XML, DTD, XSD

A cura del team WebML
Politecnico di Milano

1. Concetti generali

XML: eXtensible Markup Language

- **Formato di file proposto dal W3C per distribuire documenti elettronici sul World Wide Web**

  Evoluzione:

  - 1986: Standard Generalized Markup Language (SGML) ISO 8879-1986
  - Agosto 1997: XML Working Draft
  - Dicembre 1997: XML 1.0 Proposed Recommendation
  - Febbraio 1998: Standard

XML

- **HTML: insieme fisso di tag**
- **XML: standard per creare linguaggi di markup con tag personalizzati (erede di SGML); possono essere usati in qualunque dominio**
- **HTML vs XML**

```
<h1> The Idea Methodology </h1>
 <ul>
  <li> di S. Ceri, P. Fraternali
  <li> Addison-Wesley
  <li> US$ 49
 </ul>
```

```
<book>
<title>The Idea
      Methodology </title>
<author> S. Ceri </author>
<author> P. Fraternali </author>
<ed> Addison-Wesley </ed>
<price> US$ 49 </price>
</book>
```

XML vs HTML

- XML non rimpiazza HTML!

XML e HTML sono nati con scopi diversi:

- XML progettato per descrivere DATI → cosa sono i dati
- HTML progettato per visualizzare i dati → come appaiono i dati

Uso di XML

- Separare i dati dal modo con cui vengono visualizzati
- Scambiare i dati tra sistemi incompatibili
- Scambiare informazioni in sistemi B2B
- Condividere dati
- Memorizzare dati
- Creare nuovi linguaggi (WML, MathML…)

Accessibilità:

- Supporto nei moderni browsers per Visualizzazione, validazione, Embedding in documenti HTML,Trasformazione con XSL, Visualizzazione con CSS
- è un documento di testo → il software che tratta documenti testuali tratta anche XML

Esempio di documento XML

```
<?xml version="1.0"?>
<elenco>
 <prodotto codice="123">
    <descrizione> Forno </descrizione>
    <prezzo> 1040000 </prezzo>
 </prodotto>
 <prodotto codice="432">
    <descrizione> Frigo </descrizione>
</prodotto>
</elenco>
```

Attributi

Tag con contenuti

Sintassi XML

- Tutti gli elementi hanno un tag di chiusura
- I tag sono "case sensitive"
- I tag devono essere annidati correttamente
- Un documento XML deve avere un tag radice

Elementi

```
<prodotto codice="123kl14">
    <descrizione> Forno </descrizione>
    <prezzo> 1040000 </prezzo>
</prodotto>
```

- Si possono estendere
- Hanno relazioni (padre-figlio)
- Hanno contenuto

Attributi

```
<prodotto codice="123">
    <descrizione> Forno </descrizione>
    <prezzo> 1040000 </prezzo>
</prodotto>
```

- Gli elementi possono avere degli attributi
- I valori vanno racchiusi tra " "
- Differiscono dagli elementi perchè non possono contenere elementi figli

Attributi

Problemi nell'uso di attributi

- **Non possono contenere valori multipli**
- **Non sono facilmente estendibili**
- **Non possono descrivere strutture**
- **Sono difficilmente manipolabili da programmi**

Vanno bene per memorizzare metadati

Namespace

- **Metodo per evitare conflitti di nome**

```
<table>
   <tr>...</tr>
</table>

<table>
   <product>...</product>
</table>
```

- **Si introduce un prefisso**

```
<h:table>
   <h:tr>...</h:tr>
</h:table>

<my:table>
   <my:product>...
   </my:product>
</my:table>
```

• Un **XML namespace** è una collezione di nomi (attributi, elementi,…), identificata da un URI

• URL HTML diventa URI (Uniform Resource Identifier) XML

  • un documento XML può far riferimento ad oggetti diversi (DTD, documenti XML, etc.)

  • è più importante identificare un oggetto che localizzarlo

Tipi di marcature

- Elementi: <prodotto>
- Entità: < (sta per <), ℞ (Unicode)
- Commenti: <!- - qualsiasi testo - - >
- Istruzioni: <? Nome-istruzione dati ?>
- Sezioni CDATA (character data)

```
<![CDATA[
        *p = &q;
         b = (i <= 3);
 ]]>
```


2. DTD

Document Type Definition (DTD)

- Detta il tipo di un documento, cioè:
  - i tag ammessi
  - le regole di annidamento dei tag
- Scopi:
  - Accordarsi su formato/struttura dei documenti
  - Validare documenti XML secondo certe regole
- Esempio di dichiarazione di un elemento:
  <!ELEMENT PRODOTTO
  (DESCRIZIONE, PREZZO)>
- L'elemento prodotto contiene al suo interno un elemento descrizione seguito da un elemento prezzo

Validazione di un documento XML

- Un documento XML la cui sintassi è corretta(cioè soddisfa le regole di scrittura di documenti XML) è detto "well-formed" (ben formato)
- Un documento validato rispetto ad un DTD è detto "valid" (valido)

Modello di contenuto

- Elementi contenenti elementi figli

  <!ELEMENT PRODOTTO (DESCRIZIONE))>
  - <prodotto> <descrizione>…</descrizione></prodotto>

- Elementi con PCDATA (parsed character data = brano di testo qualunque)

  <!ELEMENT DESCRIZIONE #PCDATA)>
  - <descrizione> testo </descrizione>

- Elementi vuoti

  <!ELEMENT ARTICOLO EMPTY)>
  - <articolo/>

Modello di contenuto

- Contenuto misto

  <!ELEMENT ARTICOLO (#PCDATA | PRODOTTO)>

  - <articolo> testo </articolo>
  - <articolo><prodotto>..</prodotto><articolo>

- Qualsiasi contenuto

  <!ELEMENT PARTE ANY)>

  - <parte><sottoparte></sottoparte><parte>
  - <parte><prodotto></prodotto></parte>

Occorrenze di un elemento

- 1 volta
  <!ELEMENT PRODOTTO (DESCRIZIONE))>
- 1 o più volte
  <!ELEMENT LISTA (PRODOTTO+))>
- 0 o più volte
  <!ELEMENT LISTA (PRODOTTO*))>
- 0 o 1 volta
  <!ELEMENT PRODOTTO (DESCRIZIONE?))>

Esempio di DTD

```
<!ELEMENT ELENCO (PRODOTTO+)>
<!ELEMENT PRODOTTO (DESCRIZIONE, PREZZO?)>
<!ELEMENT DESCRIZIONE #PCDATA>
<!ELEMENT PREZZO #PCDATA>
```

```
<elenco>
    <prodotto codice="123">
        <descrizione> Forno </descrizione>
        <prezzo> 1040000 </prezzo>
    </prodotto>
    <prodotto codice="432">
        <descrizione> Frigo </descrizione>
    </prodotto>
</elenco>
```

NOTA: un DTD NON è un documento XML

Dichiarazioni di attributi

- **Per ogni elemento dice:**
  - **quali attributi può avere il tag**
  - **che valori può assumere ciascun attributo**
  - **qual è il valore di default**

- **Esempio di dichiarazione di attributo:**

```
<!ATTLIST PRODOTTO
    codice   ID                          #REQUIRED
    label    CDATA                       #IMPLIED
    status   (disponibile | terminato)   'disponibile'>
```

- **Il tag PRODOTTO può contenere 3 attributi**

Tipi di attributi

- **CDATA: stringa**
- **ID: identificatore**
- **IDREF, IDREFS: valore di un attributo di tipo ID nel documento (o insieme di valori)**
- **ENTITY, ENTITIES: nome (nomi) di entità**
- **NMTOKEN, NMTOKENS: caso ristretto di CDATA (una sola parola o insieme di parole)**

```
codice   ID                          #REQUIRED
label    CDATA                       #IMPLIED
status   (disponibile | terminato)   'disponibile'
```

Vincoli sugli attributi

- **#REQUIRED: il valore deve essere specificato**
- **#IMPLIED: il valore può mancare**
- **#FIXED "valore": se presente deve coincidere con "valore"**
- **"valore": il valore può non essere specificato, nel qual caso si assume "valore" come default**

codice	ID	#REQUIRED
label	CDATA	#IMPLIED
status	(disponibile | terminato)	'disponibile'

Dichiarazioni di entità

- **Analoghe alle dichiarazioni di macro con #define in C – esempio:**

```
<!ENTITY   ATI    "ArborText, Inc.">
<!ENTITY boilerplate   SYSTEM   "/standard/legalnotice.xml">
```

- **Le entità possono essere**
  - **interne (&ATI;)**
  - **esterne (&boilerplate; &ATIlogo;)**

Documenti con DTD

```
<?XML version="1.0" standalone="no"?>

<!DOCTYPE capitolo SYSTEM ”libro.dtd" [
      <!ENTITY %ulink.module "IGNORE">
       <!ELEMENT ulink (#PCDATA)*>
       <!ATTLIST ulink
            xml:link          CDATA  #FIXED "SIMPLE"
            xml-attributes CDATA  #FIXED "HREF URL"
            URL               CDATA  #REQUIRED>
 ]>

<capitolo>...</capitolo>
```

DTD esterno

DTD interno

HTML e XML

- HTML è un caso particolare di XML
  - è possibile scrivere un DTD per XML
  - lo stile sintattico è leggermente diverso
  - è in uscita una revisione di HTML per uniformarsi a XML (XHTML)

3. XML Schema

XML Schema

Storia: - inizialmente proposto da Microsoft
- divenuto W3C recommendation (maggio '01)

Scopo: definire gli elementi e la composizione di un documento XML

Un XML Schema definisce regole riguardanti:

- Elementi
- Attributi
- Gerarchia degli elementi
- Sequenza di elementi figli
- Cardinalità di elementi figli
- Tipi di dati per elementi e attributi
- Valori di default per elementi e attributi

Vantaggi di XML Schema rispetto a DTD

XML Schema (noti anche come XSchema o XSD= XML Schema Definition):

- **Sono estendibili (per future aggiunte)**
- **Sono anch'essi files XML**
- **Sono più ricchi e completi di DTD**
- **Supportano tipi di dati**
- **Gestiscono namespaces**

Esempio: documento XML

```
<?xml version="1.0"?>
<note xmlns:xsi=
      "http://www.w3.org/2001/XMLSchema-instance"
      xsi:noNamespaceSchemaLocation=
      "http://www.lol-si.com/Xschema/note.xsd">

      <to> Tove </to>
      <from> Jani </from>
      <head> Reminder </head>
      <body> this weekend! </body>
 </note>
```

Note:
1° attributo: URI che dichiara che si vuole la validazione del documento attraverso XSD
2° attributo: dichiara dove reperire il file XSD (sempre attraverso URI)

Esempio: XML Schema

```
<?xml version="1.0"?>
<xs:schema
  xmlns:xs="http://www.w3.org/2001/XMLSchema">
<xs:element name="note">
  <xs:complexType>
     <xs:sequence>
       <xs:element name="to" type="xs:string"/>
       <xs:element name="from" type="xs:string"/>
       <xs:element name="head" type="xs:string"/>
       <xs:element name="body" type="xs:string"/>
      </xs:sequence>
   </xs:complexType>
 </xs:element>
</xs:schema>
```

xs: namespace per XSchema (contiene tutti i tag XSD)

Elementi semplici (Simple elements)

- Possono contenere solo testo (no elem. o attributi)
- Definizione di elementi semplici:

  `<xs:element name="nome" type="tipo"/>`

  `<xs:element name="nome" type="tipo" default="xyz" />`

  `<xs:element name="nome" type="tipo" fixed="xyz" />`
- Esempi di definizione di elementi semplici in XSD

  `<xs:element name="età" type="xs:integer"/>`

  `<xs:element name="cognome" type="xs:string"/>`
- Esempi di elementi semplici XML

  `<età> 65 </età>`

  `<cognome> Rossi </cognome>`

Attributi in XSD

- Definizione di attributi

  <xs:attribute name="name" type="type"/>

  <xs:attribute … default|fixed="xyz"
  use="required|optional"/>

Valore di default o fisso

Obbligatorio o opzionale

- Esempio di definizione di attribute

  <xs:attribute name="lang" type="xs:string"/>

- Esempio di attributo

  <lastname lang="it"> qwerty </lastname>

Elementi complessi (Complex elements)

4 tipi di elementi complessi:

- **Vuoti (empty)**
- **Contenenti solo altri elementi**
- **Contenenti solo testo**
- **Contenenti testo e/o altri elementi**

- Sintassi di elementi complessi:

```
<xs:element name="name">
        <xs:complexType>
                . . element content . .
        </xs:complexType>
</xs:element>
```

Riferimento ad altri elementi

Attraverso l'attributo ref="nome_elemento"

```
…
<xs:element name="note">
  <xs:complexType> <xs:sequence>
  <xs:element ref="to"/>
  </xs:sequence> </xs:complexType>
 </xs:element>
…

<xs:element name="to" type="xs:string"/>
```


Restrizioni

- **Restrizioni sui valori(min&max)**
- **Sull'enumerazione di insiemi (set)**
- **Su lunghezza, numero di spazi, ecc: lenght, blanks, …**

Indicatori (Indicators)

- **Ordinamento:**
  - Any: qualunque elemento, in qualunque ordine
  - All: tutti gli elementi, in qualunque ordine
  - Choice: uno e un solo elemento
  - Sequence: tutti gli elementi, nell'ordine specificato
- **Numero di occorrenze:**
  - maxOccurs: max numero di occorrenze
  - minOccurs: min numero di occorrenze

  Se non specificati: 1 e 1 sola occorrenza
- **Raggruppamento:**

  Per definire gruppi di elementi, tra loro correlati
  - Group name
  - Group reference

Es.:
```
<xs:element name="persona">
    <xs:complexType>
      <xs:sequence>
            <xs:element name="Cognome" type="xs:string"/>
             <xs:element name="Nome" type="xs:string"
            maxOccurs="4" minOccurs="1"/>
      </xs:sequence>
    </xs:complexType>
</xs:element>
```


Interrogare e trasformare documenti XML

A cura del team WebML
Politecnico di Milano

XQuery

(modulo 1)

XQuery

- Proposta W3C: 15 Febbraio 2001
- Interrogazioni:
  - Path expressions
    - Sintassi abbreviata di XPath
  - Espressioni FLWR
    - Clausole FOR, LET, WHERE, RETURN

Esempio

libri.xml

```
<?xml version="1.0"?>
<Elenco>
      <Libro disponibilità='S'>
            <Titolo>Il Signore degli Anelli</Titolo>
            <Autore>J.R.R. Tolkien</Autore>
            <Data>2002</Data>
            <ISBN>88-452-9005-0</ISBN>
            <Editore>Bompiani</Editore>
      </Libro>
     <Libro disponibilità='N'>
            <Titolo>Il nome della rosa</Titolo>
            <Autore>Umberto Eco</Autore>
            <Data>1987</Data>
            <ISBN>55-344-2345-1</ISBN>
            <Editore>Bompiani</Editore>
      </Libro>
      <Libro disponibilità='S'>
            <Titolo>Il sospetto</Titolo>
            <Autore>F. Dürrenmatt</Autore>
            <Data>1990</Data>
            <ISBN>88-07-81133-2</ISBN>
            <Editore>Feltrinelli</Editore>
      </Libro>
</Elenco>
```


Path expressions

- . Nodo corrente
- .. Nodo padre del nodo corrente
- / nodo radice, o figlio del nodo corrente
- // discendente del nodo corrente
- @ attributo del nodo corrente
- * qualsiasi nodo
- [ ] predicato
- [n] posizione

Esempi di path expressions

- Una path expression può iniziare con

  document(stringa_documento)

  Restituisce la radice del documento specificato

- A partire dalla radice del documento si possono specificare delle espressioni per estrarre il contenuto desiderato
- Esempio:

  document("libri.xml")/Elenco/Libro

  Restituisce l'insieme di tutti i libri contenuti nell'elenco che si trovano nel documento libri.xml

Esempi di path expressions

```
<?xml version="1.0"?>
<Elenco>
    <Libro disponibilità='S'>
        <Titolo>Il Signore degli Anelli</Titolo>
        <Autore>J.R.R. Tolkien</Autore>
        <Data>2002</Data>
        <ISBN>88-452-9005-0</ISBN>
        <Editore>Bompiani</Editore>
    </Libro>
    <Libro disponibilità='N'>
        <Titolo>Il nome della rosa</Titolo>
        <Autore>Umberto Eco</Autore>
        <Data>1987</Data>
        <ISBN>55-344-2345-1</ISBN>
        <Editore>Bompiani</Editore>
    </Libro>
    <Libro disponibilità='S'>
        <Titolo>Il sospetto</Titolo>
        <Autore>F. Dürrenmatt</Autore>
        <Data>1990</Data>
        <ISBN>88-07-81133-2</ISBN>
        <Editore>Feltrinelli</Editore>
    </Libro>
</Elenco>
```

document("libri.xml")/Elenco/Libro

```
<Libro disponibilità='S'>
    <Titolo>Il Signore degli Anelli</Titolo>
    <Autore>J.R.R. Tolkien</Autore>
    <Data>2002</Data>
    <ISBN>88-452-9005-0</ISBN>
    <Editore>Bompiani</Editore>
</Libro>
<Libro disponibilità='N'>
    <Titolo>Il nome della rosa</Titolo>
    <Autore>Umberto Eco</Autore>
    <Data>1987</Data>
    <ISBN>55-344-2345-1</ISBN>
    <Editore>Bompiani</Editore>
</Libro>
<Libro disponibilità='S'>
    <Titolo>Il sospetto</Titolo>
    <Autore>F. Dürrenmatt</Autore>
    <Data>1990</Data>
    <ISBN>88-07-81133-2</ISBN>
    <Editore>Feltrinelli</Editore>
</Libro>
```

Esempi di path expressions

- Esempio:

  document(“libri.xml”)/Elenco/Libro[Editore=‘Bompiani’ AND @disponibilità=‘S’]/Titolo

  Restituisce l’insieme di tutti i titoli dei libri dell’editore Bompiani che sono disponibili e che si trovano nel documento libri.xml

```
<Titolo>Il Signore degli Anelli</Titolo>
```


Esempi di path expressions

- Esempio:

  document(“libri.xml”)//Autore

  Restituisce l’insieme di tutti gli autori che si trovano nel documento libri.xml annidati a qualunque livello

```
<Autore>J.R.R. Tolkien</Autore>
<Autore>Umberto Eco</Autore>
<Autore>F. Dürrenmatt</Autore>
```

Esempi di path expressions

- Esempio:

  document("libri.xml")/Elenco/Libro[2]/*

  Restituisce gli elementi contenuti nel secondo libro del documento libri.xml

```
<Titolo>Il nome della rosa</Titolo>
<Autore>Umberto Eco</Autore>
<Data>1987</Data>
<ISBN>55-344-2345-1</ISBN>
<Editore>Bompiani</Editore>
```


Espressioni FLWR

- FOR per l'iterazione
- LET per collegare variabili
- WHERE per esprimere predicati
- RETURN per generare il risultato

Espressioni FOR

- Esempio:

  FOR $l IN document(“libri.xml”)//Libro
  RETURN $l

- La clausola FOR valuta l’espressione sulla destra (//Libro) che è un insieme, e itera all’interno di questo set assegnando il nodo di turno alla variabile $l
- L’interrogazione restituisce l’insieme di tutti i libri che si trovano nel documento libri.xml

Espressioni FOR

- Esempio:

  FOR $l IN distinct(document(“libri.xml”)//Libro)
  RETURN $l

- La clausola FOR valuta l’espressione sulla destra (//Libro) che è un insieme, e itera all’interno di questo set assegnando il nodo di turno alla variabile $l
- L’interrogazione restituisce l’insieme di tutti i libri che si trovano nel documento libri.xml

Espressioni FOR

- Le espressioni FOR possono essere annidate:

```
FOR $l IN document("libri.xml")//Libro
    FOR $a IN $l/Autore
 RETURN $a
```

```
<Elenco>
  <Libro disponibilità='S'>
    <Titolo>Designing Data intensive Web
            Applications</Titolo>
    <Autore>Stefano Ceri</Autore>
    <Autore>Piero Fraternali</Autore>
    <Autore>Aldo Bongio</Autore>
    <Autore>Marco Brambilla</Autore>
    <Autore>Sara Comai</Autore>
    <Autore>Maristella Matera</Autore>
    …
  </Libro>
  …
</Elenco>
```


Espressioni LET

- Esempio:

```
LET $l := document("libri.xml")//Libro
RETURN $l
```

- La clausola LET valuta l'espressione (//Libro) e assegna l'intero insieme di libri trovati alla variabile $l
- Il risultato di una clausola LET produce un singolo binding per la variabile: l'intero set viene assegnato alla variabile

Clausola WHERE

- La clausola WHERE esprime una condizione: solamente le tuple che soddisfano tale condizione vengono utilizzate per invocare la clausola RETURN
- Le condizioni nella clausola WHERE possono contenere diversi predicati connessi da AND, OR, NOT
- Esempio:

  FOR $l IN document("libri.xml")//Libro

  WHERE $l/Editore="Bompiani"

  AND $l/@disponibilità="S"

  RETURN $l
- Restituisce tutti i libri pubblicati da Bompiani che sono disponibili

Clausola WHERE

- La clausola WHERE esprime una condizione: solamente le tuple che soddisfano tale condizione vengono utilizzate per invocare la clausola RETURN
- Le condizioni nella clausola WHERE possono contenere diversi predicati connessi da AND, OR, NOT
- Esempio:

  FOR $l IN

  document("libri.xml")//Libro[Editore="Bompiani"

  AND @disponibilità="S"]

  RETURN $l

- Restituisce tutti i libri pubblicati da Bompiani che sono disponibili

http://www.webml.org

Clausola RETURN

- Genera l'output di un'espressione FLWR che può essere:
  - Un nodo

```
<Autore>F. Dürrenmatt</Autore>
```

  - Un foresta ordinata di nodi

```
<Autore>J.R.R. Tolkien</Autore>
<Autore>Umberto Eco</Autore>
<Autore>F. Dürrenmatt</Autore>
```

  - Un valore

```
F. Dürrenmatt
```

- Può contenere dei costruttori di elementi, riferimenti a variabili definite nelle parti FOR e LET ed espressioni annidate

Clausola RETURN

- Un costruttore di elemento consiste di un tag iniziale e di un tag finale racchiudenti una lista opzionale di espressioni che determinano il contenuto dell'elemento
- Esempio:

```
FOR $l IN document("libri.xml")//Libro
 WHERE $l/Editore="Bompiani"
 RETURN  <Libro-Bompiani>
                $l/Titolo
         </Libro-Bompiani>
```

nuovo elemento

```
<Libro-Bompiani><Titolo>Il Signore degli Anelli</Titolo></Libro-Bompiani>
<Libro-Bompiani><Titolo>Il nome della rosa</Titolo></Libro-Bompiani>
```

Clausola RETURN

- Un costruttore di elemento consiste di un tag iniziale e di un tag finale racchiudenti una lista opzionale di espressioni che determinano il contenuto dell'elemento
- Esempio:

```
FOR $l IN document("libri.xml")//Libro
 WHERE $l/Editore="Bompiani"
 RETURN  <Libro-Bompiani>
                    $l/Titolo/text()
              </Libro-Bompiani>
```

```
<Libro-Bompiani>Il Signore degli Anelli</Libro-Bompiani>
<Libro-Bompiani>Il nome della rosa</Libro-Bompiani>
```


Funzioni aggregate

- Esempio:

```
FOR $e IN
  document("libri.xml")//Libro/Editore
LET $l:=
  document("libri.xml")//Libro[Editore
   = $e]
WHERE count($l) > 100
RETURN $e
```

- Restituisce gli editori che nell'elenco hanno almeno 100 libri

Ordinare il risultato

- Esempio:

```
FOR $l IN document("libri.xml")//Libro
RETURN
    <Libro>
      $l/Titolo,
      $l/Editore
    </Libro> SORTBY(Titolo ASCENDING)
```

- I libri vengono ordinati rispetto al titolo

XSL

(modulo 2)

XSL

- Standard W3C: 16 novembre 1999

- XSL = eXtensible Stylesheet Language
- Un foglio di stile è un file in cui sono condensate le specifiche e modalità di presentazione
- si può quindi separare (più o meno nettamente) la definizione dei contenuti dalla loro resa grafica

Fogli di stile

- 1996: DSSSL - standard che definisce fogli di stile per SGML (ISO/IEC 10179:1996)
- 1996-1998: CSS - fogli di stile per HTML - sono attualmente disponibili due specifiche: CSS1 e CSS2 (approvati dal W3C rispettivamente nel Dicembre 1996 e nel Maggio 1998)
- Agosto 1997: Microsoft, Arbortext e Inso Corp. sottopongono la specifica XSL al W3C (sottoinsieme di DSSSL) per dati XML altamente strutturati - e' diventato standard nel 1999

XSL

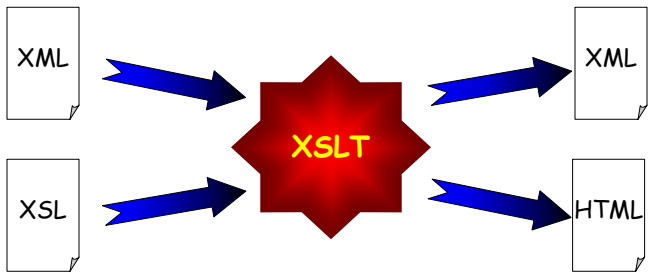
- Linguaggio per rappresentare l'output di XML formattato
- e per convertire un documento XML in un altro documento XML, HTML etc.
- Usa la notazione XML

XSLT

XSL = XSLT + XSL FO

- XSLT (XSL Transformation) Trasforma un documento XML in un altro documento XML o altro tipo di documento (HTML, ecc.)
- Può:
  - aggiungere nuovi elementi;
  - rimuovere elementi presenti; riorganizzare gli elementi;
  - decidere quali visualizzare, ecc.
- XSL FO (Formatting Object) contiene le istruzioni per formattare l'output di un documento XML

XSLT



XSLT

- Trasforma un documento XML in un altro documento XML o altro tipo di documento (HTML, ecc.)
- Può aggiungere nuovi elementi, rimuovere elementi presenti; può riorganizzare gli elementi, decidere quali visualizzare, ecc.

XSLT

- Utilizza Xpath per definire le parti del documento sul quale effettuare le trasformazioni
- Per gli elementi sui quali devono essere applicate le trasformazioni vengono definiti dei template

Template

- Per assegnare uno stile ad un particolare elemento XML oppure per applicare delle trasformazioni si usa un template nel foglio di stile
- Il foglio di stile può essere eseguito da un processore XSLT che scandisce il documento XML sorgente, identifica gli elementi per i quali è stato definito un template nel foglio di stile, ed effettua le azioni specificate nel template.

Esempio di template (1)

```
<xsl:template match=paragrafo>
…
<xsl:template>
```

- La clausola match definisce su quali elementi si applica il template
- Per ogni elemento si possono specificare più template (si applica il più specifico oppure si assegna una priorità per l'applicazione)
- Un template può specificare lo stile per più elementi

Esempio di template (2)

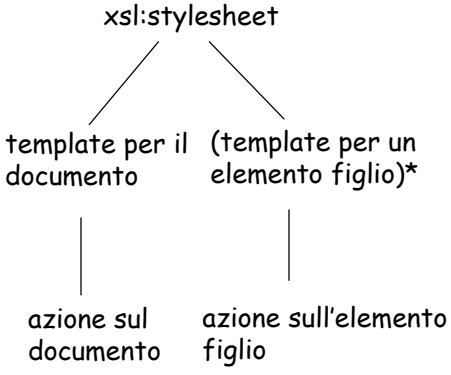
```
<xsl:template match=paragrafo>
  <fo:blocco font-size="10pt" space-before="12pt">
     <xsl:apply-templates/>
  </fo:blocco>
</xsl:template>
```

- All'interno del template si specifica come si devono processare gli elementi figli (xsl:apply-templates)
- Nell'esempio: gli elementi "paragrafo" diventano oggetti di formattazione "blocco" che vengono visualizzati con 10 punti e 12 punti di spazio dopo ogni blocco

Match

- Si possono specificare gli elementi figli da processare utilizzando i seguenti simboli:
- | operatore or
- . Elemento corrente
- // discendenti
- / figlio
- .. Padre
- @ identifica un attributo
- first-of-any(), first-of-type(), last-of-any(), last-of-type()

Struttura di un documento XSL



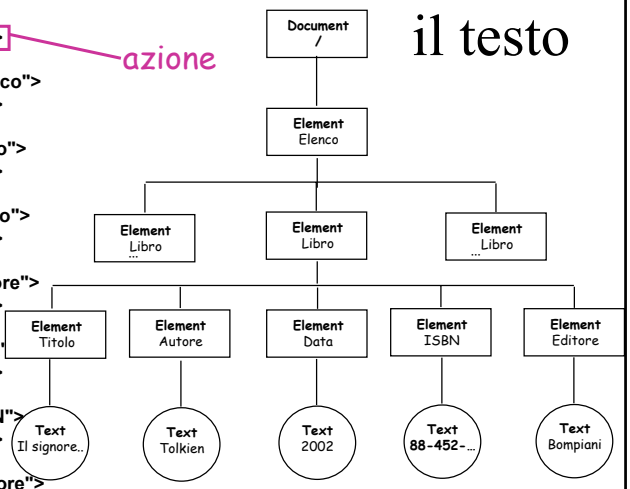

```
<?xml version="1.0"?>
<xsl:stylesheet>
    <xsl:template match="/">
            [azione]
    </xsl:template>
    <xsl:template match="Elenco">
            [azione]
    </xsl:template>
    <xsl:template match="Libro">
            [azione]
    </xsl:template>
    ...
</xsl:stylesheet>
```

Estrarre il testo

```
<?xml version="1.0"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
 version="1.0">
    <xsl:template match="/">
        <xsl:apply-templates/>
    </xsl:template>
    <xsl:template match="Elenco">
        <xsl:apply-templates/>
    </xsl:template>
    <xsl:template match="Libro">
        <xsl:apply-templates/>
    </xsl:template>
    <xsl:template match="Titolo">
        <xsl:apply-templates/>
    </xsl:template>
    <xsl:template match="Autore">
        <xsl:apply-templates/>
    </xsl:template>
    <xsl:template match="Data">
        <xsl:apply-templates/>
    </xsl:template>
    <xsl:template match="ISBN">
        <xsl:apply-templates/>
    </xsl:template>
    <xsl:template match="Editore">
        <xsl:apply-templates/>
    </xsl:template>
    <xsl:template match="text()">
        <xsl:value-of select="."/>
    </xsl:template>
</xsl:stylesheet>
```

azione

azione



http://www.webml.org

Trasformazione

```
<?xml version="1.0"?>
<Elenco>
    <Libro disponibilità='S'>
        <Titolo>Il Signore degli Anelli</Titolo>
        <Autore>J.R.R. Tolkien</Autore>
        <Data>2002</Data>
        <ISBN>88-452-9005-0</ISBN>
        <Editore>Bompiani</Editore>
    </Libro>
    <Libro disponibilità='N'>
        <Titolo>Il nome della rosa</Titolo>
        <Autore>Umberto Eco</Autore>
        <Data>1987</Data>
        <ISBN>55-344-2345-1</ISBN>
        <Editore>Bompiani</Editore>
    </Libro>
    <Libro disponibilità='S'>
        <Titolo>Il sospetto</Titolo>
        <Autore>F. Dürrenmatt</Autore>
        <Data>1990</Data>
        <ISBN>88-07-81133-2</ISBN>
        <Editore>Feltrinelli</Editore>
    </Libro>
</Elenco>
```

```
Il Signore degli Anelli  J.R.R. Tolkien
2002   88-452-9005-0  Bompiani
Il nome della rosa    Umberto Eco
1987   55-344-2345-1   Bompiani
Il sospetto   F. Dürrenmatt   1990
88-07-81133-2   Feltrinelli
```

http://www.webml.org

Creare un documento HTML

```
<?xml version="1.0"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                version="1.0">
 <xsl:template match="/">
        <HTML>
        <HEAD>
        <TITLE>Elenco libri</TITLE>
        </HEAD>
        <BODY>
          <xsl:apply-templates/>
        </BODY>
        </HTML>
   </xsl:template>
   <xsl:template match="Elenco">
        <xsl:apply-templates/>
   </xsl:template>
   <xsl:template match="Libro">
        <xsl:apply-templates/>
   </xsl:template>
   …
</xsl:stylesheet>
```


Trasformazione

```
<?xml version="1.0"?>
<Elenco>
    <Libro disponibilità='S'>
        <Titolo>Il Signore degli Anelli</Titolo>
        <Autore>J.R.R. Tolkien</Autore>
        <Data>2002</Data>
        <ISBN>88-452-9005-0</ISBN>
        <Editore>Bompiani</Editore>
    </Libro>
    <Libro disponibilità='N'>
        <Titolo>Il nome della rosa</Titolo>
        <Autore>Umberto Eco</Autore>
        <Data>1987</Data>
        <ISBN>55-344-2345-1</ISBN>
        <Editore>Bompiani</Editore>
    </Libro>
    <Libro disponibilità='S'>
        <Titolo>Il sospetto</Titolo>
        <Autore>F. Dürrenmatt</Autore>
        <Data>1990</Data>
        <ISBN>88-07-81133-2</ISBN>
        <Editore>Feltrinelli</Editore>
    </Libro>
</Elenco>
```

```
<HTML>
<HEAD>
<TITLE>Book Catalogue</TITLE>
</HEAD>
<BODY>
Il Signore degli Anelli  J.R.R. Tolkien
2002   88-452-9005-0  Bompiani
Il nome della rosa    Umberto Eco
1987   55-344-2345-1   Bompiani
Il sospetto   F. Dürrenmatt   1990
88-07-81133-2   Feltrinelli
</BODY>
</HTML>
```

Il Signore degli Anelli J.R.R. Tolkien 2002 88-452-9005-0 Bompiani Il nome della rosa Umberto Eco 1987 55-344-2345-1 Bompiani Il sospetto F. Dürrenmatt 1990 88-07-81133-2 Feltrinelli

Esempio: creare una tabella

```
<HTML>
<HEAD><TITLE>Elenco libri</TITLE></HEAD>
<BODY>
<TABLE BORDER="1" WIDTH="100%">
 <TR> <TD> Il Signore degli Anelli </TD>
      <TD> J.R.R. Tolkien </TD>
      <TD> 2002 </TD>
      <TD> 88-452-9005-0 </TD>
      <TD> Bompiani </TD>
 </TR>
 <TR> <TD> Il nome della rosa </TD>
      <TD> Umberto Eco </TD>
      <TD> 1987 </TD>
      <TD> 55-344-2345-1 </TD>
      <TD> Bompiani </TD>
 </TR>
 <TR> <TD> Il sospetto </TD>
      <TD> F. Dürrenmatt </TD>
      <TD> 1990 </TD>
      <TD> 88-07-81133-2 </TD>
      <TD> Feltrinelli </TD>
 </TR>
</TABLE>
</BODY>
</HTML>
```

Il Signore degli Anelli	J.R.R. Tolkien	2002	88-452-9005-0	Bompiani
Il nome della rosa	Umberto Eco	1987	55-344-2345-1	Bompiani
Il sospetto	F. Dürrenmatt	1990	88-07-81133-2	Feltrinelli

Esempio: creare una tabella

```
<?xml version="1.0"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                version="1.0">
  <xsl:template match="Elenco">
    <HTML>
    <HEAD>
    <TITLE>Elenco libri</TITLE>
    </HEAD>
    <BODY>
      <TABLE BORDER="1" WIDTH="100%">
        <xsl:apply-templates/>
      </TABLE>
    </BODY>
    </HTML>
  </xsl:template>
  <xsl:template match="Libro">
    <TR>
      <xsl:apply-templates/>
    </TR>
  </xsl:template>
  <xsl:template match="Titolo | Autore | Data | ISBN | Editore">
    <TD>
      <xsl:apply-templates/>
    </TD>
  </xsl:template>
</xsl:stylesheet>
```

Trasformazione XML-XML

```
<?xml version="1.0"?>
<Elenco>
    <Libro disponibilità='S'>
        <Titolo>Il Signore degli Anelli</Titolo>
        <Autore>J.R.R. Tolkien</Autore>
        <Data>2002</Data>
        <ISBN>88-452-9005-0</ISBN>
        <Editore>Bompiani</Editore>
    </Libro>
    <Libro disponibilità='N'>
        <Titolo>Il nome della rosa</Titolo>
        <Autore>Umberto Eco</Autore>
        <Data>1987</Data>
        <ISBN>55-344-2345-1</ISBN>
        <Editore>Bompiani</Editore>
    </Libro>
    <Libro disponibilità='S'>
        <Titolo>Il sospetto</Titolo>
        <Autore>F. Dürrenmatt</Autore>
        <Data>1990</Data>
        <ISBN>88-07-81133-2</ISBN>
        <Editore>Feltrinelli</Editore>
    </Libro>
</Elenco>
```

```
<?xml version="1.0"?>
<NuovoElenco>
    <Libro>
        <Titolo>Il Signore degli Anelli</Titolo>
        <Editore>Bompiani</Editore>
    </Libro>
    <Libro>
        <Titolo>Il nome della rosa</Titolo>
    </Libro>
    <Libro>
        <Titolo>Il sospetto</Titolo>
        <Editore>Feltrinelli</Editore>
    </Libro>
</NuovoElenco>
```

Il nuovo elenco per ogni libro contiene solo il loro titolo e, se questo è disponibile, anche la sua casa editrice.

http://www.webml.org

Trasformazione XML-XML

```
<?xml version="1.0"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
        version="1.0">
<xsl:template match="/">
 <NuovoElenco>
   <xsl:for-each select="Elenco/Libri">
     <Libro>
       <Titolo><xsl:value-of select="Titolo"/> </Titolo>
       <xsl:if test="@disponibilità = 'S'">
         <Editore><xsl:value-of select="Editore"/></Editore>
       </xsl:if>
     </Libro>
   </xsl:for-each>
 </NuovoElenco>
</xsl:template>
```

http://www.webml.org

Associare un foglio di stile ad un documento XML

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl"
  href="file://localhost/EsempiXML-XSL/libri.xsl"?>
<!DOCTYPE Elenco SYSTEM
  "file://localhost/EsempiXML-XSL/libri.dtd">
<Elenco>
      <Libro disponibilità='S'>
            <Titolo>Il Signore degli Anelli</Titolo>
            <Autore>J.R.R. Tolkien</Autore>
            <Data>2002</Data>
            <ISBN>88-452-9005-0</ISBN>
            <Editore>Bompiani</Editore>
      </Libro>
      …
</Elenco>
```

http://www.webml.org

Foglio di stile nel documento XML

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="#stileLibri"?>
<Elenco>
      <xsl:stylesheet id="stileLibri"
            xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
            version="1.0">
        <xsl:template match="xsl:stylesheet"></xsl:template>
        <xsl:template match="/">
          <HTML>
          <HEAD>
          <TITLE>Book Catalogue</TITLE>
          </HEAD>
          <BODY>
            <TABLE BORDER="1" WIDTH="100%">
              <xsl:apply-templates/>
            </TABLE>
          </BODY>
          </HTML>
        </xsl:template>
        ...
      </xsl:stylesheet>
      <Libro disponibilità='S'>
            <Titolo>Il Signore degli Anelli</Titolo>
            <Autore>J.R.R. Tolkien</Autore>
            <Data>2002</Data>
            <ISBN>88-452-9005-0</ISBN>
            <Editore>Bompiani</Editore>
      </Libro>
      …
</Elenco>
```

si aggiunge l'istruzione per inserire il foglio di stile, che si riferisce ad un elemento interno.

Il foglio di stile viene racchiuso all'interno di questo elemento.