Tecnologie Cloud e Mobile

Lez. 15

XML Avanzato

Giuseppe Psaila

Università di Bergamo

giuseppe.psaila@unibg.it

Namespace

Namespace: Spazio dei Nomi

- Un «Namespace» raccoglie una serie di nomi o simboli
- Il concetto viene usato in molti ambiti, per esempio nel C++
- In XML, un namespace definisce elementi specifici
- Per poter usare questi elementi, occorre indicare a quale namespace appartengono

Prefisso del Namespace

- Ogni namespace usato nel documento ha un prefisso
- Il prefisso deve precedere il nome dell'elemento
- Prefisso e nome sono separati da «:»
- Esempio (da specifica SOAP)

`soap:Envelope`

prefisso`:`nome elemento

Prefisso e Namespace

- Ma i prefissi vanno definiti
- La prima volta che un prefisso viene usato, occorre dire a quale namespace appartiene
- Con uno strano attributo:
  `xmlns:`*prefisso*
- Esempio:

```
xmlns:soap="http://www.w3.org/2003/05/soap-envelope/"
```

xmlns:*prefisso* e suo valore

- Questo attributo dice che il *prefisso* verrà usato da lì in avanti
- Ma che valore ha?
- Il valore è l'URI dello standard
- URI: Uniform Resource Identifier
  Si tratta dell'evoluzione del concetto di URL (Uniform Resource Locator) per **identificare** una risorsa, non per trovarla

URI

- Ma un URI ha la stessa forma di un URL
- L'idea è questa:
  - Il processore analizza il documento
  - Vede l'URI associato a **`xmlns:`**
  - Se lo conosce, è in grado di processare gli elementi appartenenti a quel namespace
  - Se non li conosce, li scarta/ignora

Perché i Namespace?

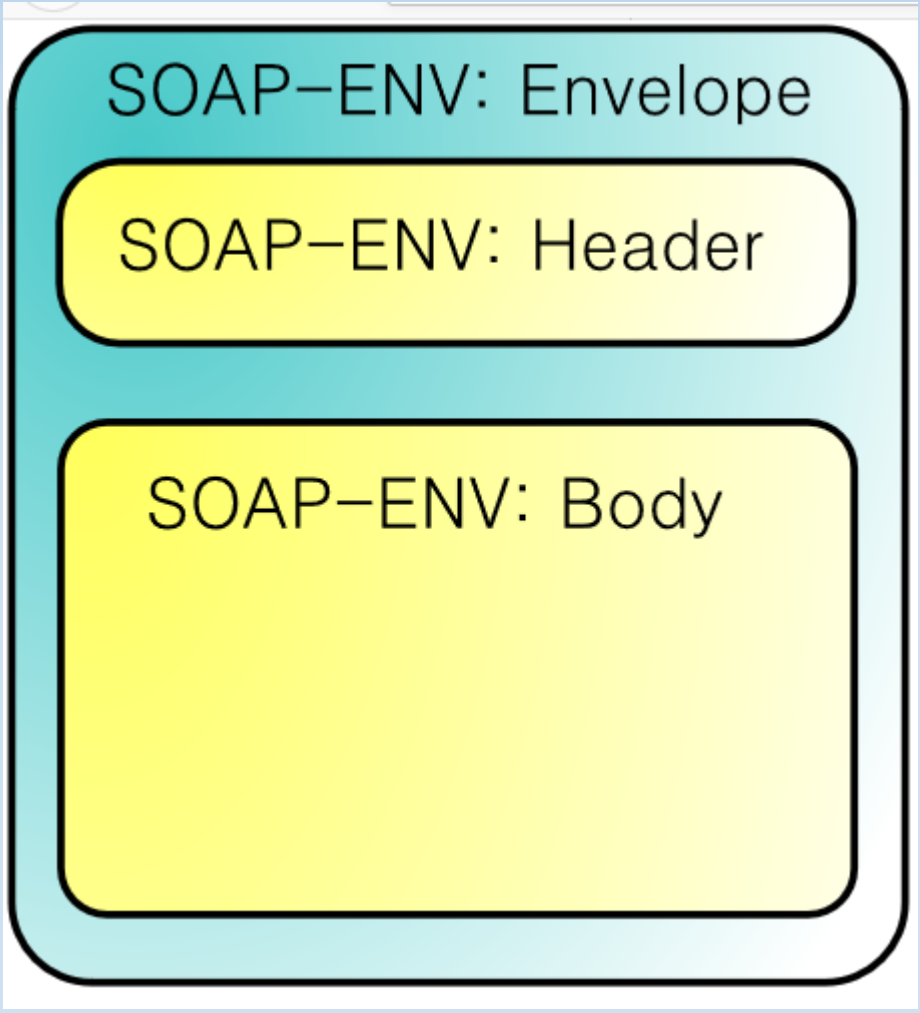
- Perché così si possono integrare nello stesso documento elementi appartenenti a namespace diversi
- Cosa impossibile da fare con il DTD
- Inoltre, si può dichiarare esplicitamente a quale definizione del documento si fa riferimento

Esempio: SOAP

- Simple
  Object
  Application
  Protocol
- È un protocollo del W3C usto per lo scamabio di messaggi tra sistemi informativi
- I messaggi sono documenti XML

Messaggi SOAP

- Envelope
  che contiene
  - Header
  - Body



SOAP BODY

- Nel corpo del messaggio viaggia il contenuto effettivo della comunicazione
- Ma il formato da inviare non è relativo al protocollo SOAP, che è generico
- Un po’ come la busta della posta tradizionale: è neutra rispetto al contenuto

Esempio SOAP

```
<soap:Envelope xmlns:soap="http://...">
 <soap:Body>
  <getProductDetailsResponse xmlns=
     "http://magazzino.example.com/ws">
…
```

Esempio SOAP

```
…
   <getProductDetailsResponse xmlns=
    "http://magazzino.example.com/ws">
    <getProductDetailsResult>
     <productName>Matita</productName> …
    </getProductDetailsResult>
   </getProductDetailsResponse>
 </soap:Body>
</soap:Envelope>
```

XmlNs Senza Prefisso

- Definisce il namespace per gli elementi senza prefisso
- Serve per dire a quale specifica/standard fanno riferimento

Ricapitoliamo

- Il Server del destinatario riceve un messaggio SOAP
- L’handler del protocollo SOAP riceve il contenuto XML del messaggio
- Sa gestire i suoi elementi, non gli altri
- Estrae il frammento nel body e lo passa al componente software del sistema informativo, che è in grado di processarlo

XSLT e FO

Alle Origini

- XML nasce per descrivere i contenuti, non il modo in cui questi vengono presentati
- Però potrebbe dover succedere di dover presentare i contenuti
- Idea: definire il concetto di Foglio di Stile

Foglio di Stile

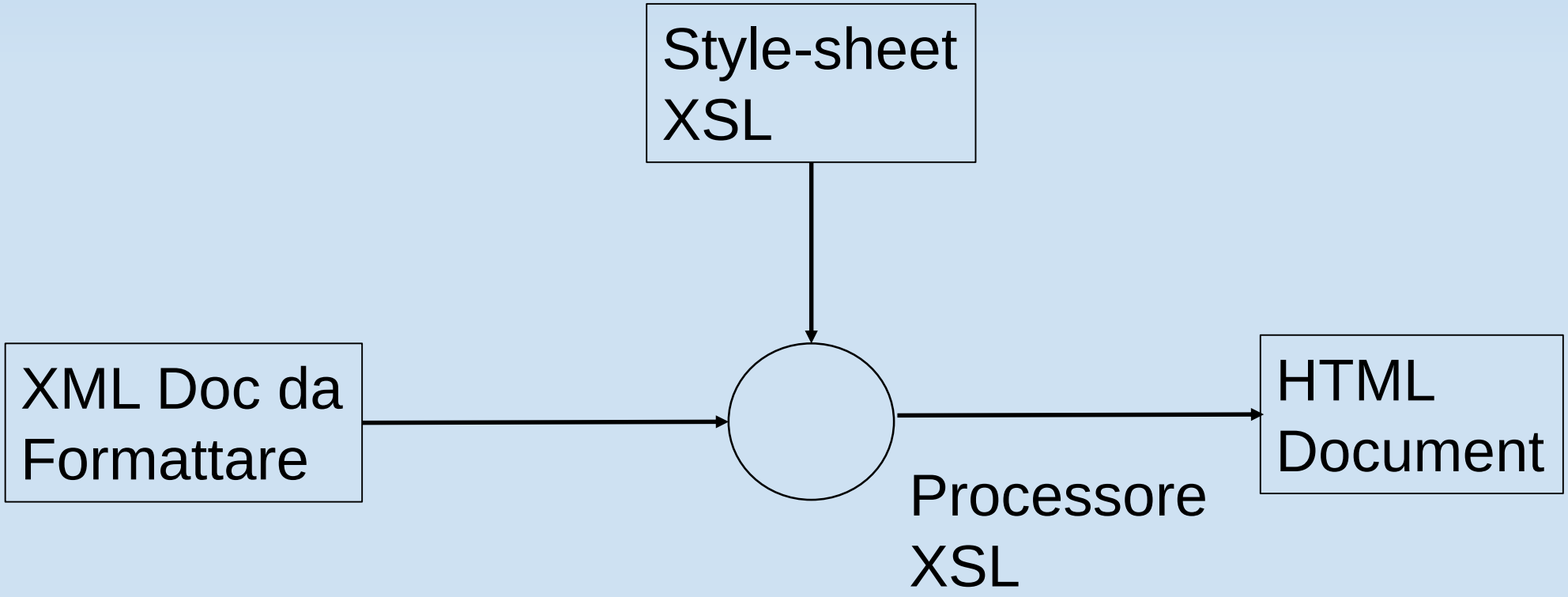
- Il «Foglio di Stile» (Style-sheet) contiene le regole per dare uno stile al documento
- Che linguaggio usare?
- Ovviamente XML
- Nasce XSL, «eXtensible Style-sheet Language»

Idea Dietro XSL

- Un insieme di elementi che descrivono come formattare gli elementi di altri documenti XML
- Ottenendo la versione HTMNL del documento originale

XML per Processare XML?

- Esatto, un documento XML specifica come processare un altro documento XML



A un Certo Punto

- Il W3C ha cambiato approccio
- O meglio, non lo ha cambiato, lo ha esteso
- Perché limitarsi a «formattare» un documento XML in HTML?
- Le necessità reali sono molto più varie:
  - Ristrutturare (trasformare) documenti XML in altri documenti XML
  - Generare documenti HTML come trasformazioni complesse di documenti XML

Nasce XSLT

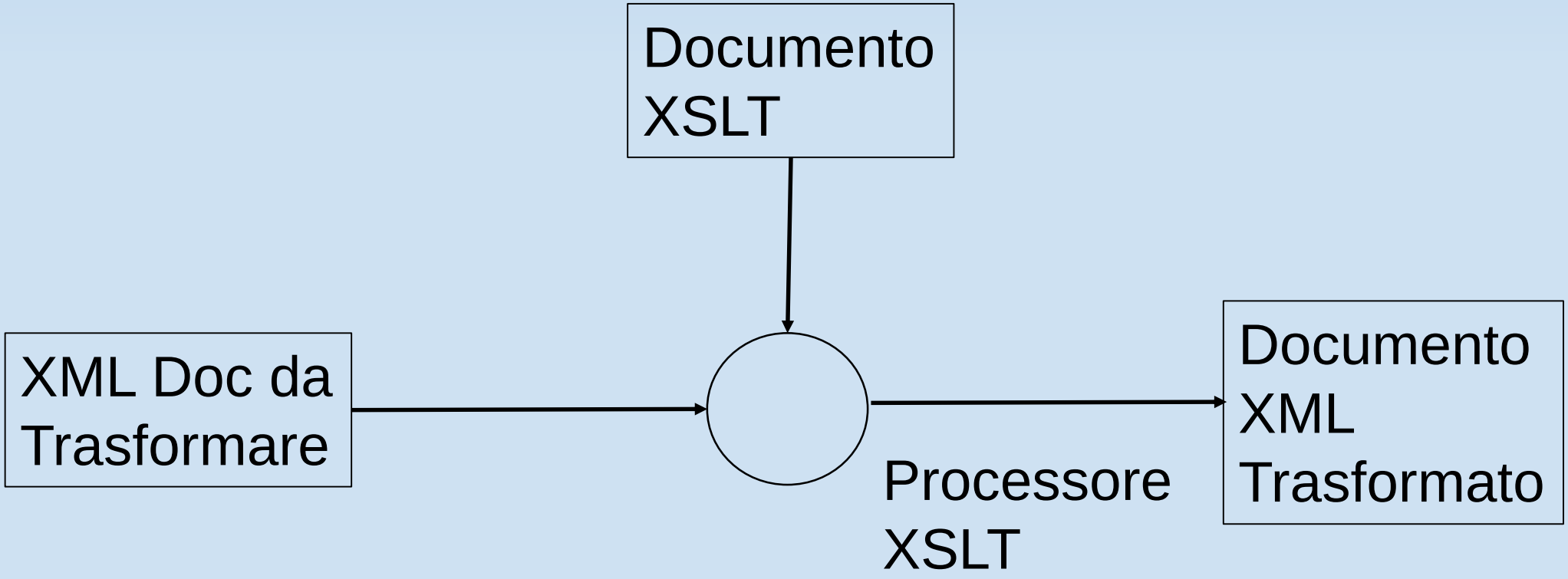
- La prima specifica di XSL viene rimossa
- Nasce XSLT
  eXtensible
  Style-sheet
  Language
  Transformation

Idea Dietro XSLT

- Rispetto a XSL, XSLT non è più un semplice foglio di stile
- È un vero e proprio linguaggio dichiarativo per specificare trasformazioni complesse dei documenti XML
- Ovviamente, usando la sintassi di XML

XML per Processare XML?

- Ancora vero, ma adesso per ottenere nuovi documenti XML



Scopo

- Ma perché trasformare un documento XML in un altro documento XML?
- Per fare adattamenti di formato
- Per sfruttare altre tecnologie XML
- Esempio:
  l Formatting Objects (o FO)

I Formatting Objects

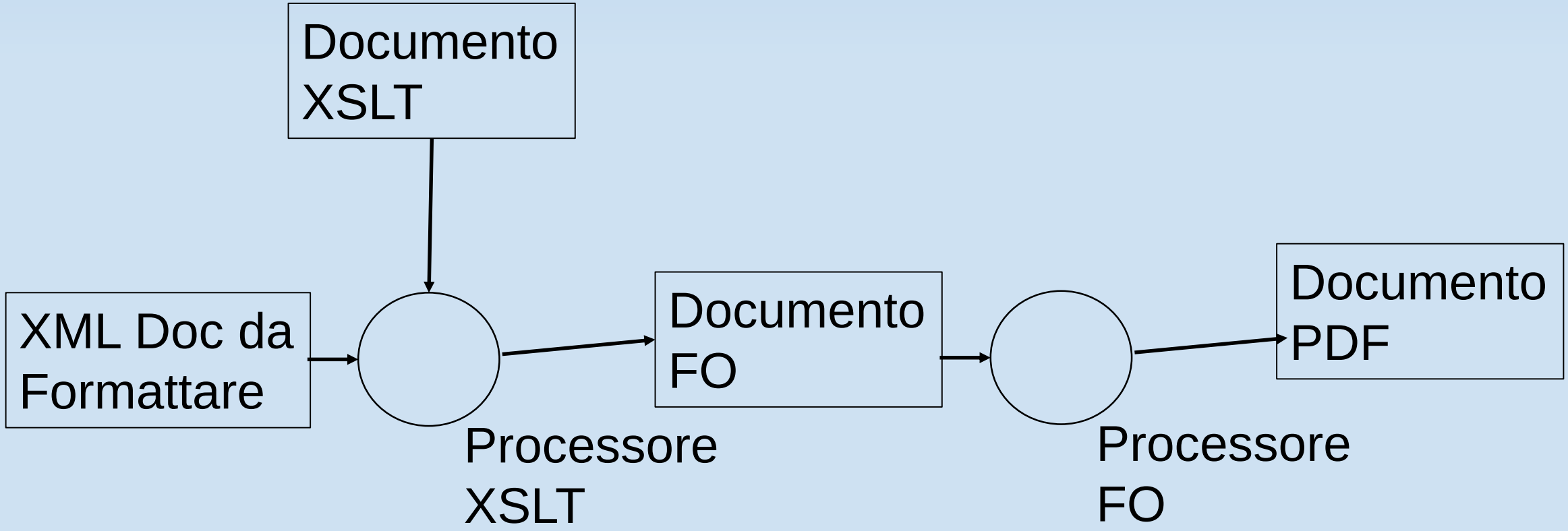
- Come specificare la struttura estetica di un documento senza essere legati ad uno specifico formato?
  - PDF
  - PNG
  - RTF (Word)
- Usando dei concetti di alto livello, indipendenti dal formato target

I Formatting Objects

- Lo standard FO consente di fare questo
- Definisce ciò che compone un documento, come
  - Testi
  - Aree/box
  - Dimensioni
  - Linee
- Usando, ancora una volta, XML come linguaggio ospite

XSLT + FO

- Il documento XSLT contiene i frammenti FO da generare



Processore FO

- Il processore FO è parametrico rispetto al formato target
- Cambiando il parametro di funzionamento, cambia il formato generato

FO e XSLT

- Una curiosità:
- Il formato FO è previsto già dalla definizione di XSLT
- Il documento ufficiale è disponibile al link seguente: **https://www.w3.org/TR/2001/REC-xsl-20011015/slice6.html**

Esempio di FO

```
<?xml version="1.1" encoding="utf-8"?>
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
 <fo:layout-master-set>
  <fo:simple-page-master master-name="my_page" margin="0.5in">
   <fo:region-body/>
  </fo:simple-page-master>
 </fo:layout-master-set>
 <fo:page-sequence master-reference="my_page">
  <fo:flow flow-name="xsl-region-body">
   <fo:block>Hello world!</fo:block>
  </fo:flow>
 </fo:page-sequence>
</fo:root>
```

FOP: Formatting Objects Processor

- Progetto di Apache
- Processa i documenti FO
- Facilmente scaricabile:

`https://xmlgraphics.apache.org/fop/`

Come Usare FOP

- Portarsi nella cartella che contiene i batch
- Su Windows, fop.bat
- Avendo un file prova.fo da formattare
  `fop prova.fo prova.pdf`
- Per ottenere un png
  `fop prova.fo –png prova.png`
- Per inviare alla stampante
  `fop prova.fo -print`

Esempio

```
<?xml version="1.1" encoding="utf-8"?>
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
 <fo:layout-master-set>
  <fo:simple-page-master master-name="my_page" margin="0.5in">
   <fo:region-body/>
  </fo:simple-page-master>
 </fo:layout-master-set>
 <fo:page-sequence master-reference="my_page">
  <fo:flow flow-name="xsl-region-body">
   <fo:block>Hello world!</fo:block>
  </fo:flow>
 </fo:page-sequence>
</fo:root>
```

Risultato

Hello world!

Altro Esempio

- Nel file Lez_15.zip trovate allegati due esempi:
- prova.fo e prova.pdf
- prova2.fo, prova2.pdf e prova2.png
  Questo è più complesso, non lo riporto nelle slide, ma il suo risultato è nella slide seguente

Esempio: prova2.pdf

Chapter title

First section title

Section one's first paragraph.

Section one's second paragraph.

Second section title

Section two's only paragraph.

XSLT in Dettaglio

Struttura di un Foglio di Stile

- Il foglio di stile è contenuto nell'elemento **`xsl:stylesheet`**

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

Struttura di un Foglio di Stile

- Un documento XSLT contiene delle regole
- Le regole vengono applicate per gli elementi del documento da formattare
- Specificando opzioni di selezione compplesse
- Le regole sono chiamate «template»

Template

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

…

```
</xsl:template>
```

L’attributo **`match`** indica su quali elementi deve scattare la regola (con il linguaggio XPath, vedi dopo)

Contenuto del Template

- Elementi di XSL ed elementi del formato target sono mescolati
- Grazie al namespace, il processore XSLT riesce a distinguere ciò che deve elaborare e ciò che deve lasciare inalterato.

Il Linguaggio XPath

- Il linguaggio XPath viene usato per specificare su quali elementi del documento sorgente applicare i template
- Fa parte dei cosiddetti XML Query Languages
- Sono linguaggi pensati per interrogare documenti XML e generare altri documenti XML

Il Linguaggio XPath

- XPath tipicamente è la base per gli altri XML Query Language
- Il linguaggio che è diventato il riferimento in questo settore si chiama XQuery (potente ma difficile da usare)

Il Linguaggio XPath

- XPath prende il nome dal fatto che permette di specificare dei path (percorsi) sia relativi che assoluti
- I path consentono di selezionare gli elementi del documento

Espressioni XPath

- **`/`** Fa riferimento all'elemento radice, qualunque esso sia
- **`/A`** Fa riferimento all'elemento radice A, se l'elemento radice è diverso, il match non avviene
- **`//A`** Fa riferimento ad un qualunque element A, in qualsiasi posizione nel documento

Espressioni XPath

- **`/A/B/C`** Specifica l'annidamento, la radice A contiene B che a sua volta contiene C; il match avviene su C
- **`/A/B//C`** Il match avviene su un qualunque C, purchè al di sotto di B contenuto nella radice A-
- **`A`** Se vi è modo di contestualizzare la ricerca su un elemento, cerca un figlio A dell'elemento corrente

Espressioni XPath

- **`/A/B/*`** Cerca qualsiasi elemento figlio di B contenuto sotto la radice A
- **`/A/B/@a`** Espressione che preleva il valore dell'attributo **`a`** di B

Espressioni XPath

- **`//B[C='v']`** Cerca tutti gli elementi B il cui figlio C contiene il testo 'v'
- **`//B[@at='v']`** Cerca tutti gli elementi B il cui attributo **`at`** vale **`'v'`**

Funzioni XPath

- XPath prevede anche funzioni che forniscono valori relativi agli elementi selezionati
- Molte servono per fare conversioni di tipo o manipolare stringhe
- Questo link riporta una lista completa **`https://developer.mozilla.org/en-US/docs/Web/XPath/Functions`**

Funzioni XPath

- **`position()`** Fornisce la posizione di un elemento nella lista degli elementi selezionati
- Esempio:
  `//A[position() = 5]`

Seleziona l'elemento A in posizione 5 tra tutti gli A selezionati

Funzioni XPath

- **`last()`** Fornisce il numero di elementi nella lista degli elementi selezionati
- Esempio:
  `//A[position() = last()]`

Seleziona l'elemento A in ultima posizione tra tutti gli A selezionati

Funzioni XPath

- **`count(`***selezione***`)`** Conta il numero di elementi selezionati
- Esempio:
  `count(//A[position() = last()])`

Conta quanti elementi A sono in ultima posizione nella lista, cioè 1

Funzioni XPath

- **`current()`** Fa riferimento all'elemento corrente
- **`.`** Shortcut per **`current()`**
- Esempio:
  `count(c`urrent()**`/*)`**
  count(./*)

Conta quanti sono i figli dell'elemento corrente

Funzioni XPath

- **`name()`** Il nome completo (Q-Name) del primo elemento tra quelli selezionati (o quelo corrente)
- Esempio: **`//myns:A/mame()`**

Restituisce il nome completo dell'elemento selezionato, cioè **`'myns:A'`**

Funzioni XPath

- **`local-name()`** Il nome senza prefisso (Local Name) del primo elemento tra quelli selezionati (o quello corrente)
- Esempio:
  `//myns:A/local-mame()`

Restituisce il nome locale dell'elemento, cioè **`'A'`**

Funzioni XPath

- **`namespace-uri()`** Restituisce l'URI del namespace dell'elemento selezionato
- Esempio:
  `//myns:A/mamespace-uri()`

Funzioni XPath

- **`text()`** Restituisce il nodo DOM corrispondente al contenuto testuale dell'elemento selezionato
- Esempio: **`//myns:A/text()`**

Costrutti di XSLT

- XSLT fornisce diversi costrutti che possono essere utilizzati per specificare trasformazioni complesse
- Approccio **dichiarativo**: XSLT non è procedurale, quindi non è possibile cambiare il valore delle variabili o scrivere cicli

Named Template

- I Template possono avere un nome

```
<xsl:template name="Interno">
```

- Invece dell'attributo **`match`**, l'attributo **`name`** dà un nome al template
- Questo consente di invocare l'esecuzione del template quando serve

Invocazione di un Template

- Per invocare un template, si usa **`xsl:call-template`**

- Esempio:

```
<xsl:call-template name="Interno"/>
```

Effetto della Chiamata

- **`xsl:call-template`** sta all'interno di xsl:template
- Il contenuto di un template (esclusi gli elementi con prefisso xsl) viene mandato in output
- Il template chiamato produce un output, quindi questo output fa parte dell'output del template corrente

Applicare i Template Ricorsivamente

- Quando un template viene processato, la visita dell'albero DOM si ferma, se non si dice di procedere ricorsivamente
- L'elemento **`<xsl:apply-templates/>`** dice al processore XSLT di visitare i discendenti, cercando di applicare i template definiti
- L'output di tutti i template processati viene raccolto nel punto in cui **`xsl:apply-templates`** è specificato

Estrarre Valori dal DOM

- Per poter svolgere le trasformazioni, è necessario estrarre valori (di attributi o del contenuto testuale) dal DOM
- Per esempio, un attribtuto dell'elemento su cui il template è stato attivato
- **`<xsl:value-of select="@Nome"/>`** estrae il valore dell'attributo Nome

Estrarre Valori dal DOM

- Esempio

```
<xsl:template match="PRODUTTORE">
<p>In questo catalogo sono
    riportati i modelli
    del produttore
<xsl:value-of select="@Nome"/>.
…
</xsl:template>
```

Estrarre Valori dal DOM

- Il template precedente si attiva sull'elemento **`PRODUTTORE`**
- Genera del testo HTML, in particolare un paragrafo
- Nel testo, serve inserire il nome del produttore, che è l'attributo **`Nome`** dell'elemento selezionato

Estrarre Valori dal DOM

- Esempio

```
<xsl:template match="CATALOGO/text()">
<h1>  <xsl:value-of select="."/> </h1>
</xsl:template>
```

Estrarre Valori dal DOM

- Il template precedente si attiva non sull'elemento **`CATALOGO`** ma sul nodo testo che descrive il suo contenuto testuale
- L'attributo **`select="."`** di **`xsl:value-of`** prende il valore del nodo corrente, che è il testo
- Il valore viene inserito all'interno dell'elemento HTML **`h1`**

Variabili

- XSLT fornisce il concetto di «variabile»
- Sono variabili dichiarative, cioè non possiamo cambiarne il valore, una volta definite
- **`<xsl:variable name="conteggio">`**
- Questo elemento definisce la variabile il cui nome è indicato dall'attributo **`name`**

Variabili

- Il valore della variabile è il contenuto dell'elemento **`xsl:variable`**
- Infatti questo elemento non è vuoto
- Il valore può essere costante (es., un frammento HTML)
- Il valore può essere ottenuto dall'elemento **`xsl:value-of`**

Variabili

- Esempio

```
<xsl:variable name="conteggio">
   <xsl:value-of select="count(AUTO)"/>
</xsl:variable>
```

- Il valore della variabile **`conteggio`** è il numero di figli **`AUTO`** del nodo su cui il template è attivato

Variabili

- Per far riferimento alle variabili, si usa la notazione $nome

```
<p> I modelli sono in totale:
<xsl:value-of select="$conteggio"/>
</p>
```

- Il valore della variabile viene inserito nel paragrafo
- Non può essere messo direttamente nell'output, server **`xsl:value-of`**

Condizioni

- XSLT fornisce elementi condizionali
- Primo elemento: **`xsl:if`**
- **`<xsl:if test="position()=1">`**
- L'attributo t**`est`** contiene la condizione da verificare (su variabili, funzioni, attributi)
- Se la condizione è vera, il contenuto viene messo in output, altrimenti no.

Condizioni

- Esempio:

```
<xsl:template match="CATALOGO/text()">
 <xsl:if test="position()=1">
  <h1>  <xsl:value-of select="."/>
  </h1>
 </xsl:if>
</xsl:template>
```

Condizioni

- Il template precedente si attiva sui nodi testuali contenuti in un elemento **`CATALOGO`**
- Poiché questi potrebbero essere più d’uno, ci interessa solo il primo
- La condizione è vera solo sul primo nodo testuale
- Il cui valore viene preso e mandato in output dentro un blocco **`h1`** di HTML

Condizioni

- L'elemento **`xsl:choose`** consente di gestire le alternative

```
<xsl:choose>
  <xsl:when test= …>
  </xsl:when>
  …
  <xsl:otherwise>
  </xsl:otherwise>
</csl:choose>
```

Condizioni

- Un blocco **`xsl:when`** gestisce un'alternativa
- Vi possono essere molti blocchi XSL:when
- Al termine, se nessun blocco **`xsl:when`** è stato attivato, il blocco **`xsl:otherwise`** viene attivato.

Condizioni

- Esempio: assegnamento condizionato

```
<xsl:variable name="colore">
 <xsl:choose>
  <xsl:when test="@Carburante='Benzina'">
   <xsl:value-of select="string('#F0F000')"/>
  </xsl:when>
  <xsl:otherwise>
   <xsl:value-of select="string('#FF0000')"/>
</xsl:otherwise>
 </xsl:choose>
</xsl:variable>
```

Condizioni

- La variabile **`colore`** deve essere giallo se il carburante dell'auto è la benzina
- Altrimenti deve essere rosso
- Il blocco **`xsl:when`** si attiva se il valore dell'attributo **`carburante`** dell'elemento su cui il template è attivo è **`'Benzina'`**
- Altrimenti si attiva ilblocco **`xsl:otherwise`**

Processare Liste di Elementi

- L'elemento **`xsl:for-each`** consente di processare una lista di elementi
- Ottenuta attraverso un'espressione XPath
- Processando un elemento per volta
- Non è un ciclo

Processare Liste di Elementi

- Esempio:

```
<table>
<xsl:for-each select="MARCA">
   <tr> <td>
   <xsl:value-of select="."/>
   </td> </tr>
</xsl:for-each>
</table>
```

Processare Liste di Elementi

- Il frammento precedente crea una tabella
- Ogni riga della tabella è derivata dalla lista degli elementi **`MARCA`** figli dell'elemento su cui il template è stato attivato
- Il blocco **`xsl:value-of`** lavora sull'elemento corrente per prenderne il valore, cioè prende il contenuto testuale
- Si produce una lista di marche

Esempio: catalogo.xml

- In Lez_09.zip, troverete una cartella Esempi
- File catalogo.xml, catalogo delle auto da processare
- File catalogo_stile.xsl, con il foglio di stile per processare il cataologo
- Come processarlo: con fop

```
fop -xml catalogo.xml
    -xsl catalogo_stile.xsl
    -foout catalogo.html
```

Esempio: catalogo.xml

- In alternativa, si può scaricare il processore msxsl di Microsoft
- Scaricabile da:
  https://www.microsoft.com/en-us/download/details.aspx?id=21714
- Come eseguirlo

```
msxsl catalogo.xml
      catalogo_stile.xsl
      -o catalogo.html
```

XSLT e FO

- Se l’output del foglio XSLT è il formato FO
- Il processore XSLT genera il formato FO, che a sua volta viene processato per generare, per esempio, il PDF
- Che aspetto ha un template con elementi FO all’interno?
- Lo vedremo nella prossima slide

XSLT e FO

```
<xsl:template match="/root">
 <fo:root>
  <fo:layout-master-set>
   <fo:simple-page-master master-name="my-page"
            page-width="8.5in" page-height="11in">
    <fo:region-body margin="1in" margin-top="1.5in"
            margin-bottom="1.5in"/>
    </fo:simple-page-master>
   </fo:layout-master-set>
```

XSLT e FO

```
  <fo:page-sequence master-reference="my-page">
    <fo:flow flow-name="xsl-region-body">
          <xsl:apply-templates/>
    </fo:flow>
  </fo:page-sequence>
 </fo:root>
</xsl:template>
```