UNIVERSITÀ
DEGLI STUDI
DI MILANO

Trasformazioni di Formato

Paolo Ceravolo

paolo.ceravolo@unimi.it

Editoria Digitale

SESAR LAB

UNIVERSITÀ
DEGLI STUDI
DI MILANO

TRASFORMAZIONI

UNIVERSITÀ DEGLI STUDI DI MILANO

- La gestione di un workflow implica la gestione di passaggi di trasformazione da un formato a un altro
- Idealmente si parte da formati semplici e portabili per poter ottenere molteplici formati di destinazione
- Insieme al contenuto testuale è necessario gestire file multimediali e riferimenti ad altri documenti (include, citazioni, link)
- Le trasformazioni avvengono attraverso la rappresentazione di un formato in un modello intermedio (spesso resta solo nella memoria del programma di trasformazione) che poi può essere trasformato in un formato di destinazione.
  - Diventa quindi essenziale conoscere la struttura e l'espressività del modello intermedio di riferimento
  - Tipicamente i formati usati per i sorgenti dei contenuti sono meno espressivi o al più espressivi come il formato intermedio
  - I formati di destinazione potranno essere più espressivi del formato intermedio e potranno essere quindi necessari degli interventi manuali (es. formattazione tabelle)

XSLT

XML Stylesheet Language Transformation

XML STYLESHEET LANGUAGE

UNIVERSITÀ DEGLI STUDI DI MILANO

XSL è uno strandard per definire *fogli di stile* XML in modo da poter rappresentare i contenuti di un documento in **altri formati**

XSL è uno standard dal 2006, si appoggia su altri tre standard

- XSLT 1.0 (un linguaggio per la trasformazione di documenti XML) Recommendation dal 1999
- XPath (un linguaggio per identificare parti di un documento XML) Versione 1.0 Recommendation dal 1999
- XSL Formatting Objects (un vocabolario per formattare documenti XML) Inserito nella Recommendation dal 1999 di XSLT

PROCEDIMENTO

UNIVERSITÀ DEGLI STUDI DI MILANO

Il processore XSLT segue la struttura gerarchica di un documento XML, individua ciascun nodo grazie alle istruzioni (eventualmente ricorsive) contenute nel foglio di stile XSLT e ai percorsi XPath

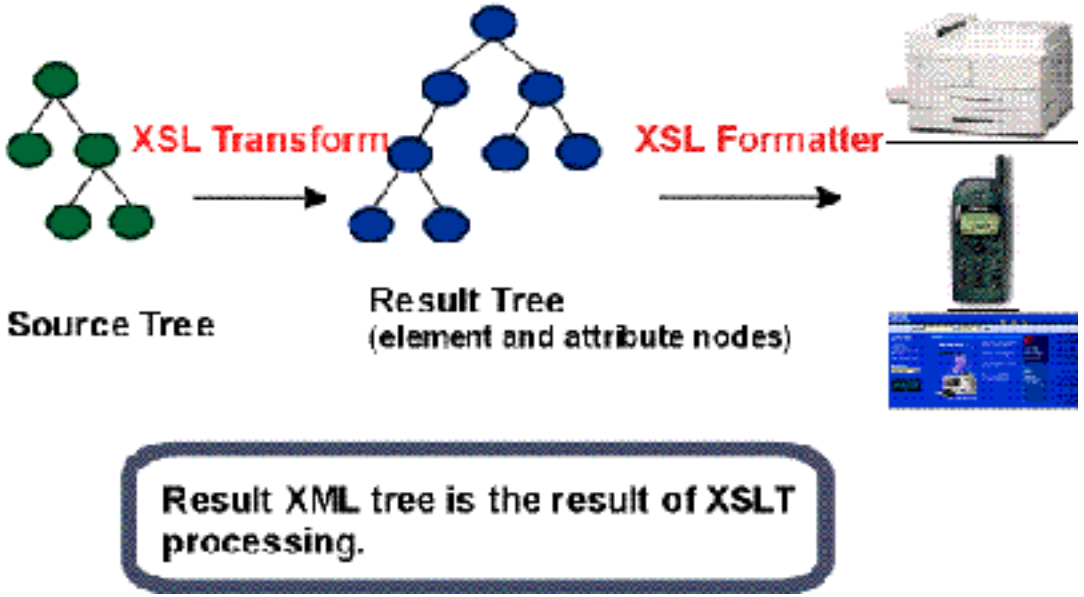
Quando il processore trova una corrispondenza, *matching*, al contenuto dell'elemento XML vengono applicate le regole, *template*, contenute nel foglio di stile

In pratica ci sono diversi usi:

- Filtrare ed ordinare dati XML
- Formattare i dati in base al loro valore
- Riorganizzare la struttura dei documenti
- Scrivere l'output verso differenti periferiche (carta, web, ebook, video, voce…)

FUNZIONI

Si definiscono due funzioni di XSL: Trasformazione e Formattazione



PROCESSORI

UNIVERSITÀ DEGLI STUDI DI MILANO

Esistono due processori XML che sono orami standard de facto: SAX (Simple API for XML) e DOM (Document Object Model)

- Differenze: modalità di interazione tra le API (Application Programming Interface) e l'applicazione che ne fa uso
- SAX invia all'applicazione eventi che descrivono il riconoscimento di un particolare elemento; in questo modo l'applicativo chiede solo gli elementi di cui ha bisogno

  - menoria, - tempo di caricamento, + tempo per singolo accesso

- DOM fornisce direttamente all'applicazione una descrizione ad albero dell'intero documento in oggetto

  + menoria, + tempo di caricamento, - tempo per singolo accesso

TECNOLOGIE XML

UNIVERSITÀ DEGLI STUDI DI MILANO

DOM (Document Object Model): definisce la rappresentazione in memoria (struttura ad albero), le interfacce e metodi per manipolare documenti XML

- libxml2 parser e toolkit per il linguaggio C
- Saxon processore per Java, JavaScript, .NET
- Xerces progetto Apache per C++ Java e Perl
- xslt-processor pacchetto per Node.js
- Lxml liberaria basata su libxml e libxslt per Python

SAX (Simple API for XML): è un'interfaccia per leggere e manipolare file XML attraverso eventi

- SAX progetto IBM per manipolazione di documenti XML
- Expat libreria stream per il linguaggio C
- Cocoon sever Apacher per gestione di XML, supporta pipeline XSLT

ESEMPIO DI TRASFORMAZIONE

UNIVERSITÀ DEGLI STUDI DI MILANO

- Partiamo da questo frammento

```
<div type="recipe" n="34">
<head>Pasta for beginners</head> <list>
<item>Pasta</item>
<item>Grated cheese</item>
</list>
<p>Cook the pasta and mix with the cheese</p>
</div>
```

- Per ottenere questo frammento

```
<html>
<h1>34: Pasta for beginners</h1> <p>Ingredients:
Pasta Grated cheese</p> <p>Cook the pasta and mix
with the cheese</p>
</html>
```

ESEMPIO DI TRASFORMAZIONE

UNIVERSITÀ DEGLI STUDI DI MILANO

- Usiamo un parser o processore XSLT
  - Con la libreria C *libxslt* da linea di comando possiamo eseguire una trasformazione in questo modo:

```
xsltproc simple.xsl origin.xml > dest.xml
```

  - Con Python possiamo importare l'opportuna libreria, i file richiesti e generare la trasformazione in questo modo:

```
from lxml import etree

### load input
dom = etree.parse('origin.xml')
transform = etree.XSLT(etree.parse('simple.xsl'))
### apply XSLT on loaded dom
s = str(transform(dom))
print(s)
```

ESEMPIO DI TRASFORMAZIONE

UNIVERSITÀ DEGLI STUDI DI MILANO

- Nel nostro esempio le regole di trasformazione incluse nel file XSLT potranno essere espresse come:

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

<xsl:template match="div">
   <html>
      <h1><xsl:value-of select="@n"/>: <xsl:value-of
select="head"/></h1>
      <p>Ingredients:<xsl:apply-templates select="list/
item"/></p>
      <p><xsl:value-of select="p"/></p>
   </html>
</xsl:template>

</xsl:stylesheet>
```

ESEMPIO DI TRASFORMAZIONE

UNIVERSITÀ DEGLI STUDI DI MILANO

- Con XSLT è possibile definire anche dei parametri che modificheranno la trasformazione
  - Con la libreria C *libxslt* da linea di comando abbiamo
    - *stringparam* per passare una copia nome del parametro e valore in formato stringa:

      ```
      xsltproc --stringparam someVariable Value template.xsl
      example.xml > output.xml
      ```

    - *param* per passare una copia nome del parametro e identificatore di un nodo:

      ```
      xsltproc --param anotherVariable /foo/bar template.xsl
      example.xml > output.xml
      ```

    - nel nostro esempio andranno uniti in questo modo:

      ```
      xsltproc --stringparam someVariable Value --param
      anotherVariable /foo/bar template.xsl example.xml >
      output.xml
      ```

DICHIARAZIONE

UNIVERSITÀ DEGLI STUDI DI MILANO

- L'elemento radice di un foglio di stile può essere `<xsl:stylesheet>` oppure `<xsl:transform>`
- Secondo lo standard del W3C, la dichiarazione corretta dovrebbe essere:

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://
www.w3c.org/1999/XSL/Transform">
```

- Per la versione 2 di XSLT, tipicamente supportata da Saxon, richiede una diversa dichiarazione. Nella dichiarazione è possibile inserire ulteriori namespace da usare nel foglio di stile.

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/
Transform" version="2.0"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:dc="http://purl.org/dc/elements/1.1">
```

RIFERIMENTO A XSL

UNIVERSITÀ DEGLI STUDI DI MILANO

- Secondo lo standard sarebbe necessario indicare nel file XML a quale file XSL esso fa riferimento

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

- Questa dichiarazione di fatto può essere omessa se l'applicazione su cui ci si basa riceve come parametri un XML di input, un XSL per la trasformazione e un output per il risultato

XSL:TEMPLATE

UNIVERSITÀ DEGLI STUDI DI MILANO

- Un foglio di stile XSL è formato da un insieme di template
- Ogni elemento `<xsl:template>` contiene delle regole da applicare quando uno specifico nodo viene trovato
- L'attributo `match` è usato per associare il template con un elemento XML utilizzando *XPath*

  `/` l'elemento radice

  `*` qualsiasi elemento

  `text()` il testo contenuto in un elemento

  `name` un elemento di nome name

  `@name` un attributo di nome name

- L'XSL che segue contiene un solo template e fornisce un output in HTML

ESEMPIO

UNIVERSITÀ DEGLI STUDI DI MILANO

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/
   Trasform">
<xsl:template match="/">
   <html> <body>
      <h2>La mia collezione di CD</h2>
      <table border="1">
      <tr bgcolor="#9acd32">
       <th align="left">Titolo</th>
       <th align="left">Autore</th>
      </tr>
      <tr>
       <td>.</td>
       <td>.</td>
      </tr>
      </table>
   </body></html>
</xsl:template>
</xsl:stylesheet>
```

associo un template XSLT, composto da tag HTML, all'elemento root del documento XML

DETTAGLI

UNIVERSITÀ DEGLI STUDI DI MILANO

- I documenti XSL sono documenti XML, quindi iniziano con la dichiarazione tipica:
  `<?xml version="1.0" encoding="ISO-8859-1"?>`
- L'elemento `<xsl:stylesheet>` definisce l'inizio del documento
- L'elemento `<xsl:template>` definisce l'inizio di un template
- L'attributo `match="/"` definisce la corrispondenza con l'elemento radice `/` del documento XML sorgente
- L'elemento `<xsl:value-of>` può essere usato per estrarre un valore da un elemento XML e aggiungerlo al flusso di output della trasformazione
- Nell'esempio che segue notiamo l'utilizzo di XPath al fine di identificare specifici elementi del file XML sorgente

ESEMPIO

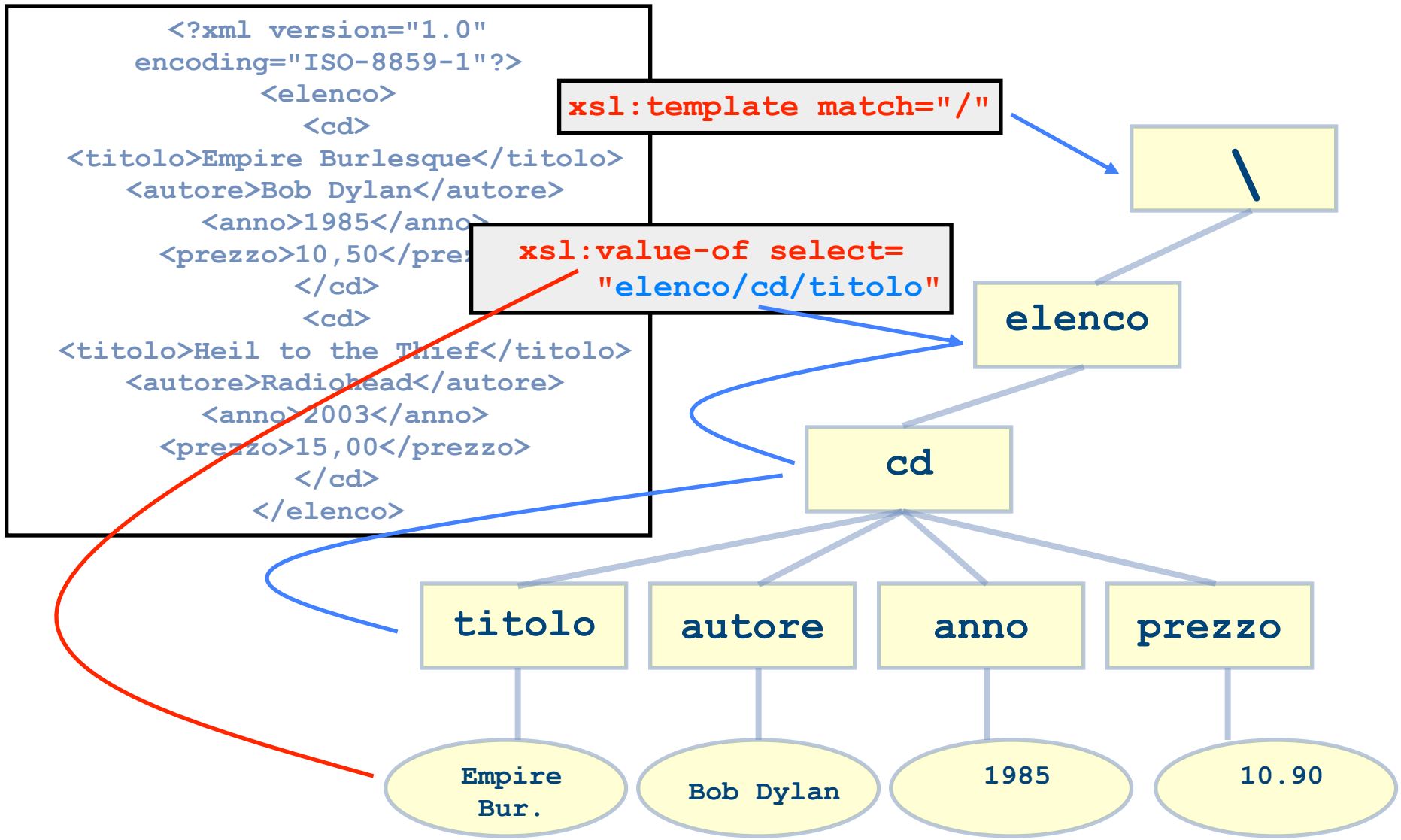
UNIVERSITÀ DEGLI STUDI DI MILANO

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0" xmlns="http://www.w3c.org/1999/XSL/
   Trasform">
<xsl:template match="/">
   <html> <body>
      <h2>La mia collezione di CD</h2>
      <table border="1">
      <tr bgcolor="#9acd32">
       <th align="left">Titolo</th>
       <th align="left">Autore</th>
      </tr>
      <tr>
       <td><xsl:value-of select="elenco/cd/titolo"/></td>
       <td><xsl:value-of select="elenco/cd/autore"/></td>
      </tr>
      </table>
   </body> </html>
</xsl:template>
</xsl:stylesheet>
```

percorso XPath all'interno dell'albero XML

NOZIONE DI NODO CORRENTE

UNIVERSITÀ DEGLI STUDI DI MILANO



OSSERVAZIONI

UNIVERSITÀ DEGLI STUDI DI MILANO

- Il risultato della valutazione di un'espressione XPath è un insieme di elementi, quindi zero, uno o più d'uno
- L'output è ancora un po' deludente in quanto viene visualizzato solo il primo cd

```
<html> <body>
  <h2>La mia collezione di CD</h2>
  <table border="1">
  <tr bgcolor="#9acd32">
   <th align="left">Titolo</th>
   <th align="left">Autore</th>
  </tr>
  <tr>
   <td>Empire Burlesque</td>
   <td>Bod Dylan</td>
  </tr>
  </table>
</body></html>
```

- PERCHE' VIENE VISUALIZZATO SOLO IL PRIMO CD?

XSL:FOR-EACH

UNIVERSITÀ DEGLI STUDI DI MILANO

- L'elemento `<xsl:for-each>` può essere usato per selezionare ogni elemento XML corrispondente ad un dato XPath
  - Quello che segue è il listato corretto relativo al nostro esempio
  - A questo punto tutti gli elementi essenziali sono stati aggiunti e l'output dovrebbe essere soddisfacente

ESEMPIO

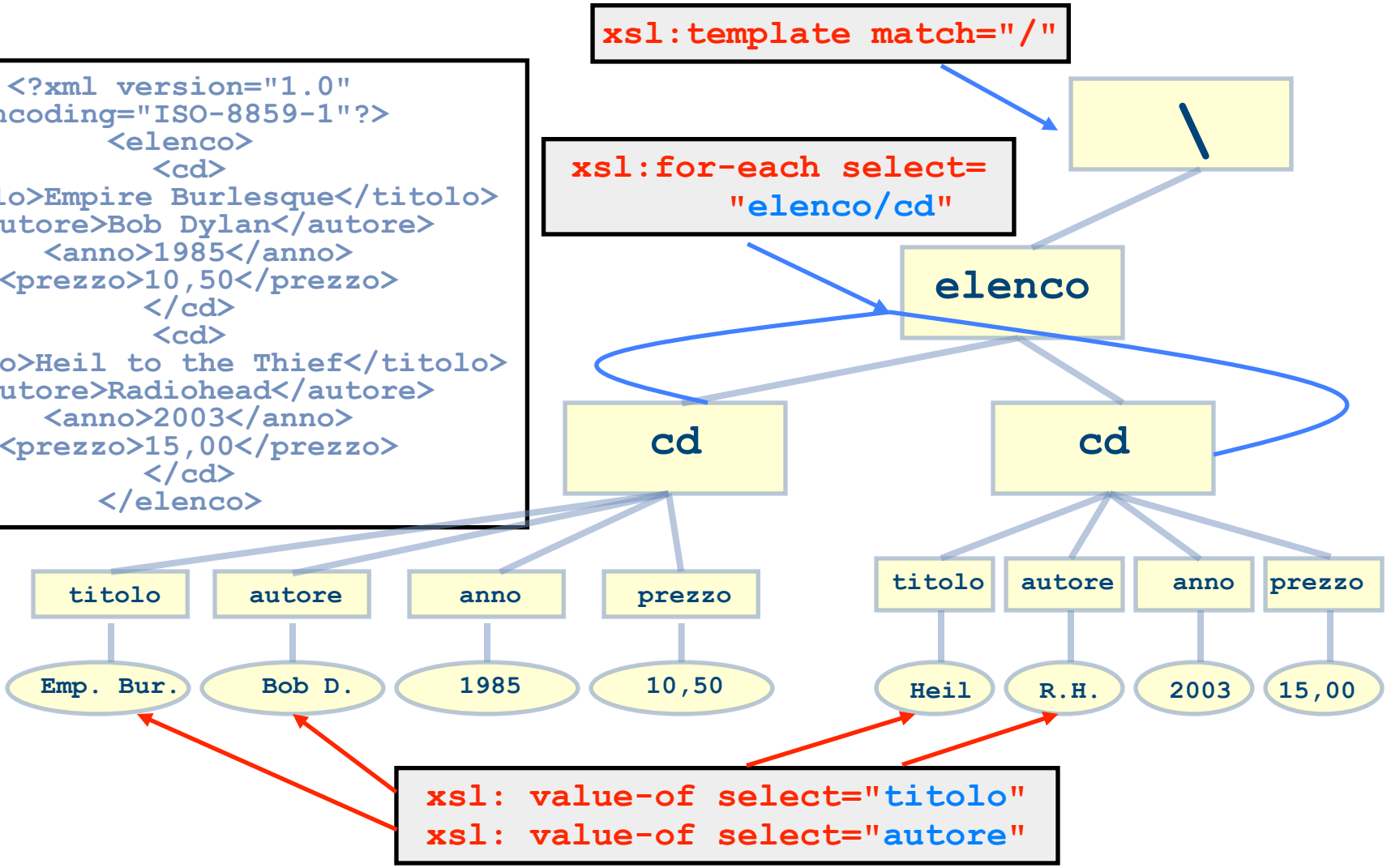
UNIVERSITÀ DEGLI STUDI DI MILANO

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0" xmlns=http://www.w3c.org/1999/XSL/
   Trasform>
<xsl:template match="/">
   <html> <body>
      <h2>La mia collezione di CD</h2>
      <table border="1">
      <tr bgcolor="#9acd32">
       <th align="left">Titolo</th>
       <th align="left">Autore</th>
      </tr>
      <xsl:for-each select="elenco/cd">
       <tr>
         <td><xsl:value-of select="titolo"/></td>
         <td><xsl:value-of select="autore"/></td>
       </tr>
      </xsl:for-each>
      </table>
   </body></html>
</xsl:template>
</xsl:stylesheet>
```

ESEMPIO

UNIVERSITÀ DEGLI STUDI DI MILANO

```
<?xml version="1.0"
encoding="ISO-8859-1"?>
<elenco>
<cd>
<titolo>Empire Burlesque</titolo>
<autore>Bob Dylan</autore>
<anno>1985</anno>
<prezzo>10,50</prezzo>
</cd>
<cd>
<titolo>Heil to the Thief</titolo>
<autore>Radiohead</autore>
<anno>2003</anno>
<prezzo>15,00</prezzo>
</cd>
</elenco>
```



FILTRARE L'OUPUT

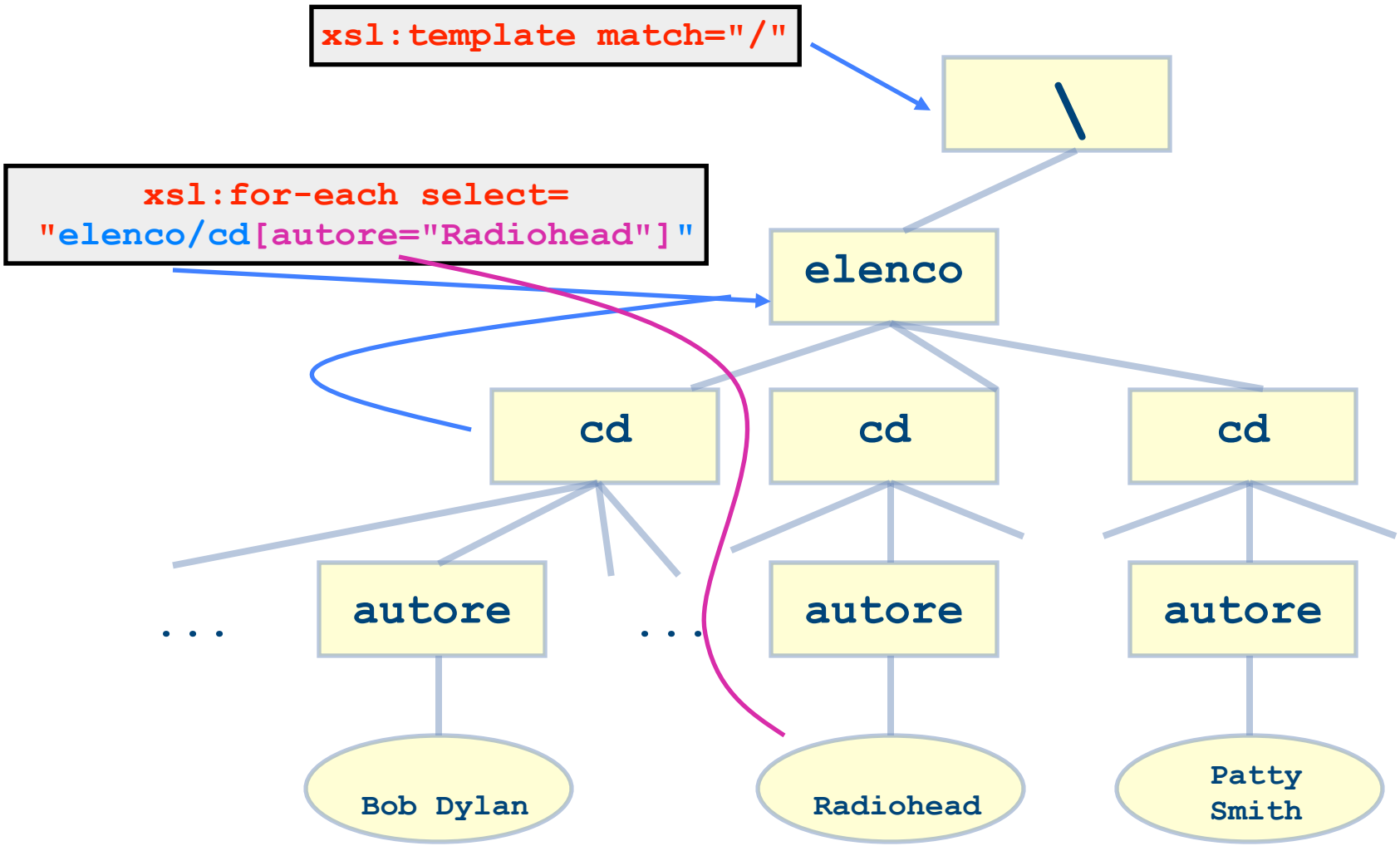
UNIVERSITÀ DEGLI STUDI DI MILANO

- Aggiungiamo qualche criterio di controllo (ancora come espressione XPath)

```
 <xsl:for-each select='elenco/
cd[autore="Radiohead"]'>
```

- Con questa condizione otteniamo in output solo i cd dei Radiohead

FILTRARE L'OUTPUT



XSL:SORT

UNIVERSITÀ
DEGLI STUDI
DI MILANO

- L'elemento `<xsl:sort>` serve per ordinare l'output
- L'output e l'ordinamento si ottenere in un unico passo inserendo l'elemento `<xsl:sort>`
- Quella che segue è la versione modificata del file XSL

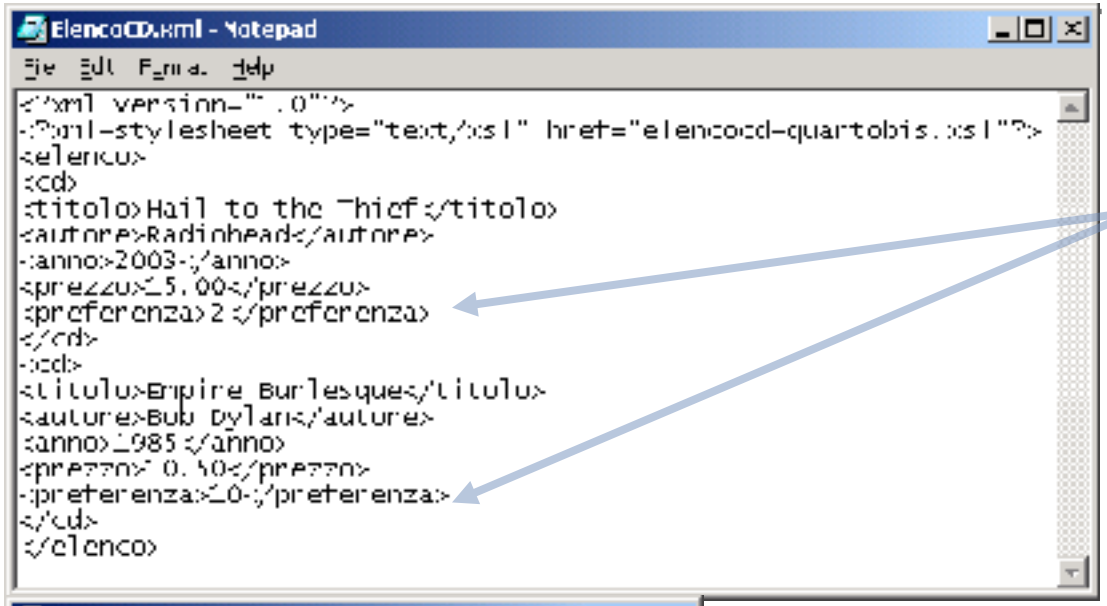
ESEMPIO

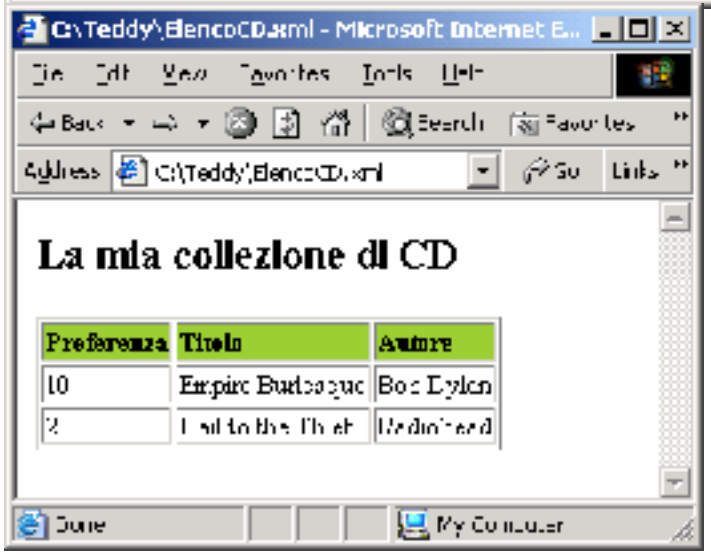
UNIVERSITÀ DEGLI STUDI DI MILANO

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<xsl:stylesheet version="1.0" xmlns=http://www.w3c.org/1999/XSL/Trasform>

<xsl:template match="/">

    <html> <body>

        <h2>La mia collezione di CD</h2>

        <table border="1">

        <tr bgcolor="#9acd32">

          <th align="left">Titolo</th>

          <th align="left">Autore</th>

        </tr>

        <xsl:for-each select="elenco/cd">

        <xsl:sort select="autore"/>

          <tr>

            <td><xsl:value-of select="titolo"/></td>

            <td><xsl:value-of select="autore"/></td>

          </tr>

        </xsl:for-each>

....
```

I valori verranno visualizzati ordinati per autore

PROBLEMA





L'output html ordina gli elementi **cd** secondo il valore dell'elemento **preferenza** interpretato come stringa di caratteri

SOLUZIONE: DATA-TYPE


```
<xsl:for-each select="elenco/cd">
  <xsl:sort data-type="number" select="preferenza"/>
  <tr>
    <td><xsl:value-of select="titolo"/></td>
    <td><xsl:value-of select="autore"/></td>
  </tr>
</xsl:for-each>
```

Ora l'ordinamento è corretto

XSL:IF

UNIVERSITÀ DEGLI STUDI DI MILANO

- L'elemento <xsl:if> contiene una serie di istruzioni che verranno eseguite solo se la condizione specificata è vera
- La condizione viene messa sotto forma di attributo e funziona come segue:

```
<xsl:if test="prezzo &gt; 10">
   qualche istruzione
</xsl:if>
```

- Otterremo in output i soli cd il cui prezzo è maggiore di 10

XSL:IF

UNIVERSITÀ DEGLI STUDI DI MILANO

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0">
<xsl:template match="/">
    <html> <body>
        <h2>La mia collezione di CD</h2>
        <table border="1">
        <tr bgcolor="#9acd32">
         <th align="left">Titolo</th>
         <th align="left">Autore</th>
        </tr>
        <xsl:for-each select="elenco/cd">
        <xsl:if test="prezzo &gt; 11">
         <tr>
           <td><xsl:value-of select="titolo"/></td>
           <td><xsl:value-of select="autore"/></td>
         </tr>
        </xsl:if>
        </xsl:for-each>
....
```

Solo i `cd` con valori degli elementi `prezzo` maggiori di 11 verranno visualizzati

XSL:CHOOSE

UNIVERSITÀ DEGLI STUDI DI MILANO

- L'elemento `<xsl:choose>` è usato insieme agli elementi `<xsl:when>` e `<xsl:otherwise>` per esprimere condizioni multiple
- La sintassi da usare è la seguente:

```
<xsl:choose>
   <xsl:when test="condizione">
     fai qualcosa
   </xsl:when>
   <xsl:otherwise>
     fai altro
   </xsl:otherwise>
</xsl:choose>
```

- Nell'esempio i cd di prezzo maggiore a 11, verranno visualizzati in colore diverso dagli altri

ESEMPIO

UNIVERSITÀ DEGLI STUDI DI MILANO

```
<xsl:for-each select="elenco/cd">
    <tr>
      <td><xsl:value-of select="autore"/></td>
      <xsl:choose>
        <xsl:when test="prezzo &gt; 11">
          <td bgcolor="#ff00ff">
            <xsl:value-of select="titolo"/>
          </td>
        </xsl:when>
        <xsl:otherwise>
          <td> <xsl:value-of select="titolo"/> </td>
        </xsl:otherwise>
      </xsl:choose>
    </tr>
```

ESEMPIO

UNIVERSITÀ DEGLI STUDI DI MILANO

Command Prompt

```
C:\Teddy>saxon ElencoCD.xml elencocd-sesto.xsl
<html xmlns:fo="http://www.w3.org/1999/XSL/Format">
   <body>
      <h2>La mia collezione di CD</h2>
      <table border="1">
         <tr bgcolor="#9acd32">
            <th align="left">Titolo</th>
            <th align="left">Autore</th>
         </tr>
         <tr>
            <td>Radiohead</td>
            <td bgcolor="#ff00ff">Hail to the Thief</td>
         </tr>
         <tr>
            <td>Bob Dylan</td>
            <td>Empire Burlesque</td>
         </tr>
      </table>
   </body>
</html>
```

VALORI DI UN ATTRIBUTO

UNIVERSITÀ DEGLI STUDI DI MILANO

- Nel caso in cui volessi partire da un sorgente come

```
<ref target="http://www.it.ox.ac.uk/">IT
Services</ref>
```

  - per ottenere l'output

```
<a href="http://www.it.ox.ac.uk/">IT Services</a>.
```

  - non possiamo definire il templare in questo modo

```
<xsl:template match="ref"> <a href="@target">
<xsl:apply-templates/> </a>
</xsl:template>
```

  - Perché nell'attributo `@href` sarebbe stampato il valore `'@target'`

VALORI DI UN ATTRIBUTO

UNIVERSITÀ DEGLI STUDI DI MILANO

- dovremo servirci delle { } per indicare che l'espressione deve essere valutata

```
<xsl:template match="ref"> <a
href="{@target}">

<xsl:apply-templates/> </a>

</xsl:template>
```

XSL:APPLY-TEMPLATES

UNIVERSITÀ DEGLI STUDI DI MILANO

- L'elemento `<xsl:apply-templates>` applica una regola di template all'elemento corrente oppure ai nodi figli dell'elemento corrente
- Se all'elemento `<xsl:apply-templates>` si aggiunge l'attributo select, le regole verranno applicate solamente ai nodi che corrispondono al valore dell'attributo

PUSHING E PULLING

UNIVERSITÀ DEGLI STUDI DI MILANO

- Due stili di parsing sono possono caratterizzare i fogli di stile XSLT
- **pull**: in questo caso le trasformazioni si organizzano attorno a un elemento principale (di solito corrispondente alla radice /) attraverso indicazioni specifiche, come `<xsl:for-each>` o `<xsl:value-of>`, è possibile definire trasformazioni per i suoi sottoelementi. Più semplice da capire ma vincola le trasformazioni alla struttura del documento sorgente
- **push**: in questo caso viene costruito un template diverso per ogni elemento, l'innestamento dei diversi template e la richiesta di applicazione tramite `<xsl:apply-templates>` produce il risultato complessivo. Più difficile visualizzare il risultato finale delle trasformazioni ma l'approccio è adatto a documenti in cui la struttura può variare

PUSHING E PULLING

UNIVERSITÀ DEGLI STUDI DI MILANO

- Ad esempio nella costruzione di una lista useremo l'approccio **pull** in questo modo

```
<xsl:template match="listPerson">
  <ul>
      <xsl:for-each select="person">
          <li> <xsl:value-of select="persName"/> </li>
      </xsl:for-each>
  </ul>
</xsl:template>
```

PUSHING E PULLING

UNIVERSITÀ DEGLI STUDI DI MILANO

- Useremo l'approccio **push** in questo modo

```
<xsl:template match="listPerson">
    <ul>
        <xsl:apply-templates select="person"/>
    </ul>
</xsl:template>
<xsl:template match="person">
    <li>
        <xsl:value-of select="persName"/>
    </li>
</xsl:template>
```

ESEMPIO

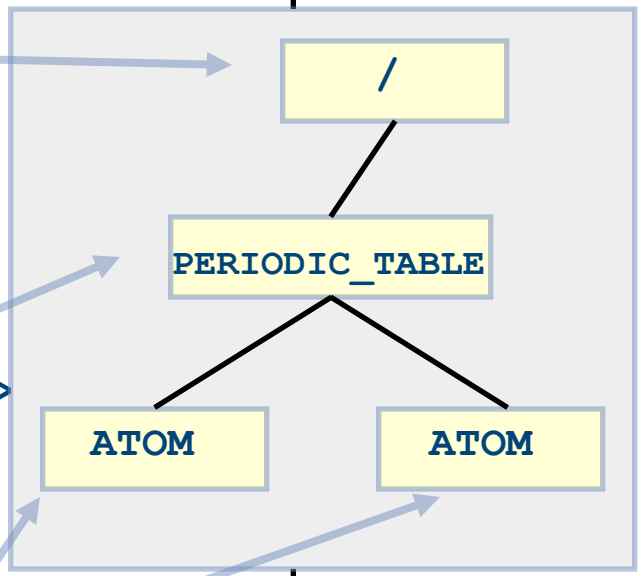
UNIVERSITÀ DEGLI STUDI DI MILANO

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xml" href="ptable-...xsl"?>
<PERIODIC_TABLE>
  <ATOM STATE="GAS">
    <NAME>Hydrogen</NAME>
    <SYMBOL>H</SYMBOL>
    <ATOMIC_NUMBER>1</ATOMIC_NUMBER>
    <ATOMIC_WEIGHT>1.00794</ATOMIC_WEIGHT>
    <BOILING_POINT UNITS="Kelvin">20.28</BOILING_POINT>
    <MELTING_POINT UNITS="Kelvin">13.81</MELTING_POINT>
  </ATOM>
  <ATOM STATE="GAS">
    <NAME>Helium</NAME>
    <SYMBOL>He</SYMBOL>
    <ATOMIC_NUMBER>2</ATOMIC_NUMBER>
    <ATOMIC_WEIGHT>4.0026</ATOMIC_WEIGHT>
    <BOILING_POINT UNITS="Kelvin">4.216</BOILING_POINT>
    <MELTING_POINT UNITS="Kelvin">0.95</MELTING_POINT>
  </ATOM>
</PERIODIC_TABLE>
```

ESEMPIO

UNIVERSITÀ DEGLI STUDI DI MILANO

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

<xsl:template match="PERIODIC_TABLE">
<h1>Tavola Periodica</h1>
<xsl:apply-templates/>
</xsl:template>

<xsl:template match="ATOM">
<P>
<xsl:value-of select="."/>
</P>
</xsl:template>
</xsl:stylesheet>
```



ESEMPIO

UNIVERSITÀ DEGLI STUDI DI MILANO

Select Command Prompt

```
C:\>java org.apache.xalan.xslt.Process -in Teddy\Ptable.xml -xsl Teddy\ptable-pr
imo.xsl
<html xmlns:fo="http://www.w3.org/1999/XSL/Format">
<body>
<h1>Tavola Periodica</h1>

<P>
    Hydrogen
    H
    1
    1.00794
    20.28
    13.81
  </P>

<P>
    Helium
    He
    2
    4.0026
    4.216
    0.95
  </P>

</body>
</html>
```

Command Prompt

```
C:\Teddy>saxon Ptable.xml ptable-primo.xsl
<html xmlns:fo="http://www.w3.org/1999/XSL/Format">
   <body>
      <h1>Tavola Periodica</h1>

      <P>
            Hydrogen
             H
             1
             1.00794
             20.28
             13.81

      </P>

      <P>
             Helium
             He
             2
             4.0026
             4.216
             0.95

      </P>

   </body>
</html>
```

XSL:CALL-TEMPLATE

UNIVERSITÀ DEGLI STUDI DI MILANO

- Un elemento `xsl:call-template` invoca un template identificandolo attraverso il nome
- Diversamente da `xsl:apply-templates`, `xsl:call-template` NON cambia il nodo corrente

```
<xsl:call-template name='nome template'>
```

XSL:COPY-OF & XSL:VARIABLE

UNIVERSITÀ DEGLI STUDI DI MILANO

- L'elemento `xsl:copy-of` può essere usato per copiare un insieme di nodi nel l'albero risultante senza che vengano applicate conversioni a stringe come accadrebbe se si usasse `xsl:value-of`

```
<xsl:copy-of select=”espressione”>
```

- L'elemento `xsl:variable` può essere usato per definire una variabile globale ad uno stylesheet (se definito come elemento di primo livello) oppure una variabile locale ad un contesto (se definita a livello inferiore)

```
<xsl:variable name="publishers" select="//
publisher>

...

<xsl:for-each select="$publishers">
```

RIFERIMENTI INCROCIATI

UNIVERSITÀ DEGLI STUDI DI MILANO

- Esiste un meccanismo per questo: gli attributi ID, IDREF e IDREFS definiti da un DTD, ed utilizzabili grazie alla funzione id di XPath. Questo meccanismo ha diversi limiti, tuttavia:
  - deve esistere un DTD esterno
  - solo un attributo puo' essere definito di tipo ID in un documento
  - Un elemento non può avere più di un ID e un particolare ID può essere associato ad un solo elemento

CHIAVI XSL

UNIVERSITÀ DEGLI STUDI DI MILANO

- Una chiave è composta da una tripla contenente:
  - il nodo che possiede la chiave `match`;
  - il nome della chiave `name`;
  - il valore della chiave `use`.
- Le chiavi XSL, definite con `<xsl:key name="..." match="..." use="..."/>`, sono un modo per gestire riferimenti incrociati tra elementi di un documento XML senza necessità di uno schema

FUNZIONE KEY

UNIVERSITÀ DEGLI STUDI DI MILANO

- La funzione key si esprime con la seguente sintassi: `node-set key(string, object)`
  - L'argomento string: il nome di una chiave definita da un elemento `xsl:key`
  - L'argomento object: viene convertito dalla funzione string in valore alfanumerico
  - Viene restituito come valore l'insieme dei nodi che hanno una chiave dal valore use uguale al risultato della conversione di `object`
- Esempio: `key('idkey', @id)`

XSLT 2.0

UNIVERSITÀ DEGLI STUDI DI MILANO

- Nel Gennaio 2007 XSLT 2.0 è stato promosso a recommendation W3C
- Le principali novità introdotte sono:
  - istruzioni per raggruppare gli elementi
  - istruzioni per gestire il testo tramite espressioni regolari
  - istruzioni per gestire la formattazione di numeri e date
  - istruzioni per generare documenti

XSL:FOR-EACH-GROUP

UNIVERSITÀ DEGLI STUDI DI MILANO

```
<files>

    <file nome="tanto.xml" versi="16" autore="Dante-Alighieri" />

    <file nome="voidonne.xml" versi="16" autore="Dante-Alighieri" />

    <file nome="lapo.xml" versi="16" autore="Dante-Alighieri" />

    <file nome="canz1.xml" versi="68" autore="Ariosto" />

    <file nome="son3.xml" versi="16" autore="Ariosto" />

    <file nome="canz1.xml" versi="14" autore="Lorenzo-DeMedici" />

</files>
```

XSL:FOR-EACH-GROUP

UNIVERSITÀ DEGLI STUDI DI MILANO

```
<xsl:template match="files">
  <xsl:for-each-group select="file" group-by="@autore">

    <xsl:for-each select="current-group()">
      <xsl:value-of select="@name"/>, <xsl:value-of
select="@versi"/>
      <xsl:text></xsl:text>
    </xsl:for-each>

    <xsl:text>Media di versi per  </xsl:text>
    <xsl:value-of select="current-grouping-key()"/>
    <xsl:text> uguale: </xsl:text>
    <xsl:value-of select="avg(current-group()/@size)"/>
    <xsl:text>

</xsl:text>
  </xsl:for-each-group>
</xsl:template>
```

XSL:FOR-EACH-GROUP

UNIVERSITÀ DEGLI STUDI DI MILANO

```
tanto.xml, 16
voidonne.xml, 16
lapo.xml, 16
Media di versi per Dante-Alighieri uguale: 16

can1.xml, 68
son3.xml, 16
Media di versi per Ariosto uguale: 42

canz1.xml, 14
Media di versi per Ariosto uguale: 14
```

XSL:DOCUMENT

UNIVERSITÀ DEGLI STUDI DI MILANO

```
<xsl:template match="/">
<xsl:document method="html" encoding="UTF-8" href="index.html">
       <html>
         <head>
           <title><xsl:value-of select="/poesia/verso[1]"/></title>
         </head>
         <body>
   <h1 align="center"><xsl:value-of select="/poesia/verso[1]"/></h1>
           <div class=”/poesia”>
    <xsl:for-each select="/poesia">
    <xsl:copy-of select="."></xsl:copy-of>
    </xsl:for-each>
           </div>
           <hr/>
         </body>
       </html>
     </xsl:document>
</xsl:template>
```

PROGETTI LEGATI A XSLT

UNIVERSITÀ DEGLI STUDI DI MILANO

- TEI XSL Stylesheets
  - una famiglia di fogli di stile XSLT 2.0 per trasformare i documenti XML TEI in vari formati, tra cui XHTML, LaTeX, XSL Formatting Objects, ePub, testo semplice, RDF, JSON; e da/verso Word OOXML (docx) e OpenOfice (odt)

    https://github.com/TEIC/Stylesheets

    https://github.com/TEIC/Jenkins
- TEIGarage
- un servizio web e RESTful per trasformare, convertire e validare vari formati, con particolare attenzione al formato TEI

  https://teigarage.tei-c.org/

  https://github.com/TEIC/TEIGarage
- XSLTJSON
  - per trasformare XML in JSON

    https://github.com/bramstein/xsltjson
- DocBook XSL
  - DocBook è una raccolta di standard e strumenti per la pubblicazione tecnica

    http://www.sagehill.net/docbookxsl/preface.html#WhatIsDocbook