Laboratorio di sistemi interattivi

Lezione 11: Introduzione a XML

Terminologia

- XML = eXtensible Markup Language
- DTD = Document Type Definition
- XSL = eXtensible Stylesheet Language
- XSLT = eXtensible Stylesheet Language Transformation

XML: descrivere caratteristiche arbitrarie di un documento

- HTML ha insieme fisso di tag. Descrivono:
  - struttura del documento
  - deleghe a gestori di comportamenti (applet o script)
- XML permette di definire e usare tag arbitrari
- XSL permette di specificare cosa fare in presenza di determinati tag

XML è un (meta)linguaggio di markup

<someTag someAttribute = someValue>
Contenuto di someTag
</someTag>

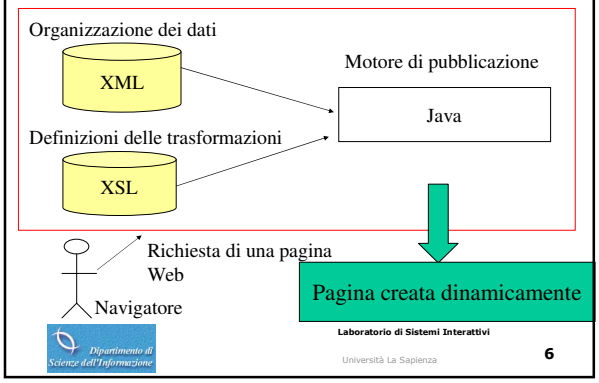
- Linguaggi specifici hanno insieme di tag predefiniti e un interprete in grado di decodificarli (e.g. HTML Browser)
- Metalinguaggi permettono definizione e utilizzo di tag arbitrari
- Interprete in grado di comprendere le definizioni.
- Definizioni devono rispettare una certa sintassi.

Differenze fra XML e HTML

XML		HTML
Descrizione informazioni in un documento	⇒	Informazioni non possono essere descritte in modo efficace
Permette definizione di nuovi tag e attributi	⇒	Tag e attributi fissati
Un documento XML può contenere descrizione della propria grammatica	⇒	Sintassi fissa

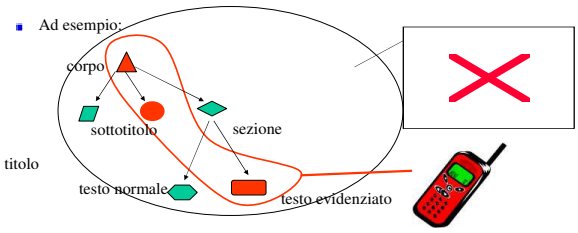
Linguaggio per creazione di altri linguaggi

Esempio di utilizzo di XML sul web



Esempio di utilizzo di XML su Web

- In funzione delle dimensioni dello schermo o delle esigenze dell'utente è possibile selezionare le informazioni desiderate (tipo, rilevanza)
  - Ad esempio:



Struttura XML

Esempio

Regole Sintassi A
(DTD o XMLSchema)

Contenuti A1

Contenuti A2

Contenuti A3

Un articolo deve essere composto al max da 2 parole

Il cane

Il gatto

Sdjh sdj

Il topo mangia

Sintatticamente giusto

sbagliato

Verifica dati

Regole Sintassi A

Contenuti A1

Contenuti A2

Contenuti A3

Livello intensionale
Organizzazione del documento

Livello estensionale
Contenuti

Documento ben formato se rispetta le specifiche XML

Documento è valido se segue tutte le regole imposte al livello estensionale ed è ben formato

Well-formedness e validity

Documento è ben formato se scritto in accordo con la grammatica di XML

```
<?xml version="1.0"?> <greeting>Hello, world!</greeting>
```

Documento è valido se l'interprete è in grado di verificare la correttezza dell'uso dei tipi

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE greeting [
  <!ELEMENT greeting (#PCDATA)>
]>
<greeting>Hello, world!</greeting>
```


Grammatica di XML

- document ::= prolog element Misc*
- prolog ::= XMLDecl? Misc* (doctypedecl Misc*)?
- XMLDecl ::= '<?xml' VersionInfo EncodingDecl? SDDecl? S? '?>'
- VersionInfo ::= S 'version' Eq ("'" VersionNum "'" | '"' VersionNum '")/* */
- Eq ::= S? '=' S?
- VersionNum ::= ([a-zA-Z0-9_.:] | '-')+
- Misc ::= Comment | PI | S

Grammatica di XML

- Elementdecl ::= '<!ELEMENT' S Name S contentspec S? '>'
- element ::= EmptyElemTag| STag content ETag
- S = white spaces
- PI = Processing instructions

Vincoli di validità

element ::= EmptyElemTag| STag content ETag
[WFC: Element Type Match]
[VC: Element Valid]

Well-formedness constraint: Element Type Match

Il Name nel tag finale di un elemento deve corrispondere al tipo dell'elemento nel tag di partenza.

Validity constraint: Element Valid I

- Un elemento è valido se c'è una dichiarazione corrispondente a elementdecl, in cuii il Name corrisponde al tipo dell'elemento e si verifica uno dei seguenti casi:

1. La dichiarazione è **EMPTY** e l'elemento non ha contenuto.

Validity constraint: Element Valid II

2. La dichiarazione corrisponde a children e la sequenza dei child elements appartiene al linguaggio definito dall'espressione regolare nel modello del contenuto, con spazi bianchi opzionali tra il tag di partenza e il primo elemento figlio, tra gli elementi figlio o tra l'ultimo elemento figlio e il tag finale. Una sezione CDATA contenente solo spazio bianchi non corrisponde al nonterminale S, e quindi non può apparire in queste posizioni.

Validity constraint: Element Valid III

3. La dichiarazione corrisponde a Mixed e il contenuto consiste di character data e child elements il cui tipo corrisponde a nomi nel modello del contenuto.
4. La dichiarazione corrisponde a **ANY**, e i tipi di tutti i child elements sono stati dichiarati.

Tipi di attributi

- AttType ::= StringType | TokenizedType | EnumeratedType
- StringType ::= 'CDATA'
- TokenizedType ::= 'ID' [VC: ID]
- [VC: One ID per Element Type]
- [VC: ID Attribute Default]
- | 'IDREF' [VC: IDREF]
- | 'IDREFS' [VC: IDREF]
- | 'ENTITY' [VC: Entity Name]
- | 'ENTITIES' [VC: Entity Name]
- | 'NMTOKEN' [VC: Name Token]
- | 'NMTOKENS' [VC: Name Token]

Tipi di attributi

EnumeratedType ::= NotationType | Enumeration

NotationType ::= 'NOTATION' S '(' S? Name (S? '|' S? Name)* S? ')'
[VC: Notation Attributes]
[VC: One Notation Per Element Type]
[VC: No Notation on Empty Element]

Enumeration ::= '(' S? Nmtoken (S? '|' S? Nmtoken)* S? ')'
[VC: Enumeration]

Document Type Definition

Definizione degli elementi utilizzati nel documento XML (cioè dei nuovi tag).

Può essere contenuta nel documento stesso o in un documento a parte. (Diversi documenti possono basarsi sulla stessa DTD.)

```
<?xml version="1.0"?> <!DOCTYPE greeting SYSTEM "hello.dtd">
<greeting>Hello, world!</greeting>
```

La DTD deve essere il primo elemento del documento.

XML processors

- Non validating:
  - controllano solo la correttezza sintattica
- Validating:
  - controllano la conformità del documento al DTD
- La validazione richiede di potere risolvere ogni riferimento a definizioni, anche esterne.

Un esempio di documento che non richiede validazione

```
<?xml version="1.0" standalone="yes"?>
<library>
   <book call_no="PZ3.S8195Gr6">
      <cover href="grapes.gif" alt="Grapes of Wrath"/>
      <title>The Grapes of Wrath</title>
      <author>
        <last_name>Steinbeck</last_name>
        <first_name>John</first_name>
      </author>
      <publisher>Viking Press</publisher>
      <pub_year>1939</pub_year>
   </book>
</library>
```

Uso di DTD (interna al documento)

```
<?xml version="1.0" standalone="yes"?>

<!DOCTYPE library [

 <!ELEMENT      library   (book+)                              >
 <!ELEMENT      book      (title,(author|editor)+,publisher,year) >
 <!ATTLIST      book      call_no     CDATA     #REQUIRED      >
 <!ELEMENT      title     (#PCDATA)                            >
 <!ELEMENT      author    (last_name,first_name)               >
 <!ELEMENT      editor    (last_name,first_name)               >
 <!ELEMENT      last_name (#PCDATA)                            >
 <!ELEMENT      first_name (#PCDATA)                           >
 <!ELEMENT      publisher (#PCDATA)                            >
 <!ELEMENT      year      (#PCDATA)                            >
]>
```


Uso di DTD (interna al documento)

```
<library>
   <book call_no="PZ3.S8195Gr6">
      <title>The Grapes of Wrath</title>
      <author>
         <last_name>Steinbeck</last_name>
         <first_name>John</first_name>
      </author>
      <publisher>Viking Press</publisher>
      <year>1939</year>
   </book>
</library>
```


Uso di DTD (esterna al documento)

File library.dtd:

```
<!ELEMENT   library (book+)                              >
<!ELEMENT   book (title,(author|editor)+,publisher,year) >
<!ATTLIST   book call_no CDATA #REQUIRED                 >
<!ELEMENT   title (#PCDATA)                              >
<!ELEMENT   author (last_name,first_name)                >
<!ELEMENT   editor (last_name,first_name)                >
<!ELEMENT   last_name   (#PCDATA)                        >
<!ELEMENT   first_name  (#PCDATA)                        >
<!ELEMENT   publisher   (#PCDATA)                        >
<!ELEMENT   year    (#PCDATA)                            >
```

cdata = character data
pcdata = parsed character data

Uso di DTD (esterna al documento)

File myLibrary.xml

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE  library  SYSTEM  "library.dtd" >
<library>
   <book call_no="PZ3.S8195Gr6">
        <title>The Grapes of Wrath</title>
        <author>
              <last_name>Steinbeck</last_name>
              <first_name>John</first_name>
        </author>
        <publisher>Viking Press</publisher>
        <year>1939</year>
   </book>
</library>
```


XML Schema

- Definisce il tipo di elementi presenti in un documento XML
- Espresso usando la sintassi XML
- Permette di utilizzare spazi di nomi (si possono fondere documenti XML diversi)

Sintassi XML-Schema

Riferimento alla definizione corrente

```
<xsd:schema
xmlns:xsd="http://www.w3.org/1999/XMLSchema"
```

Definizione dello spazio dei nomi

```
xmlns:JavaXML="http://www.oreylly.com/catalog/javaxml"
```

Spazio dei nomi per il documento bersaglio

```
targetNamespace =
"http://www.oreylly.com/catalog/javaxml"
>
```

Definizione degli elementi

```
<element
      name = "[Name of Element]"
      type = "[Type of Element]"
      [Options ….]
>
```

e.g.

```
<element name = "Title" type = "String">
<element name = "Heading" type = "String">
```

Tipi possono essere predefiniti o user-defined

Tipi definiti dall'utente

```
<complexType
      name = "ChapterType"
      <element name = "Title" type = "String">
      <element name = "Heading" type =
  "String">
</complexType>
```

Uso

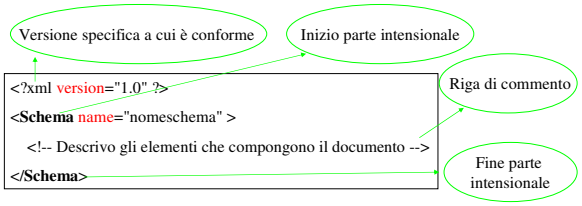
```
<element name = "Chapter" type =
  "JavaXML:ChapterType" />
```


Tipi impliciti

- Tipi con nome possono essere usati ovunque
- Tipi impliciti possono essere usati in riferimento a un solo elemento

```
<complexType name = "ContentsType">
   <element name = "Chapter">
     <complexType>
          <element name = "Title" type = "String">
          <element name = "Heading" type = "String">
     </complexType>
   </element>
   <element name = "SectionBreak">
     <complexType content = "empty" />
   </element>
</complexType>
```

Livello intensionale



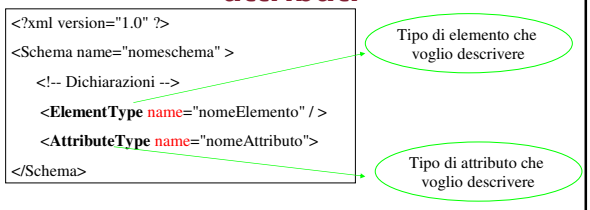
Well-formedness: Ogni tag aperto deve essere anche chiuso, Tag semplice: **<NomeTag ……. />**

Tag composto: **<NomeTag …. >**

… altre dichiarazioni…

</NomeTag>

Dichiarazione di elementi e attributi



È obbligatorio sia in *ElementType* che in *AttributeType* specificare *name*.

Le dichiarazione degli attributi sono indipendenti dalle dichiarazione degli elementi.

Dichiarazione di attributi

- Una dichiarazione di un tipo attributo può includere anche un attributo default, che indica il valore predefinito dell'attributo.
- Una dichiarazione di un tipo attributo può includere un attributo required, che specifica se è necessario che l'attributo abbia un valore

```
<?xml version="1.0" ?>
<Schema name="nomeschema" >
    <ElementType name="nomeElemento" / >
    <AttributeType name="nomeAttributo" default="…" required="yes" >
</Schema>
```

Validità dichiarazioni

Le dichiarazioni possono essere globali o locali.

Una dichiarazione è globale se è dichiarata all'interno dell'elemento <Schema>.

Una dichiarazione che appare all'interno di un'altra dichiarazione è considerata locale ed è valida solo all'interno della dichiarazione in cui è contenuta

Nidificazione

È il processo che consente di incorporare un oggetto (un elemento) all'interno di un'altro. Un elemento secondario deve essere completamente contenuto nell'elemento primario.

Corretto

<Elemento Primario……. />
<Elemento Secondario …. >
… altre dichiarazioni…
</Elemento Secondario>
</Elemento Primario>

Errato

<Elemento Primario……. />
<Elemento Secondario …. >
… altre dichiarazioni…
</Elemento Primario>
</Elemento Secondario>

Esempio schema

C:\appoggio\libro1.xml

Parte intensionale:
Descrizione caratteristiche libro

```
<?xml version="1.0" ?>
<Schema name="libro" xmlns="urn:schemas-microsoft-com:xml-data">
    <AttributeType name="tipoCarattere">
    <ElementType name="titolo"  >
        <attribute type="tipoCarattere"/>
    </ElementType >
    <ElementType name="indice"  / >
<ElementType name="libro"  / >
    <element  name="titolo" />
</ElementType >
</Schema>
```

Elemento *titolo* ha attributo *tipoCarattere*

Chiusura tag *titolo*

Fine parte intensionale.
Chiudo tag Schema

Esempio contenuti (fase estensionale)

ContenutiA.xml

```
<?xml version="1.0" ?>
<libro xmlns="x-schema:c:/appoggio/libro1.xml">
    <titolo tipoCarattere="Arial">La pesca</titolo>
</libro>
```

Imposto regole

Meta tag

ContenutiB.xml

```
<?xml version="1.0" ?>
<libro xmlns="x-schema:c:/appoggio/libro1.xml">
    <titolo tipoCarattere="Courier">La caccia</titolo>
</libro>
```

Ci sono più istanze di contenuti che soddisfano le stesse regole.

Esempio contenuti

```
<?xml version="1.0" ?>
<libro xmlns="x-schema:c:/appoggio/libro1.xml">
    <titolo tipoCar="Arial">La pesca</titolo>
</libro>
```

Non è valido ! (Perché?)
È ben formato?

```
<?xml version="1.0" ?>
<libro xmlns="x-schema:c:/appoggio/libro1.xml">
    <titolo tipoCarattere="Arial">La pesca</titolo>
<libro>
```

Non è ben formato! (Perché?)

(allora neanche valido)

```
<?xml version="1.0" ?>
<libro xmlns="x-schema:c:/appoggio/libro1.xml">
    <titolo tipoCarattere="aaaa">papapap</titolo>
</libro>
```

Questo è corretto?

Contenuto di un elemento

- Tramite l'attributo content di ElementType posso indicare il tipo di contenuto:
  - **empty -**non contiene alcun tipo di contenuto
  - **textOnly -** contiene solo testo
  - **eltOnly -** contiene solo sottoelementi
  - **mixed -** contiene una combinazione di testo e sottoelementi

```
<?xml version="1.0" ?>
<Schema name="nomeschema" >
    <!-- Dichiarazioni -->
    <ElementType name="nomeElemento" content=".."/ >
</Schema>
```

Ordinamento del contenuto

- Tramite l'attributo order di ElementType posso indicare la sequenza dei contenuti:
  - **seq -** stessa sequenza dichiarazione (default per eltOnly)
  - **one -** un sottoelemento deve essere dichiarato nell'elemento principale
  - **all -** un elemento di ogni tipo dichiarato deve comparire come sottoelemento, sottoelementi possono avere ordine qualsiasi
  - **many -** ogni elemento dichiarato può avere ordine qualsiasi (default per mixed)

```
<Schema name="nomeschema" >
    <ElementType name="nomeElemento" content=".." order=".."/ >
</Schema>
```


Occorrenze

Attributi di element
- minOccurs
- maxOccurs

Assenza di restrizione con valore "*"

- Valori di attributi possono essere #IMPLIED, #FIXED, #REQUIRED

Numero occorrenze

```
<ElementType name="nomeelemento" content=".." order=".."/ >
    <element type="elementoInterno" minOccurs=".." maxOccurs=".."/>
</ElementType >
```

minOccurs	maxOccurs	Ripetizioni
1 o nulla	1 o nulla	1 (necessario)
>1	>n	minOccurs < x < maxOccurs
>1	<1	0
0	"*"	Qualsiasi
1	"*"	Almeno 1
>0	"*"	Almeno minOccurs
Qualsiasi	0	0

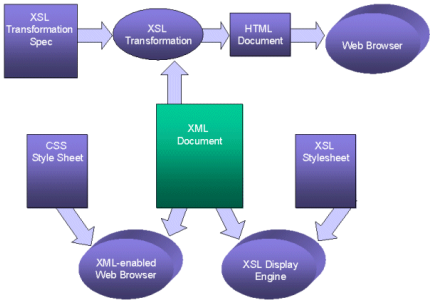
Linguaggi derivati da XML

- Si riferiscono a domini specifici
- Sono basati su un DTD o su uno Schema
- Sono associati a un interprete

XSL: traduzione del contenuto

- Due componenti:
  - traduzione verso altri linguaggi (oggi chiamata XSLT)
  - formattazione delle componenti

XSL: traduzione del contenuto



Specifica di XSL

<xsl:stylesheet
  id = *id*
  extension-element-prefixes = *tokens*
  exclude-result-prefixes = *tokens*
  version = *number*>
  <!-- Content: (xsl:import*, *top-level-elements*) -->
</xsl:stylesheet>

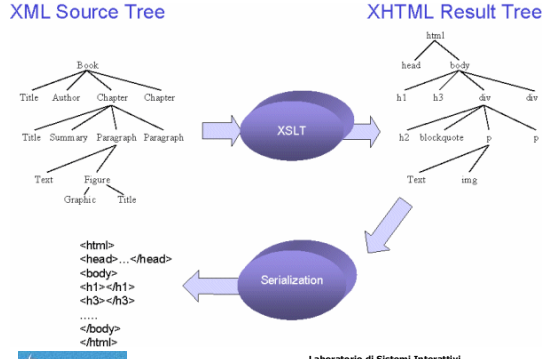
Usa template specificati con espressioni XPath per localizzare elemento sull'albero XML

ancestor'| 'ancestor-or-self'| 'attribute'| 'child'| 'descendant'| 'descendant-or-self'| 'following'| 'following-sibling'| 'namespace'| 'parent'| 'preceding'| 'preceding-sibling'| 'self'

Specifica di XSL

<xsl:import **href** = *uri-reference* />
<xsl:include **href** = *uri-reference* />
<xsl:strip-space **elements** = *tokens* />
<xsl:preserve-space **elements** = *tokens* />
xsl:output … // caratteristiche dell'albero di output
xsl:key … // possibilità di riferimenti ad altri nodi
xsl:decimal-format … // informazioni su formattazione dei numeri
xsl:namespace-alias …
xsl:attribute-set … // definizione di un insieme di attributi
xsl:variable … // valori definiti da espressioni
xsl:param … // si può definire un default
<xsl:template
  match = *pattern*
  name = *qname*
  priority = *number*
  mode = *qname*
  <!-- Content: (xsl:param*, *template*) -->
</xsl:template>

L'albero della traduzione



XSL per l'esempio della biblioteca

```
<?xml version="1.0"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/TR/WD-xsl">
<xsl:template match="/"> <xsl:apply-templates/> </xsl:template>

<xsl:template match="library">
 <html>
  <head>
   <title>XML Library Book List</title>
  </head>
  <body bgcolor="#ffffff">
   <h2 align="center">Library Books in XML Database</h2>
   <xsl:apply-templates/>
  </body>
 </html>
</xsl:template>
```


XSL per l'esempio della biblioteca

```
<xsl:template match="book">
  <b><xsl:value-of select="title"/></b><br />
  Call #: <xsl:value-of select="@call_no"/><br />
  <xsl:for-each select="author|editor">
    by  <xsl:value-of select="first_name"/>
        <xsl:value-of select="last_name"/><br />
  </xsl:for-each>
  <xsl:apply-templates/>
</xsl:template>

<xsl:template match="year"> <xsl:value-of select="."/><p></p>
</xsl:template>

</xsl:stylesheet>
```


XML modificato per usare XSL

```
<?xml version="1.0" standalone="no"?>
<?xml-stylesheet type="text/xsl" href="library.xsl"?>
<!DOCTYPE  library  SYSTEM  "library.dtd" >
<library>
  <book call_no="PZ3.S8195Gr6">
    <title>The Grapes of Wrath</title>
    <author>
     <last_name>Steinbeck</last_name>
     <first_name>John</first_name>
    </author>
    <publisher>Viking Press</publisher>
    <year>1939</year>
    </book>
</library>
```