XSL
eXtensible Stylesheet Language

Massimo Martinelli

Massimo.Martinelli@isti.cnr.it

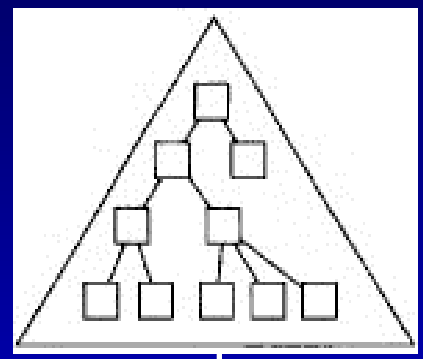
Consiglio Nazionale delle Ricerche - CNR
Istituto di Scienza e Tecnologie della Informazione - ISTI

Le componenti di XML

Contenuto

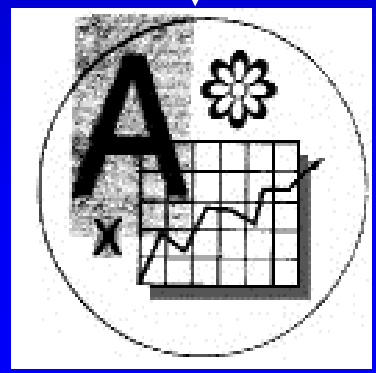
"La Discovery stava accelerando verso Giove lungo un'orbita complessa calcolata alcuni mesi prima dagli astronomi sulla Terra e controllata costantemente da Hal"
A.C. Clarke

Struttura



Specifiche di formattazione dell'output

Formattazione

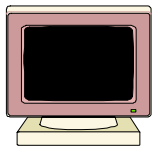


Stile

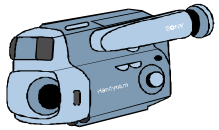
Processo completo di codifica XML

Visualizzazione di markup XML

tastiera
Braille



Monitor
(tipi differenti
Molteplici risoluzioni)



filmati

CD-DVD

Audio

televisione

dispositivi di Input/output

stile

Documento
XML

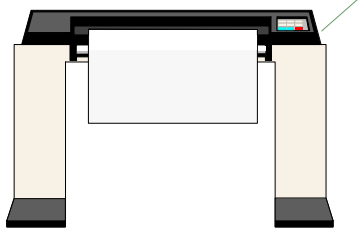
Altri dispositivi
Disponibili in fututo

lavatrice ?

orologio

comunicazioni

plotter

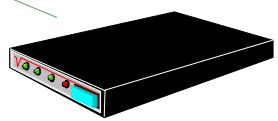


stampante

codice a barre

telefono

modem



palmare

cellulare

Esempio d'uso di CSS

```
<?xml version="1.0"?>
<biblioteca>
<?xml-stylesheet type="text/css" href="biblioteca.css"?>
<libro codice="R414">
        <titolo>2001: Odissea nello spazio</titolo>
        <autore>
                <cognome>Clarke</cognome>
                <nome>Arthur Charles</nome>
        </autore>
<editore>Rizzoli</editore>
        <parola_chiave>romanzo</parola_chiave>
        <parola_chiave>fantascienza </parola_chiave>
</libro>
</biblioteca>
```


```
titolo {        display: block;
                text-align: center;
                background: blue;
                color: white;
                font-family: Arial;
                font-size: 20pt
        }
autore { display: block;
                margin-left: 10%;
                text-align: left;
                color: red;
                font-family: Arial;
                font-style: italic;
                font-size: 14pt
        }
```

```
cognome, nome { display: inline; }
editore { display: block;
                margin-left: 15%;
                color: green;
                font-family: Arial;
                font-size: 14pt
        }
parola_chiave { display: block;
                        margin-left: 5%;
                        color: black;
                        font-family: "Times New Roman";
                        text-align: justify;
                        font-size: 14pt
                }
```

CSS non è sufficiente

- Le nuove versioni di CSS hanno aggiunto nuove possibilità
- questi *Stylesheet* sono comunque limitati:
  - non consentono modifiche al documento!
- Le potenzialità di XSL *eXtensible Stylesheet Language*, sono nettamente superiori a CSS

eXtensible Stylesheet Language (XSL)

Costituito da 3 parti:

- XSL-T(ransformations) linguaggio di trasformazione
- XPath linguaggio per indirizzare parti di documenti XML
- Formatting Objects vocabolario di formattazione

Raccomandazioni W3C

- XSL-T http://www.w3.org/TR/xslt 16 Novembre 1999
- XPath http://www.w3.org/TR/xpath
- XSL (FO) http://www.w3.org/TR/xsl

A cosa serve XSLT

- Consente di trasformare documenti XML in un altro formato:
  - **da XML a XML (HTML, XHTML, Formatting Objects, SVG, …)**
  - **da XML a testo (privo di markup XML, ma non di altro markup: es PDF, RTF, …)**

Trasformazioni

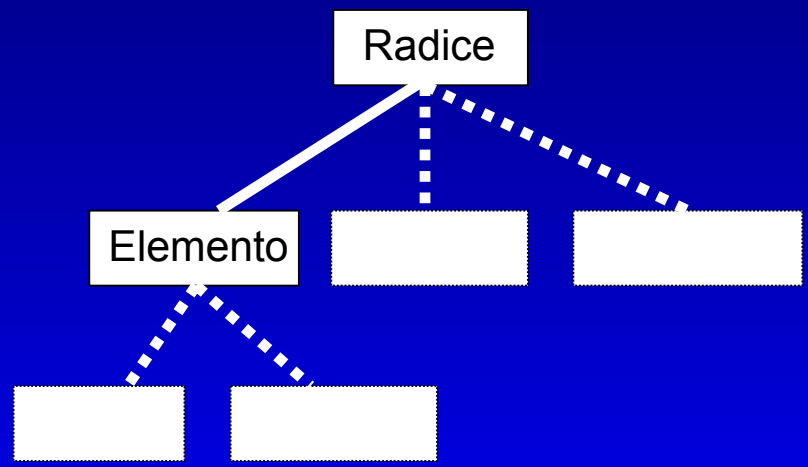
- Lato client
  - Browser applica lo stylesheet
- Lato server
  - servlet o cgi applicano lo stylesheet
- Programma
  - La trasformazione avviene prima che il file venga messo a disposizione
- Negli esercizi useremo un processore chiamandolo da linea di comando (3° metodo). E' possibile usare lo stesso processore chiamando le API da una servlet

Alberi

- Ogni documento XML ben formato può essere visto come un albero

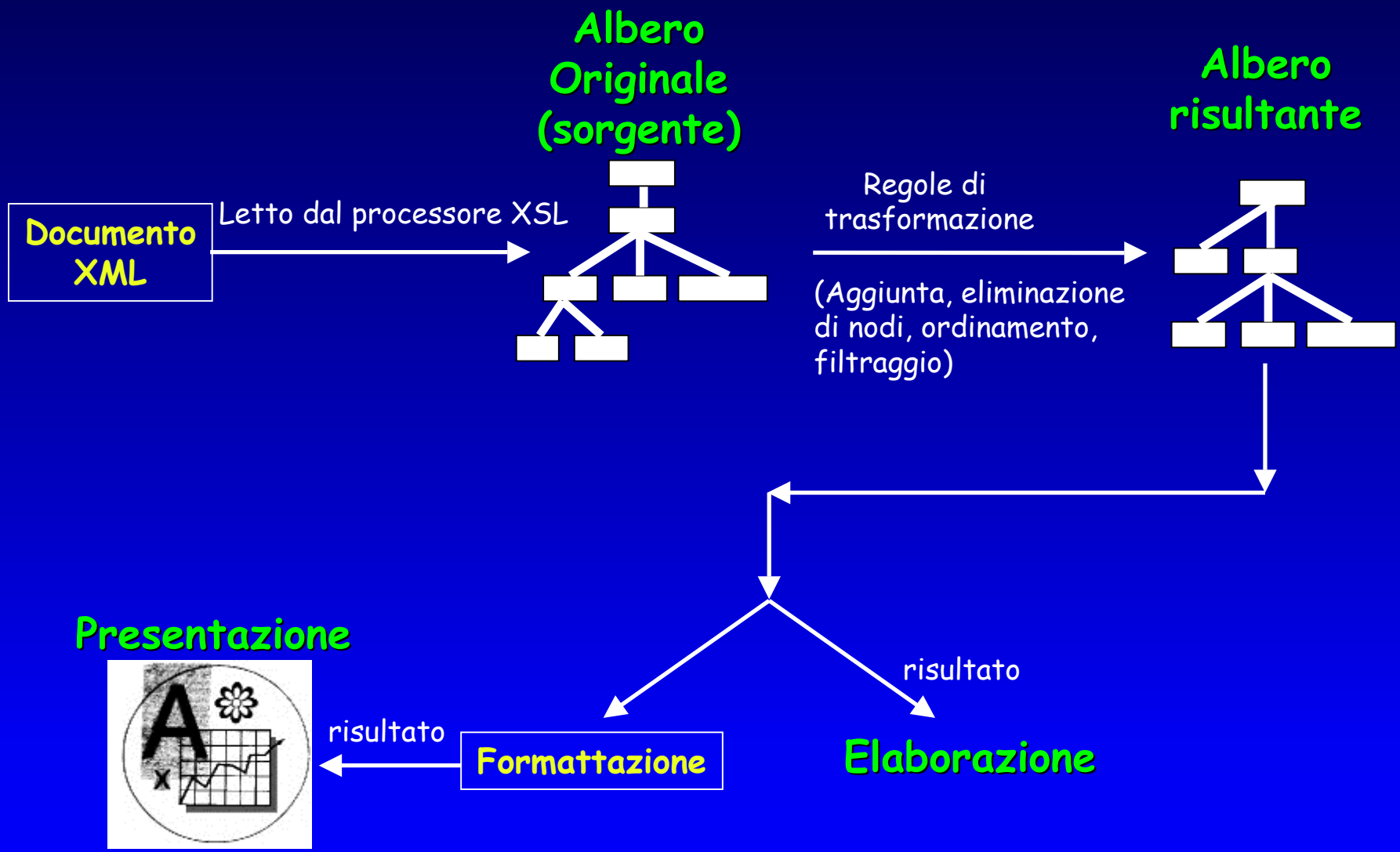
Nodi dell'albero:

- Radice
- Elementi
- Testo
- Attributi
- Namespace
- Istruzioni di processo
- Commenti



- Utile proprietà degli alberi:
  - ogni elemento con i propri figli costituisce un albero

Il processo di trasformazione

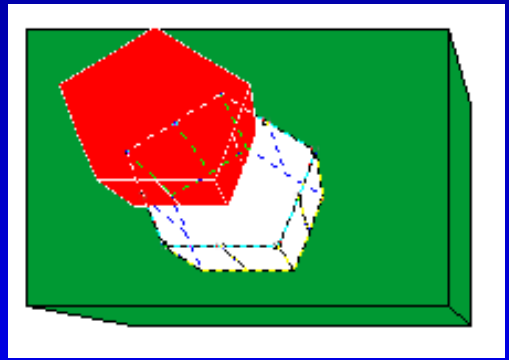


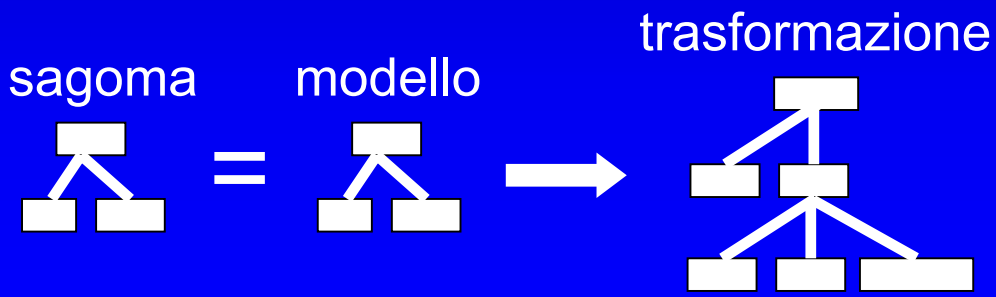
Regole XSL-T

Template (sagoma)

- Il processore scorre i nodi di cui è composto il documento XML
- Quando un albero (nodo o insieme di nodi) combacia (match) con il modello (pattern) di una regola di template (sagoma)
  viene applicata la trasformazione specificata

```
<xsl:template match="modello">
        … trasformazione …
</xsl:template>
```





Esempio di template

```
<xsl:template match="nodo">
    testo <tag> <xsl:value-of select="." /> </tag>
</xsl:template>
```

- *Se un nodo del documento XML combacia con quello della regola dell'esempio viene prodotto come risultato*

testo <tag> valore dell'elemento corrente </tag>

- *Dato il seguente frammento XML*

```
<cognome>Mario Rossi</cognome>
```

- *e il seguente frammento XSLT*

```
<xsl:template match="cognome">
    Cognome: <strong><xsl:value-of select="."/></strong>
</xsl:template>
```

- *Risultato trasformazione:*

Cognome: Mario Rossi

- *Visualizzato da un browser in questo modo:*

Cognome: **Mario Rossi**

Esempio di trasformazione

Aprire il file xml con un editore testi

Esempio lato client

- Da Internet Explorer 6 aprire il file "nuovi-esempi/esempioIE/library2.xml"
  (non perfettamente compatibile con le raccomandazioni W3C)

Esempio da programma

- Aprire il file "nuovi-esempi/esempioIE/Risultato.html" ottenuto con Xalan
  (compatibile con le raccomandazioni W3C)

- Differenze:
  - *xmlns:xsl*
  - *apply-templates (romanzo fantascienza romanzo narrativa)*

Il foglio di stile lo esamineremo più avanti…

Abbiamo visto che una regola di template specifica un nodo come sagoma, ma come si specificano i nodi ? …

XPath

- Linguaggio che attraverso una stringa specifica un insieme di nodi, un **percorso per localizzare** (location-path) questi nodi
- Utilizzato da XSL-T per selezionare i nodi da processare
- Parte della sintassi è simile a quella adottata dai sistemi operativi per individuare un file all'interno di una gerarchia di directory
  /Corsi/ComuneLivorno/XSLT-150503.ppt
- La stringa è formata da espressioni che possono restituire questi tipi oggetti
  - Insiemi di nodi
  - booleani true, false
  - numeri interi e decimali positivi e negativi
  - stringhe "questa è una stringa", 'anche questa'
  - frammenti dell'albero risultante
- XPath mette a disposizione alcune funzioni di base per manipolare i tipi di oggetti sopra elencati

Contesto

- In XPath ogni cosa è interpretata in riferimento ad un **contesto**
- In generale il contesto è il nodo dell'albero sorgente a partire dal quale viene valutata una espressione
- Il contesto è costituito da 6 parti
  1. il nodo di contesto (per fare un parallelo: la directory corrente)
  2. la posizione di contesto (es: dato un elenco <li> quale oggetto nell'elenco, il 1°, o il 2°, …)
  3. la dimensione di contesto (es: dato un elenco quanti oggetti ne fanno parte)
  4. l'insieme di variabili visibili
  5. le funzioni disponibili
  6. l'insieme di namespace visibili

Passi di un percorso di localizzazione

- Il percorso serve a localizzare insiemi di nodi (node-sets) contenuti all'interno di documenti XML
- Un percorso di localizzazione può essere scomposto in uno o più **passi**
  esempio: biblioteca/libro/titolo 3 passi separati dal carattere "/"
- Può essere un assoluto o relativo:
  - **Assoluto**: inizia con il simbolo "/" /biblioteca/libro
  - **Relativo**: non inizia con il simbolo "/ " libro/titolo
- Un passo di un percorso può essere scomposto in 3 ulteriori parti
  asse::nodo di test [predicati] (zero o più predicati)
- Esempio:

child::anagrafe [position() = 3]

asse **node-test** **predicato**

Individua il 3° elemento "anagrafe", figlio del nodo contesto

Esempi (sintassi non abbreviata)

- **child::para** seleziona tutti gli elementi para figli del nodo di contesto (NDC)
- **child::*** tutti gli elementi figli del NDC
- **child::text()** tutti i nodi teto figli del NDC
- **child::node()** tutti i figli del NDC, qualsiasi tipo di nodo siano
- **attribute::name** l'attributo name del NDC
- **attribute::*** tutti gli attributi del NDC
- **descendant::para** gli elementi para discendenti del NDC
- **ancestor::div** tutti gli elementi div avi del NDC
- **ancestor-or-self::div** come precedente più il NDC stesso se è un elemento div
- **descendant-or-self::para** tutti gli elementi para discendenti compreso il NDC se è un elemento para
- **self::para** seleziona il NDC se è un elemento para (niente altrimenti)
- **child::capitolo/descendant::para** gli elementi para discendenti dell'elemento capitolo figlio del NDC
- **child::*/child::para** gli elementi para nipoti del NDC
- **/** l'elemento radice del documento (sempre genitore del "document element")
- **/descendant::olist/child::item** tutti gli elementi item che hanno un padre olist e che sono nello stesso documento del NDC
- **child::para[position()=1]** il primo figlio para del NDC
- **child::para[position()=last()-1]** il penultimo elemento para figlio del NDC
- **child::para[position()>1]** tutti i figli para del NDC eccetto il primo
- **following-sibling::capitolo[position()=1]** il primo elemento capitolo fratello successivo del NDC
- **preceding-sibling::capitolo[position()=1]** il primo elemento capitolo fratello precedente del NDC
- **/child::doc/child::capitolo[position()=5]/child::sezione[position()=2]** il secondo elemento del 5° capitolo dell'elemento doc figlio dell'elemento radice
- **child::para[attribute::tipo='attenzione'][position()=5]** il 5° figlio para del NDC che ha un attributo tipo con il valore attenzione
- **child::para[position()=5][attribute::type="warning"]** identico al precedente
- **child::capitolo[child::titolo='Introduzione']** i figli capitolo del NDC che hanno uno o più figli titolo con valore uguale a Introduzione
- **child::capitolo[child::titolo]** i figli capitolo del NDC che hanno uno o più figli titolo
- **child::*[self::capitolo or self::appendice][position()=last()]** l'ultimo capitolo o appendice figli di NDC

Esempi (sintassi abbreviata)

- **para** seleziona gli elementi para figli del NDC
- ***** tutti gli elementi figli del NDC
- **text()** tutti i nodi testo figli del NDC
- **@nome** l'attributo nome del NDC
- **@*** tutti gli attributi del NDC
- **para[1]** il primo elemento para figlio del NDC
- **para[last()]** l'ultimo
- ***/para** tutti gli elementi para nipoti del NDC
- **/doc/capitolo[5]/sezione[2]** il 2° figlio sezione del 5° figlio capitolo dell'elemento doc figlio della radice
- **chapter//para** tutti gli elementi para discendenti dell'elemento capitolo figlio del NDC
- **//para** tutti gli elementi para discendenti dell'elemento radice (stesso documento del NDC)
- **//olist/item** tutti gli elementi item figli di olist discendenti della radice item
- **.** il nodo di contesto
- **.//para** gli elementi para discendenti del NDC
- **..** il genitore del NDC
- **../@lang** l'attributo lang del genitore del NDC
- **para[@tipo="attenzione"]** tutti i figli para che hanno un attributo tipo con valore attenzione
- **para[@tipo="attenzione"][5]** il 5° figlio para del NDC con un attributo tipo con valore attenzione
- **para[5][@tipo="attenzione"]** identico al precedente
- **capitolo[titolo="Introduzione"]** il figlio capitolo figlio del NDC che ha almeno un elemento figlio titolo con valore uguale a Introduzione
- **capitolo[titolo]** tutti i figli del NDC capitolo con almeno un figlio titolo
- **impiegato[@codice and @tipo]** tutti i figli del NDC impiegato con un attributo codice e uno tipo

Asse

Specifica la **direzione** verso cui spostarsi a partire dal nodo di contesto

- ancestor — corrisponde agli elementi **avi** del nodo corrente
- ancestor-or-self — avi e nodo corrente
- attribute — attributo del nodo corrente (abbreviato con “@”)
- child — figli del nodo corrente (asse di default)
- descendant — tutti i discendenti del nodo corrente
- descendant-or-self — discendenti e nodo corrente
- following — tutti i nodi che seguono quello corrente (in ordine di apparizione all’interno del documento)
- following-sibling — nodi fratelli successivi al nodo corrente
- namespace — i nodi namespace del nodo corrente
- parent — il genitore del nodo corrente (abbreviato con “..”)
- preceding — i nodi che precedono quello corrente (in ordine di apparizione all’interno del documento)
- preceding-sibling — nodi fratelli precedenti al nodo corrente
- self — il nodo corrente (abbreviato con “.”)

Test sui nodi

- Il node-test determina il tipo di nodo da selezionare

Node-Test

• nome	combacia con i nodi elementi <nome>
• *	qualsiasi nodo di tipo elemento
• namespace:nome	ogni elemento <nome> con il namespace specificato
• namespace:*	qualsiasi elemento con il namespace specificato
• comment()	i nodi di tipo commento
• node()	qualsiasi nodo
• text()	I nodi di tipo testo
• processing-instruction()	le istruzioni di processo

XPath: Predicati

- Un predicato **filtra** un insieme di nodi rispetto ad un asse per produrre un nuovo insieme di nodi
  - Se l'espressione è true il nodo viene inserito nel nuovo insieme di nodi
  - Se l'espressione restituisce un numero e questo numero corrisponde alla posizione del nodo di contesto nella lista dei nodi questo nodo viene inserito nel nuovo insieme
  - Se restituisce una stringa non vuota il nodo di contesto viene inserito nel nuovo insieme

Esempi

- **nodetest[1]** combacia con il primo nodo

  I test sui nodi restituiscono i nodi secondo l'ordine nel documento, ma nel caso degli avi (ancestor) o precedenti (preceding) l'ordine è ovviamente inverso.

  Il primo nodo è sempre quello più vicino al nodo di contesto

- **nodetest[position()=last()]** l'ultimo nodo
- **nodetest[position() mod 2 = 0]** nodo pari
- **element[@id='val']** elementi con un attributo id con valore "val"
- **element[not(@id)]** elementi che non hanno un attributo id
- **persona[nome="Mario"]** elementi <persona> che hanno un elemento figlio nome con valore "Mario".
- **persona[normalize-space(nome)="Mario"]** identico al precedente senza tenere di conto gli spazi

Predicati - Insiemi di nodi - Booleani

Insiemi di nodi

- last() restituisce il numero di nodi in un insieme di nodi
- position() posizione del nodo di contesto
- count(node-set) il numero di nodi
- local-name(node-set) il nome locale (senza namespace) del primo nodo
- namespace-uri(node-set) URI del namespace del primo nodo
- name(node-set) il nome qualificato (con namespace) del primo nodo

Booleani

- !=
- <
- <=
- =
- >
- >=

- and
- or
- true()
- false()

- ***Esempio:***

```
<xsl:template match="libro[position() > 2]">
...
</xsl:template>
```

Predicati: numeri e stringhe

Numeri

- \+ Addizione
- \- Sottrazione
- * Moltiplicazione
- div Divisione
- mod Modulo
- ceiling() Intero più piccolo
- floor() Intero più grande
- round() Intero più vicino (arrotondamento)
- sum() Somma di numeri

Stringhe

- starts-with(string string1, string string2) - Boolean
- contains(string string1, string string2) - Boolean
- substring(string string1, number offset, number length) - String
- substring-before(string string1, string string2) - String
- substring-after(string string1, string string2) - String
- string-length(string string1) - Number
- normalize-space(string string1) - String
- translate(string string1, string string2,string string3) - String
- concat(string string1, string string2,...) - String

Esempio di trasformazione di XML in HTML

- *Documento XML*

```
<?xml version="1.0" encoding="UTF-8" ?>
<saluto>
    Benvenuto nel mondo di XSL-T
</saluto>
```

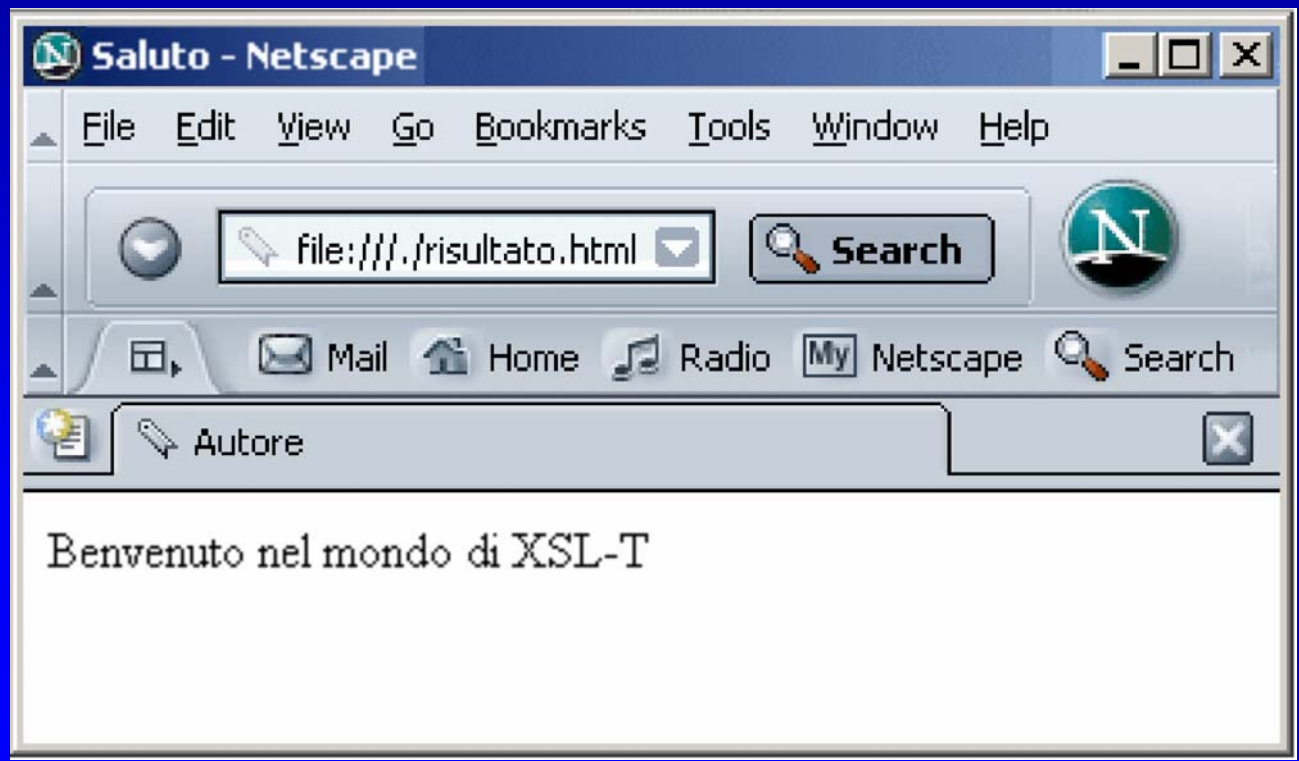
- *Documento XSL-T*

```
<?xml version="1.0" encoding='UTF-8'?>
<xsl:stylesheet version="1.0"
      xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match="/">
  <html>
    <head><title>Saluto</title></head>
    <body>
       <xsl:apply-templates select="saluto"/>
    </body>
  </html>
</xsl:template>

<xsl:template match="saluto">
    <xsl:value-of select="."/>
</xsl:template>

</xsl:stylesheet>
```

Risultato della trasformazione

```
<html>
<head>
<META http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Saluto</title>
</head>
<body>
  Benvenuto nel mondo di XSL-T
</body>
</html>
```

aggiunta dal processore



Elementi di XSL-T

- **<xsl:template match="/">**
  - Template per l'elemento radice

  indica come trasformare l'elemento radice

- **<xsl:template match="x">**
  - Template per tutti gli elementi "x"

  indica come trasformare l'elemento "x"

  Nel successivo esempio vedremo questo template
- **<xsl:template match="n/x">**
  - Template per tutti gli elementi "x" figli di elementi "n"

  indica come trasformare gli elementi n/x

Esempio di trasformazione di XML in HTML

- *Documento XML*

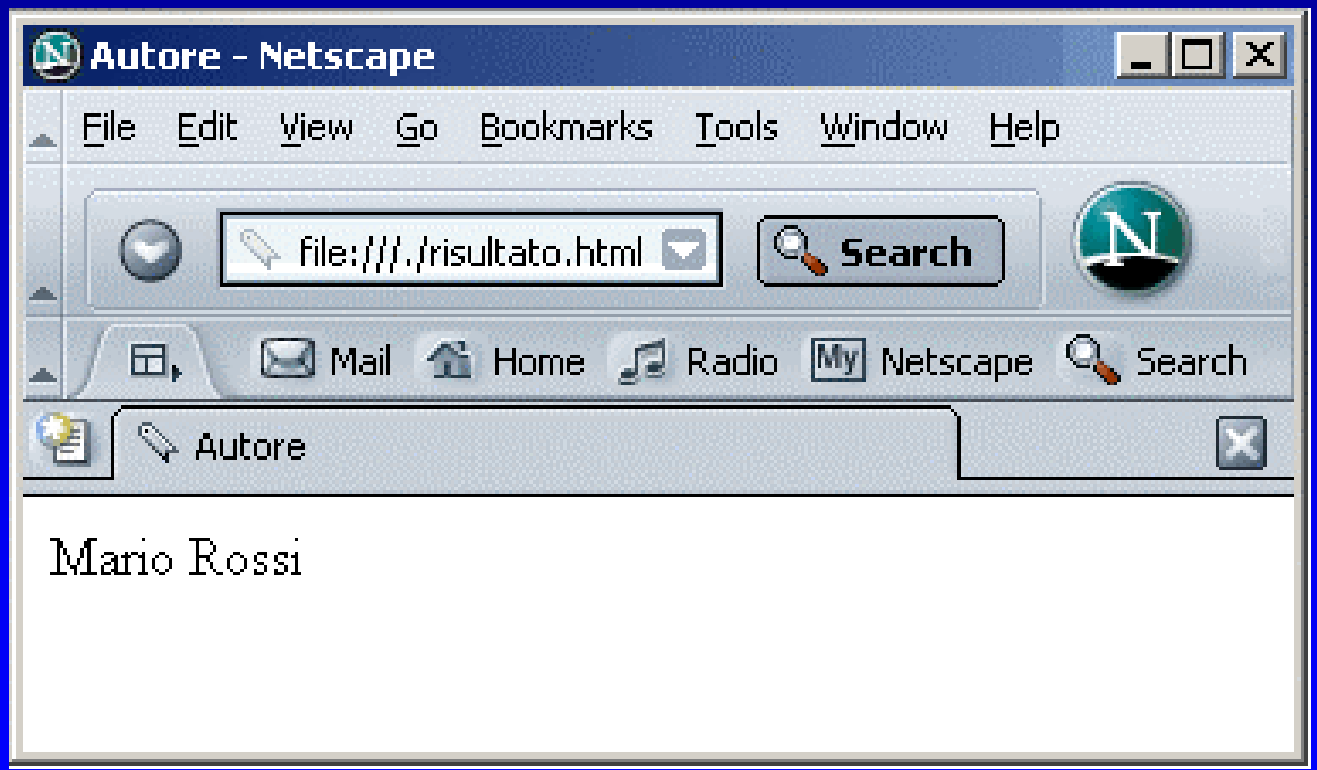
```
<?xml version="1.0" encoding="UTF-8" ?>
<documento>
    <autore>Mario Rossi</autore>
</documento>
```

- *Documento XSL-T*

```
<?xml version="1.0" encoding='UTF-8'?>
<xsl:stylesheet version="1.0"
      xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match="/">
  <html>
    <head><title>Autore</title></head>
    <body>
       <xsl:apply-templates select="documento/autore"/>
    </body>
  </html>
</xsl:template>

<xsl:template match="autore">
    <xsl:value-of select="."/>
</xsl:template>

</xsl:stylesheet>
```

Risultato della trasformazione

```
<html>
<head>
<META http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Autore</title>
</head>
<body>Mario Rossi</body>
</html>
```



Elementi di XSL-T

- **<xsl:apply-template select="x">**
  - Applica le regole di trasformazione relative a tutti gli elementi “x” contenuti nell'elemento corrente

- **<xsl:value-of select=".">**
  - Restituisce il valore dell'elemento corrente

Un altro esempio

- Documento XML

```
<?xml version="1.0" ?>
<documento>
    <autore>Mario Rossi</autore>
</documento>
```

- Documento XSLT

```
<?xml version="1.0" ?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
        version="1.0">
    <xsl:template match="autore">
        Autore: <strong> <xsl:apply-templates /> </strong>
    </xsl:template>
</xsl:stylesheet>
```

Applica i template relativi agli elementi figli (ove ve ne siano) … e non solo ….

Il risultato della trasformazione

nuovi-esempi/2-da XML in HTML/Risultato.html

Attenzione: non funziona come ci aspetteremo perché esistono delle regole predefinite……

Esempio di template

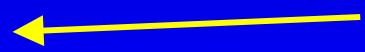
- Perché il processore ha prodotto in output il contenuto dell'elemento autore ?
- Perché non avendo specificato un template per il contenuto degli elementi viene usato un template di default

- Anche apply-templates senza select fa si che vengano applicati i template di default: applica i template per gli elementi, testi, commenti e istruzioni di processo contenuti nell'elemento corrente (tutti i nodi che sono figli dell'elemento corrente)

```
<xsl:template match="nodo">
      testo <tag> <xsl:apply-templates /> </tag>
</xsl:template>
```

- **Se un nodo del documento XML combacia con quello della regola dell'esempio viene prodotto come risultato**

```
testo  <tag>
  valore dell'elemento corrente
  applicazione delle altre regole di template
</tag>
```



regola di default

Regole XSLT precostituite, per difetto (default)

- XSL-T usa i *template* per specificare come devono essere trasformati gli elementi di un documento XML

- Il processore inizia l'elaborazione partendo dal template dell'elemento *radice*
- Ogni elemento riferito dal template dell'elemento radice sarà elaborato a sua volta

- Esistono 5 template di default, uno per ogni tipo di nodo, nel caso nessun template combaci un qualsiasi tipo di nodo viene applicata la regola di default,
  o anche se usiamo apply-templates senza attributo "select"

1.
```
<xsl:template match="/ | *">
    <xsl:apply-templates/>
</xsl:template>
```
Applica le regole per qualsiasi elemento

2.
```
<xsl:template match="text()">
    <xsl:value-of select="."/>
</xsl:template>
```
per qualsiasi testo

3.
```
<xsl:template match="@*">
    <xsl:value-of select="."/>
</xsl:template>
```
La regola per gli attributi è "ingannevole" Se non specifichiamo il template per gli attributi non vengono prodotti in output i valori

4.
```
<xsl:template match="comment()"/>
```
Non applicare le regole per i commenti

5.
```
<xsl:template match="processing-instruction()"/>
```
Ne per le istruzioni di processo

Sostituzione delle regole precostituite

<xsl:template match="nodo">
</xsl:template>

Non applica alcuna trasformazione per il nodo specificato

<xsl:template match="*|text()" />

Non applica alcuna per gli elementi e per i testi (a meno di regole più specifiche)

Tipo di Output

- XML per difetto
  <xsl:output method="xml"/>

- HTML 4.0
  <xsl:output method="html"/>

- Testo
  <xsl:output method="text"/>

- Inserimento di spazi e indentazione
  <xsl:output indent="yes"/>
- Specifica del DOCTYPE nel documento prodotto con
  <xsl:output doctype-system="nomefile.dtd"/>
  si ottiene
  <!DOCTYPE elementoradice SYSTEM "nomefile.dtd">

apply-templates

<xsl:apply-templates/>

- Indica di avviare il processamento in modo ricorsivo su tutti i figli di un nodo, controllando ogni regola che possa essere applicata.

<xsl:apply-templates select="nodo" />

- L'attributo select permette di selezionare il nodo di partenza per il processamento ricorsivo

Output XHTML

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
   xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:output method="xml"
   doctype-public="-//W3C//DTD XHTML 1.0 Transitional//EN"
   doctype-system="http://www.w3.org/TR/xhtml1/DTD/xhtml1-
    transitional.dtd"/>

<xsl:template match="/">
  <html xmlns="http://www.w3.org/1999/xhtml">
    <head>
      <title>esempio di output XHTML</title>
    </head>
    <body>
      <xsl:apply-templates />
    </body>
    </html>
</xsl:template>
  ...
</xsl:stylesheet>
```

xsl:comment

- Commenti in XSL-T

<xsl:comment>commento</xsl:comment>

Risorse su XSLT

- *"**XSL Transformations (XSLT)"*** - http://www.w3.org/TR/xslt
- *"**XML Path Language (XPath)"*** - http://www.w3.org/TR/xpath
- *"**Extensible Stylesheet Language (XSL)**"* - http://www.w3.org/TR/xsl

- *"Xalan" http://xml.apache.org/xalan-j*
- *"Xalan extensions" http://xml.apache.org/xalan-j/extensions.html*

- *XML Bible 2nd Edition* - Capitolo 17 : "XSL Transformations" http://metalab.unc.edu/xml/books/bible/updates/17.html

- "*XSL Concepts and Practical Usage*" Paul Grosso, Norman Walsh http://www.nwalsh.com/docs/tutorials/xsl/xsl/slides.html

- "XSL by Example" Marc Colan ibm.com/developerWorks/speakers/colan

- G. Ken Holman tutorial "*Practical Trasformations with XSL-T and XPath"* http://www.CraneSoftwrights.com

- Improve your XSLT coding five ways. Tips to make you a better XSLT programmer. Benoit Marchal – IBM DeveloperWorls

- "*XSLT Working with XML and HTML*" Fung. Addison Wesley

- XSLT – Doug TidWell - O'Reilly