PROGRAMMAZIONE WEB

EXTENSIBLE MARKUP LANGUAGE (XML)

Prof. Ada Bagozi
ada.bagozi@unibs.it

Ing. Paola Magrino
paola.magrino@unibs.it

Introduzione a 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: W3C Recommendation

HTML vs XML (I)

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

```
<h1> Il cosmo di Einstein </h1>
 <ul>
  <li> Autore: Kaku Michio
  <li> Editore: Codice
  <li> Anno: 2005
  <li> ISBN: 8875780153
 </ul>
```

```
<book>
 <title>Il cosmo di
    Einstein</title>
 <author>Kaku Michio</author>
 <editor>Codice</editor>
 <year>2005</year>
 <isbn>8875780153</isbn>
</book>
```

HTML vs XML (II)

XML e HTML non sono in alternativa, sono nati con scopi diversi!

L'HTML è un caso particolare di XML

- ✓ XML progettato per descrivere i dati (cosa sono i dati)
  - ✓ scambiare e condividere dati e informazioni tra sistemi eterogenei (B2B, B2C,etc.)
  - ✓ creare nuovi linguaggi (WML, MathML…)
- ✓ HTML progettato per visualizzare i dati (come appaiono i dati)
  - ✓ separare i dati dal modo con cui vengono presentati

Recentemente è stato introdotto l'XHTML, che combina l'HTML con le rigide regole sintattiche dell'XML, permettendone una maggiore interoperabilità e l'utilizzo anche su dispositivi a minore capacità

Sintassi XML

- ✓ Ogni documento deve iniziare con `<?xml version="1.0"?>`
- ✓ E' un linguaggio di markup, quindi ne rispetta tutte le regole generali
- ✓ I tag sono "case sensitive"
- ✓ Un documento XML deve avere un tag radice

- ✓ Un documento XML è *ben formato* se soddisfa le regole di sintassi dell'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>
```

Dichiarazione iniziale

Elemento radice

Attributo

Elemento con solo testo

Elemento con altri tag annidati

Elementi: modello di contenuto

- ✓ Gli elementi possono avere diversi tipi di contenuto
  - ✓ possono contenere altri elementi annidati (*element content*)

```
<book>
  <publishing year="2005"/>
  <title> Il cosmo di Einstein </title>
  <author> Kaku Michio </author>
  <part> Traduzione a cura di P. Bonini
    <chapter> L'eredità di Albert Einstein </chapter>
  </part>
</book>
```

Elementi: modello di contenuto

- ✓ Gli elementi possono avere diversi tipi di contenuto
  - ✓ possono contenere altri elementi annidati (*element content*)
  - ✓ possono contenere solo testo (*text content*)

```
<book>
  <publishing year="2005"/>
  <title> Il cosmo di Einstein </title>
  <author> Kaku Michio </author>
  <part> Traduzione a cura di P. Bonini
    <chapter> L'eredità di Albert Einstein </chapter>
  </part>
</book>
```

Elementi: modello di contenuto

- ✓ Gli elementi possono avere diversi tipi di contenuto
  - ✓ possono contenere altri elementi annidati (*element content*)
  - ✓ possono contenere solo testo (*text content*)
  - ✓ possono contenere sia elementi annidati che testo (*mixed content*)

```
<book>
  <publishing year="2005"/>
  <title> Il cosmo di Einstein </title>
  <author> Kaku Michio </author>
  <part> Traduzione a cura di P. Bonini
    <chapter> L'eredità di Albert Einstein </chapter>
  </part>
</book>
```

Elementi: modello di contenuto

- ✓ Gli elementi possono avere diversi tipi di contenuto
  - ✓ possono contenere altri elementi annidati (*element content*)
  - ✓ possono contenere solo testo (*text content*)
  - ✓ possono contenere sia elementi annidati che testo (*mixed content*)
  - ✓ possono essere vuoti (*empty content*) con o senza attributi

```
<book>
  <publishing year="2005"/>
  <title> Il cosmo di Einstein </title>
  <author> Kaku Michio </author>
  <part> Traduzione a cura di P. Bonini
    <chapter> L'eredità di Albert Einstein </chapter>
  </part>
</book>
```

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
- ✓ Limitazioni nell’uso di attributi
  - ✓ non possono contenere valori multipli
  - ✓ non possono descrivere strutture

Document Type Definition (DTD)

- ✓ Detta il formato comune per una classe di documenti XML, cioè:
  - ✓ gli elementi ammessi
  - ✓ le regole di annidamento degli elementi, gli attributi e il contenuto ammesso per ciascun elemento
- ✓ Scopi:
  - ✓ accordarsi su formato/struttura dei documenti
  - ✓ validare documenti XML secondo certe regole

Un documento XML è **valido** rispetto ad un DTD se rispetta il formato specificato

Tipi di dichiarazioni in un DTD

ELEMENT: introduce il nome dell'elemento e il suo contenuto ammissibile

ATTLIST: specifica gli attributi ammessi per un dato elemento e le proprietà di questi attributi (tipo e vincoli sugli attributi)

ENTITY: simile ad una dichiarazione di costante, si riferisce ad una particolare porzione di documento XML

Dichiarazione di elementi (I)

Elementi contenenti altri elementi figli

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

Elementi con PCDATA (*parsed character data* = porzione testo qualsiasi)

```
<!ELEMENT DESCRIZIONE (#PCDATA)>
<descrizione> testo </descrizione>
```

Elementi vuoti

```
<!ELEMENT ARTICOLO EMPTY>
<articolo/>
```

Dichiarazione di elementi (II)

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?)>
```

Dichiarazione di attributi

Per ogni elemento il DTD dice:

- ✓ quali attributi può avere il tag
- ✓ che valori può assumere ciascun attributo
- ✓ eventuali vincoli sulla cardinalità degli attributi
- ✓ qual è il valore di default

Esempio di dichiarazione di attributo:

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

Tipi di attributi

CDATA: stringa

ID: chiave unica

IDREF, IDREFS: riferimento ad uno o più ID nel documento

ENTITY, ENTITIES: nome di una o più entità

NMTOKEN, NMTOKENS: caso ristretto di CDATA (una stringa di una o più parole separate da spazi)

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

Vincoli sugli attributi

#REQUIRED: il valore deve essere specificato

#IMPLIED: l'attributo può avere un valore e il valore di default non è definito

"val": il valore dell'attributo è "val" se nient'altro è specificato

#FIXED "val": il valore, se presente, deve coincidere con "val"

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

Dichiarazione di un'entità

Analoghe alle dichiarazioni di macro con **#define** in C

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

Le entità possono essere

- ✓ interne (`&ATI;`)
- ✓ esterne (`&boilerplate;`)

Documenti XML con DTD


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

 <!DOCTYPE elenco SYSTEM ”libro.dtd“>
 <!DOCTYPE elenco [
    <!ELEMENT ELENCO (PRODOTTO+)>
    <!ELEMENT PRODOTTO (DESCRIZIONE, PREZZO?)>
    <!ELEMENT DESCRIZIONE (#PCDATA)>
    <!ELEMENT PREZZO (#PCDATA)>
    <!ATTLIST PRODOTTO codice ID #REQUIRED>
   ]>

 <elenco>...</elenco>
```

DTD esterno

Esempio di DTD


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

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

Esempio di DTD


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

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

NOTA: un DTD NON è un documento XML

XML Schema

Storia: inizialmente proposto da Microsoft, è divenuto W3C recommendation (maggio 2001)

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

Elementi semplici

Possono contenere solo testo (nessun sottoelemento o attributo)

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"/>
```

Elementi semplici in un documento XML

```
<età> 65 </età>
<cognome> Rossi </cognome>
```

Valore di default o fisso

Elementi complessi


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

Vincoli di cardinalità

Numero di occorrenze:
maxOccurs: max numero di occorrenze
minOccurs: min numero di occorrenze

Se non specificati: 1 e 1 sola occorrenza

```
<xs:element name="Nome" type="xs:string" maxOccurs="4" minOccurs="1"/>
```

Raggruppamento

Per definire gruppi di elementi (o attributi), tra loro correlati (group name)

```
<xs:group name="completeName">
   <xs:sequence>
      <xs:element name="Cognome" type="xs:string"/>
      <xs:element name="Nome" type="xs:string"/>
   </xs:sequence>
<xs:group>
...
<xs:group ref="completeName"/>
```

Restrizioni

E' possibile imporre delle restrizioni sui valori dei tipi semplici tramite l'uso di *"facet"*

- ✓ Valore max/min
- ✓ Enumerazione di valori
- ✓ Pattern di caratteri ammessi
- ✓ Lunghezza di liste di valori

Le restrizioni sono applicate al range di valori ammessi per questo tipo

```
<xs:attribute name="format">
   <xs:simpleType>
      <xs:restriction base="xs:string">
         <xs:enumeration value="paperback"/>
         <xs:enumeration value="hardback"/>
      <xs:restriction>
   <xs:simpleType>
</xs:attribute>
```

Facet + valore

Riferimenti ad altri elementi

Attraverso l'attributo **"ref"** ci si può riferire ad un elemento definito altrove

```
...
<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"/>
```

Esempio di 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>
```

XHTML

Si tratta di una versione più “controllata” dell'HTML, che segue le regole dell'XML

- ✓ annidamento corretto degli elementi `<i><b>...</b></i>`
- ✓ tag di chiusura `<p>...</p>`
- ✓ gestione ordinata dei tag vuoti `<br/>`
- ✓ *case sensitivity* per i nomi dei tag e degli attributi (minuscolo)
- ✓ Valori di attributi tra virgolette ed espliciti `checked="checked"`

Compatibilità cross-browser (I)

I Browser supportano due modalità di rendering: Standards mode and Quirks mode

- ✓ Lo Standards mode lavora seguendo il più possibile le specifiche del W3C, quindi in maniera (quasi) indipendente dal browser
- ✓ Il Quirks mode segue le regole di formattazione dello specifico browser, con le sue limitazioni ed estensioni.

La modalità Quirks esiste per rendere i browser compatibili con i vecchi siti web, che erano sviluppati con codice molto browser-dipendente. Oggi, è necessario sviluppare i nuovi siti in modalità Standards

Compatibilità cross-browser (II)

(!) Di default i browser usano la modalità Quirks

Per entrare in modalità Standards occorre inserire all'inizio del documento una dichiarazione doctype come quella che segue

- ✓ Per usare l'XHTML transitional:

HTML5 introduce una semplificazione

```
<!DOCTYPE html>
```

- ✓ Per usare l'XHTML strict:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
```

Per saperne di più…

- ✓ Tutorial W3C
  - Su XML: https://www.w3schools.com/xml
  - Su DTD: https://www.w3schools.com/xml
  - Su XSD: https://www.w3schools.com/xml/schema_intro.asp
  - Su XHTML: https://www.w3.org/xhtml1/

PROGRAMMAZIONE WEB

EXTENSIBLE MARKUP LANGUAGE (XML)

Prof. Ada Bagozi
ada.bagozi@unibs.it

Ing. Paola Magrino
paola.magrino@unibs.it