XPath & XSLT

Ombretta Gaggi
Università di Padova

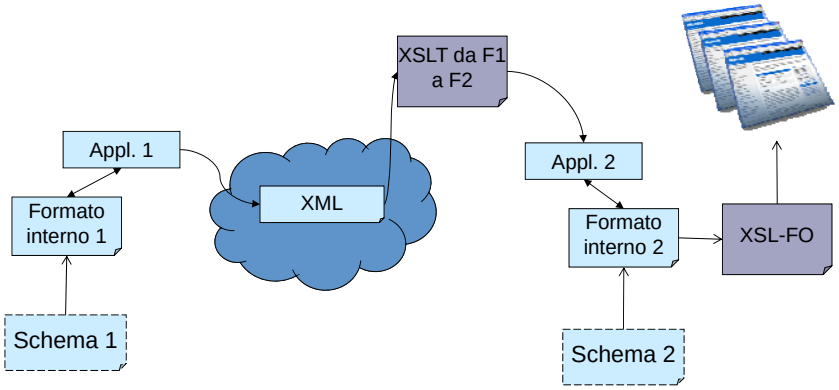
Aggiungere stile con XSL

- XHTML è un linguaggio XML che rappresenta la struttura di una pagina web. Lo stile viene aggiunto con CSS.
- Essendo un linguaggio XML, con XHTML si possono utilizzare tutti gli standard a disposizione per XML.
- In particolare esiste un'alternativa per la creazione di fogli di stile, le trasformazioni XSL (XSLT)

Cos'è XSL

- *Extensible Stylesheet Language* (XSL) è un linguaggio per fogli di stile per XML che permettere:
  - trasformazioni semplici o complesse della struttura XML
  - formattazione di un documento XML
- Una trasformazione, modifica un documento XML in un altro documento XML. Una formattazione impagina un documento XML (e quindi, in generale, il risultato non è un documento XML)
- È una raccomandazione del W3C e si divide in due parti:
  - una procedura di trasformazione (specifiche XSLT)
  - una procedura di formattazione (specifiche XSL o XSL-FO)
- La dicitura XSL è più generica di XSLT e può riferirsi sia ad un foglio di trasformazione che ad un foglio di stile

Applicazioni di XSL



Regole XSL

- Sono composte da due parti: un modello ed un'istruzione
- Il modello localizza il nodo al quale applicare l'istruzione
- L'istruzione descrive l'azione da intraprendere
- Per localizzare il nodo su cui agire, XSL utilizza lo standard XPath
  - È una raccomandazione W3C per la navigazione in documenti XML

Un passo indietro: XPath

- È una raccomandazione W3C (16 novembre 1999) usata come base da altri standard
  - XSL
  - XLink
  - XQuery
  - XPointer, etc
- È una sintassi per definire percorsi all'interno di un documento XML. Permette quindi di navigarci all'interno e di identificare frammenti contigui o non contigui
- Contiene un set di funzioni predefinite
- Usa una sintassi non XML
- XPath definisce due sintassi, una compatta ed una estesa. La prima è quella più utilizzata

Evoluzioni più recenti (23 gennaio 2007)

- XPath 2.0
  - generalizza il concetto di insieme di nodi restituendo non più questi insiemi ma sequenze (che quindi possono contenere anche cose diverse)
  - Introduce più di 90 funzioni per la manipolazione delle sequenze
- XSLT 2.0
  - rispetto alla prima versione permette di creare documenti multipli e funzioni definite dall'utente
- XQuery 1.0
  - linguaggio per effettuare query su documenti XML formattando il risultato
  - Fa fortemente uso del linguaggio XPath

Espressioni XPath

- Un'espressione XPath restituisce:
  - una selezione (un insieme) di nodi
  - un valore semplice
    - un booleano
    - una stringa
    - un numero
- Un *insieme di nodi* è una collezione di oggetti del documento XML.
- Le espressioni XPath si possono annidare: l'insieme di nodi risultante da una prima espressione XPath può essere il parametro di input per una seconda espressione

Terminologia

- XPath definisce 7 tipi di nodi:
  - elemento
  - attributo
  - testo
  - namespace
  - processing-instruction
  - commento
  - nodo documento
- I documenti XML sono visti come alberi, la cui radice è il nodo documento (o nodo radice)
- Valori atomici: nodi senza figli

Esempio

```
<?xml version="1.0" encoding="UTF-8"?>

<bookstore>
   <!– up to 2008-12-31 -->
   <book>
        <title lang="en">Harry Potter</title>
        <author>J K. Rowling</author>
        <year>2005</year>
        <price>29.99</price>
   </book>
</bookstore>
```

Nodo radice

Elemento radice

Nodo commento

Nodo attributo

Nodo testo

Nodo elemento

Relazioni tra i nodi

- Parent: ogni elemento ha un unico genitore
- Children
- Siblings (fratelli)
- Ancestors (antenati)
- Descendants (discendenti)

```
<bookstore>
  <book>
    <title lang="en">Harry Potter</title>
    <author>J K. Rowling</author>
    <year>2005</year>
    <price>29.99</price>
  </book>
</bookstore>
```


Il nodo contesto (o nodo corrente)

- Ogni passo di un'espressione di percorso va valutato sull'insieme di nodi da cui parte la computazione, detto nodo contesto o nodo corrente
- Il nodo contesto
  - è il nodo radice se si tratta del primo passo di un path assoluto
  - viene stabilito dall'applicazione se si tratta del primo passo di un path relativo
  - uno dei nodi restituiti dal passo precedente se non si tratta del primo passo di un path

Espressioni XPath

- Un'espressione Xpath è un percorso di locazione costruito da una sequenza di passi
- Ogni passo di locazione è composto da:
  - un asse di locazione
  - un test di nodo
  - zero o più predicati

/step/asse::test-di-nodo[predicato][predicato]/step/…

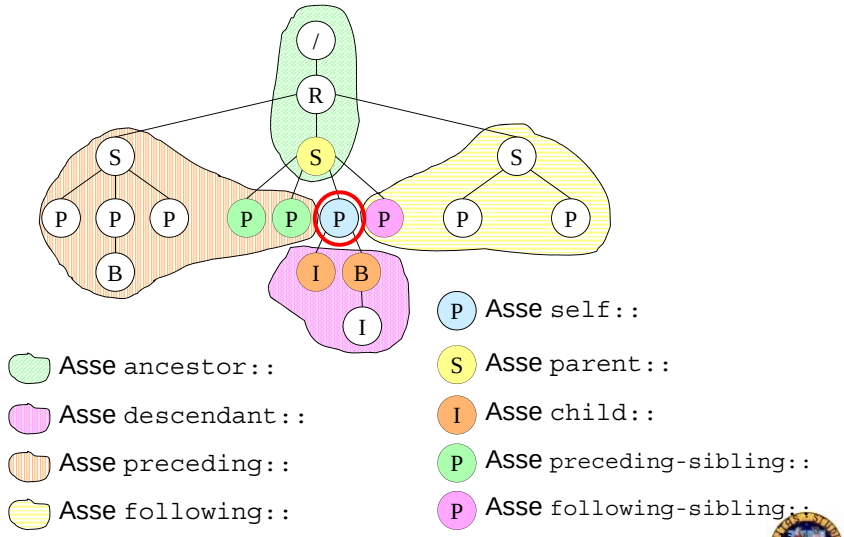
Gli assi Xpath – 1 (sintassi estesa)

- Un asse definisce un insieme di nodi in relazione al nodo corrente
- Possiamo dire che l'asse indica la direzione in cui cercare:
  - ancestor: seleziona tutti gli antenati (padre, nonno, etc) del nodo corrente
  - ancestor–or-self: seleziona tutti gli antenati (padre, nonno, etc) del nodo corrente e il nodo corrente
  - attribute: seleziona tutti gli attributi del nodo corrente
  - child: seleziona tutti i figli del nodo corrente
  - descendant: seleziona tutti i discendenti (figli, nipoti, etc) del nodo corrente
  - descendant-or-self: seleziona tutti i discendenti (figli, nipoti, etc) del nodo corrente e il nodo corrente

Gli assi Xpath – 2 (sintassi estesa)

- Un asse definisce un insieme di nodi in relazione al nodo corrente:
  - following: seleziona tutto ciò che, nel documento, segue il tag di chiusura del nodo corrente
  - following-sibling: seleziona tutti i fratelli, dopo il nodo corrente
  - namespace: seleziona tutti i nodi di tipo namespace del nodo corrente
  - parent: seleziona il padre del nodo corrente
  - preceding: seleziona tutto ciò che, nel documento, precede il tag di apertura del nodo corrente
  - preceding-sibling: seleziona tutti i fratelli, prima del nodo corrente
  - self: seleziona il nodo corrente

Gli assi Xpath - 3



Come esprimere un percorso nella sintassi estesa

- Un location path può essere assoluto o relativo. Un'espressione assoluta comincia con "**/**"
- Ogni espressione si compone di più passi (*location step*), divisi da "**/**" e letti da sinistra a destra
  - espressione assoluta: /step/step/…
  - espressione relativa: step/step/…
- Ogni step è valutato sull'insieme di nodi correnti e produce un nuovo insieme di nodi
- Ogni step può essere rappresentato così:

nomeasse::nodotest[predicato]

stabilisce la relazione con il nodo corrente	identifica un elemento in un asse	I predicati raffinano ulteriormente l'insieme di nodi

Esempi

- child::book
  - seleziona tutti i nodi book che siano figli del nodo corrente
- attribute::lang
  - seleziona l'attributo lang del nodo corrente
- child::node()
  - seleziona tutti i nodi figli del nodo corrente
- child::text()
  - seleziona il contenuto testuale del nodo corrente (seleziona tutti i nodi figli di tipo testuale del nodo corrente)

Selettori (sintassi compatta)

Espressione	Descrizione
Nome del nodo	Seleziona tutti i figli del nodo corrente con tale nome
/	Seleziona il nodo radice
//	Seleziona i nodi del documento, a partire dal nodo corrente che corrispondono alla selezione, non importa dove si trovino
.	Seleziona il nodo corrente
..	Seleziona il nodo padre
@	Seleziona gli attributi

Sintassi compatta

- In alcuni casi esistono delle forme abbreviate usabili invece della sintassi completa:
  - **`Child::x`** si può abbreviare con **`x`**
  - **`Attribute::a`** si può abbreviare con **`@a`**
  - **`Descendant-or-self::node()`** si può abbreviare con '**`//`**'
    - **`child::x/descendant-or-self::node()/y`** diventa **`x//y`**
    - **`/descendant-or_self::node()/y`** diventa semplicemente **`//y`**
    - **`child::x/attribute::k`** diventa semplicemente **`x/@k`**
  - **`Self::node()/`** si può abbreviare con '**`.`**',
  - **`Parent::node()/`** si può abbreviare con '**`..`**'

Esempi

- child::book → book
  - seleziona tutti i nodi book che siano figli del nodo corrente
- attribute::lang → @lang
  - seleziona l'attributo lang del nodo corrente
- child::node() → node()
  - seleziona tutti i nodi figli del nodo corrente
- child::text() → text()
  - seleziona il contenuto testuale del nodo corrente (seleziona tutti i nodi figli di tipo testuale del nodo corrente)

Esempio

```
<bookstore>
   <book>
         <title lang="en">Harry Potter</title>
         <author>J K. Rowling</author>
         <year>2005</year>
         <price>29.99</price>
   </book>
   <book>
         <title lang="it">Un cappello pieno di ciliege</title>
         <author>Oriana Fallaci</author>
         <year>2008</year>
         <price>20.00</price>
   </book>
</bookstore>
```

bookstore

/bookstore/

Esempio

```
<bookstore>
   <book>
         <title lang="en">Harry Potter</title>
         <author>J K. Rowling</author>
         <year>2005</year>
         <price>29.99</price>
   </book>
   <book>
         <title lang="it">Un cappello pieno di ciliege</title>
         <author>Oriana Fallaci</author>
         <year>2008</year>
         <price>20.00</price>
   </book>
</bookstore>
```

/bookstore/book

//book

Esempio

```
<bookstore>
   <book>
         <title lang="en">Harry Potter</title>
         <author>J K. Rowling</author>
         <year>2005</year>
         <price>29.99</price>
   </book>
   <book>
         <title lang="it">Un cappello pieno di ciliege</title>
         <author>Oriana Fallaci</author>
         <year>2008</year>
         <price>20.00</price>
   </book>
</bookstore>
```

/bookstore/book/title

//book/title

//title

Nodo di test

- Il nodo di test identifica attraverso il nome ed il tipo l'oggetto da restituire
- Il test può essere:
  - un nome di elemento (o attributo)
  - text(), processing-instruction(), comment()

Wildcard

- XPath definisce due test di nodo di tipo wildcard
  - node(): seleziona tutti i nodi, di qualunque tipo, nell'asse, inclusi commenti, testo e processing instruction
  - *: seleziona tutti i gli elementi nodi specifici definiti dall'asse

```
<bookstore>
  <!– up to 2008-12-31 -->
  <book>
      <title lang="en">Harry Potter</title>
      <author>J K. Rowling</author>
      <year>2005</year>
      <price>29.99</price>
  </book>
</bookstore>
```

/bookstore/node()
/bookstore/*

Predicati - 1

- I predicati sono sempre inseriti tra parentesi quadre
  - /bookstore/book[1]
- I predicati servono per selezionare un nodo specifico o nodi che contengono un valore dato
- Predicati:
  - /padre/nodo[**n**]: seleziona l'ennesimo elemento nodo figlio dell'elemento padre
  - /padre/nodo[**last()**]: seleziona l'ultimo elemento nodo figlio dell'elemento padre
  - /padre/nodo[**last() - 1**]: seleziona il penultimo elemento nodo figlio dell'elemento padre
  - /padre/nodo[**position()<n**]: seleziona i primi n-1 elementi nodo figli dell'elemento padre

Predicati - 2

- Altri predicati:
  - //nodo[**elem**]: seleziona tutti gli elementi nodo che contengono un elemento elem
  - //nodo[**@attr**]: seleziona tutti gli elementi nodo con un attributo attr
  - //nodo[**@attr='valore'**]: seleziona tutti gli elementi nodo con un attributo attr uguale a valore
  - //nodo[**@attr>n**]: seleziona tutti gli elementi nodo con un attributo attr con valore maggiore di n
  - //nodo[**@attr>n**]/figlio: seleziona tutti gli elementi figlio figli di elementi nodo con un attributo attr maggiore di n
  - ***: seleziona qualsiasi elemento nodo
  - @*: seleziona qualsiasi nodo di tipo attributo (qualsiasi attributo, ma ce ne deve essere almeno uno, ad esempio //nodo[@*] non seleziona i nodi privi di attributi)

Operatori

Operatore	Descrizione
|	Unisce due insiemi di nodi
+,-,*,div	Addizione, sottrazione, moltiplicazione, divisione
=, !=	Uguaglianza e disuguaglianza
>, >=, <, <=	
or, and	Operatori logici (restituiscono true o false)
mod	Modulo

Esempi

```
<bookstore>
   <book>
        <title lang="en">Harry Potter</title>
        <author>J K. Rowling</author>
        <year>2005</year>
        <price>29.99</price>
   </book>
   <book>
        <title lang="it">Un cappello pieno di ciliege</title>
        <author>Oriana Fallaci</author>
        <year>2008</year>
        <price>20.00</price>
   </book>
</bookstore>
```

//book/title[@lang="en"]

Esempi

```
<bookstore>
   <book>
        <title lang="en">Harry Potter</title>
        <author>J K. Rowling</author>
        <year>2005</year>
        <price>29.99</price>
   </book>
   <book>
        <title lang="it">Un cappello pieno di ciliege</title>
        <author>Oriana Fallaci</author>
        <year>2008</year>
        <price>20.00</price>
   </book>
</bookstore>
```

//book [price>20]/title

Esempi

```
<bookstore>
   <book>
        <title lang="en">Harry Potter</title>
        <author>J K. Rowling</author>
        <year>2005</year>
        <price>29.99</price>
   </book>
   <book>
        <title lang="it">Un cappello pieno di ciliege</title>
        <author>Oriana Fallaci</author>
        <year>2008</year>
        <price>20.00</price>
   </book>
</bookstore>
```

//book [year="2005" | year="2008"]/title

//book [year="2005" or year="2008"]/title

(//book [year="2005"] | //book[year="2008"])/title

Ricerca su più percorsi

- Con l'operatore | è possibile creare espressioni XPath che seguono percorsi diversi
- Esempio:
  - //book/title | //book/price: seleziona tutti i titoli e tutti i prezzi di tutti gli elementi book
  - //title | //price

Funzioni

- XPath 1.0 fornisce un insieme di funzioni su nodi, numeri e stringhe:
  - last(): restituisce il numero di fratelli del nodo corrente
  - position(): restituisce la posizione del nodo corrente
  - count(node-set): restituisce il numero di nodi di node-set
  - id(string): restituisce il nodo con id string (richiede uno schema)
  - Funzioni stringa: string-lenght(), concat(), starts-with(), contains(), etc
  - Funzioni booleane: boolean(), true(), false(), not()
  - Funzioni numeriche: sum(), number(), round(), floor(), ceiling()

http://www.w3schools.com/xpath/xpath_functions.asp

Esempio

I codici identificativi di tutte le divisioni:

//divisione/@id

```
<azienda nome="ACME, Inc">
    <divisione id="01">
        <descr> Prima divisione </descr>
        <manager>Den Raphaely
            <qualifica>manager</qualifica>
            <salario>10000</salario>
        </manager>
        <ufficio id="1"> <descr>Marketing<descr>
            <dipendente>…</dipendente>
        </ufficio>
        <reparto id="21">
            <dipendente>John Russel
                <qualifica>operaio</qualifica>
                <salario>10000</salario>
            </dipendete>
        </reparto>
    </divisione>
</azienda>
```


Esempio

Tutti gli uffici della divisione che compare per prima nell'elenco:

//divisione[1]/ufficio

```
<azienda nome="ACME, Inc">
    <divisione id="01">
        <descr> Prima divisione </descr>
        <manager>Den Raphaely
            <qualifica>manager</qualifica>
            <salario>10000</salario>
        </manager>
        <ufficio id="1"> <descr>Marketing<descr>
            <dipendente>…</dipendente>
        </ufficio>
        <reparto id="21">
            <dipendente>John Russel
                <qualifica>operaio</qualifica>
                <salario>10000</salario>
            </dipendete>
        </reparto>
    </divisione>
</azienda>
```


Attenzione

- L'espressione precedente si può anche scrivere in questo modo:
  - //divisione[position()=1]/ufficio
- Che differenza esiste tra le due espressioni di seguito?
  - //divisione[position()=3] [position()=1]/ufficio
  - //divisione[position()=1] [position()=3]/ufficio

Esempio

I dipendenti del primo ufficio di ogni divisione:

//divisione/ufficio[position()=1]/dipendente

```
<azienda nome="ACME, Inc">
    <divisione id="01">
        <descr> Prima divisione </descr>
        <manager>Den Raphaely
            <qualifica>manager</qualifica>
            <salario>10000</salario>
        </manager>
        <ufficio id="1"> <descr>Marketing<descr>
            <dipendente>…</dipendente>
        </ufficio>
        <reparto id="21">
            <dipendente>John Russel
                <qualifica>operaio</qualifica>
                <salario>10000</salario>
            </dipendete>
        </reparto>
    </divisione>
</azienda>
```


Esempio

I dipendenti di manager con salario superiore a 9000:

//divisione[manager/salario >9000]//dipendente

```
<azienda nome="ACME, Inc">
    <divisione id="01">
        <descr> Prima divisione </descr>
        <manager>Den Raphaely
            <qualifica>manager</qualifica>
            <salario>10000</salario>
        </manager>
        <ufficio id="1"> <descr>Marketing</descr>
            <dipendente>…</dipendente>
        </ufficio>
        <reparto id="21">
            <dipendente>John Russel
                <qualifica>operaio</qualifica>
                <salario>10000</salario>
            </dipendete>
        </reparto>
    </divisione>
</azienda>
```


Esempio

I dipendenti di Den Raphaely:

//divisione[manager/text()= "Den Raphaely"] //dipendente

```
<azienda nome="ACME, Inc">
    <divisione id="01">
        <descr> Prima divisione </descr>
        <manager>Den Raphaely
            <qualifica>manager</qual
            <salario>10000</salario>
        </manager>
        <ufficio id="1"> <descr>Marketing<descr>
            <dipendente>…</dipendente>
        </ufficio>
        <reparto id="21">
            <dipendente>John Russel
                <qualifica>operaio</qualifica>
                <salario>10000</salario>
            </dipendete>
        </reparto>
    </divisione>
</azienda>
```


Possibili errori con la sintassi compatta

- Quali sono le differenze tra le seguenti espressioni?
  - //reparto[1]
  - /descendant::reparto[1]
  - /descendant-or-self::node()/reparto[1]
- E tra queste?
  - /azienda/divisione[//ufficio]
  - /azienda/divisione[.//ufficio]

Regole XSL

- Sono composta da due parti: un modello ed un'istruzione
- Il modello localizza il nodo al quale applicare l'istruzione
- L'istruzione descrive l'azione da intraprendere
- Per localizzare il nodo su cui agire, XSL utilizza lo standard Xpath
- Browser che supportano XSLT
  - Firefox 3 supporta XML, XSLT e XPath
  - Internet Explorer 6 supporta XML, XSLT e XPath
  - Google Chrome 1 supporta XML, XSLT e XPath
  - Opera 9 supporta XML, XSLT e XPath
  - Safari 3 supporta XML e XSLT ma non Xpath
  - iOs supporta XML e XSLT dalla versione 3
  - Android supporta XML e XSLT dalla versione 2.1

Dichiarazione iniziale

- Per le trasformazioni:

```
<xsl:stylesheet version="1.0"
   xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
```

- Per le formattazioni

```
<xsl:transform version="1.0"
   xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
```

- Per aggiungere un foglio di trasformazione o di stile ad un file xml si usa il codice seguente:

```
<?xml-stylesheet type="text/xsl" href="fogliodistile.xsl"?>
```


<xsl:template>

- I fogli di stile XSL sono un insieme di chiamate a template
- Un template contiene le regole di formattazione da applicare ad un determinato nodo di un file XML

```
<xsl:template match="espressione XPath">
   …formattazione da applicare…
</xsl:template>
```

- Attributi: match, name, priority, mode

Caso di studio: trasformare XML in XHTML

```
<?xml version="1.0" encoding="UTF-8"?>
<catalog>
   <cd>
        <title>Empire Burlesque</title>
        <artist>Bob Dylan</artist>
        <country>USA</country>
        <company>Columbia</company>
        <price>10.90</price>
        <year>1985</year>
   </cd>
   <cd>...</cd>
</catalog>
```

Caso di studio: trasformare XML in XHTML

```
<?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> <body>
        <h2>My CD Collection</h2>
        <table border="1">
        <tr bgcolor="#9acd32"> <th>Title</th> <th>Artist</th>
        </tr>
        <tr> <td>...</td> <td>..</td> </tr>
        </table>
      </body> </html>
   </xsl:template>
</xsl:stylesheet>
```

<xsl:value-of>

- L'elemento <xsl:value-of> serve per estrarre il valore da un elemento XML ed inserirlo nell'output della trasformazione
- Anche in questo caso, l'elemento XML è selezionato tramite XPath
- Esempio:

```
<td><xsl:value-of select="catalog/cd/title"/></td>
<td><xsl:value-of select="catalog/cd/artist"/></td>
```

- Attributi: select, disable-output-escaping

<xsl:for-each>

- L'elemento <xsl:for-each> permette di scorrere gli elementi selezionati da un template. In pratica serve per creare dei loop
- Esempio:

```
<xsl:for-each select="catalog/cd">
      <tr> <td><xsl:value-of select="title"/></td>
           <td><xsl:value-of select="artist"/></td> </tr>
</xsl:for-each>
```

- Attributi: select
- È possibile filtrare l'output utilizzando espression XPath più complesse
  - <xsl:for-each select="catalog/cd[artist='Bob Dylan']">

<xsl:sort>

- L'elemento <xsl:sort> permette di ordinare l'output secondo alcuni criteri
- Viene inserito all'interno dei template, in particolare dentro gli elementi:
  - <xsl:for-each>
  - <xsl:apply-templates>
- Esempio:

```
<xsl:for-each select="catalog/cd">
      <xsl:sort select="artist"/>
      <tr> <td><xsl:value-of select="title"/></td>… </tr>
</xsl:for-each>
```

- Attributi: select, lang, data-type, order, case-order

<xsl:if>

- L'elemento <xsl:if> permette di creare un'istruzione condizionale
- È simile alle istruzioni if dei linguaggi di programmazione ma non esiste il ramo else
- Esempio:

```
<xsl:for-each select="catalog/cd">
      <xsl:if test="price &gt; 10">
      <tr> <td><xsl:value-of select="title"/></td>
         <td><xsl:value-of select="artist"/></td> </tr>
      </xsl:if>
</xsl:for-each>
```


<xsl:choose>

- L'elemento <xsl:choose> viene usato insieme a <xsl:when> e <xsl:otherwise> per esprime test condizionali multipli
- È simile all'istruzione switch dei linguaggi di programmazione
- L'istruzione <xsl:when> può comparire più volte e la clausola <xsl:otherwise> può essere omessa
- Attributi per <xsl:when>: test

Esempio

```
<xsl:for-each select="catalog/cd">
   <tr> <td><xsl:value-of select="title"/></td>
   <xsl:choose>
      <xsl:when test="price &gt; 10">
        <td bgcolor="#f0f"> <xsl:value-of select="artist"/></td>
      </xsl:when>
      <xsl:when test="price &gt; 9">
        <td bgcolor="#ccc"> <xsl:value-of select="artist"/></td>
       </xsl:when>
       <xsl:otherwise>
        <td><xsl:value-of select="artist"/></td>
       </xsl:otherwise>
   </xsl:choose> </tr> </xsl:for-each>
```


Invocazione delle regole

- Se un template ha un attributo name è possibile richiamarlo senza far uso del meccanismo di pattern matching tramite il tag call-template

```
<xsl:for-each select="catalog/cd">
   <xsl:call-template name="title"/>
   <xsl:call-template name="artist"/>
</xsl:for-each>
<xsl:template name="title">
   <xsl:value-of select="title"/>
</xsl:template>
<xsl:template name="artist">
   <xsl:value-of select="artist"/>
</xsl:template>
```


Template ricorsivi

- L'elemento <xsl:apply-templates> serve per applicare un template all'elemento corrente o ai nodi figli dell'elemento corrente
- Permette una definizione modulare della trasformazione
- Mentre l'elemento <xsl:for-each> permette di applicare una trasformazione in modo iterativo, <xsl:apply-templates> permette anche elaborazioni ricorsive: <xsl:apply-templates> può essere paragonato alla chiamata ricorsiva di funzione nei linguaggi di programmazione
- Attributi: select (per selezionare solo alcuni figli), mode

Template predefiniti

```
<xsl:template match="*|/">
   <xsl:apply-templates/>
</xsl:template>

<xsl:template match="text()|@*"  mode="#all">
   <xsl:value-of select="."/>
</xsl:template>

<xsl:template match="comment ()
                |processing-instruction()"  mode="#all"/>
```


Esempio - 1

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
   xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:output method='html' version='1.0' encoding='UTF-8'
   indent='yes'/>
<xsl:template match="/">
   <html>
    <body>
       <h2>My CD Collection</h2>
       <xsl:apply-templates/>
    </body>
   </html>
</xsl:template>
```


Esempio - 2

```
<xsl:template match="cd">
    <p>
        <xsl:apply-templates select="title"/>
        <xsl:apply-templates select="artist"/>
    </p>
</xsl:template>
<xsl:template match="title"> Title: <span style="color:#ff0000">
    <xsl:value-of select="."/></span> <br />
</xsl:template>
<xsl:template match="artist"> Artist: <span style="color:#0f0">
    <xsl:value-of select="."/></span>
</xsl:template>
</xsl:stylesheet>
```


Output

My CD Collection

Title: Empire Burlesque
Artist: Bob Dylan

Title: Hide your heart
Artist: Bonnie Tyler

Title: Greatest Hits
Artist: Dolly Parton

Title: Still got the blues
Artist: Gary Moore

Title: Eros
Artist: Eros Ramazzotti

Title: One night only
Artist: Bee Gees

Title: Sylvias Mother
Artist: Dr.Hook

Title: Maggie May
Artist: Rod Stewart

Title: Romanza
Artist: Andrea Bocelli

Title: When a man loves a woman
Artist: Percy Sledge

Esercizio

- Provare ad ottenere lo stesso output con l'elemento <xsl:for-each>

Soluzione

```
<xsl:template match="/">
    <html><body>
        <h2>My CD Collection</h2>
        <xsl:for-each select="catalog/cd">
        <p> Title: <span style="color:#ff0000">
                <xsl:value-of select="title"/></span> <br/>
            Artist: <span style="color:#0f0">
                <xsl:value-of select="artist"/></span>
        </p>
        </xsl:for-each>
    </body> </html>
</xsl:template>
</xsl:stylesheet>
```


Altro esempio: utilizzo dei modi

```
<?xml version="1.0" encoding="UTF-8"?>
<catalog>
   <cd>
        <title>Empire Burlesque</title>
        <artist>Bob Dylan</artist>
        <country>USA</country>
        <company>Columbia</company>
        <price>10.90</price>
        <year>1985</year>
   </cd>
   <cd>...</cd>
</catalog>
```

Altro esempio: utilizzo dei modi

```
<?xml version="1.0" encoding="UTF-8"?>
<index>
   <titles>
        <title>Empire Burlesque</title>
        <title>...</title>
   </titles>
   <artists>
        <artist>Bob Dylan</artist>
        <artist>...</artist>
   </artists>
</index>
```

Utilizzo dei modi - 1

```
<xsl:template match="/catalog">
   <index>
     <titles>
        <xsl:apply-templates mode="title" select="cd">
                <xsl:sort select="title" />
        </xsl:apply-templates>
     </titles>
     <artists>
        <xsl:apply-templates mode="artist" select="cd">
                <xsl:sort select="artist" />
        </xsl:apply-templates>
     </artists>
   </index>
</xsl:template>
```

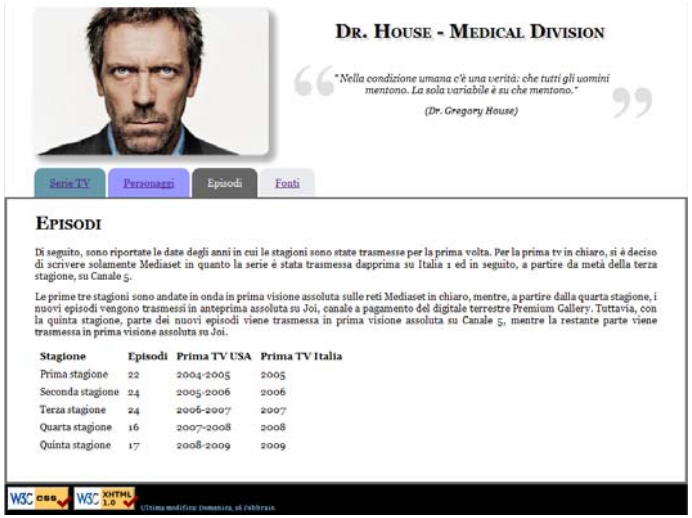
Utilizzo dei modi - 2

```
...
<xsl:template mode="artist" match="cd">
   <artist><xsl:value-of select="artist"/></artist>
</xsl:template>

<xsl:template mode="title" match="cd">
   <title><xsl:value-of select="title"/></title>
</xsl:template>
...
```

Utilizzo dei modi - 3

```
...
<xsl:template mode="artist" match="cd">
    <xsl:element name="artist">
        <xsl:value-of select="artist"/>
    </xsl:element>
</xsl:template>

<xsl:template mode="title" match="cd">
    <xsl:element name="title">
        <xsl:value-of select="title"/>
    </xsl:element>
</xsl:template>
...
```


Creazione di "template" per html



File XML - 1

```
<?xml version="1.0" encoding="UTF-8"?>

<pagina>
   <header/>
   <navigazione>
      <ul>
         <li id="posizione"><span>Serie TV</span></li>
         <li><span><a href="personaggi.html">Personaggi</a>
             </span></li>
         <li><span><a href="episodi.html">Episodi</a></span></li>
         <li><span><a href="fonti.html">Fonti</a></span></li>
      </ul>
   </navigazione>
```


File XML - 2

```
   <corpo>
         <h1>Dr. House - Medical Division</h1>

         <p><em>Dr. House - Medical Division (House, MD) </em> è
         una serie televisiva statunitense del 2004
         ambientata nel reparto di clinica medica dell'ospedale
         universitario fittizio Princeton-Plainsboro
         Teaching Hospital, situato nel New Jersey. La serie è
         incentrata sulle vicende di un'equipe di
         ...</p>
         ...
   </corpo>
   <piede/>
</pagina>
```


File XML - 3

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="trasforma.xsl"?>
<pagina>
   <header/>
   <navigazione>
      <ul>
         <li id="posizione"><span>Serie TV</span></li>
         <li><span><a href="personaggi.html">Personaggi</a>
             </span></li>
         <li><span><a href="episodi.html">Episodi</a></span></li>
         <li><span><a href="fonti.html">Fonti</a></span></li>
      </ul>
   </navigazione>
```


Foglio di trasformazione - 1

```
<xsl:template match="/">
<html xmlns="http://www.w3.org/1999/xhtml"  xml:lang="en" lang="en">
<head>
   <title>Dottor House - Serie Televisiva </title>
   <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
   <meta name="title" content="Dottor House - Serie Televisiva - Home
    page" />
   ...
   <link href="house1.css" rel="stylesheet" type="text/css"
    media="screen"/>
   <link href="house_print.css" rel="stylesheet" type="text/css"
    media="print"/>
</head>
```


Foglio di trasformazione - 2

```
<body>
   <xsl:apply-templates/>
</body>
</html>
</xsl:template>

<xsl:template match="pagina">
    <xsl:apply-templates/>
</xsl:template>
```


Foglio di trasformazione - 3

```
<xsl:template match="header">
    <div id="header">
      <img id="logo" src="img/house_title.jpg" alt="Attore che
   interpreta House (Hugh Laurie) "/>
      <h1>Dr. House - Medical Division</h1>
      <div id="citazione">
         <blockquote> <p>&quot; Nella condizione umana c'è una
   verità: che tutti gli uomini mentono. La sola variabile è su che
   mentono.&quot;</p> </blockquote>
         <cite>(Dr. Gregory House)</cite>
      </div>
   </div>
</xsl:template>
```


Foglio di trasformazione - 4

```
<xsl:template match="piede">
    <div id="piede">
      <img src="img/css.bmp" class="valid" alt="CSS Valid!"/>
      <img src="img/xhtml.bmp" class="valid" alt="XHTML 1.0 Valid!"/>
      Ultima modifica: <script type="text/javascript"
   src="ultima_modifica.js"></script>
   </div>
</xsl:template>
```


Foglio di trasformazione - 5

```
<xsl:template match="corpo">
    <div id="corpo">
      <xsl:for-each select="node()">
         <xsl:copy-of select="." />
      </xsl:for-each>
   </div>
</xsl:template>
<xsl:template match="navigazione">
    <div id="nav">
      <xsl:for-each select="node()">
         <xsl:copy-of select="." />
      </xsl:for-each>
   </div>
</xsl:template></xsl:stylesheet>
```


Copia di elementi

- Ci sono due tag preposti alla copia di elementi:
  - xsl:copy-of: crea una copia profonda dei nodi che hanno come radice i nodi elementi specificati da un attributo select
  - xsl:copy: crea una copia superficiale del nodo contesto, escludendo gli attributi
- Esempio: per aggiungere un attributo ad un nodo

```
<xsl:template match="nodo">
   <xsl:copy>
        <xsl:attribute name="nuovoAttributo">
              <xsl:value-of select="espressione" />
        </xsl:attribute>
        <xsl:copy-of select="*" />
    </xsl:copy>
</xsl:template>
```


Cenni al linguaggio XSL-FO

- XSL-FO (formatting objects) è un linguaggio per la formattazione di dati XML. Pur considerando anche lo schermo, si applica bene principalmente ai media paginati
- XSL-FO distingue tra le proprietà visive e uditive
- XSL-FO è una raccomandazione del dicembre 2006 che si basa su XPath 1 e 2
- XSL-FO non cambia l'albero del documento XML ma lo decora con proprietà di formattazione

Esempio

```
<?xml version="1.0" encoding="UTF-8"?>
   <fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
   <fo:layout-master-set>
        <fo:simple-page-master master-name="A4">
             <fo:region-body /> <!-- Template della pagina -->
        </fo:simple-page-master>
   </fo:layout-master-set>
   <fo:page-sequence master-reference="A4">
        <!-- Contenuto della pagina -->
        <fo:flow flow-name="xsl-region-body">
             <fo:block>Hello W3Schools</fo:block>
        </fo:flow>
   </fo:page-sequence>
</fo:root>
```


Riferimenti bibliografici

- Tutorial XPath W3C
  - http://www.w3schools.com/xpath/
- Tutorial XSLT W3C
  - http://www.w3schools.com/xsl/
- XSLT Elements Reference
  - http://www.w3schools.com/xsl/xsl_w3celementref.asp
- Libro
  - M. Anders, M. Schwartzbach, *Introduzione a XML*, Pearson, 2007