UNIVERSITÀ DEGLI STUDI DI PALERMO

dipartimento di ingegneria unipa

Tecnologie XML

CORSO DI

PROGRAMMAZIONE WEB E MOBILE

a.a. 2021/2022

Prof. Roberto Pirrone

LABORATORIO DI INTERAZIONE UOMO-MACCHINA
CHILAB

XPath

- **XPath** (XML Path Language) è un linguaggio usato per indirizzare porzioni di un documento XML, o per calcolare valori (stringhe, numeri, booleani) basati sul contenuto di un documento XML.
- Il linguaggio XPath è basato su una rappresentazione ad albero del documento XML e fornisce la capacità di navigare lungo l'albero, selezionando i nodi sulla base di una varietà di criteri.
- XPath è spesso adoperato come un linguaggio di query semplice e veloce.

XPath

- Per stabilire quali elementi del documento attivano una regola, si ricorre a XPath
- XPath permette di localizzare, in modo efficace parti specifiche di un documento XML
- In XPath, un documento XML viene considerato come una struttura ad albero, in cui ogni parte del documento è rappresentato da un nodo

XPath

- XPath ha sette tipi di nodi:
  - Radice
  - Elemento
  - Attributo
  - Testo
  - Commento
  - Istruzione di elaborazione
  - Namespace
- Eccetto la radice, un nodo dell'albero può essere genitore (parent) di nodi figlio (child)

XPath: operatori

- XPath ha 3 operatori principali per *creare percorsi nel DOM* e ricercare gruppi di nodi:
  - | (unisce due gruppi di nodi)
  - / (separa i passaggi di posizione)
  - // (raggiunge un nodo direttamente senza specificare il percorso: abbrevia il path **`/descendant-or-self::node()/`**)

- Gli operatori consentono di manipolare gruppi di nodi per formare altri gruppi

XPath: specifica dei percorsi

- Un generico percorso Xpath può essere assoluto o relativo e ha la sintassi:

  [/]<step>[/<step>…]

- Ogni step può essere definito come

  step → <nome_asse>::<test di nodo>[<predicato>]

XPath: specifica dei percorsi

step → <nome_asse>::<test di nodo>[<predicato>]

- L'asse è una espressione che ricerca le relazioni del nodo corrente con gli altri nel DOM
- Il test di nodo seleziona esplicitamente un nodo o attributo in base al nome o a una wildcard che fa riferimento a gruppi di nodi predefiniti
- Il predicato usa funzioni e operatori per stabilire ulteriori condizioni di scelta

XPath: selezioni di nodi

Expression	Description
nodename	Selects all nodes with the name "*nodename*"
/	Selects from the root node
//	Selects nodes in the document from the current node that match the selection no matter where they are
.	Selects the current node
..	Selects the parent of the current node
@	Selects attributes

In the table below we have listed some path expressions and the result of the expressions:

Path Expression	Result
bookstore	Selects all nodes with the name "bookstore"
/bookstore	Selects the root element bookstore **Note:** If the path starts with a slash (/) it always represents an absolute path to an element!
bookstore/book	Selects all book elements that are children of bookstore
//book	Selects all book elements no matter where they are in the document
bookstore//book	Selects all book elements that are descendant of the bookstore element, no matter where they are under the bookstore element
//@lang	Selects all attributes that are named lang

Credits w3schools.com

XPath: selezioni di nodi

Wildcard	Description
*	Matches any element node
@*	Matches any attribute node
node()	Matches any node of any kind

Credits w3schools.com

Xpath: assi

AxisName	Result
ancestor	Selects all ancestors (parent, grandparent, etc.) of the current node
ancestor-or-self	Selects all ancestors (parent, grandparent, etc.) of the current node and the current node itself
attribute	Selects all attributes of the current node
child	Selects all children of the current node
descendant	Selects all descendants (children, grandchildren, etc.) of the current node
descendant-or-self	Selects all descendants (children, grandchildren, etc.) of the current node and the current node itself
following	Selects everything in the document after the closing tag of the current node
following-sibling	Selects all siblings after the current node
namespace	Selects all namespace nodes of the current node
parent	Selects the parent of the current node
preceding	Selects all nodes that appear before the current node in the document, except ancestors, attribute nodes and namespace nodes
preceding-sibling	Selects all siblings before the current node
self	Selects the current node

Credits w3schools.com

XPath: predicati

- Indicizzazione diretta del nodo: **`[<numero posizione>]`**

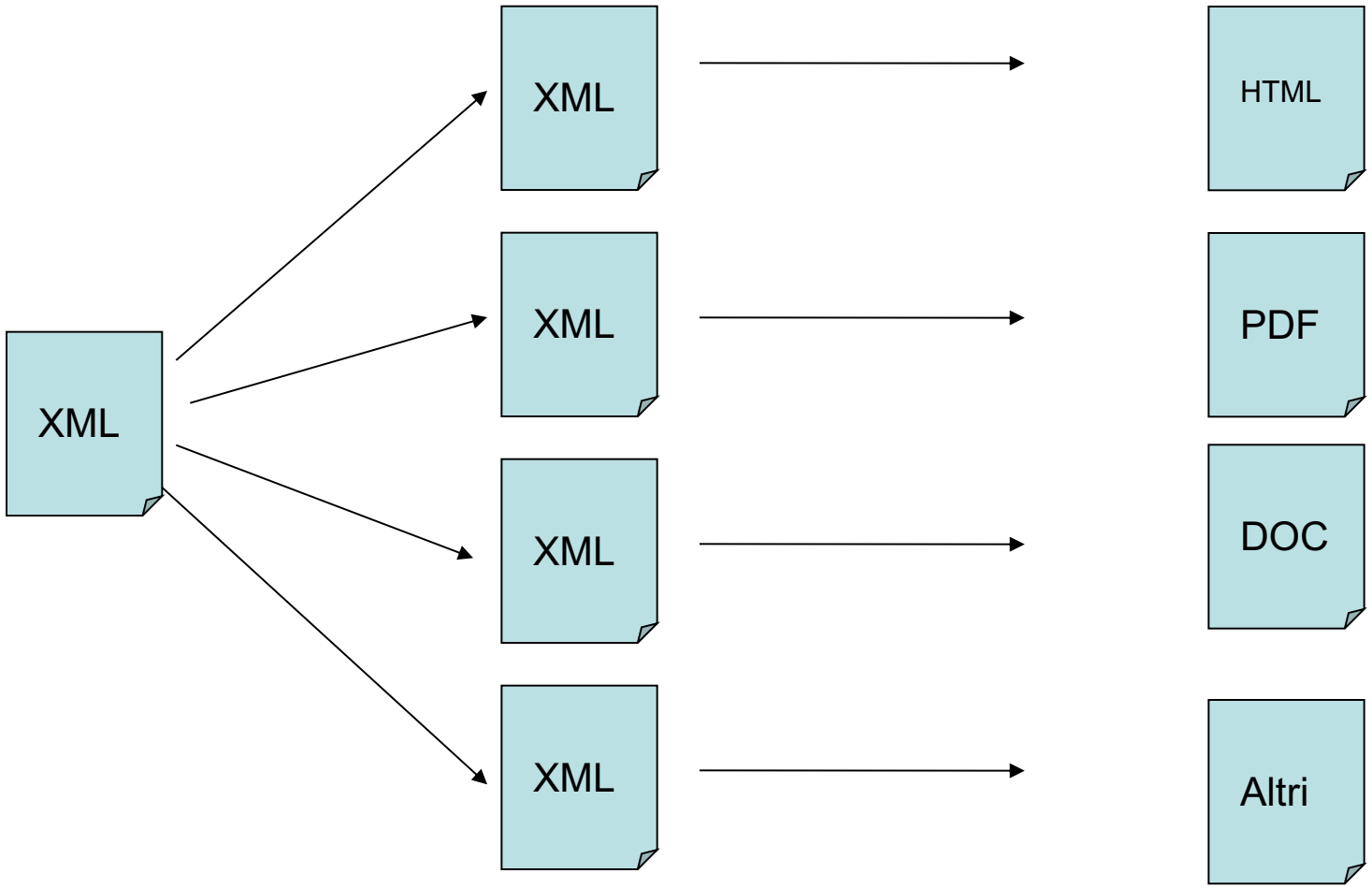
- Operatori
  - Aritmetici: **`+, -, *, div, mod`**
  - Relazionali: **`>, <, =, !=, >=, <=`**
  - Logici: **`and, or`**

- Funzioni
  - Di nodo: **`position(), count(), last(), name(), text()`**
  - Stringa: **`contains(), starts-with(), ends-with()`**
  - Booleane: **`not(<expr>)`**

XSL Transformations

- Linguaggio utilizzato per manipolare strutture o documenti XML
- La trasformazione è il processo di creazione di un nuovo documento basato sul documento originale
- Il processo non modifica il documento originale

XSL Transformations



XSL Transformations

- *Programmazione dichiarativa*
  - Diciamo alla macchina *cosa fare* e non come farlo

- Esecuzione basata sui dati
  - Il codice viene eseguito quando il parser incontra un certo dato quando analizza il documento XML

XSL Transformations

Ciascun documento XSLT ha il prologo dell'XML

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

Ciascun documento XSLT è compreso tra i tag

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

In tal modo, i tag dell'XSLT fanno riferimento al namespace xsl

XSL Transformations

Il codice XSLT è prevalentemente composto da una serie di regole, dette modelli.

Ciascun modello indica un pattern ed un template

Ogni volta che il parser incontra un pattern nel documento, esso restituisce in output il template

Il pattern viene indicato come valore dell'attributo match

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

In questo caso, quando il parser incontra il tag *dessert*, stampa la parola *torta*

XSL Transformations

- Il match viene effettuato tramite espressioni Xpath

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

In tal caso, quando il parser comincia a leggere il documento, stampa la parola yo

XSL Transformations

- Le regole possono essere concatenate

```
<xsl:template match"/">
 <xsl:apply-template />
</xsl:template>
```

Il tag <xsl:apply-template /> indica al parser che deve procedere nella lettura del documento ed applicare le regole attivate

```
<?xml version="1.0" encoding="UTF-8" ?>
<pets>
 <pet type="cat">
      Max
 <pet>
 <pet type="parrot" color="red">
      Peter
 <pet>
</pets>
```

XSL Transformations

```
<xsl:template match”/”>
  <xsl:apply-template />
</xsl:template>
```

- Se il parser incontra la radice del documento
  - Continua la lettura del documento

```
<xsl:template match=”pets”>
  <xsl:apply-template />
</xsl:template>
```

- Il parser incontra il tag pets
  - Continua a leggere il documento

```
<xsl:template match=”pet”>
  animale
</xsl:template>
```

- Il parser incontra il tag pet
  - Stampa animale

XSL Transformations

Un modello può essere identificato esplicitamente tramite un nome

```
<xsl:template match="cars">
 <xsl:call-template name="car" />
</xsl:template>
<xsl:template name="car">
 Model: <xsl:value-of select="@model" />
</xsl:template>
```

XSL Transformations

- Posso assegnare a due modelli distinti lo stesso match distinguendoli tramite l'attributo mode

```
<xsl:template match="person" mode="generic">
 <xsl:value-of select="./name" />
 <xsl:value-of select="./surname" />
</xsl:template>
<xsl:template match="person" mode="work">
 <xsl:value-of select="./role" />
</xsl:template>

<xsl:template match="/">
 <xsl:apply-templates mode="generic" />
 <xsl:apply-templates mode="work" />
</xsl:template>
```

XSL Transformations

- Inserire in output di elementi
- È sufficiente inserire l'elemento nel modello

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

Il parser stampa in uscita `<el>ciao</el>`

XSL Transformations

- Inserire in output di elementi con nome generato

```
<xsl:template match="/">
 <xsl:element name="el">ciao</xsl:element>
</xsl:template>
```

Il parser stampa in uscita **<el>ciao</el>**

XSL Transformations

- Inserire in output di attributi

```
<xsl:template match="persons">
 <table>
      <xsl:attribute name="border">1</xsl:attribute>
      <xsl:apply-templates />
 </table>
</xsl:template>
```

Il parser stampa in uscita

```
<table border="1">
 …
</table>
```

XSL Transformations

- Inserire in output un gruppo di attributi
- Definisco un insieme di attributi

```
<xsl:attribute-set name="tableattr">
  <xsl:attribute name="border">1</xsl:attribute>
  <xsl:attribute name="width">2</xsl:attribute>
</xsl:attribute-set>
```

- Uso l'insieme definito

```
<xsl:template match="persons">
  <xsl:element name="table" use-attribute-sets="tableattr" />
  <xsl:apply-templates />
</xsl:template>
```

Il parser stampa in uscita

```
<table border="1" width="2">
  …
</table>
```

XSL Transformations

- Inserire in output il nodo contest
  - Due varianti

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

In questo caso si fa una copia dei nodi risultati dalla chiamata di apply-templates

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

In questo caso si fa una copia del nodo persons

XSL Transformations

- copy fa una copia del solo nodo di contesto, ignorando i nodi eventualmente contenuti
- Per fare una *deep copy*, usare copy-of
  - Due varianti

```
<xsl:template match="persons">
 <xsl:copy-of select="." />
</xsl:template>
```

In questo caso si fa una copia del nodo persons e di tutti i nodi contenuti

XSL Transformations

- Inserire in output un commento

```
<xsl:template match="persons">
 <xsl:comment>
     questo sarà inserito come commento
 </xsl:comment>
  <xsl:apply-templates />
</xsl:template>
```

XSL Transformations

- Inserire in output un'istruzione di elaborazione

```
<xsl:template match="persons">
 <xsl:processing-instruction name="nome">
      href="prova"
 </ xsl:processing-instruction >
  <xsl:apply-templates />
</xsl:template>
```

- Il sistema stampa

```
<?nome href="prova" ?>
```

Controllo del flusso

- **<xsl:value-of select="…" />**
  - seleziona un elemento o attributo (con la notazione XPath @)
- **<xsl:for-each select="…"> … </xsl:for-each>**
  - Itera per tutti gli elementi/attribute corrispondenti alla selezione
- **<xsl:if test="…"> … </xsl:if>**
  - Selezione sotto condizione

XSL Transformations

- Posso definire più condizioni

```
<xsl:for-each select="phonenumber">
 <xsl:choice>
      <xsl:when type="@type='mobile'">
            mobile: <xsl:value-of select="." />
       </xsl:when>
      <xsl:when type="@type='fax'">
            fax: <xsl:value-of select="." />
       </xsl:when>
      <xsl: otherwise>
            phone: <xsl:value-of select="." />
       </otherwise>
 </xsl:choice>
</xsl:for-each>
```