Un Sistema Desktop per l'accesso a librerie digitali distribuite

Loredana Versienti[1]

[1] ISTI-CNR, Via G. Moruzzi 1, 56124 Pisa, versienti@isti.cnr.it

Abstract. Questo articolo descrive l'architettura di un componente software che permette la gestione di Digital Library distribuite, mediante l'uso delle tecnologie sviluppate da IBM all'interno del framework Eclipse. Eclipse rappresenta uno standard de facto nella comunità degli sviluppatori Java ed è una piattaforma adatta per sviluppare componenti orientati alle applicazioni in quanto è basato sulle specifiche del modello OSGi. Tali specifiche consentono la scrittura di software modulare ed estendibile, favorendo così un incremento robusto ed affidabile del nostro componente al crescere delle esigenze degli utilizzatori.

Keywords: Eclipse, Digital Library, OSGi, Java RMI

Introduzione

Questo documento descrive le interfacce grafiche e l'architettura software del plugin "BRICKS Desktop Demonstrator" e fornisce inoltre le linee guida per gli sviluppatori di software che desiderano aggiungere nuovi plugin. Il documento è strutturato come segue: il Capitolo 2 presenta una panoramica dello stato dell'arte per quanto riguarda lo sviluppo del Rich Internet Applications e introduce il framework tecnologico dell'applicazione desktop; il Capitolo 3 descrive come installare ed aggiornare il software desktop e come utilizzare le interfacce grafiche desktop per interagire con la "BRICKS Digital Library"; il Capitolo 4 descrive una vista d'insieme dell'architettura con degli snippet delle principali funzionalità.

1. Concetti base

Il BRICKS PILLAR è un'applicazione che utilizza i servizi implementati dai componenti della fondazione BRICKS per permettere agli utenti di accedere e gestire una Digital Library (DL).

La biblioteca digitale BRICKS è distribuita su un insieme di nodi, chiamati BNodi. Un BNodo è una nozione astratta che rappresenta un'installazione di un sottoinsieme (possibilmente tutti) di componenti. I nodi sono interconnessi da una infrastruttura di comunicazione (rete P2P) e trasparentemente cooperano per fornire le funzionalità di una DL.

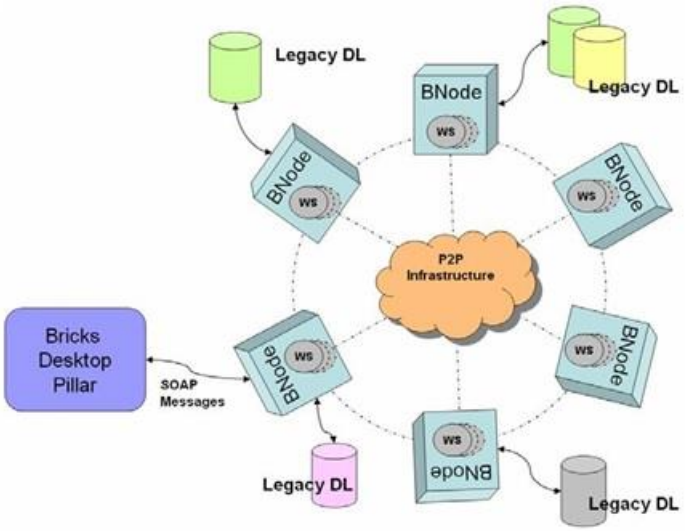


Figura 1: Infrastruttura di rete usata dai Desktop Pillar

BRICKS Desktop fornisce una serie di funzionalità per organizzare, gestire, controllare e riferire il contenuto della biblioteca digitale. Usando una interfaccia grafica semplice, ma potente l'utente ha la possibilità di creare e successivamente aggiornare collezioni di oggetti digitali (contenuti e metadati), navigare e ricercare nello spazio delle DL, importare ed esportare contenuti e metadati. Altre funzionalità personalizzate possono essere facilmente aggiunte come plugin esterni.

2. Stato dell'arte

La nozione di servizio non è una novità nel settore IT, ma l'idea di una Service-oriented architecture SOA [1] è evoluta nel corso degli ultimi due anni. Si tratta di uno stile architettonico per la costruzione di applicazioni software, che promuove accoppiamento lasco tra i componenti, in modo che possano essere reimpiegati in modo facile e naturale. E' un nuovo modo di costruire applicazioni con le seguenti caratteristiche:

- ✓ I servizi sono componenti software in grado di pubblicare funzionalità/interfacce; le funzionalità sono indipendenti dalla piattaforma, dalla lingua e dal sistema operativo.
- ✓ I consumatori possono dinamicamente scoprire i servizi.
- ✓ I servizi sono interoperabili.

SOA fornisce una soluzione chiara a questi problemi di integrazione delle applicazioni, consentendo ai sistemi di esporre la loro funzionalità tramite interfacce di interoperabilità standard. L'utilizzo di SOA offre diversi vantaggi chiave al fine di:

- ✓ adattare le applicazioni alle tecnologie che cambiano.
- ✓ integrare facilmente applicazioni con altri sistemi.
- ✓ creare rapidamente e facilmente un processo di business da servizi esistenti.

La maggior parte degli sviluppatori pensano spesso che i servizi web e SOA sono sinonimi. Molti pensano anche che non è possibile costruire applicazioni orientate ai servizi senza l'utilizzo di Web Services. Per chiarire la SOA è un principio di progettazione, mentre i servizi web sono solo una

tecnologia di implementazione. XML e il Simple Object Access Protocol (SOAP) [2] utilizzato dai servizi Web sono solo tecnologie abilitanti per la SOA.

L' applicazione service-oriented può essere costruita senza l'utilizzo di servizi Web per esempio, utilizzando altre tecnologie tradizionali come Java RMI. Il tema principale dietro SOA è quello di trovare la modularità appropriata e creare delle relazioni flessibili tra i moduli. Possiamo costruire un'applicazione in cui i moduli non hanno una stretta connessione con i componenti con cui interagiscono, come ad esempio un'applicazione in cui il livello di presentazione JSP non è strettamente integrato con il modello di dati e l'accesso ad esso avviene via EJB.

Come tutte le applicazioni distribuite, le applicazioni orientate ai servizi sono applicazioni multi-tier caratterizzate dallo strato di presentazione, dalla logica di business e dalla persistenza. L'architettura del sistema BRICKS segue il paradigma SOA e i componenti base implementano i tre strati chiave come mostrato in Figura 2.

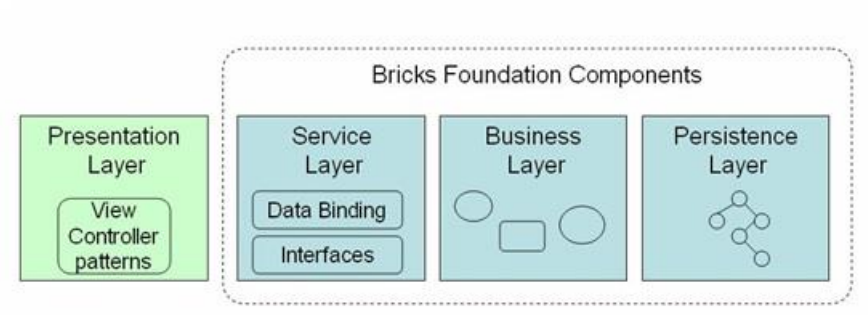


Figura 2. Architettura N-Tier del SOA BRICKS

Lo scopo del Bricks Desktop Demonstrator è di implementare il livello di presentazione della libreria digitale BRICKS. Il livello di presentazione è molto importante dal punto di vista dell'utente. Questo livello può essere costruito con diverse tecnologie client, come JSP, JSF, portlet, standard Java client.

In generale, a seconda di dove si trova il motore di presentazione, ci riferiamo a queste applicazioni come applicazioni Web standard o Rich Internet Applications (RIA) [3]. Le applicazioni web tradizionali centravano tutte le attività intorno all'architettura client-server, dove il client tipicamente era un web browser. Con questo sistema tutti i processi vengono eseguiti sul server e il client viene utilizzato solo per visualizzare il contenuto HTML. Lo svantaggio di questo sistema è che tutte le interazioni con l'applicazione passano dal server: questo richiede che i dati vengono inviati al server e dopo che il server risponde, la pagina viene ricaricata sul client con la risposta. Al contrario, utilizzando una tecnologia lato client in grado di eseguire le istruzioni sul computer del cliente, è possibile ridurre i tempi di attesa. RIA è la tecnologia utilizzata per la nostra applicazine.

Wikipedia definisce Rich Internet Application come le applicazioni Web che hanno le caratteristiche e le funzionalità delle applicazioni desktop tradizionali. L'elenco delle tecnologie abilitanti RIA sono:

- JavaScript (Ajax, DHTML)
- Adobe Flash

- Windows Presentation Foundation and Silverlight
- ActiveX Controls
- JavaFX
- Java applets
- Java applications (Java Web Start, XUI)
- User Interface languages (XUL, SVG)
- Altre tecniche (XForms, XSLT)

Le soluzioni disponibili sono molto più vaste dell'elenco sopra riportato, difatti sono disponibili numerosissimi framework per lo stesso standard.

La distinzione tra i differenti approcci è una questione molto dibattuta, ma possiamo certamente dividerla tra l'approccio "web browser based" e "desktop based". Il primo è fondato sulla convinzione che il browser web sia il futuro; una credenza basata principalmente sull'osservazione che il browser web è visto come lo strumento predominante per accedere a Internet. Il secondo approccio ha lo scopo di dimostrare come le tecnologie si siano evolute nel tempo, ad esempio se prendiamo in considerazione le librerie di Java siamo passati dalla libreria AWT utilizzata per lo sviluppo di applicazioni desktop a Swing una libreria dove sono stati aggiunti nuovi componenti grafici e nuove funzionalità. Quindi abbiamo due tendenze opposte: alcune piattaforme sfruttano l'ubiquità del browser web ma mancano di funzionalità, risultando soluzioni "leggere", altre invece sono ricche di funzionalità ma possono risultare molto più lente.

Il Google Web Toolkit (GWT) appartiene alla prima categoria, è un framework open source, sviluppato da Google, che permette di creare applicazioni Web con AJAX, scrivendo le proprie pagine esclusivamente in linguaggio JAVA. Sarà compito della libreria GWT tradurre il codice Java e produrre le pagine HTML e JavaScript corrispondenti.

Sebbene il Desktop Demostrator può essere distribuito come una applicazione Java Web Start, esso appartiene alla seconda categoria. Abbiamo deciso di sviluppare un'applicazione Desktop ad hoc, in quanto non abbiamo trovato una Rich Internet Application che soddisfacesse le nostre necessità. Gli utenti finali devono essere messi nelle condizioni di poter manipolare gli oggetti digitali come farebbero con le loro applicazioni desktop preferite. Inoltre, l'utente deve essere messo nelle condizioni di aggiungere nuovi tool qualora, per esempio, la richiesta di visualizzare gli oggetti digitali secondo un schema particolare non è coperto dal livello di presentazione delle GUI.

Eclipse è il framework che abbiamo adottato per lo sviluppo desktop. Esso rappresenta uno standard de facto nella comunità degli sviluppatori Java ed è una piattaforma adatta per sviluppare componenti orientati alle applicazioni in quanto è basato sulle specifiche del modello OSGi[4].

2.1 Eclipse framework e le specifiche OSGI

Nel novembre 2001 IBM e sette società hanno lanciato Eclipse [5] come progetto open source. Nel giro di poco tempo il framework Eclipse è diventato lo standard "de facto" nel campo della piattaforma di integrazione con più di 10 milioni di download. Negli ultimi anni il progetto Eclipse è cresciuto notevolmente, il suo successo è dovuto non solo al fatto che gli sviluppatori possono utilizzare un ambiente integrato ma anche espandibile con nuove funzionalità grazie all'aggiunta di plugin. Questo ha dato agli sviluppatori la possibilità di concentrare le loro energie su come

affrontare i componenti specifici, invece di perdere tempo a reinventare una serie di componenti di base. Il quadro Rich Client Platform [6] è il risultato di questo importante sforzo.

Riassumendo l'adozione del framework Eclipse RCP porta l'applicazione Desktop in linea con lo stato dell'arte: orientato allo sviluppo del software a componenti. Il Desktop dal punto di vista dell'integrazione può considerarsi un'applicazione client-server che collega i servizi Web delle BRICKS DL. In ogni caso dobbiamo sottolineare che il desktop può essere distribuito sia come applicazione Java Web Start che come applicazione web.

3. Documentazione per l'utente

3.1 Come Installare il Desktop

L' installazione del Desktop Pillar è semplice. E' disponibile sia per la piattaforma Windows che Linux. Il file scaricato è un archivio standard (.zip per Windows, .tar.gz per Linux) che deve essere estratto in una directory del file-system locale. L'eseguibile contenuto nella directory estratta è direttamente utilizzabile per eseguire l'applicazione e accedere alla libreria digitale BRICKS.

La Figura 3 mostra il contenuto della directory dopo l'estrazione dell'archivio su una piattaforma Windows. Il file BricksDesktop.exe è l'eseguibile che deve essere lanciato al fine di eseguire l'applicazione. Il file Readme.txt contiene le ultime istruzioni e tiene traccia delle modifiche rispetto alla versione precedente. LGPL-License.text e notice.txt sono la licenza del sistema operativo e informazioni sul copyright del prodotto. Il resto di file e directory sono file di configurazione utilizzati dall'applicazione.

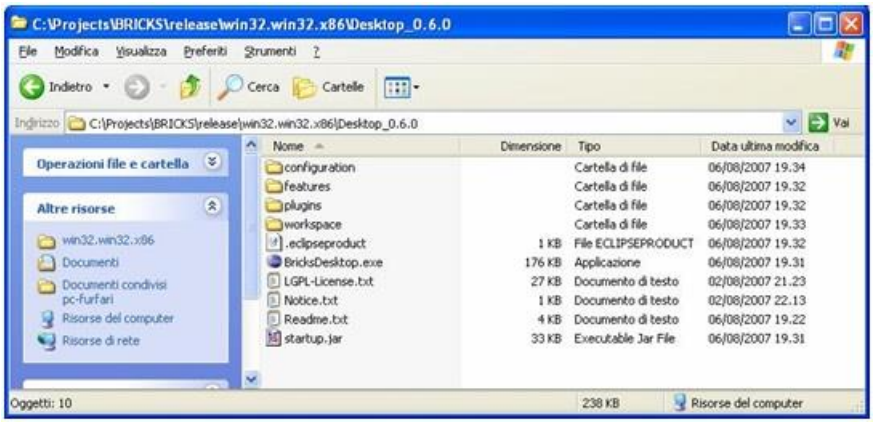


Figura 3: Directory di installazione per il Bricks Desktop

3.2 Aggiornamento e Estensione del desktop

L'installazione desktop iniziale fornisce solo il set minimo di funzionalità per accedere a una DL BRICKS; in sostanza si consente all'utente di sfogliare le collezioni e gli oggetti digitali. Per poter installare altre funzionalità, dopo l'avvio dell'applicazione (vedi "Iniziamo con il Desktop Pillar" paragrafo 3.3), l'utente deve collegarsi al sito di aggiornamento desktop. Esso può essere ottenuto selezionando il sottomenu "Search for extension..." del menu di livello superiore "Help". Una finestra di dialogo utente verrà visualizzata per installare le funzionalità adeguate.

La figura 4 mostra le caratteristiche attualmente disponibili. Le caratteristiche import/export consentono all'utente di popolare la biblioteca digitale Bricks e di salvare le collezioni e gli oggetti digitali sul file-system locale. Le funzione di ricerca aggiungono una serie di GUI per la ricerca di oggetti digitali e collezioni.

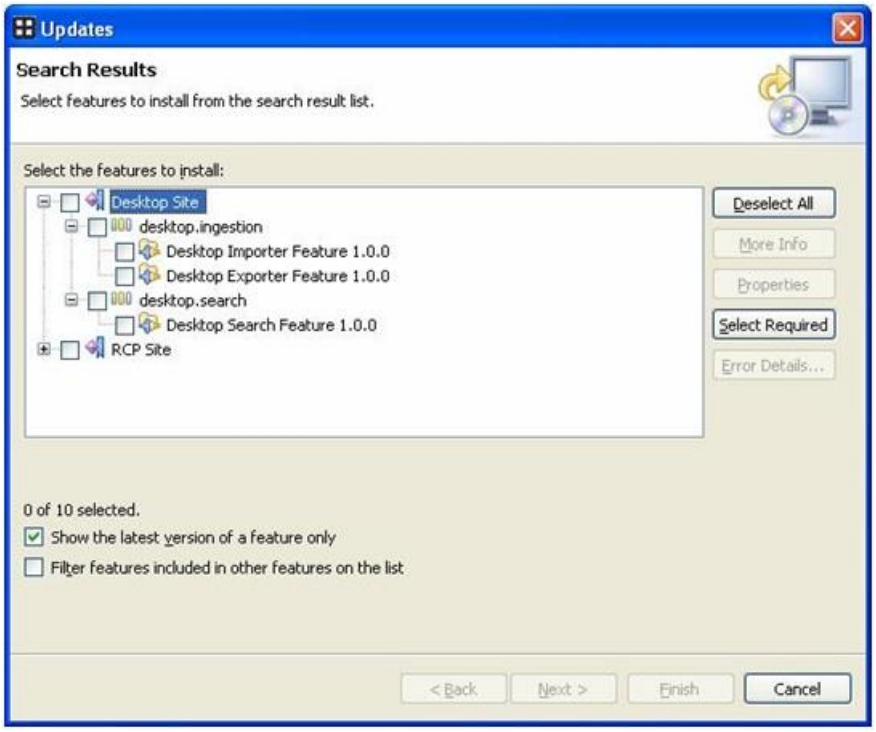


Figura 4: Finestra di dialogo per la "Search for extension..." menu

Il dialogo tra l'utente e il sistema durante il processo di installazione è riportato dalla figura 5 alla figura 9. Cliccando sul pulsante all'utente viene richiesto di accettare i termini del contratto di licenza (Figura 5). Nel passo successivo l'Update Manager Desktop scarica i plugin selezionati dall'utente (Figura 6) e verifica che il software richiesto non abbia bisogno di ulteriori componenti software per poter funzionare (Figura 7). Procedendo con l'installazione i plugin richiesti sono installati (Figura 8) e pronti per essere utilizzati (Figura 9).

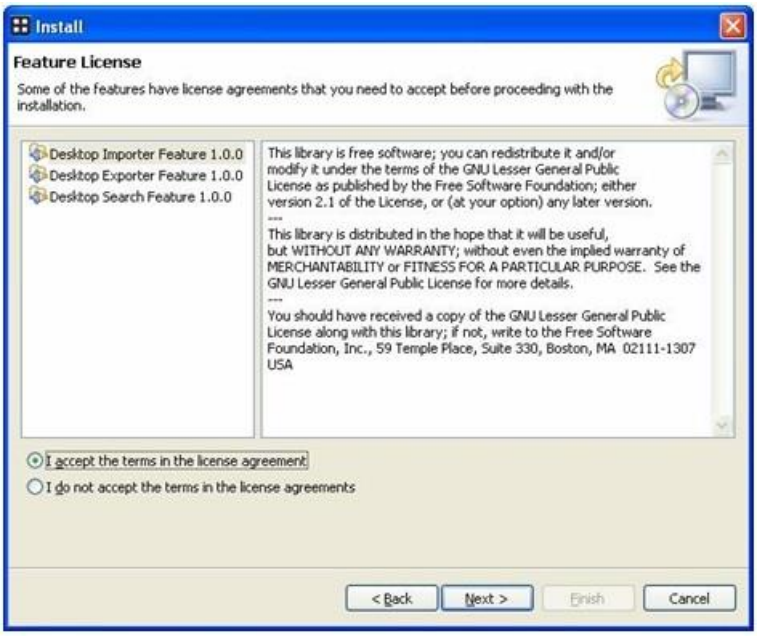


Figura 5: Termini per la licenza

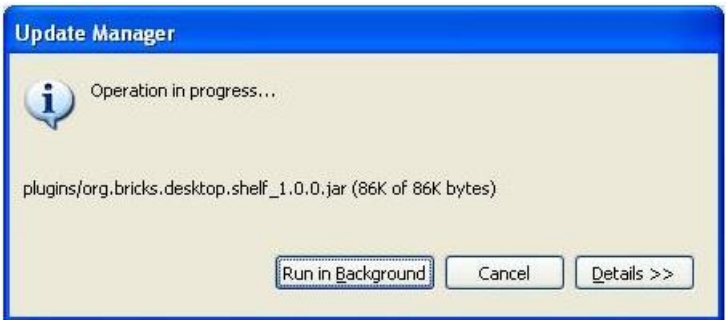


Figura 6: Progress monitor per l'installazione di aggiornamenti

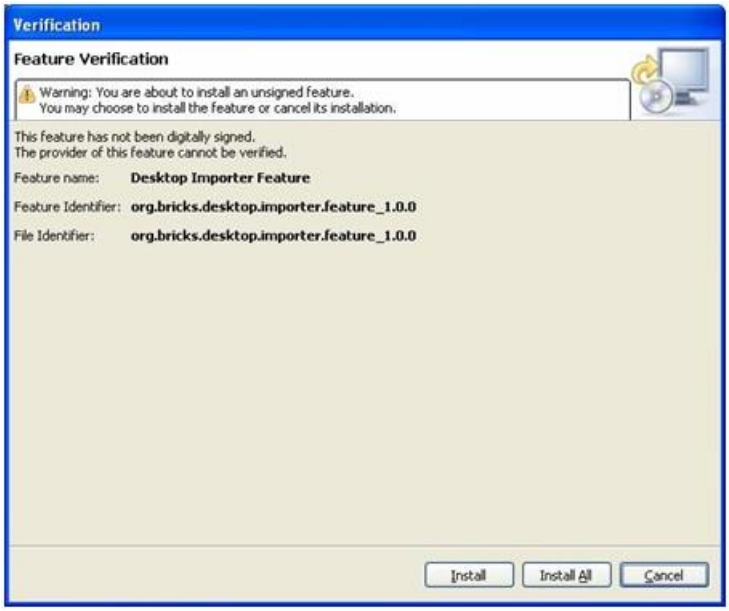


Figura 7: Messaggio di warning per le caratteristiche non segnalate

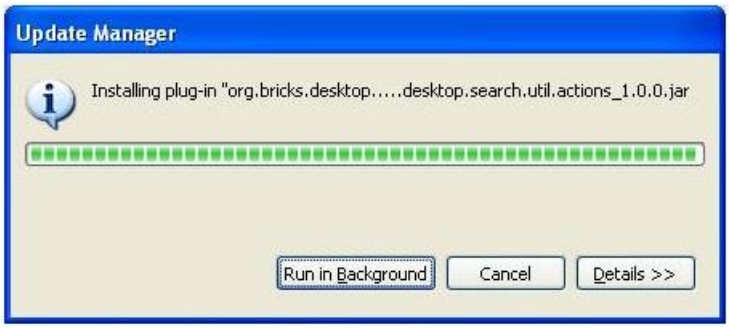


Figura 8: Update manager

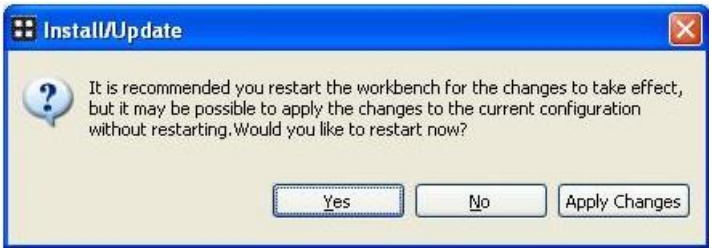


Figura 9: Attivazione delle caratteristiche di installazione

Al termine del processo di installazione l'Update Manager chiede opzionalmente di riavviare l'applicazione; allo stato attuale del Desktop non è necessario riavviare in quanto i componenti sviluppati sono *full-compliant* con il modello a plugin di Eclipse, ma a volte potrebbe essere richiesto di riavviare l'applicazione, al fine di aggiornare i componenti grafici.

Nella figura 10 è mostrato lo stato della GUI prima e dopo l'installazione. L'installazione minimale del desktop non fornisce la tool-bar mostrata nella figura 10b, il menu di ricerca (Figura 10c) e il menu di importazione ed esportazione (Figura 10d).

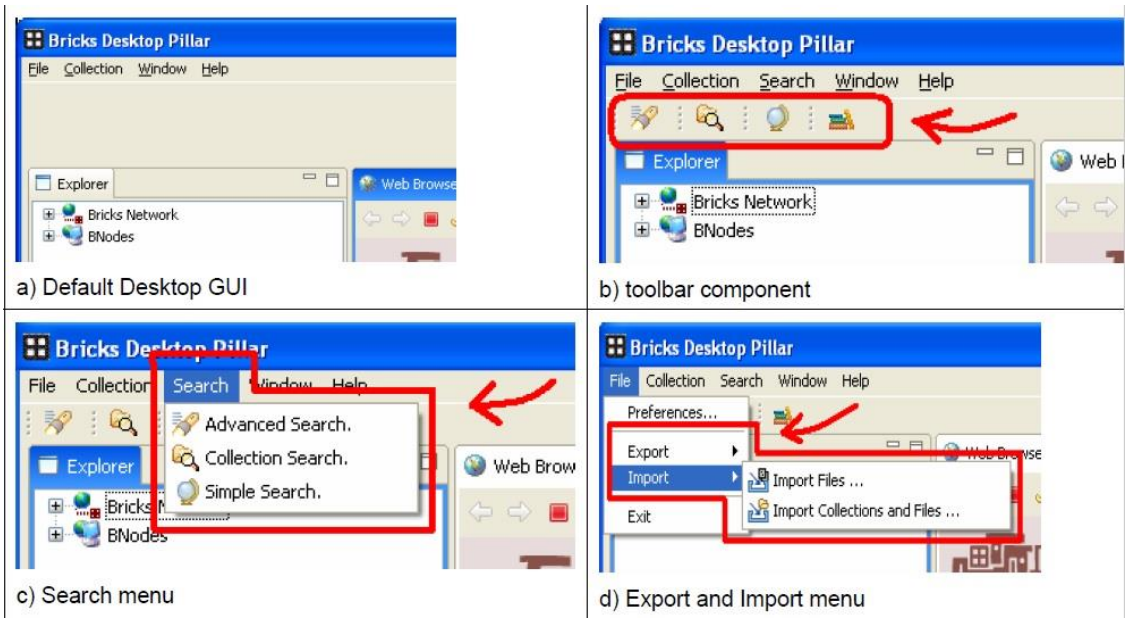


Figura 10: Componenti grafici installati

3.3 Iniziamo con il Desktop Pillar

Quando l'applicazione viene avviata si apre una finestra di accesso (si veda la Figura 11). L'utente deve inserire la login e la password ricevuti durante il processo di registrazione da parte del Bricks DL System Manager. Il nodo primario (BNodo) è l'URL del peer (vedi Figura 1), in cui è stato creato l'account utente. E' possibile mettere un segno di spunta sulla casella "Login automatically at startup", per evitare di loggarsi le volte successive. Per disattivare l'opzione di login automatico l'utente può utilizzare l'accesso Preferenze pagina, accessibile dal menu File/Preferenze.

Figura 11: Login Dialog

Non appena l'utente è autenticato compare una finestra di dialogo che mostra all'utente le informazioni di avanzamento (Figura 12) della richiesta di connessione. In caso di errori di rete o indisponibilità dei servizi richiesti il Desktop torna alla finestra di accesso, al fine di consentire all'utente il cambio di URL del nodo primario.

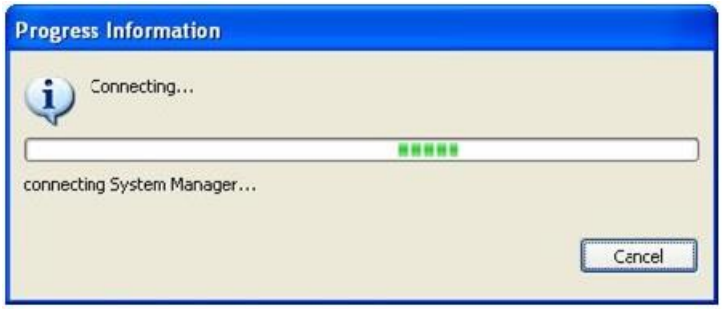


Figura 12 Finestra di dialogo

Non appena viene stabilita una connessione con il Bnodo primario, l'interfaccia grafica Desktop GUI (Figura 13) viene visualizzata all'utente. La configurazione iniziale della GUI parte con la web page della comunità di Bricks, al fine di mettere in rilievo le ultime news riguardanti la comunità. La pagina viene caricata nella zona centrale della GUI, e grazie ad un web browser integrato è possibile la navigazione in Internet direttamente all'interno dell'ambiente desktop evitando di passare così ad un'altra applicazione. Il browser Internet è ampiamente utilizzato all'interno del desktop per visualizzare gli oggetti digitali multimediali disponibili nella biblioteca digitale BRICKS.

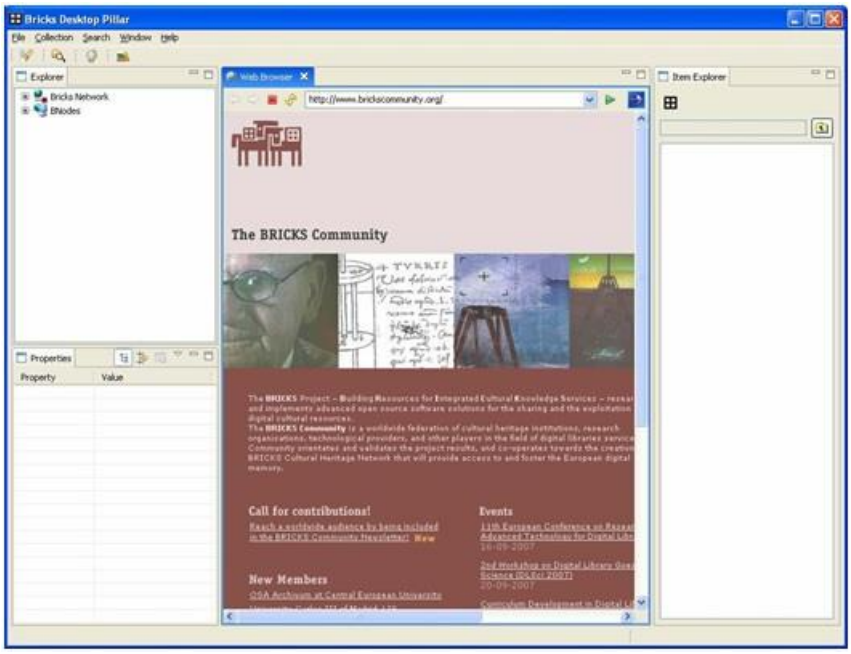


Figura 13: Desktop GUI

Il layout grafico del desktop è logicamente composto da due tipi di componenti: **View** e **Editors**.

Editors: sono raggruppati nella zona centrale della GUI. All'interno di questa zona il posizionamento degli editor può essere riorganizzata dall'utente. Un esempio di GUI con la zona centrale (riquadro rosso) diviso in tre parti è mostrato in Figura 14. Ogni parte contiene uno o più editor raggruppati in schede (riquadri gialli).

Views: vengono visualizzati nelle aree oblique della GUI. In Figura 14 sono circondati da rettangoli verdi. Anche le viste possono essere raggruppate in schede e spostate nell'area centrale dall'utente. Sono di solito utilizzate per fornire informazioni sintetiche in forma tabellare o gerarchica del contenuto attualmente visualizzato nella zona centrale. Per spostare e riposizionare gli editor e le view è sufficiente cliccare sulla barra del titolo e trascinare l'oggetto in un'altra posizione. La GUI darà un feedback, cambiando l'icona del mouse nei luoghi abilitati a ricevere l'oggetto trascinato. Se tutti i tab-editor non possono essere visualizzati a causa dello spazio limitato, una parte di essi è accessibile con un pulsante nella barra degli strumenti. Se si clicca sulla parte sinistra del pulsante un menu contestuale si aprirà per passare ad un altro editor aperto (Figura 15), se si clicca invece sulla parte destra del pulsante un menu generale si aprirà per gestire tutti gli editor.

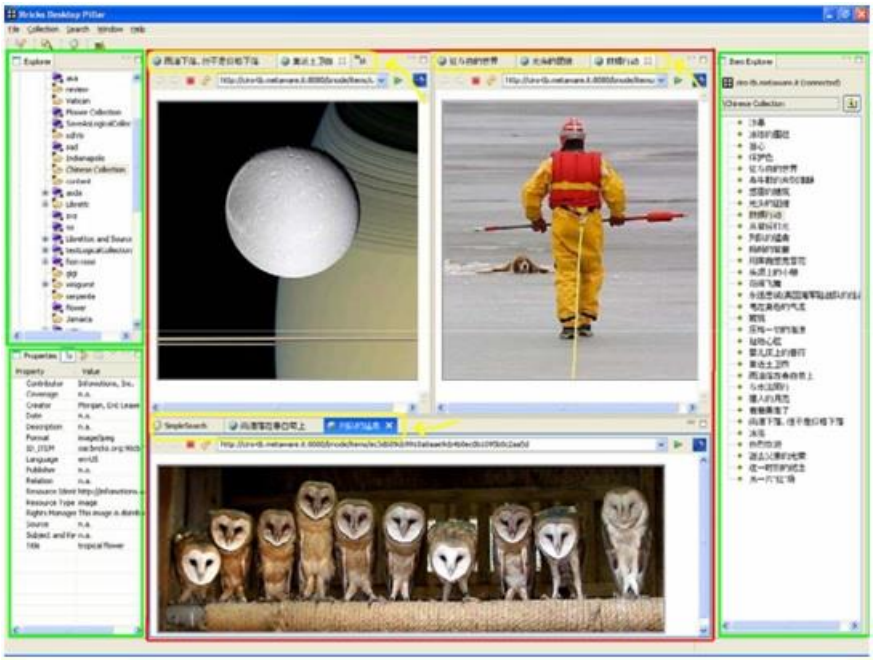


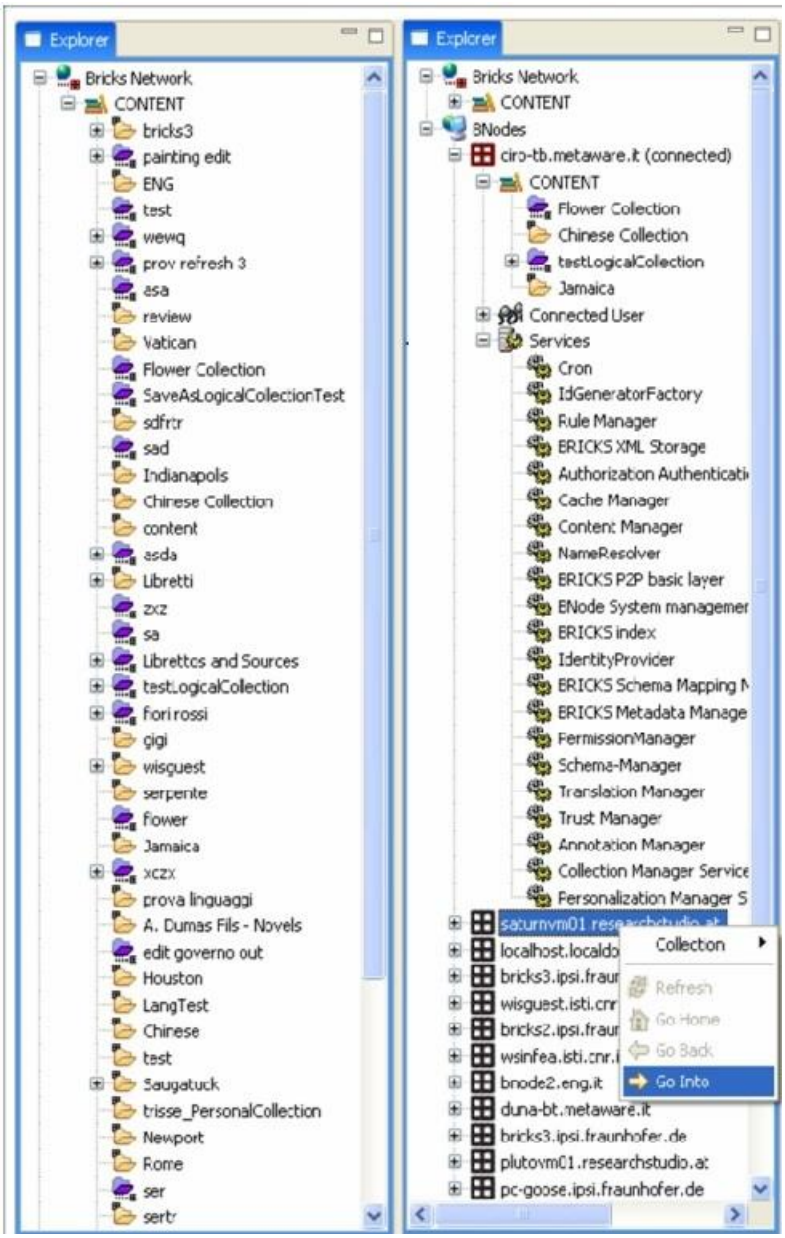
Figura 14: Riorganizzazione degli editor nella parte centrale

Figura 15: Menu per andare su un'altro editor

3.4 Navigazione e ricerca

3.4.1 Navigazione di Collezioni e di Oggetti Digitali

L'interfaccia grafica Desktop si basa sulla metafora "PC Desktop". Offre due modi diversi di esplorare la rete Bricks: la prima presenta tutte le collezioni e il loro contenuto come file system. Il secondo mostra l'elenco delle collezioni raccolte sotto ogni Bnodo. Due snapshot dell'Explorer View, si trovano in basso a sinistra dell'interfaccia grafica e sono illustrate nella Figura 16. La Figura 16-a mostra le collezioni trovate sulla rete BRICKS. Le collezioni gialle sono collezioni fisiche, mentre in colore blu sono raccolte le collezioni logiche che contengono riferimenti a oggetti memorizzati in altre collezioni fisiche.



a-Vista delle collezioni b-Vista dei Bnodi

Figura 16: Explorer View

Nella figura 16-b viene mostrata la lista delle collezioni che appartengono al Bnodo principale. Le collezioni allocate nel BNodo sono accessibili sotto il nodo grafico "Content", mentre i servizi disponibili sono accessibili tramite il nodo denominato "Services". L'utente può filtrare le collezioni e i BNodi utilizzando il comando "Go Into" mostrato nel menu contestuale.
La vista Explorer viene utilizzata per attivare altre viste che raccolgono in modo asincrono i dati dalla rete quando l'utente seleziona e apre un elemento grafico. Ad esempio, selezionando una collezione la "Property View" (Figura 17) viene attivata per visualizzare le proprietà associate a quell'elemento. In questo caso i metadati della collezione vengono recuperati e visualizzati come coppia attributo-valore.

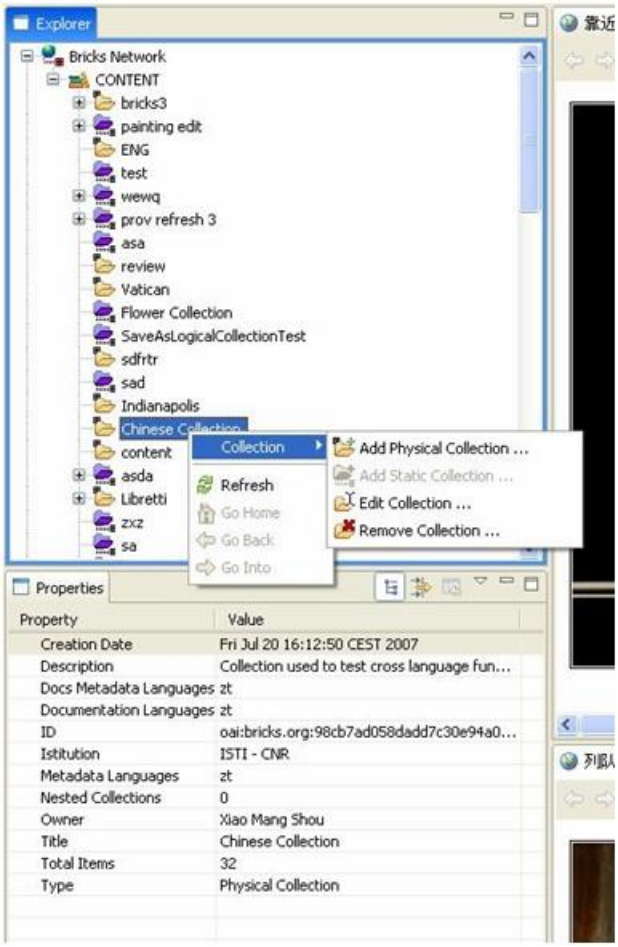


Figura 17: Explorer context e Property View

Scelto un elemento dal menu di contesto con il tasto destro viene aperto un menu contestuale a seconda del tipo di oggetto selezionato (Figura 17). In caso di collezione tutte le funzionalità per creare o modificare la collezione sono abilitate. Per sfogliare il contenuto di una collezione l'utente può navigare attraverso le collezioni nidificate cliccando sul "+" vicino all'elemento grafico.

3.4.2 Ricerca di Oggetti Digitale e Collezioni

Dentro la biblioteca digitale BRICKS sono state implementate tre funzionalità di ricerca: la ricerca semplice, dove l'utente inserisce una parola chiave, la ricerca avanzata che si basa su una ontologia e su uno schema di metadati, la ricerca per collezioni.

3.4.2.1 Ricerca Semplice

La figura 18 mostra l'area centrale della GUI desktop, dove sono state aperte due istanze dell'Editor Ricerca semplice. Quello di sinistra mostra i risultati di una query ottenuti usando come parola chiave il termine "flower", mentre quello di destra utilizza un carattere jolly (*) per trovare tutti gli oggetti che iniziano con "flow". Il numero di risultati viene visualizzato nella parte inferiore della GUI: 12 per la prima ricerca e 19 per una più generale. La figura dà anche un'idea dei diversi modi per la rappresentazione dei risultati. Nella prima ricerca è attivata una visualizzazione tabulare dei risultati che consente all'utente di manipolarli; difatti selezionando un elenco di oggetti è possibile applicare le azioni presenti nel menu contestuale (tasto destro). Il diagramma, invece, è utile per vedere le informazioni statistiche dei record trovati (ad esempio, la percentuale di immagini o file

PDF che corrispondono alla query). L'utente può selezionare diversi tipi di diagrammi (a torta o a barre), così come i tipi di metadati (tipo di formato, lingua dei metadati, istituzione, ecc.).

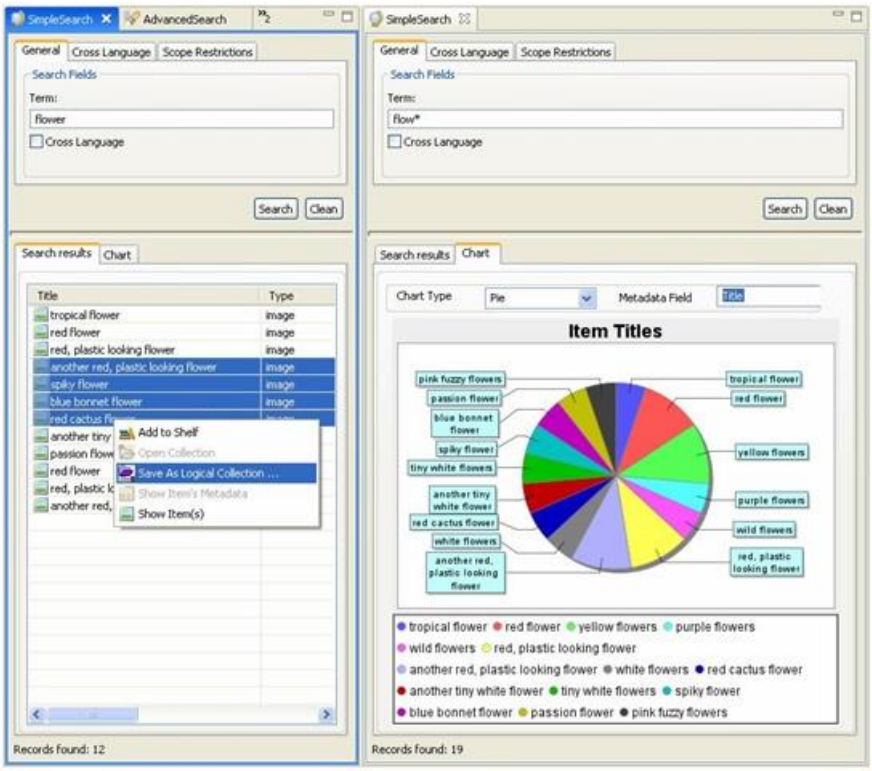


Figura 18: Ricerca Semplice

3.4.2.2 Ricerca avanzata e ricerca su collezioni

La Ricerca Avanzata e la Ricerca su Collezioni condividono lo stesso layout del pannello per esprimere le condizione per la query.

La ricerca avanzata consente all'utente di specificare lo Schema dei Metadata da utilizzare nella query. La prima volta che l'Editor per la ricerca avanzata viene aperto, viene interrogato il servizio Metadata Manager dei peer connessi, per recuperare gli schemi dei metadati disponibili. Quindi l'utente deve selezionare uno schema utilizzando la finestra di popup denominata "Select Metadata Schema". A questo punto una o più condizioni in forma (Attributo, Operatore, Value) possono essere aggiunti al pannello (con l'ausilio del pulsante "add"). Per attivare l'operatore logico AND/OR si seleziona la voce "Match all" e "Match at least one". I nomi degli attributi consentiti dallo schema selezionato vengono caricati automaticamente dall'editor e visualizzati all'utente in forma di lista. Gli operatori applicabili alla query sono: contiene, non contiene, uguale e non-uguale.

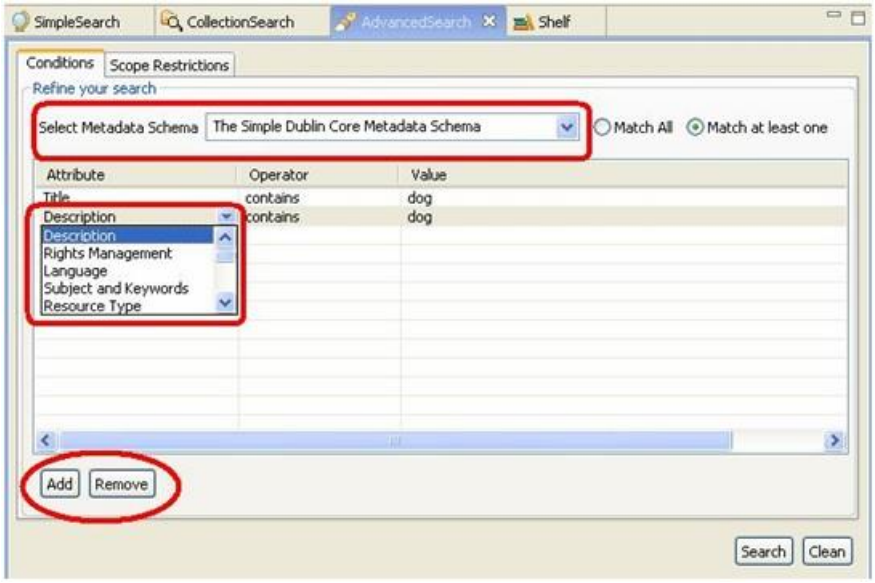


Figura 19: Ricerca Avanzata

Lo stesso meccanismo è usato per la ricerca su Collezioni, con la differenza che lo schema utilizzato si riferisce a collezioni e non ad oggetti digitali.

Quando viene eseguita la query (pulsante "Search") non appena c'è una corrispondenza, i risultati vengono visualizzati nel pannello inferiore e la vista tabellare comincia ad essere popolata. L'utente può utilizzare i risultati parziali, senza attendere che tutti i dati di risposta alla query vengano visualizzati, utilizzando il menu contestuale.

3.5 Importa esporta dati

Le funzionalità di Importazione ed Esportazione sono implementate nel BRICKS Desktop per popolare la libreria digitale e/o salvare in locale una copia di collezioni o oggetti digitali. L'utente può importare una gerarchia di sottodirectory e file a partire da una directory selezionata del file-system. Se i file e le directory allegati sono accompagnati da un file manifesto che esprime i metadati in Dublin Core (DC), gli oggetti vengono immessi nella libreria senza alcun intervento da parte dell'utente; altrimenti il desktop richiede all'utente di inserire i metadati DC obbligatoriamente. Il sistema di esportazione consente all'utente di salvare una collezione con tutte le sotto-collezioni annidate (sia fisiche che logiche) come una directory di file correlata con i metadati DC.

Esportazione di Collezioni e di Oggetti Digitali

Il menu di esportazione viene installato da Exporter plugin sotto il menu File. Le seguenti figure mostrano le finestre di dialogo utilizzate dall'operazione di esportazione. Inizialmente all'utente è richiesto di specificare la directory del file system locale su cui salvare le collezioni. Successivamente una nuova finestra di dialogo si apre (Figura 20) per selezionare la collezione digitale da esportare. Le collezioni sono raggruppate per bnode perché l'operazione avrà successo solo se l'utente seleziona collezioni di un bnode per il quale ha ottenuto l'autorizzazione. Facendo clic sul pulsante fine l'operazione termina. La figura 21 mostra un esempio di elenco dei file memorizzati sul file-system locale. Ogni collezione ed in modo ricorsivo tutte le sue sotto-

collezioni, vengono memorizzate come directory. Tutte le directory contengono all'interno un file di metadati con lo stesso nome della directory.

Figura 20: Collezione selezionata per l'esportazione

Importazione di directory e file

Lo stesso processo inverso è disponibile per l'importazione di oggetti del BNodo Bricks. L'utente deve prima selezionare una directory del file-system e dopo il BNodo di destinazione, o, la collezione in cui importare la directory. Se le directory importate hanno lo stesso formato di file per i file di metadati utilizzati dalla funzione di esportazione, nessun altro intervento da parte dell'utente è richiesto (Figura 21), altrimenti per ogni directory e file senza metadati si apre una finestra per facilitarne l'inserimento.

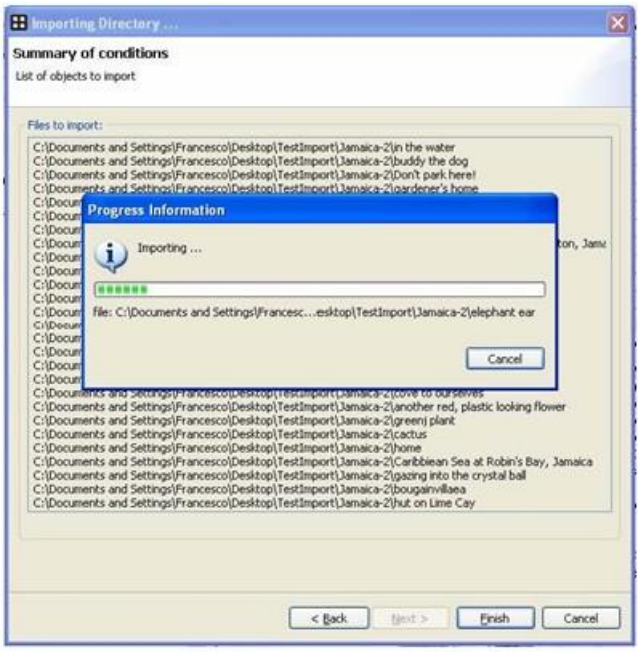


Figura 21: Importazione di directory e file

4. Architettura software

Il primo componente sviluppato è stato il "org.bricks.framework.client" che serve per comunicare con la rete Bricks. Questo bundle esporta le API dei componenti della Fondazione Bricks e di tutti i pacchetti necessari per utilizzare gli stub-client generati dal progetto. Dal momento che il progetto non ha come obiettivo la generazione di librerie Java (bundle) conformi alle specifiche OSGi, il primo passo è stato l'acquisizione di tutte le librerie necessarie di Java.

Lista dei jar usati: activation.jar, axis-schema.jar, axis.jar, bcprov-jdk13-128.jar, bnode-annotations.jar, bnode-api.jar, bnode-core-client.jar, castor-1.0.5-xml.jar, commons-codec-1.3.jar, commons-discovery- 0.2.jar, commons-io-1.1.jar, commons-logging-1.0.4.jar, xalan-2.6.0.jar, jaxrpc.jar, jcr-1.0.jar, log4j-1.2.8.jar, mail.jar, opensaml-1.0.1.jar, saaj.jar, sunxacml.jar, uddi4j.jar, wsdl4j-1.5.1.jar, wss4j-1.1.jar, xmlsec.jar, xercesImpl.jar, xml-apis.jar.

La classe astratta *ExploreNode* è la classe base dei modelli utilizzati da tutti i widget grafici del GUI Desktop. E' utilizzata per costruire la vista delle collezioni (Fig 16), la vista degli oggetti digitali e la vista dei metadati. Una instanza di ExplorerNode-derived può contenere altri oggetti Java derivati sempre dalla stessa classe base, e può implementare metodi di supporto per l'interfaccia IPropertySource richiesti per visualizzare le proprietà, sotto forma di coppie attributo-valore, delle View e delle Property.

```
public synchronized ExplorerNode addEntry(ExplorerNode entry) public
synchronized void removeEntry(ExplorerNode entry) protected abstract void
initDescriptors();
protected abstract void initProperties();
protected void addDescriptor(Object id, String display) protected
void addProperty(Object id,Object value) public ExplorerNode
search(Object obj)
public void fireNodeChanged(ExplorerNode entry, int event)
```

Snippet 1: i principali metodi di ExplorerNode

I primi due metodi sono utilizzati per gestire un modello gerarchico di nodi possibili, per essere visualizzati dentro il TreeViewer, i metodi astratti sono metodi di utilità utilizzati per inizializzare le proprietà dell'oggetto. Il resto dei metodi vengono utilizzati per ricercare una specifica istanza di un nodo nella gerarchia e per forzare un refresh delle view per sincronizzarle con il modello.

Se facciamo la supposizione di avere le informazioni dei servizi distribuiti sui BNodi collegati e vogliamo raggruppare tutti i servizi attivi come figli di un dato nodo, il codice da utilizzare per tale operazione è:

```
SystemManager SystemMgr = Session.getInstance().getSystemMgr();
Info[] info = SystemMgr.getDeployedBRICKS();
for (int i = 0; i< info.length;i++){ ServiceGroupNode.addEntry(new
      ActiveService(info[i]));
}
```

Snippet 2: come aggiungere ExplorerNode

Usiamo il metodo addEntry per aggiungere l'elenco dei servizi scoperti andando ad interrogare il servizio SystemManager. Ogni voce aggiunta all'istanza ExplorerNode è chiamata "ServiceGroupNode". Il codice per la classe ActiveService è il seguente:

```
public class ActiveService extends ExplorerNode {

      private Info info;
      public ActiveService(Info info) {
            super(info.getName()); this.info =
            info;
      }
      public ImageDescriptor getImageDescriptor(Object object) { return
            AbstractUIPlugin.imageDescriptorFromPlugin(
                        Application.PLUGIN_ID, IImageKeys.SERVICE);
      }
      protected void initDescriptors() {
            addDescriptor("service.name","Name");
            addDescriptor("service.producer","Producer");
            addDescriptor("service.version","Version");
      }
      protected void initProperties() {
            addProperty("service.name",info.getName());
            addProperty("service.producer",info.getProducer());
            addProperty("service.version",info.getVersion());
      }
}
```

Snippet 3: estendere la classe ExplorerNode

Abbiamo utilizzato i dati passati nel costruttore della classe per reperire i valori da visualizzare nella View Property. Mediante il metodo addDescriptor() diamo un nome al servizio (vale a dire "service.name") e allo stesso modo gli associamo una proprietà con il metodo addProperty (). Inoltre facendo l'override del metodo getImageDescriptor(), personalizziamo l'icona per il rendering.

Ogni nodo della gerarchia ad albero visualizzato dall'ExplorerView può essere esteso aggiungendo nodi personalizzati, che sono istanza di oggetto della classe ExplorerNode. I nodi personalizzati possono essere utilizzati o per rappresentare solo informazioni, o come punto di accesso per altri componenti grafici come le view o l'editor.

Ogni qual volta che un nuovo nodo viene aggiunto, un riferimento all'istanza del modello deve essere cercata. La classe Session fornisce differenti metodi per ottenere questi riferimenti: il metodo getRoot() fornisce l'intera gerarchia usata nell'architettura Desktop, il metodo getBnode() restituisce l'istanza del nodo connesso al BNode, il metodo getActivebnodes() restituisce la lista di tutti i BNodi rintracciati nella rete Bricks.

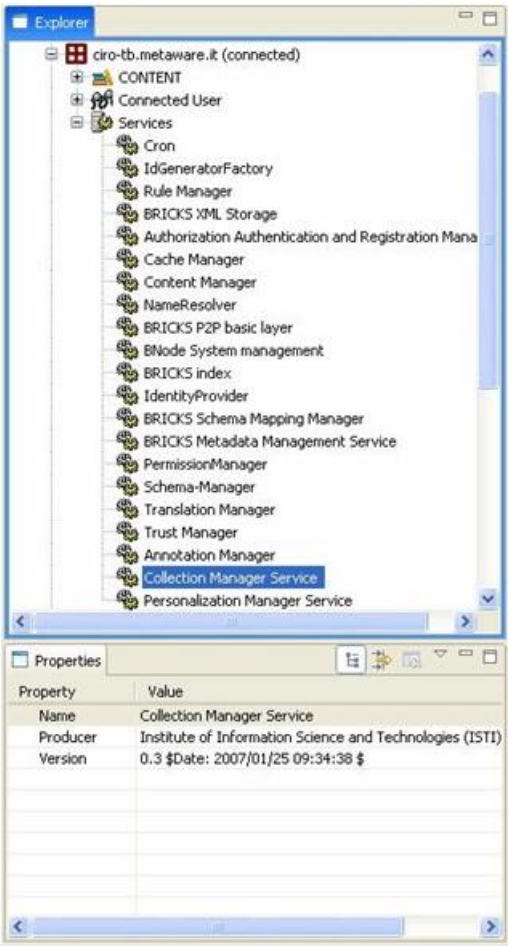


Figura 22 Aggiunta di servizi all'Explorer View

```
RootNode root = Session.getInstance().getRoot();
treeViewer = new TreeViewer(dlo_grp, SWT.BORDER | SWT.V_SCROLL);
treeViewer.addFilter(new ViewerFilter(){
      public boolean select(Viewer viewer, …, Object element) { if (
  (element instanceof UserGroupNode)
       || (element instanceof ServiceGroupNode) ) return
            false;
        else
            return true;
      }
});
      ….
treeViewer.setInput(root);
```

Snippet 4: accesso a Root Node

Fondamentalmente il modello è accessibile dall'istanza della classe Session ed è usato come input per la classe TreeView applicando un filtro che rimuove le estensioni della classe ExplorerNode che non devono essere visualizzate. Ci sono molte estensioni della classe ExplorerNode nel modello dell'applicazione Desktop; consultare il package "org.bricks.desktop.models.explorer" per un elenco completo. In figura 23 sono elencate tutte le classi del package:

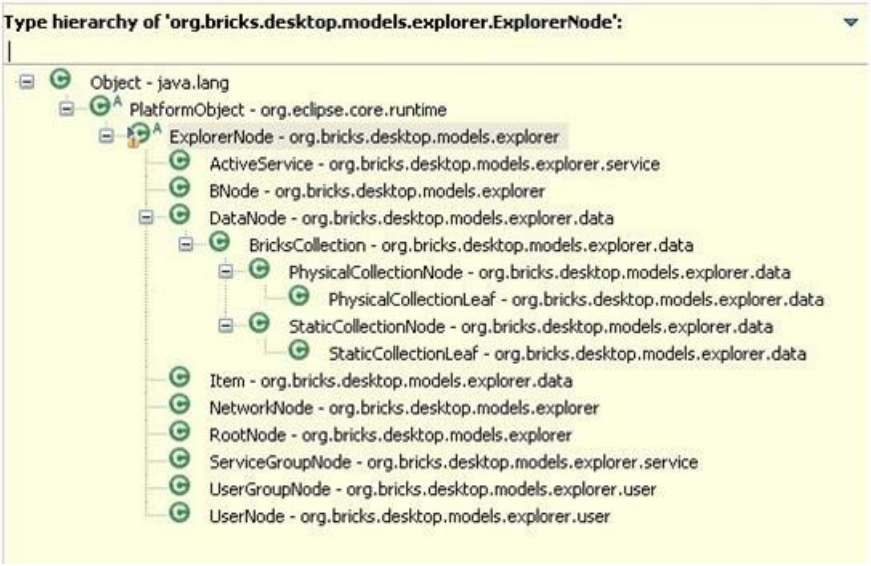


Figura 23 package org.bricks.desktop.models.explorer

5. Conclusioni

Con il Dimostratore Desktop abbiamo sperimentato un ambiente dinamico, espandibile e nel nostro piccolo abbiamo avuto la soddisfazione di entrare in un mondo software (il framework Eclipse) vasto, sviluppato da dei colossi dell' ICT come IBM. Attualmente il "Desktop Demonstrator" consiste di dodici plugin che accedono direttamente i seguenti servizi della fondazione:

Info; ServiceProxyFactory; SystemManagement; CollectionManager; NameResolver; MetadataManager; TranslationManager; RegistrationManager; RoleManager; UserManager; DRManager;

Altri servizi sono acceduti indirettamente dal "Collection Manager" e dal "Metadada Manager".

Dall'esperienza maturata con l'attuale implementazione del plugin, il Desktop è stato fonte di ispirazione per nuovi suggerimenti riguardanti il "deployment" dinamico di servizi BRICKS e nuove idee per altri potenziali mercati nei quali il sistema di gestione delle "DL BRICKS" potrebbe essere adottato.

Alcune ma importanti caratteristiche sono ancora mancanti nell'applicazione Desktop; la cattura di alcune informazioni potrebbe migliorare l'avvio dell'applicazione e la gestione di sessioni multiple consentirebbe una migliore gestione e accesso ai differenti peers della rete BRICKS.

Riferimenti

[1] https://it.wikipedia.org/wiki/Service-oriented_architecture
[2] https://www.w3.org/TR/soap/
[3] http://www.osgi.org/
[4] http://wiki.apache.org/cocoon/osgi
[5] https://eclipse.org/
[6] http://en.wikipedia.org/wiki/Rich_internet_application