         /// <returns>"true" se sono layer identici.</returns>
116         /// <remarks>Overload di "=="</remarks>
117         public static bool operator == (WMS_LayerClass a,WMS_LayerClass b )
118         {
119             if( ((object)a)==null || ((object)a)==null )
120                 if( ((object)a)==null && ((object)a)==null )
121                     return true;
122                 else return false;
123             bool r = true;
124             if ((a.Name == b.Name))
125             {
126                 r =( (a.Abstract == b.Abstract) && (a.Title == b.Title) );
127                 if(r)
128                 {
129                     r =( (a.LatLongBoundingBox == b.LatLongBoundingBox) && r);
130                     if(r) r =( (a.FlatKeywords == b.FlatKeywords) &&  (a.FeatureListURL == b.FeatureListURL) &&
131                                 (a.FlatMetadataURL == b.FlatMetadataURL) && (a.ScaleHint.max == b.ScaleHint.max) &&
132                                 (a.ScaleHint.min == b.ScaleHint.min) && (a.queryable == b.queryable) &&
133                                 (a.BBOX == b.BBOX) &&    (a.FlatSRS == a.FlatSRS));
134                 }
135             }
136             return r;
137         }
138 
139         /// <summary>
140         /// Permette di stabilire la non uguaglianza fra due Layer.
141         /// </summary>
142         /// <param name="a">Oggetto WMS_LayerClass a.</param>
143         /// <param name="b">Oggetto WMS_LayerClass b.</param>
144         /// <returns>"true" se non sono layer identici.</returns>
145         /// <remarks>Overload di "!="</remarks>
146         public static bool operator != (WMS_LayerClass a,WMS_LayerClass b )
147         {
148             if( ((object)a)==null || ((object)a)==null )
149                 if( ((object)a)==null && ((object)a)==null )
150                     return false;
151                 else return true;
152             bool r = true;
153             if ((a.Name != b.Name))
154             {
155                 r =( (a.Abstract != b.Abstract) && (a.Title != b.Title) );
156                 if(r)
157                 {
158                     r =( (!a.LatLongBoundingBox.Equals(b.LatLongBoundingBox) && r));
159                     if(r) r =( (a.FlatKeywords != b.FlatKeywords) && (a.FeatureListURL != b.FeatureListURL) &&
160                                 (a.FlatMetadataURL != b.FlatMetadataURL) &&(a.ScaleHint.max != b.ScaleHint.max) &&
161                                 (a.ScaleHint.min != b.ScaleHint.min) &&(a.queryable != b.queryable) &&
162                                 (a.BBOX != b.BBOX) &&(a.FlatSRS != a.FlatSRS) );
163                 }
164             }
165             return r;
166         }
167         #endregion
168 
169         #region Override di GetHashCode e Equals
170         /// <summary>
171         /// Override di GetHashCode
172         /// </summary>
173         public override int GetHashCode()
174         {
175             return base.GetHashCode();
176         }
177 
178         /// <summary>
179         /// Override di Equals
180         /// </summary>
181         public override bool Equals(object obj)
182         {
183             return base.Equals (obj);
184         }
185         #endregion
186     }// END CLASS DEFINITION WMS_LayerClass
187 } // OWSServiceCrawler
```

WMS_LayerHierarchyClass.cs

```
using System.Collections;
using System.Xml;

namespace OWSServiceCrawler
{
    /// <summary>
    /// Modella l'istanza di un servizio WMS.
    /// </summary>
    public class WMS_ServiceClass : ServiceClass
    {
        #region Proprietà & campi
        /// <summary>
        /// Descrive la struttura gerarchica composta da "gerarchia di layer" e lista layer.
        /// </summary>
        public OWSServiceCrawler.WMS_CapabilityLayersClass Layers = null;
        /// <summary>
        /// Informazioni inerenti l'istanza del servizio.
        /// </summary>
        public WMS_ServiceInfosClass ServiceInfo = null;
        /// <summary>
        /// Indica il tirpo di servizio; se si tratta.
        /// </summary>
        /// <value>"WMS" per Web Map Service, oppure "WFS" per Web Feature Services.</value>
        public override string IstanceType{get{return "WMS";}}
        #endregion

        #region Puntatori "Rapidi" alla gerchia ed alla lista.
        /// <summary>
        /// Puntatore alla lista di layer.
        /// </summary>
        public OWSServiceCrawler.TypedArrayList.WMS_LayerClassArrayList CLayers
        {
            get
            {
                if (null!=this.Layers)
                    if (null!=this.Layers.Layers) return this.Layers.Layers;
                return null;
            }
        }
        /// <summary>
        /// Puntatore alla gerarchia di layer.
        /// </summary>
        public OWSServiceCrawler.TypedArrayList.WMS_LayerHierarchyClassArrayList CLayersHierarchy
        {
            get
            {
                if (null!=this.Layers)
                    if (null!=this.Layers.LayersHierarchy) return this.Layers.LayersHierarchy;
                return null;
            }
        }
        #endregion

        #region Costruttori
        /// <summary>
        /// Costruttore predefinito.
        /// </summary>
        /// <remarks>
        /// Uso deprecato.
        /// </remarks>
        public WMS_ServiceClass(){}
        /// <summary>
        /// Costruttore.
        /// </summary>
        /// <param name="capabilityDoc">
        /// Documento XML contenente le "capabilities" dell'istanza del servizio.
        /// </param>
        /// <param name="URL">URL del servizio.</param>
        public WMS_ServiceClass(XmlDocument capabilityDoc, string URL) : base(capabilityDoc, URL)
        {
            XmlNode root = capabilityDoc.DocumentElement;
            XmlNode tmp = root.SelectSingleNode(@"./" + this.prefix + "Service",this.nsmgr);
            if(null!=tmp) this.ServiceInfo = new WMS_ServiceInfosClass(tmp,this.prefix,this.nsmgr);

            tmp = root.SelectSingleNode(@"./" + this.prefix + "Capability",this.nsmgr);
            if(null!=tmp) this.Layers = new WMS_CapabilityLayersClass(tmp,this.prefix,this.nsmgr);
        }
        #endregion
    }// END CLASS DEFINITION WMS_ServicesClass
} // OWSServiceCrawle
```

WMS_ServiceInfosClass.cs

```
using System.Xml;
namespace OWSServiceCrawler
{
    /// <summary>
    /// Riporta le informazioni relative ad un servizio WMS contenute identificate dall'elemento <Service>.
    /// </summary>
    public class WMS_ServiceInfosClass    : ServiceInfosClass
    {
        /// <summary>
        /// Mappa le informazioni definite all'interno dell'elemento <ContactInformation>.
        /// </summary>
        public OWSServiceCrawler.ContactsInfos ContactInformation;

        #region Costruttori
        /// <summary>
        /// Costruttore.
        /// </summary>
        /// <param name="service">Nodo XML <Service> del documento "capability". </param>
        /// <param name="prefix">Prefisso legato ai tag del documento "capabilities".</param>
        /// <example> <wms:ContactInformation> restituisce "wms" </example>
        /// <param name="nsmgr">Oggetto di tipo "XmlNamespaceManager" relativo al documento "capability".</param>
```

WMS_ServiceClass.cs

```
        public WMS_ServiceInfosClass(XmlNode service, string prefix, XmlNamespaceManager nsmgr): base
(service,prefix,nsmgr)
        {
            XmlNode tmp = service.SelectSingleNode("./" + prefix + "ContactInformation",nsmgr);
            ContactsInfos ci = null;
            if (null!=tmp)
            {
                ci = new ContactsInfos(tmp,prefix,nsmgr);
                if ((null!=ci) && (!ci.NotPresent)) {this.ContactInformation=ci;ci=null;}
            }
        }
        /// <summary>
        /// Costruttore predefinito.
        /// </summary>
        /// <remarks>
        /// Uso deprecato.
        /// </remarks>
        public WMS_ServiceInfosClass(){}
        #endregion
    }// END CLASS DEFINITION WMS_ServiceInfosClass
} // OWSServiceCrawler
```

WMS_LayerHierarchyClass.cs

```

using System.Xml;
using OWSServiceCrawler.TypedArrayList;

namespace OWSServiceCrawler
{
    public class WMS_LayerHierarchyClass : WMS_LayerClass
    {
        #region Proprietà & Campi
        /// <summary>
        /// Indica che il layer è una "Categoria"
        /// </summary>
        public bool IsCategory{get{return (""==this.Name);}}

        /// <summary>
        /// Puntatore alla lista di layer del livello della gerachia.
        /// </summary>
        public TypedArrayList.WMS_LayerClassArrayList ChildLayers = new WMS_LayerClassArrayList();

        /// <summary>
        /// Puntatore alla sotto gerarchia.
        /// </summary>
        public TypedArrayList.WMS_LayerHierarchyClassArrayList ChildHierarchy = new WMS_LayerHierarchyClassArrayList();
        #endregion

        #region Costruttori
        /// <summary>
        /// Costruttore di inizializzazione
        /// </summary>
        /// <param name="hierarchy">
        /// Nodo XML (appartenente al documento delle "capabilities") che
        /// rappresenta la gerachia da mappare.
        /// </param>
        /// <param name="prefix">
        /// Prefisso degli elementi xml del nodo.
        /// </param>
        /// <param name="nsmgr"
        /// >Risolutore dei namespace.
        /// </param>
        public WMS_LayerHierarchyClass(XmlNode hierarchy, string prefix, XmlNamespaceManager nsmgr) : base( hierarchy,
prefix,nsmgr)
        {
            XmlNodeList layerset = hierarchy.SelectNodes(@"./" + prefix + "Layer", nsmgr);
            XmlNode curr = null;
            XmlNode sub_layer = null;
            int i = 0;
            if ( (null!=layerset) && (layerset.Count>0) )
            {
                for(i=0; i<layerset.Count;i++)
                {
                    curr = layerset.Item(i);
                    sub_layer = curr.SelectSingleNode(@"./" + prefix + "Layer", nsmgr);
                    if(null!=sub_layer)
                    {
                        this.ChildHierarchy.Add(new WMS_LayerHierarchyClass(curr,prefix,nsmgr));
                        System.Console.WriteLine("GERARCHIA: " + curr.SelectSingleNode(@"./" + prefix + "Title",
nsmgr).InnerText);
                    }
                    else
                    {
                        this.ChildLayers.Add(new WMS_LayerClass(curr,prefix,nsmgr));
                        System.Console.WriteLine("FIGLIO: " + curr.SelectSingleNode(@"./" + prefix + "Name",
nsmgr).InnerText);
                    }
                }
                if (0==this.ChildLayers.Count) this.ChildLayers = null;
                if (0==this.ChildHierarchy.Count) this.ChildHierarchy = null;
            }
        }
        #endregion
    }// END CLASS DEFINITION WMS_CategoryClass
} // OWSServiceCrawler
```

```
using System;
using System.Xml;
using OWSServiceCrawler.TypedArrayList;

namespace OWSServiceCrawler
{
    /// <summary>
    /// Rappresenta i layer di un WMS.
    /// </summary>
    /// <remarks>
    /// In un WMS si possono trovare due tipi di raggruppamenti di layer:
    /// Category: Ha un elemento <Title> ma non un <Name>.
    ///Map Layer: Ha sia l'elemento <Title> che <Name>. questo tipo di gruppo può essere richiesto in blocco.
    /// Ognuno di questi due gruppi può contenere una o più istanze dell'altro
    ///o di se stesso,oltre ad un insieme di layer.
    /// Per semplificare i gruppi "Map Layer" e "Category" sono implementati
    ///allo stesso modo.
    /// Per maggiori dettagli vedere il diagramma UML relativo.
    /// </remarks>
    public class WMS_CapabilityLayersClass
    {
        /// <summary>
        /// Contiene solo layer singoli.
        /// </summary>
        public TypedArrayList.WMS_LayerClassArrayList Layers = null;

        /// <summary>
        /// Contiene sia gruppi "Category" che "Map Layer".
        /// </summary>
        public TypedArrayList.WMS_LayerHierarchyClassArrayList LayersHierarchy = null;

        #region Costruttori
        /// <summary>
        /// Costruttore
        /// </summary>
        /// <param name="capabilityNode">Elemento <Capability> contenente i layer.</param>
        /// <param name="prefix">
        /// Prefisso legato ai tag del documento "capability".<wms:ContactInformation> restituisce "wms" </param>
        /// <param name="nsmgr">Oggetto di tipo "XmlNamespaceManager" relativo al documento "capability".</param>
        public WMS_CapabilityLayersClass(XmlNode capabilityNode, string prefix, XmlNamespaceManager nsmgr)
        {
            this.Layers = new WMS_LayerClassArrayList();
            this.LayersHierarchy = new WMS_LayerHierarchyClassArrayList();
            XmlNodeList layerset = capabilityNode.SelectNodes(@"./" + prefix + "Layer", nsmgr);
            XmlNode curr = null;
            XmlNode sub_layer = null;
            int i = 0;
            if ( (null!=layerset) && (layerset.Count>0) )
            {
                for(i=0; i<layerset.Count;i++)
                {
                    curr = layerset.Item(i);
                    sub_layer = curr.SelectSingleNode(@"./" + prefix + "Layer", nsmgr);
                    if(null!=sub_layer)
                        this.LayersHierarchy.Add(new WMS_LayerHierarchyClass(curr,prefix,nsmgr));
                    else
                        this.Layers.Add(new WMS_LayerClass(curr,prefix,nsmgr));
                }
                if (0==this.Layers.Count) this.Layers = null;
                if (0==this.LayersHierarchy.Count) this.LayersHierarchy = null;
            }
        }

        /// <summary>
        /// Costruttore predefinito.
        /// </summary>
        /// <remarks>
        /// Uso deprecato.
        /// </remarks>
        public WMS_CapabilityLayersClass(){}
        #endregion
    }
}
```

```
using System.Xml;

namespace OWSServiceCrawler
{
    /// <summary>
    /// Si occupa di recuperare le informazioni riguardo il contatto primario
    /// di un servizio WMS.
    /// </summary>
    /// <remarks>
    /// Tutti i campi di questa classe rispecchiano gli elementi di <ContactInformation>.
    /// La versione di documento di "capabilities" fa riferimento alla 1.1.0. esiste
    /// comunque compatibilità con 1.1.1 e 1.3 .
    /// </remarks>
    public class ContactsInfos
    {
        private bool pNotPresent = true;

        #region Proprietà & Attributi
        /// <summary>Elemento ContactPerson <ContactPerson>.</summary>
        public string ContactPerson = "";

        /// <summary>Elemento <ContactOrganization>.</summary>
        public string ContactOrganization = "";

        /// <summary>Elemento <ContactPosition>.</summary>
        public string ContactPosition = "";

        /// <summary>Elemento <AddressType>.</summary>
        public string AddressType = "";

        /// <summary>Elemento <Address>.</summary>
        public string Address = "";

        /// <summary>
        /// Elemento <City>.
        /// </summary>
        public string City = "";

        /// <summary>
        /// Elemento <StateOrProvince>.
        /// </summary>
        public string StateOrProvince = "";

        /// <summary>
        /// Elemento <PostCode>.
        /// </summary>
        public string PostCode = "";

        /// <summary>
        /// Elemento <;Country>.
        /// </summary>
        public string Country = "";

        /// <summary>
        /// Elemento <;ContactVoiceTelephone.
        /// </summary>
        public string ContactVoiceTelephone = "";

        /// <summary>
        /// Elemento <ContactFacsimileTelephone>.
        /// </summary>
        public string ContactFacsimileTelephone = "";

        /// <summary>
        /// Elemento <ContactElectronicMailAddress>.
        /// </summary>
        public string ContactElectronicMailAddress = "";

        /// <summary>
        /// Indica se le informazioni realtive al contatto primario sono presenti.
        /// </summary>
        /// <value>
        /// True se non esiste nessuna informazione inerente al contatto primario.
        /// </value>
        public bool NotPresent{get{return this.pNotPresent;}}
        #endregion

        #region Costruttori
        /// <summary>
        /// Costruttore predefinito.
        /// </summary>
        /// <remarks>
        /// Uso deprecato.
        /// </remarks>
        public ContactsInfos(){}

        /// <summary>
        /// Costruttore.
        /// </summary>
        /// <param name="contanctinfoNode">
        /// Nodo XML "<Service> del documento "capabilities".
        /// </param>
        /// <param name="prefix">
        /// Prefisso legato ai tag del documento "capabilities". <wms:ContactInformation> restituisce "wms".</param>
        /// <param name="nsmgr">Oggetto di tipo "XmlNamespaceManager" relativo al documento "capabilities". </param>
        public ContactsInfos(XmlNode contanctinfoNode,string prefix,XmlNamespaceManager nsmgr)
        {
            XmlNode tmp=null;
            XmlNodeList ltmp;
            int i=0;
            if ((null!=contanctinfoNode) && (contanctinfoNode.HasChildNodes))
            {
                this.pNotPresent=false;
                tmp = contanctinfoNode.SelectSingleNode(@"./" + prefix + "ContactPersonPrimary",nsmgr);
```

```
                if((null!=tmp) && tmp.HasChildNodes)//ContactPersonPrimary
                {
                    ltmp=tmp.ChildNodes;
                    for (i=0;i< ltmp.Count;i++)
                        switch (ltmp.Item(i).LocalName)
                        {
                            case "ContactPerson": this.ContactPerson = ltmp.Item(i).InnerText; break;
                            case "ContactOrganization": this.ContactOrganization = ltmp.Item(i).InnerText; break;
                        }
                }//ContactPersonPrimary
                tmp = contanctinfoNode.SelectSingleNode(@"./" + prefix + "ContactPosition",nsmgr);
                if(null!=tmp) this.ContactPosition = tmp.InnerText;
                tmp = contanctinfoNode.SelectSingleNode(@"./" + prefix + "ContactAddress",nsmgr);
                if((null!=tmp) && tmp.HasChildNodes)//ContactAddress
                {
                    ltmp=tmp.ChildNodes;
                    for (i=0;i< ltmp.Count;i++)
                        switch (ltmp.Item(i).LocalName)
                        {
                            case "AddressType": this.AddressType = ltmp.Item(i).InnerText; break;
                            case "Address": this.Address = ltmp.Item(i).InnerText; break;
                            case "City": this.City = ltmp.Item(i).InnerText; break;
                            case "StateOrProvince": this.StateOrProvince = ltmp.Item(i).InnerText; break;
                            case "PostCode": this.PostCode = ltmp.Item(i).InnerText; break;
                            case "Country": this.Country = ltmp.Item(i).InnerText; break;
                        }
                }//ContactAddress
                tmp = contanctinfoNode.SelectSingleNode(@"./" + prefix + "ContactVoiceTelephone",nsmgr);
                if(null!=tmp) this.ContactVoiceTelephone = tmp.InnerText;
                tmp = contanctinfoNode.SelectSingleNode(@"./" + prefix + "ContactFacsimileTelephone",nsmgr);
                if(null!=tmp) this.ContactFacsimileTelephone = tmp.InnerText;
                tmp = contanctinfoNode.SelectSingleNode(@"./" + prefix + "ContactElectronicMailAddress",nsmgr);
                if(null!=tmp) this.ContactElectronicMailAddress = tmp.InnerText;
                this.pNotPresent=((""==this.ContactPerson) && (""==this.ContactOrganization) &&
                                  (""==this.ContactPosition) && (""==this.Address) && (""==this.AddressType) &&
                                  (""==this.City) &&(""==this.PostCode) && (""==this.Country) &&
                                  (""==this.ContactVoiceTelephone) && (""==this.ContactFacsimileTelephone) &&
                                  (""==this.ContactElectronicMailAddress));
            }
            else this.pNotPresent=true;
            ltmp=null;tmp=null;
        }
        #endregion
    }// END CLASS DEFINITION ContactsInfos
} // OWSServiceCrawler
```

MBRClass.cs

```
using System;

namespace OWSServiceCrawler
{
    /// <summary>Rappresenta un Minimun Bounding Box</summary>
    /// <remarks>Viene impiegata per rappresentare un elemento <LatLonBoundingBox> (per WMS 1.1.1 e 1.0) e
    /// <LatLongBoundingBox> (per WFS 1.0).
    /// E' impiegata inoltre per implementare gli attributi dell'elemento <BoundingBox>.
    /// </remarks>
    public class MBRClass
    {
        #region Proprietà & Attributi
        /// <summary>Attributo "minx".</summary>
        public double minx = 0;
        /// <summary>Attributo "miny".</summary>
        public double miny = 0;
        /// <summary>Attributo "maxx".</summary>
        public double maxx = 0;
        /// <summary>Attributo "maxy".</summary>
        public double maxy = 0;
        #endregion
        #region Costruttori
        /// <summary>Cosruttore predefinto. </summary>
        /// <remarks>Uso deprecato. </remarks>
        public MBRClass()   {}
        /// <summary>Costruttore della classe. </summary>
        /// <param name="minx">Attributo "miny"</param>
        /// <param name="miny">Attributo "miny"</param>
        /// <param name="maxx">Attributo "maxx"</param>
        /// <param name="maxy">Attributo "maxy"</param>
        public MBRClass(double minx, double miny, double maxx, double maxy)
            {this.minx=minx; this.miny=miny; this.maxx=maxx; this.maxy=maxy;}
        #endregion
        #region Operatori di confronto; == e !=
        /// <summary>Override dell'operatore "=="</summary>
        /// <param name="a">Primo MBR.</param>
        /// <param name="b">Secondo MBR.</param>
        /// <returns>"true" se sono identici.</returns>
        public static bool operator ==(MBRClass a, MBRClass b)
        {
            if( ((object)a)==null || ((object)b)==null )
                if( ((object)a)==null && ((object)b)==null )
                    return true;
                else return false;
            return ((a.minx == b.minx) &&(a.miny == b.miny) &&(a.maxx == b.maxx) &&(a.maxy == b.maxy));
        }
        /// <summary>Override dell'operatore "!="</summary>
        /// <param name="a">Primo MBR.</param>
        /// <param name="b">Secondo MBR.</param>
        /// <returns>"true" se sono non identici.</returns>
        public static  bool operator !=(MBRClass a, MBRClass b)
        {
            if( ((object)a)==null || ((object)b)==null )
                if( ((object)a)==null && ((object)b)==null )
```

MBRClass.cs

```
                    return false;
                else return true;
            return ((a.minx != b.minx) && (a.miny != b.miny) &&(a.maxx != b.maxx) &&(a.maxy != b.maxy));
        }
        #endregion

        #region Override di GetHashCode e Equals
        /// <summary> Override di GetHashCode</summary>
        public override int GetHashCode(){return base.GetHashCode();}
        /// <summary>Override di Equals</summary>
        public override bool Equals(object obj){return base.Equals (obj);}
        #endregion
    }// END CLASS DEFINITION MBRClass
} // OWSServiceCrawler
```

MetadataURLClass.cs

```
using System;
using System.Xml;

namespace OWSServiceCrawler
{
    /// <summary>Descrive un collegamanto ai metadati di un layer</summary>
    public class MetadataURLClass
    {
        #region Proprietà & Attributi
        /// <summary>Link URL al documento che costituisce i metadati del layer. </summary>
        public string URL = "";

        /// <summary>Tipo formalismo utilizzato per descrivere i metadati: "FGDC".</summary>
        public string type = "";
        #endregion

        #region Costruttori
        /// <summary>Costruttore predefinito. </summary>
        public MetadataURLClass(){this.URL="";this.type="";}

        /// <summary>Costruttore basato sui valori degli attributi. </summary>
        /// <param name="URL">Link URL al documento che costituisce i metadati del layer. </param>
        /// <param name="type">Tipo formalismo utilizzato per descrivere i metadati. </param>
        public MetadataURLClass(string URL, string type){this.URL=URL;this.type=type;}

        /// <summary>Costrutture basato su di un nodo che descrive loo URL ai metadati. </summary>
        /// <param name="MetadataURLnode">
        /// Elemento <MetadataURL>, del documento capabilities, che contiene le informazioni sui metadati.
        /// </param>
        /// <param name="prefix">
        /// Prefisso legato ai tag del documento "capabilities". Esempio, per <wms:ContactInformation> restituisce "wms".
        /// </param>
        /// <param name="nsmgr">Oggetto di tipo "XmlNamespaceManager" relativo al documento "capabilities".</param>
        public MetadataURLClass(XmlNode MetadataURLnode, string prefix, XmlNamespaceManager nsmgr)
        {
            XmlNode t=null;
            this.type=MetadataURLnode.Attributes.GetNamedItem("type").Value;
            t = MetadataURLnode.SelectSingleNode(@"./" + prefix + "OnlineResource",nsmgr);
            if (null!=t)
                {this.URL = t.Attributes.GetNamedItem("xlink:href").Value.ToString();t = null;}
        }
        #endregion

        #region Override
        /// <summary>Restituisce il la coppia <URL,metadataType> </summary>
        /// <returns>coppia sottoforma di stringa "URL:type"</returns>
        public override string ToString(){return this.URL + ":" +this.type;}
        #endregion
    }
}
```

ScaleHintClass.cs

```
using System;
namespace OWSServiceCrawler
{

    /// <summary>Indica il valore massimo e minimo di scala. </summary>
    public class ScaleHintClass
    {
        #region Proprietà & Attributi.
        /// <summary>Valore minimo di scala. </summary>
        public double min = 0;

        /// <summary>Valore massimo di scala. </summary>
        public double max = 0;
        #endregion

        #region Costruttori
        /// <summary>Costruttore predefinito </summary>
        public ScaleHintClass(){}

        /// <summary>Costruttore basato sugli attributi. </summary>
        /// <param name="min">Valore minimo di scala.</param>
        /// <param name="max">Valore massimo di scala.</param>
        public ScaleHintClass(double min,double max){this.min = min;  this.max = max;}
        #endregion
    }// END CLASS DEFINITION ScaleHintClass
} // OWSServiceCrawler
```

```
using System;
using System.Collections;

namespace OWSServiceCrawler.TypedArrayList
{
    /// <summary>
    /// Struttura per rappresentare le informazioni inerenti ad un campo di un layer.
    /// </summary>
    public struct LayerField
    {
        /// <summary>
        /// Nome del campo
        /// </summary>
        public string Name;
        /// <summary>
        /// Tipo del campo
        /// </summary>
        /// <remarks>
        /// I tipi supportati sono: "long", "string", "double", "Polygon", "Line",
        /// "Point".Il valore "?" indica che il tipo non è riconosciuto.
        /// </remarks>
        public string Type;
        /// <summary>
        /// Indica se il campo è un BLOB che rappresenta la geometria del layer.
        /// </summary>
        /// <value>"true" se il campo rappresenta una geomtria.</value>
        public bool isGeometry;
        /// <summary>
        /// In caso il campo rappresenti una geometria, viene restituito il tipo.
        /// </summary>
        /// <remarks>
        ///  Utilizzato per conformità di tipo con la la proprietà "GeometryType"
        /// della classe "LayerClass"
        /// </remarks>
        public OWSServiceCrawler.LayerGeometryType GetGeometryFieldType
        {
            get
            {
                if(this.isGeometry)
                    switch (int.Parse(this.Type))
                    {//Costanti di GWM
                        case -1: return LayerGeometryType.gtNA;
                        case  1: return LayerGeometryType.gtLinear;
                        case  2: return LayerGeometryType.gtAreal;
                        case  3: return LayerGeometryType.gtAny;
                        case  5: return LayerGeometryType.gtGraphicalText;
                        case 10: return LayerGeometryType.gtPoint;
                        default : return LayerGeometryType.gtNA;
                    }
                else
                    return LayerGeometryType.gtNA;
            }
        }
        /// <summary>
        /// Struttura che rappresenta un campo della relazione tabellare relativa al layer.
        /// </summary>
        /// <param name="strName">Nome del campo</param>
        /// <param name="strType">Tipo del campo</param>
        /// <param name="blisGeoemtry">
        /// valore booleano per indicare se il campo rappresenta una geometria o meno
        /// </param>
        public LayerField(string strName, string strType, bool blisGeoemtry)
        {this.Name = strName.Trim();this.Type = strType.Trim();this.isGeometry = blisGeoemtry;}
    }

    /// <summary>
    /// Rapresentazione della lista di campi di un layer.
    /// </summary>
    public class LayerFields : ICollection
    {
        private ArrayList fieldsList;
        #region Costruttori
        /// <summary>
        /// Costruttore per indicare la dimansione della collezione.
        /// </summary>
        /// <param name="capacity"></param>
        public LayerFields(int capacity) {this.fieldsList = new ArrayList(capacity);}
        /// <summary>
        /// Costruttore predefinito.
        /// </summary>
        public LayerFields(){}
        #endregion
        /// <summary>
        /// Metodo per aggiungere un campo alla collezione.
        /// </summary>
        /// <param name="field">oggetto di tipo "LayerField".</param>
        /// <returns></returns>
        public int Add(LayerField field){return this.fieldsList.Add(field);}
        /// <summary>
        /// Restituisce il numero di elementi elle collezione
        /// </summary>
        public int Count{   get{return this.fieldsList.Count;}}

        public object SyncRoot{ get{return this;}}
        public bool IsSynchronized{get{return false;}}
        /// <summary>
        /// Restituisce un campo "LayerField" tramite l'indice nella collezione.
        /// </summary>
        public LayerField this[int index]
        {
            get{return (LayerField) this.fieldsList[index];}
            set{this.fieldsList[index] = (LayerField) value;}
        }
        public IEnumerator GetEnumerator(){return this.fieldsList.GetEnumerator();}
```

```
        public void CopyTo(Array a, int index){this.fieldsList.CopyTo(a,index);}
    }
    /// <summary>
    /// Implementa una collezione di layer di istanze di servizi WFS.
    /// </summary>
    public class WFS_LayerClassArrayList : ICollection
    {
        private ArrayList layerlist;
        #region Costruttori
        /// <summary>
        /// Costruttore parametrico.
        /// </summary>
        /// <param name="capacity">Dimensione vettore.</param>
        public WFS_LayerClassArrayList(int capacity) {this.layerlist = new ArrayList(capacity);}
        /// <summary>
        /// Costruttore predefinito.
        /// </summary>
        public WFS_LayerClassArrayList(){}
        #endregion
        /// <summary>
        /// Metodo per aggiungere un layer.
        /// </summary>
        /// <param name="layer">Oggetto layer WFS.</param>
        /// <returns>Posizione del layer all'interno del vettore</returns>
        public int Add(WFS_LayerClass layer){return this.layerlist.Add(layer);}
        /// <summary>
        /// Restituisce il numero di elementi elle collezione
        /// </summary>
        public int Count{get{return this.layerlist.Count;} }
        public object SyncRoot{get{return this;} }
        public bool IsSynchronized{get{return false;}}
        /// <summary>
        /// Accessore del vettore.
        /// </summary>
        public WFS_LayerClass this[int index]
        {
            get{return (WFS_LayerClass) this.layerlist[index];}
            set{this.layerlist[index] = (WFS_LayerClass) value;}
        }
        /// <summary>
        /// Effettua l'ordinamento dei Layer in ordine crescente in base al nome.
        /// </summary>
        /// <remarks>Utilizza la classe "SortByNameASC" come criterio di ordinamento.</remarks>
        public  void Sort() {SortByNameASC s = new SortByNameASC();   this.layerlist.Sort (s);}
        public int BinarySearch(object value){return this.layerlist.BinarySearch (value,new SortByNameASC());}
        public IEnumerator GetEnumerator(){return this.layerlist.GetEnumerator();}
        public void CopyTo(Array a, int index){this.layerlist.CopyTo(a,index);}
    }
    /// <summary>
    /// Implementa una collezione di layer di istanze di servizi WMS.
    /// </summary>
    public class WMS_LayerClassArrayList : ArrayList
    {
        #region Costruttori
        /// <summary>
        /// Costruttore parametrico.
        /// </summary>
        /// <param name="capacity">Dimensione vettore.</param>
        public WMS_LayerClassArrayList(int capacity) : base(capacity){}
        public WMS_LayerClassArrayList(){}
        #endregion
        /// <summary>
        /// Metodo per aggiungere un layer.
        /// </summary>
        /// <param name="layer">Oggetto layer WMS.</param>
        /// <returns>Posizione del layer all'interno del vettore</returns>
        public int Add(WMS_LayerClass layer){return base.Add(layer);}
        /// <summary>
        /// Accessore del vettore.
        /// </summary>
        public WMS_LayerClass this[int index]
        {
            get{return (WMS_LayerClass) base[index];}
            set{base[index]=(WMS_LayerClass) value;}
        }
        /// <summary>
        /// Effettua l'ordinamento dei Layer in ordine crescente in base al nome.
        /// </summary>
        /// <remarks>Utilizza la classe "SortByNameASC" come criterio di ordinamento.</remarks>
        public override void Sort(){SortByNameASC s = new SortByNameASC();base.Sort (s);}
        public override int BinarySearch(object value){return base.BinarySearch (value,new SortByNameASC());}
    }
    public class WMS_LayerHierarchyClassArrayList : ArrayList
    {
        #region Costruttori
        /// <summary>
        /// Costruttore parametrico.
        /// </summary>
        /// <param name="capacity">Dimensione vettore.</param>
        public WMS_LayerHierarchyClassArrayList(int capacity) : base(capacity){}
        /// <summary>
        /// Costruttore Predefinito.
        /// </summary>
        public WMS_LayerHierarchyClassArrayList(){}
        #endregion
        /// <summary>
        /// Metodo per aggiungere un layer.
        /// </summary>
        /// <param name="layer">Oggetto layer WMS.</param>
        /// <returns>Posizione del layer all'interno del vettore</returns>
        public int Add(WMS_LayerHierarchyClass layerHierarchy){return base.Add(layerHierarchy);}
        /// <summary>
        /// Accessore del vettore.
        /// </summary>
```

```
209             public WMS_LayerHierarchyClass this[int index]
210             {
211                 get{return (WMS_LayerHierarchyClass) base[index];}
212                 set{base[index] = (WMS_LayerHierarchyClass) value;}
213             }
214         }
215         public class MetadataURLArrayList : ArrayList
216         {
217             #region Costruttore
218             /// <summary>
219             /// Costruttore predefinito.
220             /// </summary>
221             public MetadataURLArrayList(int capacity) :base(capacity)                  {}
222             #endregion
223             /// <summary>
224             /// Accessore del vettore.
225             /// </summary>
226             public MetadataURLClass this[int index]
227             {
228                 get{return (MetadataURLClass) base[index];}
229                 set{base[index]=(MetadataURLClass) value;}
230             }
231             public MetadataURLClass GetMetadataURL(int index){return (MetadataURLClass)this[index];}
232             #region Operatori di confronto == e !=
233             public static bool operator == (MetadataURLArrayList a, MetadataURLArrayList b)
234             {
235                 bool res = true;
236                 if ( null==(object)a || null==(object)b)
237                     if (null==(object)a && null==(object)b)
238                         return true;
239                     else
240                         return false;
241                 if(a.Count == b.Count)
242                     {int i; for (i=0;(i<a.Count && res);i++) res = (a[i]==b[i]);}
243                 else
244                     res = false;
245                 return res;
246             }
247             public static bool operator != (MetadataURLArrayList a, MetadataURLArrayList b)
248             {
249                 bool res = true;
250                 if(a.Count == b.Count)
251                     {int i; for (i=0;(i<a.Count && res);i++) res = (a[i]!=b[i]); }
252                 else
253                     res = true;
254                 return res;
255             }
256             #endregion
257         }
258
259         /// <summary>
260         /// Moedella le coppie SRS MBR
261         /// </summary>
262         public class BBOXArrayList
263         {
264             #region Attributi privati
265             private ArrayList pSRSs = null;
266             private ArrayList pBBOXs = null;
267             #endregion
268
269             #region Costruttori
270             /// <summary>
271             /// Costruttore parametrico.
272             /// </summary>
273             /// <param name="capacity">Dimensione vettore.</param>
274             public BBOXArrayList(int capacity)
275             {this.pSRSs = new ArrayList(capacity); this.pBBOXs = new ArrayList(capacity); }
276             #endregion
277
278             /// <summary>
279             /// Metodo per aggiungere un layer.
280             /// </summary>
281             /// <param name="SRS">Stringa SRS</param>
282             /// <param name="mbr">Oggetto MBR</param>
283             /// <returns>Posizione ennupla inserita.</returns>
284             public int Add(string SRS, MBRClass mbr)
285             {int i = this.pSRSs.Add(SRS); int j = this.pBBOXs.Add(mbr); return i;}
286             /// <summary>
287             /// Restituisce un oggetto MBR.
288             /// </summary>
289             /// <param name="index">Posizione dell'oggetto all'interno della lista.</param>
290             /// <returns>Oggetto MBR.</returns>
291             public MBRClass MBR(int index)
292             {
293                 if (index>=this.pBBOXs.Count)
294                     return null;
295                 else
296                     return (MBRClass) this.pBBOXs[index];
297             }
298             /// <summary>
299             /// Restituisce un oggetto MBR.
300             /// </summary>
301             /// <param name="index">
302             /// SRS associato allo MBR da ricercare..
303             /// </param>
304             /// <returns>Oggetto MBR.</returns>
305             public MBRClass MBR(string srs)
306             {
307                 if(this.pSRSs.Contains(srs))
308                     return (MBRClass) this.pBBOXs[this.pSRSs.IndexOf(srs)];
309                 else
310                     return null;
311             }
312             /// <summary>
```

```
313             /// Restituisce un SRS.
314             /// </summary>
315             /// <param name="index">Posizione del SRSall'interno della lista.</param>
316             /// <returns>Stringa SRS.</returns>
317             public string SRS(int index)
318             {
319                 if (index>=this.pSRSs.Count)
320                     return null;
321                 else
322                     return (string) this.pSRSs[index];
323             }
324             /// <summary>
325             /// Restituisce il numero di elementi elle collezione
326             /// </summary>
327             public int Count{get{return (null==this.pBBOXs) ? 0 : this.pBBOXs.Count;}}
328
329             #region Operatori di confronto == e !=
330             public static bool operator == (BBOXArrayList a, BBOXArrayList b)
331             {
332                 bool res = true;
333                 if( ((object)a)==null || ((object)b)==null )
334                     if( ((object)a)==null && ((object)b)==null )
335                         return true;
336                     else return false;
337                 if(a.Count == b.Count)
338                 {
339                     int i;for (i=0;(i<a.Count && res);i++) res = ( (a.pBBOXs[i]==b.pBBOXs[i]) && (a.pSRSs[i]==b.pSRSs[i]) );
340                 }
341                 else
342                     res = false;
343                 return res;
344             }
345
346             public static bool operator != (BBOXArrayList a, BBOXArrayList b)
347             {
348                 bool res = true;
349                 if( ((object)a)==null || ((object)b)==null )
350                     if( ((object)a)==null && ((object)b)==null )
351                         return false;
352                     else return true;
353                 if(a.Count == b.Count)
354                 {
355                     int i;for (i=0;(i<a.Count && res);i++) res = ( (a.pBBOXs[i]!=b.pBBOXs[i]) || (a.pSRSs[i]!=b.pSRSs[i]) );
356                 }
357                 else
358                     res = true;
359                 return res;
360             }
361             #endregion
362         }
363         /// <summary>
364         /// Definisce il criterio di ordinamento per i Layer.
365         /// L'ordinamento utilizza come criterio il campo "Name" in modo che la
366         /// lista sia ordinata in modo crescente.
367         /// </summary>
368         /// <remarks>Implementa l'interfaccia "IComparer".</remarks>
369         public class SortByNameASC : IComparer
370         {
371             public SortByNameASC(){}
372
373             public int Compare(object a, object b)
374                 {LayerClass aa =(LayerClass)a; LayerClass bb =(LayerClass)b; return String.Compare(aa.Name,bb.Name);}
375         }
376
377         public class LayerClassArrayList : ArrayList
378         {
379             #region Costruttori
380             /// <summary>
381             /// Costruttore parametrico.
382             /// </summary>
383             /// <param name="capacity">Dimensione vettore.</param>
384             public LayerClassArrayList(int capacity) : base(capacity){}
385             /// <summary>
386             /// Costruttore predefinito.
387             /// </summary>
388             public LayerClassArrayList(){}
389             #endregion
390
391             /// <summary>
392             /// Metodo per aggiungere un layer.
393             /// </summary>
394             /// <param name="layer">Oggetto layer WFS.</param>
395             /// <returns>Posizione del layer all'interno del vettore</returns>
396             public int Add(LayerClass layer){return base.Add(layer);}
397             /// <summary>
398             /// Accessore del vettore.
399             /// </summary>
400             public LayerClass this[int index]
401             {
402                 get{return (LayerClass) base[index];}
403                 set{base[index]=(LayerClass) value;}
404             }
405             /// <summary>
406             /// Effettua l'ordinamento dei Layer in ordine crescente in base al nome.
407             /// </summary>
408             /// <remarks>Utilizza la classe "SortByNameASC" come criterio di ordinamento.</remarks>
409             public override void Sort() {SortByNameASC s = new SortByNameASC();base.Sort (s);}
410         }
411     }
```

```
using System;
using System.Xml;
using OWSServiceCrawler.TypedArrayList;

namespace OWSServiceCrawler
{
    /// <summary>
    /// Definisce un insieme di funzionalità utili all'implementazione
    /// delle classi del namespace.
    /// </summary>
    /// <remarks>
    /// Classe definita "internal", visibile quindi solo all'interno del namespace.
    /// </remarks>
    public class Utils
    {
        public Utils(){}
        /// <summary>
        /// Recupera le parole chiave relative ad un nodo.
        /// </summary>
        /// <remarks>
        /// Nel caso di un WFS va a ricercare l'elemento <Keywords>,
        /// mentre per un WMS ricerca gli elementi <Keyword>
        /// all'interno dell'elemento <KeywordList>.
        /// </remarks>
        /// <param name="node">Nodo che contiene l'elemento <Keywords> oppure <KeywordList>.</param>
        /// <param name="prefix">Prefisso legato ai tag del documento "capabilities".</param>
        /// <example> <wms:ContactInformation> restituisce "wms" </example>
        /// <param name="nsmgr">
        /// Oggetto di tipo "XmlNamespaceManager" relativo al documento "capabilities".
        /// </param>
        /// <returns>
        /// lista di parole chiavi: la lista è un oggetto di tipo string[]
        /// null: nel caso non vi siano parole chiave.
        /// </returns>
        internal static string[] GetKeywordList(XmlNode node, string prefix, XmlNamespaceManager nsmgr)
        {
            XmlNode tmp = null;
            XmlNodeList ltmp = null;
            int i=0;
            string[] Keywords = null;
            tmp = node.SelectSingleNode(@"./" + prefix + "Keyword",nsmgr);
            if (null!=tmp)
            {
                Keywords = tmp.InnerText.Split(',');
                for (i=0; i<Keywords.Length;i++) Keywords[i]=Keywords[i].Trim();
            }//Keyword
            else
            {//Si è nel caso di un WMS e quindi ho <KeywordList><keyword>???</keyword>,... </KeywordList>
                tmp = node.SelectSingleNode(@"./" + prefix + "KeywordList",nsmgr);
                if( (null!=tmp) && tmp.HasChildNodes)//ContactAddress
                {
                    ltmp=tmp.ChildNodes;
                    Keywords = new string[ltmp.Count];
                    for (i=0;i< ltmp.Count;i++) Keywords[i] = ltmp.Item(i).InnerText;
                }//KeywordList
            }
            tmp = null; ltmp = null;
            return Keywords;
        }
        /// <summary>
        /// Restituisce il prefisso da utilizzare nelle query XPath.
        /// </summary>
        /// <param name="rootNode">Nodo root del documento</param>
        /// <param name="nsmgr">Namespace manager del documento.</param>
        /// <returns></returns>
        internal static string GetDocumentPrefix(XmlNode rootNode,XmlNamespaceManager nsmgr)
        {

            string URI="",pref="";
            string dummyprefix="";
            XmlAttributeCollection atts = rootNode.Attributes;
            if (null!=atts)
            {
                int i;
                for (i=0;i<atts.Count;i++)
                {
                    if(@"xmlns"==atts[i].Prefix.ToLower())
                    {URI=atts[i].Value.ToString();pref=atts[i].LocalName.ToString();}
                    else
                        if (@"xmlns"==atts[i].LocalName.ToLower())
                    {URI=atts[i].Value.ToString();pref=atts[i].Prefix.ToString();}
                    if(""!=URI)
                    {
                        // Questo per ovviare a problemi di prefisso.
                        // Può capitare di trovare documenti dove è specificato il namaspace di default,
                        //in questo caso non è possibile fare query XPath senza prefisso.
                        // Nel namespacemanager viene associato al namespace di default un prefisso fittizio.
                        if(""==pref) {pref=dummyprefix="prefissofinto";}
                        nsmgr.AddNamespace(pref,URI);
                    }
                }
                atts=null;
                return ( (rootNode.Prefix!="") ? rootNode.Prefix : dummyprefix);
            }
            else
                return "";
        }
        /// <summary>
        /// Appiattisce un vettore di stringhe.
        /// </summary>
        /// <param name="s">vettore di stringe di tipo string[].</param>
        /// <returns>Ritorna una stringa, dove ogni elemento è separato dal carattere ";".</returns>
        internal static string FlatternArray(string[] s)
        {
```

```
            int i = 0;
            string t = "";
            if (null!=s)
                if (s.Length>0)
                {
                    for (i=0;i<s.Length;i++)
                        t+=s[i].Trim() + ";";
                    if(';'==t[t.Length-1])//caso in cui non c'è un ";" nella strnga => un solo elemento.
                        return t.Substring(0,t.Length-1);
                    else return t;
                }
            return "";
        }

        /// <summary>
        /// Dato un documento XMl contenente le schema XSD, dei layer
        /// dell'istanza del servizio, ottenuto tramite la richiesta
        /// "SecribeFeatureType", recupera i campi di un dato layer.
        /// </summary>
        /// <param name="strLayerName">Nome del layer.</param>
        /// <param name="xd">
        /// documento di tipo "XmLDocument" contenente lo schema XSD.
        /// </param>
        /// <param name="msg">Messaggio di errore</param>
        /// <returns>La lista di campi di tipo "LayerFields".</returns>
        public static LayerFields GetFieldsFromDFT(string strLayerName, XmlDocument xd, ref string msg)
        {
            //string request = URL + "?request=describeFeatureType";
            int i = 0;
            string nm ="", tp = "", qry = "", prefix = "";
            bool isgeo = false;
            //XmlDocument xd = new XmlDocument();
            try
            {   XmlNode tmp = null, subtmp = null;
                LayerFields list = null;
                //xd.Load(request);
                XmlNode root = xd.DocumentElement;
                XmlNamespaceManager nsmgr = new XmlNamespaceManager(xd.NameTable);
                prefix = Utils.GetDocumentPrefix(root,nsmgr);
                if (""!=prefix) prefix += ":";

                qry = @"./" + prefix + "element[@name='" + strLayerName.ToLower() + "']//" + prefix + "sequence";
                tmp = root.SelectSingleNode(qry,nsmgr);
                if(null==tmp)
                {
                    qry = @"./" + prefix + "element[@name='" + strLayerName.ToUpper() + "']//" + prefix + "sequence";
                    tmp = root.SelectSingleNode(qry,nsmgr);
                    if(null==tmp)
                    {
                        strLayerName = strLayerName[0].ToString().ToUpper() + strLayerName.Substring(1);
                        qry = @"./" + prefix + "element[@name='" + strLayerName + "']//" + prefix + "sequence";
                        tmp = root.SelectSingleNode(qry,nsmgr);
                    }
                }
                int l = tmp.ChildNodes.Count;
                list = new LayerFields(l);
                for (i=0;i<l;i++)
                {
                    isgeo = false;
                    nm = tmp.ChildNodes[i].Attributes["name"].Value;
                    if (null==tmp.ChildNodes[i].Attributes["type"])// Il tipo può essere annidato in un figlio.
                    {                       //nel caso al tipo sia associato un numero di cifre (lunghezza).
                        qry = @".//" + prefix + "restriction"; // Questo accade per il tipo stringa (string), long (long)
                        subtmp = tmp.ChildNodes[i].SelectSingleNode(qry,nsmgr);
                        tp = subtmp.Attributes["base"].Value;
                        if("gml:GeometryPropertyType"==tp)//è un poligono.
                        {
                            qry = @".//" + prefix + "element";
                            subtmp = subtmp.SelectNodes(qry,nsmgr).Item(0);
                            if(null!=subtmp)
                                tp = subtmp.Attributes["ref"].Value;
                            else
                                tp = "?";
                        }
                    }
                    else
                        tp = tmp.ChildNodes[i].Attributes["type"].Value;
                    tp = tp.Replace(prefix,"");//Alla stringa che identifica il tipo è associato anche il prefisso:
"xsd:long"
                    isgeo = (tp.IndexOf("gml")>=0);
                    if(isgeo)
                    {
                        tp = tp.Replace("gml:","").ToLower();
                        if(tp.IndexOf("polygon")>-1) tp =((int) LayerGeometryType.gtAreal).ToString();
                        else
                            if(tp.IndexOf("point")>-1) tp = ((int) LayerGeometryType.gtPoint).ToString();
                            else
                            if( (tp.IndexOf("line")>-1) || (tp.IndexOf("curve")>-1)) tp = ((int)
LayerGeometryType.gtLinear).ToString();
                            else
                                tp = ((int) LayerGeometryType.gtNA).ToString();
                    }
                    list.Add(new LayerField(nm,tp,isgeo));
                }
                tmp = subtmp = root = null;
                xd = null;
                return list;
            }
            catch(Exception e)
                {msg = "GetFieldsFromDFT::\n" + e.ToString();return null;}
        }
    }
}
```

F.6. Codice Namespace CoreCatalogueInterface

```
using System;
using System.IO;
using System.Xml;
using System.Data;
using System.Data.OleDb;
using System.Collections;
//*********************************
using OWSServiceCrawler;
using OWSServiceCrawler.TypedArrayList;

namespace CoreCatalogueInterface
{
    /// <summary>
    /// Gestore del catalogo.
    /// </summary>
    /// <remarks>
    ///  L'interazione col catalogo (in questo caso rappresentatoi tramite un file
    /// MS Access2000 MDB  a cui si accede tramite una connessione Oledb) avviene tramite
    /// transazioni. Le informazioni in esso contenute sono pertanto da considerarsi complete.
    ///  Gli errori marcati con "CoreMGR(num)" sono da considerarsi "gravi" in quanto indicano un funzionamento
    /// anomalo dovuto a malfunzionamenti od errate impostazioni delle operazioni sulla base di dati.
    ///  Gli altri messaggi servono per fornire un feedback all'utente; non minano il funzionamento
    /// del catalogo.
    /// </remarks>
    public class CoreManager
    {
        #region Campi privati
        private string pDBSource = "";

        /// <summary>
        /// stringa di connessione, in questo caso è relativ alla connessione ad un motore JET 4
        /// </summary>
        private string strConnectionString = "";
        /// <summary>
        /// Ogetto connessione a visibilità privata.
        /// </summary>
        /// <remarks>La connessione DEVE essere tenuta a perta il meno possibile.</remarks>
        private OleDbConnection Conn = null;
        /// <summary>
        /// Oggetto transazione, utilizzato per gestire le transazione con il catalogo.
        /// </summary>
        private OleDbTransaction myTrans = null;
        #endregion

        #region Properties
        /// <summary>
        /// Permette di leggere il nome e percorso del catalogo.
        /// </summary>
        public string DBSource{ get{return this.pDBSource;}}

        /// <summary>
        /// Permette di sapere se il catalogo esiste nel percorso specificato.
        /// </summary>
        /// <value>"true" se esiste.</value>
        public bool CatalogExist
        {
            get
            {
                if(null!= this.pDBSource && "" !=this.pDBSource)
                    return File.Exists(this.pDBSource);
                else
                    return false;
            }
        }

        #endregion

        #region Costruttori
        /// <summary>
        /// Costruttore del gestore del catalogo
        /// </summary>
        /// <param name="strDBsource">Percorso e nome del catalogo</param>
        public CoreManager(string strDBsource)
        {
            if(null!=strDBsource && ""!=strDBsource && File.Exists(strDBsource))
            {
                this.pDBSource = strDBsource;
                this.strConnectionString = "Provider=Microsoft.Jet.OLEDB.4.0; Data Source=" + this.DBSource;
            }
        }
        /// <summary>
        /// Costruttore.
        /// </summary>
        public CoreManager(){this.pDBSource = "";}
        #endregion

        #region Metodi pubblici
        /// <summary>
        /// Aggiunge un'istanza di servizio nel catalogo.
        /// </summary>
        /// <param name="srvc">Rapresentazione dell'istanza del servizio.</param>
        /// <param name="msg">Stringa passata per riferimento. Contiene eventuali messaggi di errore.</param>
        /// <returns>"true" se il servizio è stato inserito correttamente.</returns>
        public bool AddService(ServiceClass srvc, ref string msg)//eseguo un up-cast
        {
            if (!this.CatalogExist)
                {msg += ((char)13) + "CoreMGR(1): Unable to locate the Catalog.";return false;}
            if("WMS"==srvc.IstanceType) {return AddWMS_Service((WMS_ServiceClass)srvc, ref msg);}
            else
                return AddWFS_Service((WFS_ServiceClass)srvc, ref msg);//DownCast
```

```
        }

        /// <summary>
        /// Aggiorna un servizio in catalogo in base alla rappresentazione della stessa istanza
        /// ottenuta dalla rete.
        /// </summary>
        /// <param name="onlinesrvc">Rappresentazione dell'istanza del servizio ottenuta dalla rete.</param>
        /// <param name="msg">stringa passata per riferimento. Contiene eventuali messaggi di errore.</param>
        /// <returns>
        /// Valore intero:
        /// -1:Non è stato effettuato alcun aggiornamento a causa di un errore.
        ///  0:Non è stato effettuato alcun aggiornamento.
        ///  1:Sono state aggiornate solo le informazioni relative al servizio.
        ///  2:Sono state aggiornate solo le informazioni di tutti o di alcuni layer.
        ///  3:Sono state aggiornate sia le informazioni del servizio che di tutti od alcuni layer.
        /// </returns>
        public int UpdateService(ServiceClass onlinesrvc, ref string msg)
        {
            if (!this.CatalogExist) {msg += ((char)13) + "CoreMGR(2): Unable to locate the Catalog.";return -1;}
            string pURL = onlinesrvc.URL;
            string pType = onlinesrvc.IstanceType;
            long sId = -1;
            int res = 0;
            OleDbCommand cmd = null;
            this.Conn = new OleDbConnection(this.strConnectionString);
            try
            {
                this.Conn.Open();
                cmd = this.Conn.CreateCommand();
                cmd.CommandText = "SELECT Count(Sid) FROM Tbl_OGCServices WHERE URL='" + pURL + "' AND ServiceType='" +
pType + "';";
                sId = long.Parse(cmd.ExecuteScalar().ToString());
                cmd.Dispose();this.Conn.Close();this.Conn.Dispose();
                if(1==sId)
                {
                    if("WMS"==onlinesrvc.IstanceType)
                        res = UpdateWMS_Service((WMS_ServiceClass)onlinesrvc, ref msg);
                    else
                    {
                        res = UpdateWFS_Service((WFS_ServiceClass)onlinesrvc, ref msg);//DownCast
                        ((WFS_ServiceClass)onlinesrvc).GetGeometryType(false, ref msg);
                    }
                }
            }
            catch(Exception e)  {res = -1; msg += (char)13 + "CoreMGR(3): " + e.ToString();}
            if (null!=this.Conn)
            {this.Conn.Close(); this.Conn.Dispose();this.Conn = null;}
            this.myTrans = null;return res;
        }

        /// <summary>
        /// Elimina le informazioni di un'istanzza di servizio dal catalogo.
        /// </summary>
        /// <param name="URL"></param>
        /// <param name="serviceType"></param>
        /// <param name="msg">stringa passata per riferimento. Contiene eventuali messaggi di errore.</param>
        /// <returns>
        /// -1:Eliminazione fallita a causa di un errore.
        ///  0:Nessuna eliminazione, elemento inesistente.
        ///  1:Eliminazione effettuata.
        /// </returns>
        public int DeleteService(string URL, string serviceType, ref string msg)
        {
            if (!this.CatalogExist)
            {msg += ((char)13) + "CoreMGR(4): Unable to locate the Catalog.";return -1;}
            int nup = 0;
            try
            {
                this.Conn = new OleDbConnection(this.strConnectionString);
                this.Conn.Open();
                this.myTrans = this.Conn.BeginTransaction();
                OleDbCommand cmd = this.Conn.CreateCommand();
                cmd.Transaction = this.myTrans;
                cmd.CommandText = "SELECT Count(Sid) FROM Tbl_OGCServices WHERE URL='" + URL + "' AND ServiceType='" +
serviceType +"';";
                long l = long.Parse(cmd.ExecuteScalar().ToString());
                if (1==l)
                {
                    cmd.CommandText = "SELECT Sid FROM Tbl_OGCServices WHERE URL='" + URL + "' AND ServiceType='" +
serviceType +"';";
                    l = long.Parse(cmd.ExecuteScalar().ToString());
                    cmd.CommandText = "DELETE * FROM Tbl_OGCServices WHERE sid=" + l + ";";
                    cmd.ExecuteNonQuery();  nup = 1;
                }
            }
            catch(Exception e)
                {msg = "CoreMGR(4.1): " + e.ToString();nup = -1;}
            if(null!=this.Conn)
                if (this.Conn.State == ConnectionState.Open)
                {
                    if(-1==nup)
                        this.myTrans.Rollback();
                    else
                        this.myTrans.Commit();
                    this.Conn.Close();  this.Conn.Dispose();this.Conn = null;
                }
            this.myTrans = null;;
            return nup;
        }

        /// <summary>
        /// Elimina le informazioni di un'istanzza di servizio dal catalogo.
        /// </summary>
        /// <param name="sId">Identificatore dell'istanza di servizio all'interno del catalogo.</param>
        /// <param name="msg">stringa passata per riferimento. Contiene eventuali messaggi di errore.</param>
        /// <returns>
```

```
        /// -1:Eliminazione fallita a causa di un errore.
        ///  0:Nessuna eliminazione, elemento inesistente.
        ///  1:Eliminazione effettuata.
        /// </returns>
        public int DeleteService(long sId,ref string msg)
        {
            if (!this.CatalogExist)
            {msg += ((char)13) + "CoreMGR(5): Unable to locate the Catalog.";return -1;}
            int nup = 0;
            try
            {
                this.Conn = new OleDbConnection(this.strConnectionString);
                this.Conn.Open();
                this.myTrans = this.Conn.BeginTransaction();
                OleDbCommand cmd = this.Conn.CreateCommand();
                cmd.Transaction = this.myTrans;
                cmd.CommandText = "SELECT Count(Sid) FROM Tbl_OGCServices WHERE sid=" + sId + ";";
                long l = long.Parse(cmd.ExecuteScalar().ToString());
                if (1==l)
                {
                    cmd.CommandText = "DELETE * FROM Tbl_OGCServices WHERE sid=" + sId + ";";
                    cmd.ExecuteNonQuery();  nup = 1;
                }
            }
            catch(Exception e)  {msg = "CoreMngr(5.1): " + e.ToString();nup = -1;}
            if(null!=this.Conn)
                if(this.Conn.State == ConnectionState.Open)
                {
                    if(-1==nup)
                        this.myTrans.Rollback();
                    else
                        this.myTrans.Commit();
                    this.Conn.Close();
                    this.Conn.Dispose();this.Conn = null;
                }
            this.myTrans = null;
            return nup;
        }

        /// <summary>
        /// Costruisce la rappresentazione di un servizio WFS basandosi sulle informazioni contenute
        /// nel catalogo.
        /// </summary>
        /// <param name="URL">URL dell'istanza del servizio.</param>
        /// <param name="msg">Stringa passata per riferimento. Contiene eventuali messaggi di errore.</param>
        /// <returns>"null" se si è verificato un errore durante la ricerca.</returns>
        /// <returns>rappresentazione del'istanza del servizio WFS.</returns>
        public WFS_ServiceClass GetWFSServiceFromCatalog(string URL, ref string msg)
        {
            WFS_ServiceClass res = null;
            OleDbCommand cmd = null;
            this.Conn = new OleDbConnection(this.strConnectionString);
            try
            {
                this.Conn.Open();
                cmd = this.Conn.CreateCommand();
                cmd.CommandText = "SELECT Count(Sid) FROM Tbl_OGCServices WHERE URL='" + URL + "' AND
ServiceType='WFS';";
                long l = long.Parse(cmd.ExecuteScalar().ToString());
                cmd.Dispose();
                if(1==l)
                    res = GetWFS_ServiceFromCatalog( URL, "WFS", ref msg);
                else
                {res = null;msg = "The requeste service id not present; try to change the type.";}
                cmd = null;
            }
            catch(Exception e)
                {res = null;msg = e.ToString(); cmd = null;}
            if (null!=this.Conn){this.Conn.Close();this.Conn.Dispose();this.Conn = null;}
            return res;
        }

        /// <summary>
        /// Costruisce la rappresentazione di un servizio WMS basandosi sulle informazioni contenute
        /// nel catalogo.
        /// </summary>
        /// <param name="URL">URL dell'istanza del servizio.</param>
        /// <param name="msg">Stringa passata per riferimento. Contiene eventuali messaggi di errore.</param>
        /// <returns>"null" se si è verificato un errore durante la ricerca.</returns>
        /// <returns>rappresentazione del'istanza del servizio WMS.</returns>
        public WMS_ServiceClass GetWMSServiceFromCatalog(string URL, ref string msg)
        {
            WMS_ServiceClass res = null;
            OleDbCommand cmd = null;
            this.Conn = new OleDbConnection(this.strConnectionString);
            try
            {
                this.Conn.Open();
                cmd = this.Conn.CreateCommand();
                cmd.CommandText = "SELECT Count(Sid) FROM Tbl_OGCServices WHERE URL='" + URL + "' AND
ServiceType='WMS';";
                long l = long.Parse(cmd.ExecuteScalar().ToString());
                cmd.Dispose();
                if(1==l)
                    res = GetWMS_ServiceFromCatalog( URL, "WMS", ref msg);
                else
                    {res = null;msg = "The requeste service id not present; try to change the type.";}
                cmd = null;
            }
            catch(Exception e){res = null;msg = e.ToString();cmd = null;}
            if (null!=this.Conn){this.Conn.Close();this.Conn.Dispose();this.Conn = null;}
            return res;
        }

        /// <summary>
        /// Permette di eseguire un'interrogazione di tipo SELECT sul catalogo.
```

```
        /// </summary>
        /// <param name="sfwStatment">Interrogazione da eseguire</param>
        /// <param name="msg">Stringa passata per riferimento. Contiene eventuali messaggi di errore.</param>
        /// <returns>Un dataset contenente il risultato dell'interrogazione.</returns>
        /// <return>"null" nel caso che vi sisano stati problemi di lettura o che l'interrogazione non sia
permessa.</return>
        /// <remarks>
        /// E' Questo metodo evita di operare direttamente sul catalogo, ma da la possibilità
        /// di recuperare le informazioni in esso contenute.
        /// </remarks>
        public DataSet ExecuteSFW(string sfwStatment, ref string msg)
        {
            sfwStatment = sfwStatment.ToLower();
            if (//Sono ignorate query che alterano il catalogo
                sfwStatment.LastIndexOf("insert")>-1 ||sfwStatment.LastIndexOf("update")>-1 ||
                sfwStatment.LastIndexOf("delete")>-1 || sfwStatment.LastIndexOf("alter")>-1)
            {Msg +=((char)13) + "CoreMGR(6): Forbidden query statment.";return null;}
            else
            {
                this.Conn = new OleDbConnection(this.strConnectionString);
                DataSet ds = null;
                OleDbDataAdapter da = null;
                try
                {
                    this.Conn.Open();
                    ds = new DataSet("ServizioLocale");
                    da = new OleDbDataAdapter(sfwStatment,this.Conn);
                    da.Fill(ds);da.Dispose();this.Conn.Close(); this.Conn.Dispose();this.Conn = null;
                    return ds;
                }
                catch(Exception e)
                {
                    if (null!=this.Conn)
                    {this.Conn.Close(); this.Conn.Dispose();this.Conn = null;}
                    msg += ((char)13) + "CoreMGR(7):" + e.ToString();
                    return null;
                }
            }
        }

        /// <summary>
        /// Restituisce un valore in base all'interrogazione richiesta.
        /// </summary>
        /// <param name="sfwStatment">Interrogazione da eseguire</param>
        /// <param name="msg">Stringa passata per riferimento. Contiene eventuali messaggi di errore. </param>
        /// <returns>Il valore rischiesto in formato stringa.</returns>
        /// <returns>Stringa vuota ("") in caso di errori.</returns>
        public string Get1Value(string sfwStatment, ref string msg)
        {
            sfwStatment = sfwStatment.ToLower();
            OleDbCommand cmd = null;
            if (//Sono ignorate query che alterano il catalogo
                sfwStatment.LastIndexOf("insert")>-1 ||sfwStatment.LastIndexOf("update")>-1 ||
                sfwStatment.LastIndexOf("delete")>-1 ||sfwStatment.LastIndexOf("alter")>-1)
                {msg +=((char)13) + "CoreMGR(7.1): Forbidden query statment.";return "";}
            else
            {
                this.Conn = new OleDbConnection(this.strConnectionString);
                try
                {
                    this.Conn.Open();
                    cmd = this.Conn.CreateCommand();
                    cmd.CommandText = sfwStatment;
                    string t = cmd.ExecuteScalar().ToString();
                    this.Conn.Close(); this.Conn.Dispose();this.Conn = null;
                    return t;
                }
                catch(Exception e)
                {
                    if (null!=this.Conn){this.Conn.Close();this.Conn.Dispose();this.Conn = null;}
                    msg += ((char)13) + "CoreMGR(7.2):" + e.ToString();return "";
                }
            }
        }
        #endregion

#region Update Service To Catalog
        #region Note
        //** Update Service To Catalog ********************************************************************************
        //* Tutti questi metodi privati interagiscono con il catalogo tramite transazioni, quando viene ritornato    *
        //* un errore (il metodo ritorna false) dovuto ad un problema di scrittura, è da considerarsi un problema    *
        //* relativo alla transazione. Solo i metodi "UpdateWFS_Service" e "UpdateWMS_Service" riportano l'errore di *
        //* scrittura sul catalogo (che non viene effettuata in quanto sono loro ad eseguire il roolback della       *
        //* transazione.                                                                                            *
        //*************************************************************************************************************
        #endregion

        /// <summary>Aggiorna le informazioni, di un'istanza di un servizio WMS, presenti nel catalogo.</summary>
        /// <param name="osrvc">Istanza del servizio di confronto</param>
        /// <param name="msg">stringa passata per riferimento. Contiene eventuali messaggi di errore.</param>
        /// <returns>
        /// Valore intero:
        /// -1:Non è stato effettuato alcun aggiornamento a causa di un errore.
        ///  0:Non è stato effettuato alcun aggiornamento.
        ///  1:Sono state aggiornate solo le informazioni relative al servizio.
        ///  2:Sono state aggiornate solo le informazioni di tutti o di alcuni layer.
        ///  3:Sono state aggiornate sia le informazioni del servizio che di tutti od alcuni layer.
        /// </returns>
        /// <remarks>
        /// Viene passato in input la rappresentazione del servizio WFS che si vuole confrontare
        /// con l'istanza dellostesso presente nel catalogo.
        /// </remarks>
        private int UpdateWMS_Service(WMS_ServiceClass osrvc, ref string msg)
        {

            int res = 0;
```

```
            try
            {
                this.Conn = new OleDbConnection(this.strConnectionString);
                this.Conn.Open();
                WMS_ServiceClass csrvc = GetWMS_ServiceFromCatalog(osrvc.URL, osrvc.IstanceType, ref msg);
                if(null!=csrvc)
                {
                    this.myTrans = this.Conn.BeginTransaction();
                    if (!osrvc.ServiceInfo.IsLike(csrvc.ServiceInfo))
                        {res = UpdateSrvcInf(osrvc.ServiceInfo,csrvc.sId,ref msg);}
                    if(res>-1)
                    {
                        int i=0;
                        bool b = true;
                        WMS_LayerClassArrayList oll = new WMS_LayerClassArrayList();
                        if(null!=osrvc.Layers.LayersHierarchy)//se ci sono layer non in gerarchia
                            for(i=0;(i<osrvc.Layers.LayersHierarchy.Count && (res>-1));i++)
                                b = WMS_Stuffs.UnfoldHierarchy(osrvc.Layers.LayersHierarchy[i],null,null,null,oll);
                        if (b)
                        {
                            if(null!=osrvc.Layers.Layers)//se ci sono layer non in gerarchia
                                for (i=0;i<osrvc.Layers.Layers.Count;i++) oll.Add(osrvc.Layers.Layers[i]);
                            oll.Sort();
                            int g = UpdateWMSLayers(oll,csrvc.Layers.Layers, csrvc.sId, ref msg);
                            res = ((g>-1) ? res+g : -1);
                        }
                        else res = -1;
                    }
                }
                else{msg += (char)13 + "Unable to find the service in the Catalog.";res = -1;}
            }
            catch(Exception e){res= -1;msg = e.ToString();}
            if(null!=this.Conn)
                if (this.Conn.State == ConnectionState.Open)
                {
                    if(res>-1)
                        this.myTrans.Commit();
                    else
                        {this.myTrans.Rollback();res = -1;}
                    this.Conn.Close();this.Conn.Dispose();this.Conn = null;
                }
            this.myTrans = null;
            return res;
        }

        /// <summary>
        /// Costrisce la rappresentazione di un servizio WMS basandosi sulle informazioni contenute
        /// nel catalogo.
        /// </summary>
        /// <param name="URL">URL dell'istanza del servizio.</param>
        /// <param name="serviceType">Tipologia dell'istanza del servizio :WMS.</param>
        /// <param name="msg">Stringa passata per riferimento. Contiene eventuali messaggi di errore.</param>
        /// <returns>"null" se si è verificato un errore durante la ricerca.</returns>
        /// <returns>rappresentazione del'istanza del servizio WMS.</returns>
        private WMS_ServiceClass GetWMS_ServiceFromCatalog(string URL, string serviceType, ref string msg)
        {
            string sqlf = "SELECT * FROM";
            string sqlw = "WHERE URL='" + URL + "' AND ServiceType='" + serviceType + "';";
            try
            {
                //Creo una vista dei dati relativi al servizio
                DataSet ds = new DataSet("ServizioLocale");
                OleDbDataAdapter da = new OleDbDataAdapter(sqlf + " Tbl_OGCServices " +sqlw,this.Conn);
                da.Fill(ds, "Tbl_OGCServices");
                DataRow r   = ds.Tables["Tbl_OGCServices"].Rows[0];
                long sId = long.Parse(r["sid"].ToString());
                da = new OleDbDataAdapter(sqlf + " Tbl_Layers WHERE sid=" + sId + ";"  ,this.Conn);
                da.Fill(ds, "Tbl_Layers");
                da = new OleDbDataAdapter(sqlf + " Tbl_SRSBox WHERE sid=" + sId + ";"  ,this.Conn);
                da.Fill(ds, "Tbl_SRSBox");
                da.Dispose(); da = null;
                //Inzio popolazione servizio
                OWSServiceCrawler.WMS_ServiceClass s = new WMS_ServiceClass();
                s.URL = URL;
                s.IstanceType = serviceType;
                s.sId = sId;
                s.Version = ""; //<-- in futuro
                s.ServiceInfo = new WMS_ServiceInfosClass();
                s.ServiceInfo.Name = r["name"].ToString();
                s.ServiceInfo.Title = r["Title"].ToString();
                s.ServiceInfo.Keywords = r["Keywords"].ToString().Split(';');//<! vedere come si comporta con l'ultimo
                //Creazione lista Layer
                DataRowCollection dtL = ds.Tables["Tbl_Layers"].Rows;//Collezione di layer da scandire
                DataTable dtS = ds.Tables["Tbl_SRSBox"];
                WMS_LayerClass l = null;
                s.Layers = new WMS_CapabilityLayersClass();
                s.Layers.Layers = new WMS_LayerClassArrayList(dtL.Count);
                int i,j;
                string[] metadata = null;
                for (i=0;i<dtL.Count;i++)
                {
                    l = new WMS_LayerClass();
                    l.lId = long.Parse( dtL[i]["lid"].ToString());
                    l.Name = dtL[i]["Name"].ToString();
                    l.Title = dtL[i]["Title"].ToString();
                    string c = dtL[i]["Keywords"].ToString().Trim();
                    string[] v= null;v=c.Split(';');
                    if (null!=v && c!="")   l.Keywords = v;
                    c = dtL[i]["SRS"].ToString();v=c.Split(';');
                    if(null!=v && c!="")    l.SRS = v;
                    c = dtL[i]["MetadataURL"].ToString().Trim();
                    if (null!=c && c!="")//creazione metadata
                    {
                        v = c.Split(';');
                        l.MetadataURL = new MetadataURLArrayList(v.Length);
                        string u="",t="";
```

```
                        for(j=0;j<metadata.Length;j++)
                        {
                            u=(v[j].Split(':'))[0]; //in metadata la coppia è URL:type
                            t=(v[j].Split(':'))[1];
                            l.MetadataURL.Add(new MetadataURLClass(u,t));
                        }
                    }
                    else l.MetadataURL = null;
                    //Fine creazione metadata
                    //recupero latlon
                    DataRow[] ll = dtS.Select("lid=" + l.lId);//seleziono i BBOx che hanno lo Id uguale al layer
                    if(ll.Length>1) l.BBOX = new BBOXArrayList(ll.Length-1);//solo i boundingbox e non IL latlon
                    for(j=0;j<ll.Length;j++)
                    {
                        DataRow bb = ll[j];
                        MBRClass b = new MBRClass(double.Parse(bb["minx"].ToString()),
                            double.Parse(bb["miny"].ToString()),
                            double.Parse(bb["maxx"].ToString()),
                            double.Parse(bb["maxy"].ToString())
                            );//Me ne aspetto uno solo in qaunto è un servizio WFS
                        if(bool.Parse(bb["LatLon"].ToString()))
                            l.LatLongBoundingBox = b;
                        else
                            l.BBOX.Add(bb["SRS"].ToString(),b);
                        b = null;
                    }
                    s.Layers.Layers.Add(l);
                    l = null;ll = null;
                }//Fine FOR di creazione layer.
                s.Layers.Layers.Sort();
                return s;
            }
            catch(Exception e){msg = e.ToString();return null;}
        }

        /// <summary>
        /// Aggiorna le informazioni relative ai layers contenuti nel catalogo per
        /// l'istanza di servizio WMS specificato.
        /// </summary>
        /// <param name="oll">Lista di layers dell'istanza del servizio "online".</param>
        /// <param name="cll">Lista di layers dell'istanza del servizio presente nel catalogo.</param>
        /// <param name="sId">Identificativo del servizio nel catalogo.</param>
        /// <param name="msg">Stringa passata per riferimento. Contiene eventuali messaggi di errore.</param>
        /// <returns>
        /// -1:Si è verificato un errore.</description>
        ///  0:Non è stata preparata alcuna modifica.</description>
        ///  2:E' stata preparata una o più modifiche nella sezione layer del catalogo</description>
        /// </returns>
        private int UpdateWMSLayers(WMS_LayerClassArrayList oll, WMS_LayerClassArrayList cll, long sId, ref string msg)
        {
            OleDbCommand cmd = this.Conn.CreateCommand();
            WMS_LayerClass l = null; MBRClass bx =null;
            cmd.Transaction = this.myTrans;

            int k,i = 0;
            long supLid =-1;
            bool res = true;
            #region Stringhe di comodo per le query
            //stringa SQL "L"ayer_"i"nsert_"SQL"
            string L_iSql = @"INSERT INTO Tbl_Layers
(Lid,Sid,Name,Title,Keywords,MetadataURL,FeatureListURL,SRS,PreviewURL,GeometryType) VALUES(";
            //stringa SQL "L"ayer_"u"pdate_"SQL"
            string L_uSql = "UPDATE Tbl_Layers SET Title='" ;
            //condizione per eseguire la query di UPDATE per i layer
            string L_uSqlw = " WHERE sid=" + sId + " AND lid=";
            //I bounding box vengono eliminati e inseriti i nuovi nel caso di un update del layer
            string B_iSql = "INSERT INTO Tbl_SRSBox (Sid,Bid,latlon,Lid,SRS,minx,miny,maxx,maxy) VALUES(" + sId + ",";
            string B_dSql = "DELETE * FROM Tbl_SRSBox WHERE sid=" + sId + " AND lid=";
            #endregion
            string srs = "";
            int nup = 0;
            try
            {
                #region Calcolo dell nuovo id dei layer
                cmd.CommandText = "Select MAX(Lid) FROM tbl_layers WHERE (sid=" + sId +");";
                supLid = long.Parse(cmd.ExecuteScalar().ToString())+1;
                #endregion
                for (i=0;(i<cll.Count && res);i++)
                {//Preparo la lista dei layer online per essere aggiornata
                    k=cll.BinarySearch(oll[i]);
                    if(k>=0)// esiste
                        if(cll[k].IsLike(oll[i]) )//non lo tocco
                            cll[k].Name="__XNOTDELX__"; //marco i layer in catalogo che non vanno eliminati.
                        else
                        {
                            oll[i].lId = cll[k].lId;//Va aggiornato
                            l=oll[i];//puntatore di comodo
                            #region Aggiorno il Layer
                            cmd.CommandText = L_uSql + l.Title + "', Name='" + l.Name + "', "
                                + "Keywords='" + l.FlatKeywords + "', MetadataURL='" + l.FlatMetadataURL + "', "
                                + "SRS='" + l.FlatSRS + "', FeatureListURL='', GeometryType=33, PreviewURL='" +
l.LegendURL + "' "
                                + L_uSqlw + l.lId + ";";//WHERE
                            cmd.ExecuteNonQuery();
                            #endregion
                            //Elimino il bounding box del layer.
                            cmd.CommandText = B_dSql + l.lId + ";";
                            cmd.ExecuteNonQuery();
                            //metto quello nuovo.
                            #region Creo il nuovo LatLonboundingBox
                            cmd.CommandText=B_iSql //INSERT della tabella Tbl_SRSBox
                                + "0,true," + l.lId + ",'" + l.SRS[0].ToString() + "','" + l.LatLongBoundingBox.minx +
"','"
                                + l.LatLongBoundingBox.miny + "','" + l.LatLongBoundingBox.maxx + "','"
                                + l.LatLongBoundingBox.maxy + "');";
                            cmd.ExecuteNonQuery();
```

```
                            #endregion
                            cll[k].Name="__XNOTDELX__";//Marco il layer in modo che non venga eliminato.
                            #region Se vi sono dei BoundingBox, li inserisco...
                            if(null!=l.BBOX)
                            {
                                for(k=0;k<l.BBOX.Count;k++)
                                {
                                    bx = l.BBOX.MBR(k);  srs = l.BBOX.SRS(k);
                                    cmd.CommandText=B_iSql //INSERT della tabella Tbl_SRSBox
                                        + (k+1) + ",false," + l.lId + ",'" + srs + "','" + bx.minx + "','"
                                        + bx.miny + "','" + bx.maxx + "','"          + bx.maxy + "');";
                                    cmd.ExecuteNonQuery();
                                }
                                bx = null;
                            }
                            #endregion
                            nup=2;
                        }//if-branch controllo IsLike
                    else
                    {//devo aggiungerlo
                        oll[i].lId = supLid;
                        supLid++;//creazione nuovo ID ottenuto dal massimo degli id dei layer +1
                        #region Creazione nuovo layer e LatLonBoundingBox...
                        l = oll[i];//puntatore di comodo.
                        cmd.CommandText = L_iSql //INSERT della tabella Tbl_layers
                            + l.lId + "," + sId + ",'" + l.Name + "','" + l.Title + "','"
                            + l.FlatKeywords + "','" + l.FlatMetadataURL + "','" + l.FeatureListURL + "','" + l.FlatSRS +
"','" + ((null==l.LegendURL)?"":l.LegendURL) + "',33)";
                        cmd.ExecuteNonQuery();//EPSG:4326  è fisso per l'attributo latlonboundingbox.
                        cmd.CommandText=B_iSql //INSERT della tabella Tbl_SRSBox
                            + "0,true," + l.lId + ",'EPSG:4326','" + l.LatLongBoundingBox.minx + "','"
                            + l.LatLongBoundingBox.miny + "','"+ l.LatLongBoundingBox.maxx + "','"
                            + l.LatLongBoundingBox.maxx + "');";
                        cmd.ExecuteNonQuery();
                        #endregion
                        #region Se vi sono dei BoundingBox, li inserisco...
                        if(null!=l.BBOX)
                        {//In fede, questo codice Gira!!!
                            for(k=0;k<l.BBOX.Count;k++)
                            {
                                bx = l.BBOX.MBR(k);srs = l.BBOX.SRS(k);
                                cmd.CommandText=B_iSql //INSERT della tabella Tbl_SRSBox
                                    + (k+1) + ",false," + l.lId + ",'" + srs + "','" + bx.minx + "','"
                                    + bx.miny + "','" + bx.maxx + "','"             + bx.maxy + "');";
                                cmd.ExecuteNonQuery();
                            }
                            bx = null;
                        }
                        #endregion
                        nup = 2;
                    }//end inserimento nuovo
                }//End for
                #region Eliminazione layer che non sono presenti nel servizio
                for(i=0;i<cll.Count;i++)//elimino i vecchi layer
                {
                    if("__XNOTDELX__" != cll[i].Name)
                    {
                        cmd.CommandText = "DELETE * FROM Tbl_Layers WHERE lid=" + cll[i].lId;
                        cmd.ExecuteNonQuery();
                    }
                }
                #endregion
            }//end try
            catch(Exception e)  nup = -1;msg += ((char)13) + "CoreMGR(9): " + e.ToString();}
            cmd = null;
            return nup;
        }

        /// <summary>
        /// Aggiorna le informazioni, di un'istanza di un servizio WFS, presenti nel catalogo.
        /// </summary>
        /// <param name="osrvc">Istanza del servizio di confronto</param>
        /// <param name="msg">stringa passata per riferimento. Contiene eventuali messaggi di errore.</param>
        /// <returns>
        /// Valore intero:
        /// -1:Non è stato effettuato alcun aggiornamento a causa di un errore.
        ///  0:Non è stato effettuato alcun aggiornamento.
        ///  1:Sono state aggiornate solo le informazioni relative al servizio.
        ///  2:Sono state aggiornate solo le informazioni di tutti o di alcuni layer.
        ///  3:Sono state aggiornate sia le informazioni del servizio che di tutti od alcuni layer.
        /// </returns>
        /// <remarks>
        /// Viene passato in input la rappresentazione del servizio WFS che si vuole confrontare
        /// con l'istanza dellostesso presente nel catalogo.
        /// </remarks>
        private int UpdateWFS_Service(WFS_ServiceClass osrvc, ref string msg)
        {
            int res = 0;
            try
            {
                this.Conn = new OleDbConnection(this.strConnectionString);
                this.Conn.Open();
                WFS_ServiceClass csrvc = GetWFS_ServiceFromCatalog(osrvc.URL, osrvc.IstanceType, ref msg);
                if(null!=csrvc)
                {
                    this.myTrans = this.Conn.BeginTransaction();
                    if (!osrvc.ServiceInfo.IsLike(csrvc.ServiceInfo))
                        {res = UpdateSrvcInf(osrvc.ServiceInfo,csrvc.sId,ref msg);}
                    if(res>-1)
                        {osrvc.Layers.Sort();res += UpdateWFSLayers(osrvc.Layers,csrvc.Layers, csrvc.sId, ref msg);}
                }
                else
                    {msg += (char)13 + "Unable to find the service in the Catalog."; res = -1;}
            }
            catch(Exception e)
                {res= -1;msg = e.ToString();}
```

```
            if(null!=this.Conn)
                if (this.Conn.State == ConnectionState.Open)
                {
                    if(res>0)//è stata preparata una modifica consistente al catalogo.
                        this.myTrans.Commit();
                    else
                        {this.myTrans.Rollback();res = -1;}
                    this.Conn.Close();this.Conn.Dispose();this.Conn = null;
                }
                this.myTrans = null;
            return res;
        }

        /// <summary>
        /// Aggiorna le informazioni relative ai layers contenuti nel catalogo per
        /// l'istanza di servizio WFS specificato.
        /// </summary>
        /// <param name="oll">Lista di layers dell'istanza del servizio "online".</param>
        /// <param name="cll">Lista di layers dell'istanza del servizio presente nel catalogo.</param>
        /// <param name="sId">Identificativo del servizio nel catalogo.</param>
        /// <param name="msg">Stringa passata per riferimento. Contiene eventuali messaggi di errore.</param>
        /// <returns>
        /// -1:Si è verificato un errore.</description>
        ///  0:Non è stata preparata alcuna modifica.</description>
        ///  2:E' stata preparata una o più modifiche nella sezione layer del catalogo</description>
        /// </returns>
        private int UpdateWFSLayers(WFS_LayerClassArrayList oll, WFS_LayerClassArrayList cll, long sId, ref string msg)
        {
            OleDbCommand cmd = this.Conn.CreateCommand();
            WFS_LayerClass l = null;
            cmd.Transaction = this.myTrans;
            int k,i = 0;
            long supLid =-1;
            int res = 0;
            //stringa SQL "L"ayer_"i"nsert_"SQL"
            string L_iSql = @"INSERT INTO Tbl_Layers
(Lid,Sid,Name,Title,Keywords,MetadataURL,FeatureListURL,SRS,PreviewURL,GeometryType)) VALUES(";
            //stringa SQL "L"ayer_"u"pdate_"SQL"
            string L_uSql = "UPDATE Tbl_Layers SET Title='" ;
            //condizione per eseguire la query di UPDATE per i layer
            string L_uSqlw = " WHERE sid=" + sId + " AND lid=";
            //I bounding box vengono eliminati e inseriti i nuovi nel caso di un update del layer
            //Per un WFS ho sempre un bounding box (id=0) ed è sempre un latlon (true)
            string B_iSql = "INSERT INTO Tbl_SRSBox (Bid,latlon,Sid,Lid,SRS,minx,miny,maxx,maxy) VALUES(0,true," + sId +
",";
            string B_dSql = "DELETE * FROM Tbl_SRSBox WHERE sid=" + sId + " AND lid=";
            try
            {
                cmd.CommandText = "Select MAX(Lid) FROM tbl_layers WHERE (sid=" + sId +");";
                supLid = long.Parse(cmd.ExecuteScalar().ToString())+1;
                for (i=0;(i<cll.Count && (res>-1));i++)
                {//Preparo la lista dei layer online per essere aggiornata
                    k=cll.BinarySearch(oll[i]);
                    if(k>=0)//esiste
                    {
                        if(!cll[k].IsLike(oll[i]))
                        {
                            oll[i].lId = cll[k].lId;//Va aggiornato
                            l=oll[i];//puntatore dicomodo
                            cmd.CommandText = L_uSql + l.Title + "', Name='" + l.Name + "', "
                                + "Keywords='" + l.FlatKeywords + "', MetadataURL='" + l.FlatMetadataURL + "', "
                                + "SRS='" + l.FlatSRS + "', FeatureListURL='" + l.FeatureListURL + "', GeometryType=" +
(int) l.GeometryType
                                + L_uSqlw + l.lId + ";";//WHERE
                            cmd.ExecuteNonQuery();
                            //Elimino il bounding box del layer.
                            cmd.CommandText = B_dSql + l.lId + ";";
                            cmd.ExecuteNonQuery();
                            //metto quello nuovo.
                            cmd.CommandText=B_iSql //INSERT della tabella Tbl_SRSBox
                                + l.lId + ",'" + l.SRS[0].ToString() + "','" + l.LatLongBoundingBox.minx + "','"
                                + l.LatLongBoundingBox.miny + "','" + l.LatLongBoundingBox.maxx + "','"
                                + l.LatLongBoundingBox.maxy + "');";
                            cmd.ExecuteNonQuery();
                            res = 2;
                        }
                        cll[k].Name="__XNOTDELX__";//marco gli elementi uguali in modo che non vengano eliminati dal
catalogo.
                    }
                    else
                    {//devo aggiungerlo
                        oll[i].lId = supLid;
                        supLid++;
                        l = oll[i];//puntatore di comodo.
                        cmd.CommandText = L_iSql //INSERT della tabella Tbl_layers
                            + l.lId + "," + sId + ",'" + l.Name + "','" + l.Title + "','"
                            + l.FlatKeywords + "','" + l.FlatMetadataURL + "','" + l.FeatureListURL + "','" + l.FlatSRS +
"','" + (int) l.GeometryType + ");";
                        cmd.ExecuteNonQuery();
                        cmd.CommandText=B_iSql //INSERT della tabella Tbl_SRSBox
                            + l.lId + ",'" + l.SRS[0].ToString() + "','" + l.LatLongBoundingBox.minx + "','"
                            + l.LatLongBoundingBox.miny + "','" + l.LatLongBoundingBox.maxx + "','"
                            + l.LatLongBoundingBox.maxx + "');";
                        cmd.ExecuteNonQuery();
                        res = 2;
                    }
                    l=null;
                }//Fine preparazione
                for(i=0;i<cll.Count;i++)//elimino i vecchi layer
                {
                    if("__XNOTDELX__" != cll[i].Name)
                    {
                        cmd.CommandText = "DELETE * FROM Tbl_Layers WHERE lid=" + cll[i].lId;
                        cmd.ExecuteNonQuery();
                    }
                }
```

```
        }//try
        catch(Exception e){res = -1;    msg += ((char)13) + "CoreMGR(11): " + e.ToString();}
        cmd = null; return res;
    }

    /// <summary>
    /// Costrisce la rappresentazione di un servizio WFS basandosi sulle informazioni contenute
    /// nel catalogo.
    /// </summary>
    /// <param name="URL">URL dell'istanza del servizio.</param>
    /// <param name="serviceType">Tipologia dell'istanza del servizio :WFS.</param>
    /// <param name="msg">stringa passata per riferimento. Contiene eventuali messaggi di errore.</param>
    /// <returns>"null" se si è verificato un errore durante la ricerca.</returns>
    /// <returns>rappresentazione del'istanza del servizio WFS.</returns>
    private WFS_ServiceClass GetWFS_ServiceFromCatalog(string URL, string serviceType, ref string msg)
    {
        string sqlf = "SELECT * FROM";
        string sqlw = "WHERE URL='" + URL + "' AND ServiceType='" + serviceType + "';";
        try
        {
            //Creo una vista dei dati relativi al servizio
            DataSet ds = new DataSet("ServizioLocale");
            OleDbDataAdapter da = new OleDbDataAdapter(sqlf + " Tbl_OGCServices " +sqlw,this.Conn);
            da.Fill(ds, "Tbl_OGCServices");
            DataRow r   = ds.Tables["Tbl_OGCServices"].Rows[0];
            long sId = long.Parse(r["sid"].ToString());
            da = new OleDbDataAdapter(sqlf + " Tbl_Layers WHERE sid=" + sId + ";"  ,this.Conn);
            da.Fill(ds, "Tbl_Layers");
            da = new OleDbDataAdapter(sqlf + " Tbl_SRSBox WHERE sid=" + sId + ";"  ,this.Conn);
            da.Fill(ds, "Tbl_SRSBox");
            da.Dispose(); da = null;
            //Inzio popolazione servizio
            OWSServiceCrawler.WFS_ServiceClass s = new WFS_ServiceClass();
            s.URL = URL;s.IstanceType = serviceType;
            s.sId = sId;s.GlobalOperations = null;
            s.Version = ""; //<-- in futuro
            s.ServiceInfo = new WFS_ServiceInfosClass();
            s.ServiceInfo.Name = r["name"].ToString();s.ServiceInfo.Title = r["Title"].ToString();
            s.ServiceInfo.Keywords = r["Keywords"].ToString().Split(';');//<! vedere come si comporta con l'ultimo
            //Creazione lista Layer
            DataRowCollection dtL = ds.Tables["Tbl_Layers"].Rows;//Collezione di layer da scandire
            DataTable dtS = ds.Tables["Tbl_SRSBox"];
            WFS_LayerClass l = null;
            s.Layers = new WFS_LayerClassArrayList(dtL.Count);
            int i,j;
            string[] metadata = null;
            for (i=0;i<dtL.Count;i++)
            {
                l = new WFS_LayerClass();
                l.lId = long.Parse( dtL[i]["lid"].ToString());
                l.Name = dtL[i]["Name"].ToString();
                l.Title = dtL[i]["Title"].ToString();
                string c = dtL[i]["Keywords"].ToString().Trim();
                string[] v= null;v=c.Split(';');
                if (null!=v && c!="")
                    l.Keywords = v;
                c = dtL[i]["SRS"].ToString();v=c.Split(';');
                if(null!=v && c!="")
                    l.SRS = v;
                c = dtL[i]["MetadataURL"].ToString().Trim();
                if (null!=c && c!="")//creazione metadata
                {
                    v = c.Split(';');
                    l.MetadataURL = new MetadataURLArrayList(v.Length);
                    string u="",t="";
                    for(j=0;j<metadata.Length;j++)
                    {//in metadata la coppia è URL:type
                        u=(v[j].Split(':'))[0]; t=(v[j].Split(':'))[1];
                        l.MetadataURL.Add(new MetadataURLClass(u,t));
                    }
                }
                else l.MetadataURL = null;
                //Fine creazione metadata
                //recupero latlon
                DataRow ll = dtS.Select("lid=" + l.lId)[0];//seleziono i BBOx che hanno lo Id uguale al layer
                l.LatLongBoundingBox = new MBRClass(double.Parse(ll["minx"].ToString()),
                                                    double.Parse(ll["miny"].ToString()),
                                                    double.Parse(ll["maxx"].ToString()),
                                                    double.Parse(ll["maxy"].ToString())
                                                    );//Me ne aspetto uno solo in qaunto è un servizio WFS
                ll = null;s.Layers.Add(l);
            }//Fine FOR di creazione layer.
            s.Layers.Sort();return s;
        }
        catch(Exception e){msg = e.ToString();return null;}
    }
    /// <summary>
    /// Aggiorna le informazioni relative al servizio.
    /// </summary>
    /// <param name="si">Informazioni del servizio, oggetto ServiceInfosClass.</param>
    /// <param name="sId">Identificativo del servizio nel catalogo.</param>
    /// <param name="msg">stringa passata per riferimento. Contiene eventuali messaggi di errore.</param>
    /// <returns>
    /// -1:Errore in inserimento nel catalogo.
    /// 1:Inserimento effettuato correttamente.
    /// </returns>
    private int UpdateSrvcInf(ServiceInfosClass si,long sId, ref string msg)
    {
        int res = 0;
        try
        {
            OleDbCommand cmd = this.Conn.CreateCommand();
            cmd.Transaction = this.myTrans;
            int i=0;
            string sql = "UPDATE tbl_OGCServices SET Name='";
            string sqlw = " WHERE (sid=" + sId +");";
```

```
                cmd.CommandText = sql + si.Name + "', Title='" + si.Title +
                                  "', Keywords='" + si.FlatKeywords + "'" + sqlw;
                i=cmd.ExecuteNonQuery();
                res = 1;//E' stato fatto l'update?cmd = null;
            }
            catch(Exception e)  {res = -1;  msg += (char)13 + "CoreMGR(12): " + e.ToString();}
            return res;
        }
#endregion

#region Add Service To Catalog
        /// <summary>
        /// Aggiunge alcune informazioniriguardanti l'istanza di un servizio WFS.
        /// </summary>
        /// <param name="srvc">Rappresentazione dell'istanza del servizio WFS.</param>
        /// <param name="msg">stringa passata per riferimento. Contiene eventuali messaggi di errore.</param>
        /// <returns>"true" se l'inserimento è stato effettuato.</returns>
        private bool AddWFS_Service(OWSServiceCrawler.WFS_ServiceClass srvc, ref string msg)
        {
            bool res = false;
            long sid = -1;

            OleDbCommand cmd = null;
            this.Conn = new OleDbConnection(this.strConnectionString);
            try
            {
                this.Conn.Open();
                cmd = this.Conn.CreateCommand();
                cmd.CommandText = "SELECT Count(Sid) FROM Tbl_OGCServices WHERE URL='" + srvc.URL + "' AND ServiceType='"
+ srvc.IstanceType + "';";
                sid = long.Parse(cmd.ExecuteScalar().ToString());
                if (0==sid)//Non esiste
                {
                    srvc.GetGeometryType(true,ref msg); sid = -1;
                    this.myTrans = this.Conn.BeginTransaction(IsolationLevel.ReadCommitted);
                    sid = this.DumpServiceInformations(srvc.ServiceInfo,srvc.URL,srvc.IstanceType, ref msg);
                    if (-1!=sid)
                    {
                        int j=0,i = 0;
                        WFS_LayerClass l = null;
                        cmd = this.Conn.CreateCommand();cmd.Transaction = this.myTrans;
                        for (i=0;i<srvc.Layers.Count;i++)
                        {
                            l = srvc.Layers[i];
                            cmd.CommandText=@"INSERT INTO Tbl_Layers
(Lid,Sid,Name,Title,Keywords,MetadataURL,FeatureListURL,SRS,GeometryType) VALUES("
                                + i + "," + sid + ",'" + l.Name + "','" + l.Title + "','"
                                + l.FlatKeywords + "','" + l.FlatMetadataURL + "','" + l.FeatureListURL + "','" +
l.FlatSRS + "'," + (int) l.GeometryType + ");";
                            cmd.ExecuteNonQuery();//Lo SRS per <FeatureType> è unico, non come per i <Layer> di un WMS.
                            cmd.CommandText=@"INSERT INTO Tbl_SRSBox (Bid,Lid,Sid,SRS,minx,miny,maxx,maxy,latlon)
VALUES("
                                + j + "," + i + "," + sid + ",'" + l.SRS[0].ToString() + "','" +
l.LatLongBoundingBox.minx + "','"
                                + l.LatLongBoundingBox.miny + "','" + l.LatLongBoundingBox.maxx + "','"
                                + l.LatLongBoundingBox.maxy + "',true);";
                            cmd.ExecuteNonQuery();
                        }
                        this.myTrans.Commit();res = true;
                    }
                    else
                    {this.myTrans.Rollback();res = false;}
                }//Stai davvero leggendo questo codice? Oppure ti ci è capitato l'occhio casualmente? ^_^
                else {res = false;msg += (char)13 + "The selected service is already present in the Catalog.";}
            }
            catch(Exception e)
            {
                msg += (char)13 + "CoreMGR(14): " + e.ToString();
                if(null!=this.Conn) if (this.Conn.State == ConnectionState.Open) myTrans.Rollback();
            }
            if (null!=this.Conn){this.Conn.Close();this.Conn.Dispose();this.Conn = null;}
            this.myTrans = null;return res;
        }

        /// <summary>
        /// Aggiunge alcune informazioniriguardanti l'istanza di un servizio WMS.
        /// </summary>
        /// <param name="srvc">Rappresentazione dell'istanza del servizio WMS.</param>
        /// <param name="msg">stringa passata per riferimento. Contiene eventuali messaggi di errore.</param>
        /// <returns>"true" se l'inserimento è stato effettuato.</returns>
        private bool AddWMS_Service(OWSServiceCrawler.WMS_ServiceClass srvc, ref string msg)
        {
            bool res = true;
            long sId = -1;long exs = 0;
            this.Conn = new OleDbConnection(this.strConnectionString);
            try
            {
                this.Conn.Open();
                OleDbCommand cmd = null;
                cmd = this.Conn.CreateCommand();
                cmd.CommandText = "SELECT Count(sId) FROM Tbl_OGCServices WHERE URL='" + srvc.URL + "' AND ServiceType='"
+ srvc.IstanceType + "';";
                exs = long.Parse(cmd.ExecuteScalar().ToString());
                cmd = null;
                if (0==exs)//Non esiste
                {
                    this.myTrans = this.Conn.BeginTransaction(IsolationLevel.ReadCommitted);
                    sId = this.DumpServiceInformations(srvc.ServiceInfo,srvc.URL,srvc.IstanceType, ref msg);
                    long lId = 0;
                    if (-1!=sId)
                    {
                        int i=0;
                        WMS_LayerClassArrayList ll = new WMS_LayerClassArrayList();
                        for(i=0;(i<srvc.Layers.LayersHierarchy.Count && res);i++)
                            res = WMS_Stuffs.UnfoldHierarchy(srvc.Layers.LayersHierarchy[i],null,null,null,ll);
                        for (i=0;(i<ll.Count && res);i++)
```

```
                                res = DumpWMSLayer(ll[i],sId,lId+i, ref msg);
                            }
                            else
                                res = false;
                        }
                        else{res = false;msg += (char)13 + "The selected service is already present in the Catalog.";}
                    }
                    catch(Exception e){res = false;msg += (char)13 + "CoreMGR(16): " + e.ToString();}
                    finally
                    {
                        if(null!=this.Conn)
                            if (this.Conn.State == ConnectionState.Open)
                            {
                                if (0==exs)//exs utilizzato per fare commit o reollback sse nn esiste il servizio
                                    if(res)
                                        this.myTrans.Commit();
                                    else
                                        this.myTrans.Rollback();
                                this.Conn.Close();  this.Conn.Dispose();this.Conn = null;
                            }
                    }
                    this.myTrans = null;return res;
                }

                /// <summary>
                /// Memorizza le informazioni relative ad un layer di un'istanza di servizio WMS.
                /// </summary>
                /// <param name="l">Rappresentazione del layer WMS.</param>
                /// <param name="sId">Identificatore del servizio nel catalogo.</param>
                /// <param name="lId">Identificatore del layer nel catalogo, rispetto al servizio.</param>
                /// <param name="msg">stringa passata per riferimento. Contiene eventuali messaggi di errore.</param>
                /// <returns>"true" se l'inserimento è stato effettuato</returns>
                private bool DumpWMSLayer(WMS_LayerClass l,long sId,long lId,ref string msg)
                {
                    OleDbCommand cmd = null;
                    OleDbDataAdapter da = null;
                    int i = 0;
                    bool res = false;
                    try
                    {
                        cmd = this.Conn.CreateCommand();cmd.Transaction = this.myTrans;
                        da = new OleDbDataAdapter("SELECT * FROM Tbl_Layers,Tbl_SRSBox WHERE Tbl_Layers.Lid=-1 AND
Tbl_SRSBox.Bid=-1",myTrans.Connection);

                        cmd.CommandText=@"INSERT INTO Tbl_Layers
(Lid,Sid,Name,Title,Keywords,MetadataURL,FeatureListURL,SRS,PreviewURL,GeometryType) VALUES("
                            + lId + "," + sId + ",'" + l.Name + "','" + l.Title + "','"
                            + l.FlatKeywords + "','" + l.FlatMetadataURL + "','" + l.FeatureListURL + "','" + l.FlatSRS + "','" +
((null==l.LegendURL)?"":l.LegendURL) + "',33);";
                        cmd.ExecuteNonQuery();//Lo SRS per <FeatureType> è unico, non come per i <Layer> di un WMS.
                        cmd.CommandText=@"INSERT INTO Tbl_SRSBox (Bid,Lid,Sid,SRS,minx,miny,maxx,maxy,latlon) VALUES("
                            + "0," + lId + "," + sId + ",'EPSG:4326','" + l.LatLongBoundingBox.minx + "','"
                            + l.LatLongBoundingBox.miny + "','" + l.LatLongBoundingBox.maxx + "','"
                            + l.LatLongBoundingBox.maxy + "',true);";//EPSG:4326 è di default per WMS
                        cmd.ExecuteNonQuery();
                        #region Inserimento Lista BoundingBox nella tabella Tbl_SRSBox
                        for(i=0;(null!=l.BBOX && i<l.BBOX.Count);i++)
                        {
                            cmd.CommandText=@"INSERT INTO Tbl_SRSBox (Bid,Lid,Sid,SRS,minx,miny,maxx,maxy,latlon) VALUES("
                                + (i+1) + "," + lId + "," + sId + ",'" + l.BBOX.SRS(i) + "','" + l.BBOX.MBR(i).minx + "','"
                                + l.BBOX.MBR(i).miny + "','" + l.BBOX.MBR(i).maxx + "','"
                                + l.BBOX.MBR(i).maxx + "',false);";
                            cmd.ExecuteNonQuery();
                        }
                        #endregion
                        res = true;
                    }
                    catch(Exception e){res = false;msg += (char)13 + "CoreMGR(17): " + e.ToString();}
                    cmd.Dispose();da.Dispose();
                    return res;//FAQ: Facimm'A' Queri!!!
                }

                /// <summary>
                /// Memorizza nel catalogo le informazioni generali relative all'istanza del servizio.
                /// </summary>
                /// <param name="srvc">Informazioni generali relative all'istanza del servizio. </param>
                /// <param name="URL">ULR dell'istanza di servizio.</param>
                /// <param name="IstanceType">Tipo di istanza (WMS/WFS).</param>
                /// <param name="msg">stringa passata per riferimento. Contiene eventuali messaggi di errore.</param>
                /// <returns>Ritorna lo id del servizio inserito nel catalogo, altrimenti "-1"</returns>
                /// <remarks>Lo id restituito è la chiave primaria (sId) della tabella Tbl_OGCService.</remarks>
                private long DumpServiceInformations(ServiceInfosClass srvc, string URL, string IstanceType, ref string msg)
                {
                    long t = -1;OleDbCommand cmd = null;
                    try
                    {
                        //da = new OleDbDataAdapter("SELECT * FROM Tbl_OGCServices WHERE Sid=-1", myTrans.Connection);
                        cmd = this.Conn.CreateCommand();
                        cmd.Transaction =this.myTrans;
                        cmd.CommandText = @"INSERT INTO Tbl_OGCServices (URL,ServiceType,Name,Title,Keywords) VALUES('"
                            + URL + "','" + IstanceType + "','" + srvc.Name + "','" + srvc.Title + "','"
                            + srvc.FlatKeywords + "')";
                        cmd.ExecuteNonQuery();
                        cmd.CommandText = @"SELECT Sid FROM Tbl_OGCServices WHERE (URL='" + URL + "' AND ServiceType='" +
IstanceType + "');";
                        t = long.Parse( cmd.ExecuteScalar().ToString());
                    }
                    catch(Exception e)  {t = -1;msg += (char)13 + "CoreMGR(18): " + e.ToString();}
                    cmd.Dispose(); cmd = null;  return t;
                }
#endregion
            }
        }
```

```
using System;
using System.Data;
using System.Data.OleDb;
using System.Collections;
//**********************************
using OWSServiceCrawler;
using OWSServiceCrawler.TypedArrayList;

namespace CoreCatalogueInterface
{
    /// <summary>
    /// Fornisce un metodo statico pe gestire le gerarchie di layer WMS
    /// </summary>
    public class WMS_Stuffs
    {
        /// <summary>
        /// Costruttore.
        /// </summary>
        public WMS_Stuffs(){}

        /// <summary>
        /// Srotola una gerarchia che rappresenta una "Category" on un "Layer Map" di un WMS
        /// in una lista di layer WMS dove vengono replicati gli attributo ereditati.
        /// </summary>
        /// <param name="nodo">Gerarchia</param>
        /// <param name="parentLatLon">LatLonBoundingBox erditato dal nodo padre.</param>
        /// <param name="parentBBOXlist">BoundingBox erditato dal nodo padre.</param>
        /// <param name="parentSRS">Lista di SRS erditata dal nodo padre.</param>
        /// <param name="ll">Lista di Layers</param>
        /// <returns>"true"</returns>//test
        /// <remarks>
        /// Metodo ricorsivo, la chiamata si effettua passando "null" ai tre parametri relativi a
        /// LatlonBoundingbox, BoundingBox e SRS.
        /// </remarks>
        internal static bool UnfoldHierarchy(WMS_LayerHierarchyClass nodo, MBRClass parentLatLon, BBOXArrayList
parentBBOXlist, string[] parentSRS, WMS_LayerClassArrayList ll)
        {
            int i = 0;
            MBRClass currLatLon = null; BBOXArrayList currBBOXlist = null;
            string[] currSRS = null;
            bool res = true;
            //Gestione eredità LatLonBoundingBox (SOSTITUTIVA)
            if(null==nodo.LatLongBoundingBox)//Non ha LatLon...
                currLatLon = parentLatLon;
            else
                currLatLon = nodo.LatLongBoundingBox;
            //Gestione eredità BoundingBox (SOSTITUTIVA)
            if(null==nodo.BBOX)//Non ha BoundingBox
                currBBOXlist = parentBBOXlist;
            else//cerco quelli diversi
                currBBOXlist = BuildCurrentBBOX(nodo.BBOX,parentBBOXlist);
            //Gestione eredità SRS (ADDITIVA)
            if(null==nodo.SRS)
                currSRS = parentSRS;
            else
                currSRS = BuildCurrentSRS(nodo.SRS, parentSRS);
            if(null!=nodo.ChildHierarchy)//E' una gerarchia
            {
                for (i=0;(i<nodo.ChildHierarchy.Count && res);i++)
                    res = UnfoldHierarchy(nodo.ChildHierarchy[i],currLatLon,currBBOXlist,currSRS,ll);
            }
            if(null!=nodo.ChildLayers && res)
            {
                for(i=0;(i<nodo.ChildLayers.Count);i++)
                {
                    ll.Add(AddLayer(nodo.ChildLayers[i], currLatLon,currBBOXlist,currSRS));
                }
            }
            return res;
        }

        /// <summary>
        /// Completa un layer WMS con gli attributi ereditati.
        /// </summary>
        /// <param name="l">Layer</param>
        /// <param name="parentLatLon">LatLonBoundingBox erditato dal nodo padre.</param>
        /// <param name="parentBBOXlist">BoundingBox erditato dal nodo padre.</param>
        /// <param name="parentSRS">Lista di SRS erditata dal nodo padre.</param>
        /// <returns>Il layer WMS completato, altrimenti "null"</returns>
        internal static WMS_LayerClass AddLayer(WMS_LayerClass l, MBRClass parentLatLon, BBOXArrayList parentBBOXlist,
string[] parentSRS)
        {
            MBRClass currLatLon = null; BBOXArrayList currBBOXlist = null;
            string[] currSRS = null;
            try
            {
                //Gestione eredità LatLonBoundingBox (SOSTITUTIVA)
                if(null==l.LatLongBoundingBox)//Non ha LatLon...
                    currLatLon = parentLatLon;
                else
                    currLatLon = l.LatLongBoundingBox;
                if(null==l.BBOX)//Non ha BoundingBox
                    currBBOXlist = parentBBOXlist;
                else//cerco quelli diversi
                    currBBOXlist = BuildCurrentBBOX(l.BBOX,parentBBOXlist);
                //Gestione eredità SRS (ADDITIVA)
                if(null==l.SRS)
                    currSRS = parentSRS;
                else
                    currSRS = BuildCurrentSRS(l.SRS,parentSRS);

                WMS_LayerClass tl = new WMS_LayerClass();
                tl.Name = l.Name; tl.Title = l.Title;
                tl.Abstract = tl.Abstract; tl.Keywords = l.Keywords;
                tl.queryable = l.queryable; tl.ScaleHint = tl.ScaleHint;
                tl.Style = l.Style; tl.LegendURL = l.LegendURL;
                tl.FeatureListURL = l.FeatureListURL;
                tl.SRS = currSRS;
```

```
                    tl.BBOX = currBBOXlist;
                    tl.LatLongBoundingBox = currLatLon;
                    return tl;
                }
                catch
                {
                    return null;
                }
            }

            /// <summary>
            /// Crea la lista di SRS in base alla lista ereditata e corrente.
            /// </summary>
            /// <param name="curr">Lista SRS del nodo.</param>
            /// <param name="par">Lista SRS del padre (reditata).</param>
            /// <returns>Lista SRS che tiene conto dell'eredità. </returns>
            /// <remarks>SRS subisce ereditarietà di tipo "additiva".</remarks>
            internal static string[] BuildCurrentSRS(string[] curr, string[] par)
            {
                if(null==par && null==curr) return null;
                if(null==par) return curr;
                if(null==curr) return par;
                string[] t = null;
                string[] max, min;
                int i;

                if(curr.Length>par.Length)
                {max = curr; min = par;}
                else
                {max = par; min = curr;}
                ArrayList u = new ArrayList(curr.Length+par.Length);
                for (i=0;i< min.Length;i++) u.Add(min[i]);
                for (i=0;i< max.Length;i++)
                    if (!u.Contains(max[i])) u.Add(max[i]);
                t = new string[u.Count];
                for(i=0;i<u.Count;i++) t[i]=u[i].ToString();
                return t;
            }

            /// <summary>
            /// Crea la lista di MBRClass in base alla lista ereditata e corrente.
            /// </summary>
            /// <param name="currBBOXl">Lista di MBR del layer</param>
            /// <param name="parentBBOXl">Lista di MBR ereditato</param>
            /// <returns>Lista di MBR che tiene conto dell'eredità</returns>
            /// <remarks>Sia LatLonBoundingBox che BoundingBox subiscono
            /// ereditarietà di tipo "sostitutiva".</remarks>
            internal static BBOXArrayList BuildCurrentBBOX(BBOXArrayList currBBOXl, BBOXArrayList parentBBOXl)
            {

                BBOXArrayList max = null,min = null;
                int i;
                //Prendo la lista + grande
                if(null==currBBOXl && null==parentBBOXl) return null;
                if(null==currBBOXl) return parentBBOXl;
                if(null==parentBBOXl) return currBBOXl;
                BBOXArrayList currBBOXlist = new BBOXArrayList(currBBOXl.Count + parentBBOXl.Count);
                if(currBBOXl.Count > parentBBOXl.Count)
                {max=currBBOXl;min=parentBBOXl;}
                else
                {min=currBBOXl;max=parentBBOXl;}
                //Utilizzo il max per vedere se ha elementi in min
                for (i=0;i< max.Count;i++)
                    if(null == min.MBR(max.SRS(i)))//se max non ha l'elemento in min
                        currBBOXlist.Add(max.SRS(i),max.MBR(i));//aggiungi elementi di max.
                    else
                        currBBOXlist.Add(min.SRS(i),min.MBR(i));//aggiungi l'elemento di min.
                min = max = null;
                return currBBOXlist;
            }
        }
    }
```

```
using System;
using System.Collections;
using OWSServiceCrawler.TypedArrayList;

namespace CoreCatalogueInterface
{
    /// <summary>
    /// Descrizione di riepilogo per Util.
    /// </summary>
    internal class Util
    {
        /// <summary>
        /// Costruttore predefinito.
        /// </summary>
        public Util(){}

        internal static string[] BuildCurrentSRS(string[] curr, string[] par)
        {
            if(null==par && null==curr) return null;
            if(null==par) return curr;
            if(null==curr) return par;
            string[] t = null;
            string[] max, min;
            int i;

            if(curr.Length>par.Length)
            {max = curr; min = par;}
            else
            {max = par; min = curr;}
            ArrayList u = new ArrayList(curr.Length+par.Length);
            for (i=0;i< min.Length;i++) u.Add(min[i]);
            for (i=0;i< max.Length;i++)
                if (!u.Contains(max[i])) u.Add(max[i]);
            t = new string[u.Count];
            for(i=0;i<u.Count;i++) t[i]=u[i].ToString();
            return t;
        }

        internal static BBOXArrayList BuildCurrentBBOX(BBOXArrayList currBBOXl, BBOXArrayList parentBBOXl)
        {

            BBOXArrayList max = null,min = null;
            int i;
            //Prendo la lista + grande
            if(null==currBBOXl && null==parentBBOXl) return null;
            if(null==currBBOXl) return parentBBOXl;
            if(null==parentBBOXl) return currBBOXl;
            BBOXArrayList currBBOXlist = new BBOXArrayList(currBBOXl.Count + parentBBOXl.Count);
            if(currBBOXl.Count > parentBBOXl.Count)
            {max=currBBOXl;min=parentBBOXl;}
            else
            {min=currBBOXl;max=parentBBOXl;}
            //Utilizzo il max per vedere se ha elementi in min
            for (i=0;i< max.Count;i++)
                if(null == min.MBR(max.SRS(i)))//se max non ha l'elemento in min
                    currBBOXlist.Add(max.SRS(i),max.MBR(i));//aggiungi elementi di max.
                else
                    currBBOXlist.Add(min.SRS(i),min.MBR(i));//aggiungi l'elemento di min.
            min = max = null;
            return currBBOXlist;
        }
    }
}
```

F.7. Codice Namespace LocalCatalogueService

```
using System;
using System.Collections;
using System.ComponentModel;
using System.Data;
using System.Diagnostics;
using System.Web;
using System.Web.Services;
using System.Xml;
//*************************
using CoreCatalogInterface;
using OWSServiceCrawler;

///<Author>Giulio Massei</Author>
///<LastUpdate>12/08/2005</LastUpdate>

namespace LocalCatalogService
{
/// <summary>
/// Descrizione di riepilogo per Service1.
/// </summary>

[WebService(Namespace="http://webgisserver.isti.cnr.it/LocalCatalogueService/")]
public class LocalCatalogService : System.Web.Services.WebService
   {
    private string pCatalogPath = "";

    private CoreCatalogInterface.CoreManager cm = null;

    private OWSServiceCrawler.WMS_ServiceClass sm = null;
    private OWSServiceCrawler.WFS_ServiceClass sf = null;
    private OWSServiceCrawler.ServiceClass s = null;

    private XmlDocument xd = null;

    /// <summary>
    /// Costruttore predefinito
    /// </summary>
    public LocalCatalogService()
        {InitializeComponent();this.pCatalogPath = Server.MapPath(@"LocalCatalog\local.mdb");}

    /// <summary>
    /// Permette di inserire una nuova istanza di servizo nel catalogo.
    /// </summary>
    /// <param name="URL">URL del servizio.</param>
    /// <param name="istanceType">Tipo di distanza; WFS o WMS.</param>
    /// <param name="msg">Parametro di uscita che riporta il messaggio di errore nel caso vi sia un fallimento.</param>
    /// <returns>True se l'inserimento ha avuto esito positivo; false altrimenti ed il parametro °msg° viene
inizializzato</returns>
    [WebMethod(Description="Lets to insert a new istance of service in the catalog.")]
    public bool Insert(string URL,string istanceType, ref string msg)
        {
        bool b = true;
        this.xd = new XmlDocument();
           try
            {
               try
                {
                this.xd.Load(URL + "?request=getcapabilities");
                this.cm = new CoreManager(this.pCatalogPath);
                if(this.cm.CatalogExist)
                    {
                    if("WMS"==istanceType)
                        {this.sm = new WMS_ServiceClass(this.xd,URL);this.s = this.sm;}
                      else
                        {this.sf = new WFS_ServiceClass(this.xd,URL);this.s = this.sf;}
                    b = this.cm.AddService(this.s, ref msg);
                    }
                  else
                    {b = false;msg = "CatalogDriver(1.1): Unable to locate the Catalog.";}
                }
             catch(System.Net.WebException we)
                {msg += ((char)13) + "CatalogDriver(1.2): Unable to contact the service." + we.ToString();b = false;}
            }
        catch(Exception e)
            {msg += ((char)13) + "CatalogDriver(1.3): " + e.ToString();b = false;}
        this.s= null; this.sf =null; this.sm = null;this.xd = null;this.cm = null;
        return b;
        }

    /// <summary>
    /// Permette di eliminare una istanza di servizo dal catalogo specificandone la URL.
    /// </summary>
    /// <param name="URL">URL del servizio</param>
    /// <param name="istanceType">Tipo di distanza; WFS o WMS</param>
    /// <param name="msg">Parametro di uscita che riporta il messaggio di errore nel caso vi sia un fallimento.</param>
    /// <returns>
    /// Valore intero:
    /// -1: Eliminazione fallita a causa di un errore.
    ///  0: Nessuna eliminazione, elemento inesistente.
    ///  1: Eliminazione effettuata.
    /// </returns>
    [WebMethod(Description="Lets to remove an istance of service from the catalog giving its URL.")]
    public int DeleteByURL(string URL,string istanceType, ref string msg)
        {
        this.cm = new CoreManager(this.pCatalogPath);
        if(this.cm.CatalogExist)
                return this.cm.DeleteService(URL,istanceType,ref msg);
          else
            {msg = "CatalogDriver(3): Unable to locate the catalog."; return -1;}
        this.cm = null;
```

```
        }

    /// <summary>
    /// Permette di eliminare una istanza di servizo dal catalogo specificandone l'identificatore all'interno del
catalogo.
    /// </summary>
    /// <param name="sId">Identificatore del servizio all'interno del catalogo.</param>
    /// <param name="URL">URL del servizio</param>
    /// <param name="msg">Parametro di uscita che riporta il messaggio di errore nel caso vi sia un fallimento.</param>
    /// Valore intero:
    /// -1: Eliminazione fallita a causa di un errore.
    ///  0: Nessuna eliminazione, elemento inesistente.
    ///  1: Eliminazione effettuata.
    /// </returns>
    [WebMethod(Description="Lets to remove an istance of service from the catalog giving its identifier in the catalog.")]
    public int DeleteByID(long sId, ref string msg)
        {
        this.cm = new CoreManager(this.pCatalogPath);
        if(this.cm.CatalogExist)
            return this.cm.DeleteService(sId,ref msg);
          else
            {msg = "CatalogDriver(4): Unable to locate the catalog."; return -1;}
        this.cm = null;
        }

    /// <summary>
    /// Aggiorna un servizio in catalogo in base alla rappresentazione della stessa istanza
    /// ottenuta dalla rete.
    /// </summary>
    /// <param name="URL">URL del servizio.</param>
    /// <param name="istanceType">Tipo di distanza; WFS o WMS.</param>
    /// <param name="msg">Parametro di uscita che riporta il messaggio di errore nel caso vi sia un fallimento.</param>
    /// <returns>
    /// Valore intero:
    /// -1: Non è stato effettuato alcun aggiornamento a causa di un errore.
    ///  0: Non è stato effettuato alcun aggiornamento.
    ///  1: Sono state aggiornate solo le informazioni relative al servizio.
    ///  2: Sono state aggiornate solo le informazioni di tutti o di alcuni layer.
    ///  3: Sono state aggiornate sia le informazioni del servizio che di tutti od alcuni layer.
    /// </returns>
    [WebMethod(Description="Let to update, in the catalog, the informations concerning an istance of service giving its
URL.")]
    public int UpdateByURL(string URL,string istanceType, ref string msg)
        {
        int b = 0;
        this.xd = new XmlDocument();
           try
            {
               try
                {
                this.xd.Load(URL);
                this.cm = new CoreManager(this.pCatalogPath);
                if(this.cm.CatalogExist)
                    {
                    if("WMS"==istanceType)
                        {this.sm = new WMS_ServiceClass(this.xd,URL);this.s = this.sm;}
                      else
                        {this.sf = new WFS_ServiceClass(this.xd,URL);this.s = this.sf;}
                    b = cm.UpdateService(this.s, ref msg);
                    }
                  else
                    {msg = "CatalogDriver(5): Unable to locate the catalog.";return -1;}
                }
             catch(System.Net.WebException we)
                 {msg += ((char)13) + "CatalogDriver(6): Unable to contact the service." + we.ToString();b = -1;}
            }
        catch(Exception e)
            {msg += ((char)13) + e.ToString();          b = -1;}
        this.s= null; this.sf =null; this.sm = null;this.xd = null;this.cm = null;
        return b;
        }

    /// <summary>
    /// Aggiorna un servizio in catalogo in base alla rappresentazione della stessa istanza
    /// ottenuta dalla rete.
    /// </summary>
    /// <param name="sId">Identificatore dell'istanza di servizio all'interno del catalogo.</param>
    /// <param name="msg">Parametro di uscita che riporta il messaggio di errore nel caso vi sia un fallimento.</param>
    /// <returns>
    /// Valore intero.
    /// <returns>
    /// Valore intero:
    /// -1: Non è stato effettuato alcun aggiornamento a causa di un errore.
    ///  0: Non è stato effettuato alcun aggiornamento.
    ///  1: Sono state aggiornate solo le informazioni relative al servizio.
    ///  2: Sono state aggiornate solo le informazioni di tutti o di alcuni layer.
    ///  3: Sono state aggiornate sia le informazioni del servizio che di tutti od alcuni layer.
    /// </returns>  /// </returns>
    [WebMethod(Description="Let to update, in the catalog, the informations concerning an istance of service giving its
identifier in the catalog.")]
    public int UpdateByID(long sId, ref string msg)
        {
        this.cm = new CoreManager(this.pCatalogPath);;
        int b = 0;
        string url, itype;
        DataSet ds = this.cm.ExecuteSFW("SELECT URL, ServiceType FROM TBL_OGCServices WHERE sId=" + sId, ref msg);
        if(null != ds)
            {
            url = ds.Tables[0].Rows[0]["URL"].ToString().Trim();
            itype = ds.Tables[0].Rows[0]["ServiceType"].ToString().Trim();
            ds.Dispose(); ds = null;
            }
          else
            {msg = "CatalogDriver(5.1): Service not present in the Catalog.";return -1;}
        this.xd = new XmlDocument();
          try
```

```
            {
               try
                {
                this.xd.Load(url);
                //cm = new CoreManager(this.pCatalogPath);
                if(this.cm.CatalogExist)
                    {
                    if("WMS"==itype)
                        {this.sm = new WMS_ServiceClass(this.xd,url);this.s = this.sm;}
                      else
                        {this.sf = new WFS_ServiceClass(this.xd,url);this.s = this.sf;}
                    b = this.cm.UpdateService(this.s, ref msg);
                    }
                  else
                    {msg = "CatalogDriver(5.2): Unable to locate the Catalog.";return -1;}
                }
             catch(System.Net.WebException we)
                 {msg += ((char)13) + "CatalogDriver(5.3): Unable to contact the service." + we.ToString();b = -1;}
            }
         catch(Exception e)
             {msg += ((char)13) + e.ToString();b = -1;}
         this.s= null; this.sf =null; this.sm = null;this.xd = null;this.cm = null;
         return b;
         }

    [WebMethod(Description="Recupera un dataset contenente il nome, il tipo di istanza e l'identificatore, nel catalogo,
di un istanza di servizio")]
    public DataSet GetServiceList(ref string msg)
        {
         this.cm = new CoreManager(this.pCatalogPath);
         if(cm.CatalogExist)
             return this.cm.ExecuteSFW("SELECT Name,sId,ServiceType, ServiceType + ':' + Name as cog FROM Tbl_OGCServices",
ref msg);
           else
             {msg = "CatalogDriver(7): Unable to locate the catalog.";return null;}
         this.cm = null;
         }

    [WebMethod(Description="Recupera un dataset contenente il nome e l'identificatore, nel catalogo, di un'istanza di
servizio WMS")]
    public DataSet GetWMSServiceList(ref string msg)
        {
         this.cm = new CoreManager(this.pCatalogPath);
         if(this.cm.CatalogExist)
             return this.cm.ExecuteSFW("SELECT Name,sId,ServiceType FROM Tbl_OGCServices WHERE ServiceType='WMS'", ref
msg);
           else
             {msg = "CatalogDriver(8): Unable to locate the catalog."; return null;}
         this.cm = null;
         }

    [WebMethod(Description="Recupera un dataset contenente il nome e l'identificatore, nel catalogo, di un'istanza di
servizio WFS")]
    public DataSet GetWFSServiceList(ref string msg)
        {
         this.cm = new CoreManager(this.pCatalogPath);
         if(cm.CatalogExist)
             return cm.ExecuteSFW("SELECT Name,sId,ServiceType FROM Tbl_OGCServices WHERE ServiceType='WFS'", ref msg);
           else
             {msg = "CatalogDriver(9): Unable to locate the catalog.";return null;}
         this.cm = null;
         }

    [WebMethod(Description="Recupera un dataset contenente il nome e l'identificatore, nel catalogo, di un'istanza di
servizio WMS")]
    public DataSet GetDataFromCatalog(string query, ref string msg)
        {
         this.cm = new CoreManager(this.pCatalogPath);
         if(this.cm.CatalogExist)
             return this.cm.ExecuteSFW(query, ref msg);
           else
             {msg = "CatalogDriver(10): Unable to locate the catalog.";return null;}
         this.cm = null;
         }

    [WebMethod]
    public string Get_a_Value(string strQuery,ref string msg)
        {
         string t = "";
            try
             {
             this.cm = new CoreManager(this.pCatalogPath);
             if(this.cm.CatalogExist)
                 t = this.cm.Get1Value(strQuery,ref msg).Trim();
               else
                 msg = "CatalogDriver(9.1): Unable to locate the Catalog.";
             }
         catch(Exception e)
             {msg += ((char)13) + "CatalogDriver(9.2): " + e.ToString(); t = "";}
         this.cm = null; return t;
         }
    }
}
```

Riferimenti

[1] Quinto programma di azione per le aziende "*Per uno sviluppo durevole e sostenibile*".

[2] Sesto programma di azione per le aziende "*Ambiente 2010: il nostro futuro la nostra scelta*" GU L 242 del 10.9.02.

[3] http://www.ec-gis.org/inspire/principles_it.html.

[4] Jos Van Orshoven, "*Spatial Data Infrastructure in Europe: State of Play Spring 2004*": 18/08/2004.

[5] Jos Van Orshoven, "*Spatial Data Infrastructure in Europe: State of Play Spring 2004*": pag. 13.

[6] Atlante Italiano www.atlanteitaliano.it.

[7] http://www.centrointerregionale.it/laboratorio/italia.html .

[8] Vanden Brouke, Peter Beusen, Alessandro Annoni, "*Spatial Data Infrastructure in Italy: State of Play Spring 2004*": Danny; 07/07/2004;

[9] Rete civica dell'Alto Adige, www.provincia.bz.it .

[10] Portale regione Emilia-Romagna,

http://www.regione.emilia-romagna.it/carto/reper/default.htm .

[11] Centro Interregionale, www.centrointerregionale.itlaboratorio/italia.html.

[12] Rete civica dell'Alto Adige, Ufficio informatica geografica e statistica, http://www.provincia.bz.it/informatica/0906/service_i.asp .

[13] "*SIGMA TER, Servizi Integrati catastali per il Monitoraggio Amministrativo TERritoriale, Σ3*", http://www.sigmater.it/.

[14] Cliff Kottman , "*The OpenGIS™ Abstract Specification Topic 5: Features, Version 4*": 27/03/1999;

[15] *OpenGIS® Geographic Markup Language Implementation Specification*, version 2.1.2; cap5;

[16] Jeff de La Beaujardière, "*Web Map Service Implementation Specification v1.1.1*": Capitolo 7, paragrafo 3 sezione 1, 16/01/2002;

[17] Kris Kolodziej, "*OpenGIS® Web Map* Server *Cookbook*", pcapitolo 1, paragrafo 5 sezione 3: 18/08/2003.

[18] Panagiotis A. Vretanos, "*Filter Encoding Implementation Specification v1.0*": 12/09/2001.

[19] Lou Reich, Web Registry Server Discussion Paper v0.0.2

[20] Josh Lieberman, Lou Reich, Peter Vretanos, "*OGC Web Services UDDI Experiment*": 17/02/2003

[21] Jérôme Sonnett, "*OWS 2 Common Architecture: WSDL SOAP UDDI*" 17/02/2005.

[22] David Booth, Hugo Haas, Francis McCabe, Eric Newcomer, Michael Champion, Chris Ferris, David Orchard, "*Web Services Architecture*, *W3C Working Group Note 11*"-§1.4 February 2004: http://www.w3.org/TR/2004/NOTE-ws-arch-20040211 ;

[23] Hugo Haas, Allen Brown, "*Web Services Glossary*":11/02/2004

[24] Microsoft UDDI Business Registry Node, http://uddi.microsoft.com/default.aspx ;

[25] Electronic Business using eXtensible Markup Language, ebXMl: http://www.ebxml.org;

[26] XML Schema Part 2: Datatypes Second Edition:

http://www.w3.org/TR/xmlschema-2/;

[27] Per una lista completa, http://www.uddi.org/register.html

[28] Microsoft, https://test.uddi.microsoft.com,

IBM, https://www.3-ibm.com/services/uddi/testregistry/protect;

[29] SOAP Client, http://www.soapclient.com/uddisearch.html;

[30] Henrik Frystyk Nielsen, Hervé Ruellan, W3C Working Group Note,

"*SOAP 1.2 Attachment Feature*" 8/06/200

[31] Java Web Service Developer Pack:

http://www.java.sun.com/webservice/jwsdp/index.jsp;

[32] Mono Project: http://www.mono-project.com/Main_Page ;

[33] MSDN Library: http://msdn.microsoft.com/library/;

[34] WMS Italia, http://webgisserver.isti.cnr.it/wms/wmsitalia/request.asp.

[35] Demis World Map, http://www2.demis.nl/mapserver/request.asp.

[36] Intergraph WFS, http://regis.intergraph.com/wfs/dcmetro/request.asp.