Basi di dati - Laboratorio

Corso di Laurea in Bioinformatica

Docente: Barbara Oliboni

Lezione 8

Contenuto della lezione

- eXtensible Markup Language (XML)
  - HTML vs XML
  - Documenti ben formati
  - Documenti validi
    - Document Type Definition (DTD)
    - Validazione
- XML e BIO

eXtensible Markup Language

- XML è un linguaggio di marcatura proposto dal W3C
- XML definisce una sintassi generica per contrassegnare i dati di un documento elettronico con marcatori (tag) semplici e leggibili
- La sintassi XML viene utilizzata in contesti molto diversi:
  - pagine web
  - scambio di dati elettronici
  - grafica vettoriale
  - cataloghi di prodotti
  - sistemi di gestione di messaggi vocali
  - …

`http://www.w3c.org/XML`

Caratteristiche di XML

- XML consente di
  - descrivere i dati in modo semplice e flessibile
  - ottenere documenti manipolabili automaticamente
- XML viene usato come formato di scambio di dati su Web
- XML può essere usato per descrivere l'informazione estratta da basi di dati

XML: evoluzione

- 1986: Standard Generalized Markup Language (SGML)
  - Linguaggio di marcatura strutturato, dotato di semantica, per documenti di tipo testuale
- 1995: HyperText Markup Language (HTML)
  - Applicazione di SGML che permette di descrivere come il contenuto di un documento verrà presentato
- 1998: eXtensible Markup Language (XML)
  - Versione “leggera” di SGML

HTML vs XML

- HTML
  - Insieme fisso di tag
  - Descrizione di come il documento verrà presentato
  - Usato solo per la costruzione di pagine web
- XML
  - Insieme non fisso di tag: i tag possono essere personalizzati
  - Descrizione del contenuto del documento
  - Usato in molti domini diversi

HTML vs XML: esempio

```
<h1>
   XML <br> Guida di
   riferimento
</h1>

<ul>
   <li>
        Elliotte Rusty Harold
   </li>
   <li>
        W. Scott Means
   </li>
</ul>

<i>
   APOGEO – O'REILLY
</i>
```

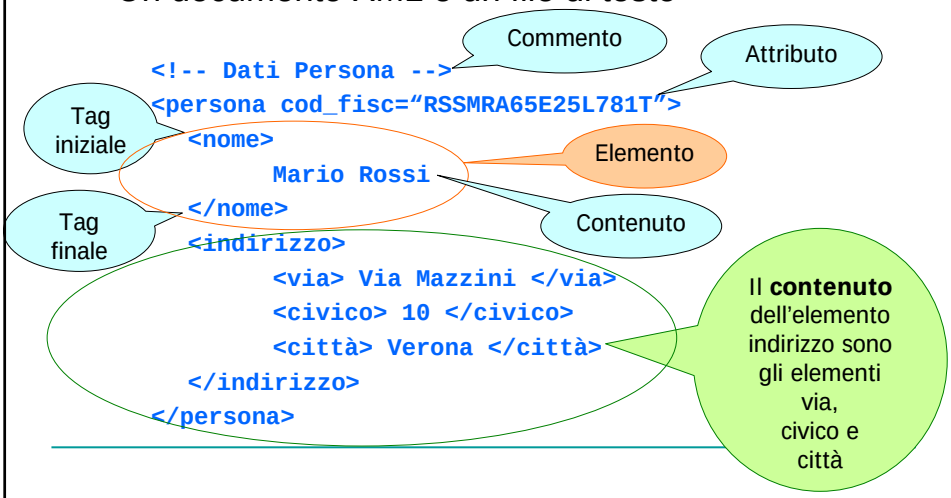
```
<titolo>
   XML Guida di riferimento
</titolo>

<autore>
   Elliotte Rusty Harold
</autore>
<autore>
   W. Scott Means
</autore>

<casa editrice>
   APOGEO – O'REILLY
</casa editrice>
```

Documenti *ben formati*

- XML è più restrittivo di HTML per quanto riguarda il posizionamento dei tag e il modo in cui vengono scritti
- Ogni documento XML deve essere ben formato
  - Ha una sola radice
  - L'innestamento dei tag deve essere corretto
  - Tutti i tag aperti devono essere chiusi
  - I valori degli attributi devono essere specificati tra virgolette

Esempio di documento XML: persona.xml

- Un documento XML è un file di testo



Sintassi dei tag

- Tag iniziale
  `<nome>`
- Tag finale
  `</nome>`
- Elementi vuoti: sono elementi privi di contenuto
  `<maschio></maschio>` oppure `<maschio/>`
- XML è sensibile alla differenza tra maiuscole e minuscole (è case-sensitive)

Contenuto di un elemento

- Contenuto di tipo carattere
  ```
  <nome>Mario Rossi</nome>
  ```
- Contenuto costituito da altri elementi (figli)
  ```
  <indirizzo>
            <via> Via Mazzini </via>
            <civico> 10 </civico>
            <città> Verona </città>
  </indirizzo>
  ```
- Contenuto misto
  ```
  <dati_anagrafici>Il signor
     <persona>
            <nome>Mario Rossi</nome>
      vive in <indirizzo>
                  <via> Via Mazzini </via>
                  <civico> 10 </civico>
                  <città> Verona </città>
            </indirizzo>
     </persona>
  </dati_anagrafici>
  ```

Attributi

- Un attributo consiste in una coppia nome-valore associata al tag iniziale di un elemento
  ```
  <persona cod_fisc="RSSMRA65E25L781T">
  ```
- Si possono usare anche gli apici singoli
  ```
  <persona cod_fisc='RSSMRA65E25L781T'>
  ```

Nomi XML

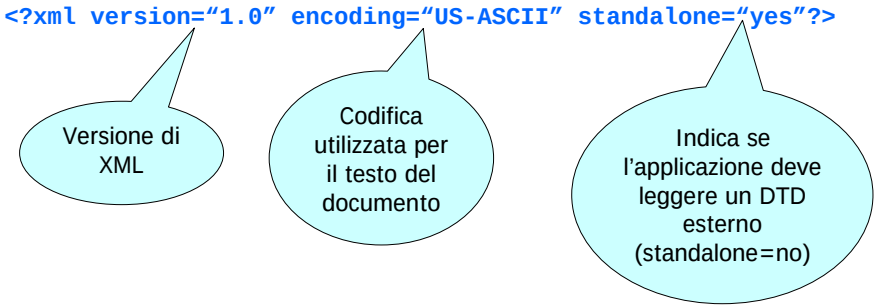
- Possono essere costituiti da qualsiasi carattere alfanumerico
- Possono includere:
  - Underscore _
  - Trattino -
  - Punto .
- Possono iniziare solo con lettere, ideogrammi o con il carattere underscore
- Non possono includere:
  - Altri caratteri di punteggiatura
  - Virgolette
  - Apostrofi
  - $ e %
  - < e >
  - Spazi

Esempi di nomi XML

- Nomi ben formati:
  - `<Nome_persona> Maria </Nome_persona>`
  - `<Giorno-Mese-Anno> 10/06/2004 </Giorno-Mese-Anno>`
  - `<_indirizzo> Via Stella 10 </_indirizzo>`
- Nomi NON ben formati:
  - `<Nome persona> Maria </Nome persona>`
  - `<Giorno/Mese/Anno> 10/06/2004 </Giorno/Mese/Anno>`
  - `<citta'> Verona </citta'>`
  - `<1_telefono> 045 1234567 </1_telefono>`
  - `<%vendita> 20 </%vendita>`

La dichiarazione XML

- I documenti XML iniziano con:



Document Type Definition (DTD)

- Un DTD descrive la struttura di un documento XML:
  - i tag ammessi
  - le regole di annidamento dei tag
- I DTD vengono utilizzati per la validazione di un documento XML
  - Un documento XML è valido quando è conforme ad un dato DTD

Esempio di DTD: elenco.dtd

```
<!ELEMENT elenco (libro+)>
<!ELEMENT libro (titolo,prezzo?)>
<!ELEMENT titolo (#PCDATA)>
<!ELEMENT prezzo (#PCDATA)>
```

- Ogni riga rappresenta la dichiarazione di elemento
- L'elemento elenco contiene uno o più elementi libro
- L'elemento libro contiene un titolo e zero o un prezzo
- titolo e prezzo possono contenere solo testo.

Dichiarazione di elementi

```
<!ELEMENT nome_elemento (modello_di_contenuto)>
```

- Il modello di contenuto può essere:
  - #PCDATA
  - Elementi figli
  - Sequenze
  - Misto
  - EMPTY
  - ANY

Modello di contenuto: #PCDATA

- Specifica che l'elemento deve contenere solamente dati di tipo carattere
- L'elemento non può contenere elementi figli di alcun tipo

ESEMPIO:

- L'elemento titolo deve contenere solo testo

```
<!ELEMENT titolo (#PCDATA)>
```

Modello di contenuto: Elementi Figli

- Specifica che un elemento deve contenere esattamente un elemento figlio di un determinato tipo

ESEMPIO:

- L'elemento libro deve contenere esattamente un elemento titolo (né più né meno di uno)

```
<!ELEMENT libro (titolo)>
```

Modello di contenuto: Sequenze

- Specifica che un elemento deve contenere più elementi figli
- I figli vengono elencati in una sequenza separati da virgole
- Gli elementi figli devono apparire all'interno dell'elemento padre nell'ordine specificato

ESEMPIO:

- L'elemento libro deve contenere un elemento titolo e un elemento prezzo

```
<!ELEMENT libro (titolo,prezzo)>
```

Il numero di figli

- Per indicare quante istanze di un elemento possono apparire si usa:
  - ? zero o una istanza
  - * zero o più istanze
  - + una o più istanze

ESEMPIO:

- L'elemento libro deve contenere un elemento titolo, uno o più elementi autore e zero o un elemento prezzo

```
<!ELEMENT libro (titolo,autore+,prezzo?)>
```

Scelte

- Una scelta è un elenco di nomi di elementi (due o più) che possono apparire nell'elemento padre
- Gli elementi della scelta vengono separati da barre verticali
- L'elemento padre non può contenere entrambi gli elementi elencati nella scelta

ESEMPIO:

- L'elemento contatto può contenere o un elemento telefono_casa o un elemento telefono_ufficio

```
<!ELEMENT contatto (telefono_casa | telefono_ufficio)>
```

Parentesi

- Per combinare scelte e sequenze si possono usare le parentesi

ESEMPIO:

- L'elemento indirizzo deve contenere un elemento tra via e piazza e un elemento civico

```
<!ELEMENT indirizzo ((via | piazza), civico)>
```

- L'elemento persona può contenere un elemento nome e un elemento cognome o un elemento cognome e un elemento nome

```
<!ELEMENT persona ((nome,cognome) | (cognome,nome))>
```

Modello di contenuto: Misto

- Specifica che un elemento deve contenere sia dati di tipo carattere che elementi figli
- Non è possibile specificare:
  - l'ordine in cui appariranno
  - quante istanze di essi appariranno
- L'elemento #PCDATA deve essere il primo della lista

ESEMPIO:

- L'elemento libro può contenere dati di tipo carattere e elementi figli titolo e prezzo

```
<!ELEMENT libro (#PCDATA|titolo|prezzo)*>
```

Modello di contenuto: EMPTY

- Specifica che un elemento deve essere vuoto e quindi senza nessun tipo di contenuto

ESEMPIO:

- L'elemento immagine deve essere un elemento vuoto

```
<!ELEMENT immagine EMPTY>
```

Modello di contenuto: ANY

- Specifica che un elemento può contenere qualsiasi cosa
  - Testo
  - Elementi figli
  - Contenuto misto
- Gli elementi che appaiono come figli devono comunque esser stati dichiarati

ESEMPIO:

- L'elemento pagina può contenere qualsiasi cosa

```
<!ELEMENT pagina ANY>
```

Dichiarazione di attributi

```
<!ATTLIST nome_elemento
          nome_attributo1 CDATA #REQUIRED
          nome_attributo2 CDATA #IMPLIED
          nome_attributo3 CDATA #FIXED valore>
```

ESEMPIO:

- L'elemento immagine ha un attributo codice obbligatorio ed un attributo titolo opzionale

```
<!ATTLIST immagine
          codice CDATA #REQUIRED
          titolo CDATA #IMPLIED>
```

Valori di default per gli attributi

- #IMPLIED: il valore dell'attributo è opzionale
- #REQUIRED: il valore dell'attributo è obbligatorio
- #FIXED: il valore dell'attributo è costante e immutabile
- Literal: indica il valore di default sotto forma di stringa tra apici

Tipi di Attributi

- CDATA
- NMTOKEN
- NMTOKENS
- Enumerazione
- ID
- IDREF
- IDREFS
- ENTITY
- ENTITIES
- NOTATION

Tipi di Attributi (1)

- CDATA: può contenere qualsiasi tipo di stringa accettabile in un documento XML ben formato

```
<!ATTLIST immagine titolo CDATA #IMPLIED>

<immagine titolo="tramonto"/>
```

- NMTOKEN: può iniziare con qualsiasi carattere

```
<!ATTLIST libro anno_pubblicazione NMTOKEN #REQUIRED>

<libro anno_pubblicazione="1950 d.c."/>
```

- NMTOKENS: può contenere uno o più token

```
<!ATTLIST esibizione date NMTOKENS #IMPLIED>

<esibizione date="10-07-2004 17-07-2004 24-07-2004/>
```

Tipi di Attributi (2)

- Enumerazione: lista di tutti i possibili valori assegnabili all'attributo

```
<!ATTLIST data giorno (1|2|3|4|5|6|7|8|9|
                       10|11|12|13|14|15|16|17|18|19|
                       20|21|22|23|24|25|26|27|28|29|
                       30|31) #REQUIRED
                 mese (gennaio|febbraio|marzo|
                       aprile|maggio|giugno|
                       luglio|agosto|settembre|
                       ottobre|novembre|dicembre) #REQUIRED
                anno (2000|2001|2002|2003|2004) #REQUIRED>

<data giorno="15" mese="agosto" anno="2003"/>
```

Tipi di Attributi (3)

- ID: contiene un nome XML che abbia valore univoco all'interno del documento

```
<!ATTLIST persona cod_fisc ID #REQUIRED>

<persona cod_fisc="RSSMRA65E25L781T"/>
```

- IDREF: riferimento all'attributo di tipo ID di un elemento del documento

```
<!ATTLIST persona cod_fisc ID #REQUIRED>
<!ATTLIST docente persona IDREF #REQUIRED>

<persona cod_fisc="RSSMRA65E25L781T"/>
<docente persona="RSSMRA65E25L781T"/>
```

Tipi di Attributi (4)

- IDREFS: contiene una lista di nomi XML ognuno dei quali deve essere un ID valido di un elemento del documento

```
<!ATTLIST persona cod_fisc ID #REQUIRED>
<!ATTLIST sposi persone IDREFS #REQUIRED>

<persona cod_fisc="RSSMRA65E25L781T"/>
<persona cod_fisc="BNCFRN63D45L781T"/>
<sposi persone="RSSMRA65E25L781T
                BNCFRN63D45L781T"/>
```

Validazione

- Un documento XML per il quale è richiesta la validazione deve includere un riferimento al DTD con cui deve essere messo a confronto
- Il riferimento deve essere fornito nella dichiarazione del tipo di documento

```
<!DOCTYPE elenco SYSTEM "http://ibiblio.org/xml/dtds/elenco.dtd">
```

- Questa dichiarazione afferma che elenco è la radice del documento e il DTD si trova all'URL `http://ibiblio.org/xml/dtds/elenco.dtd`
- La dichiarazione del tipo di documento si trova dopo la dichiarazione XML

Dichiarazione del tipo di documento

- Se il DTD di riferimento si trova ad un certo URL:

```
<!DOCTYPE elenco SYSTEM
  "http://ibiblio.org/xml/dtds/elenco.dtd">
```

- Se il DTD si trova allo stesso URL del documento:

```
<!DOCTYPE elenco SYSTEM "/dtds/elenco.dtd">
```

- Se il DTD si trova nella stessa directory del documento

```
<!DOCTYPE elenco SYSTEM "elenco.dtd">
```

Esempio di documento valido

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE elenco SYSTEM "elenco.dtd">
<elenco>
  <libro>
      <titolo>XML Guida di riferimento</titolo>
      <prezzo>35</prezzo>
  </libro>
  <libro>
      <titolo>Basi di dati</titolo>
  </libro>
</elenco>
```

Dichiarazione del tipo di documento (2)

- Il DTD può essere inserito direttamente nella dichiarazione del tipo di documento

```
<?xml version="1.0"?>
<!DOCTYPE elenco [
   <!ELEMENT elenco (libro+)>
   <!ELEMENT libro (titolo,prezzo?)>
   <!ELEMENT titolo (#PCDATA)>
   <!ELEMENT prezzo (#PCDATA)>
] >
<elenco>
   <libro><titolo>XML Guida di riferimento</titolo>
          <prezzo>35</prezzo>
   </libro>
   <libro><titolo>Basi di dati</titolo>
   </libro>
</elenco>
```

Validazione di un documento

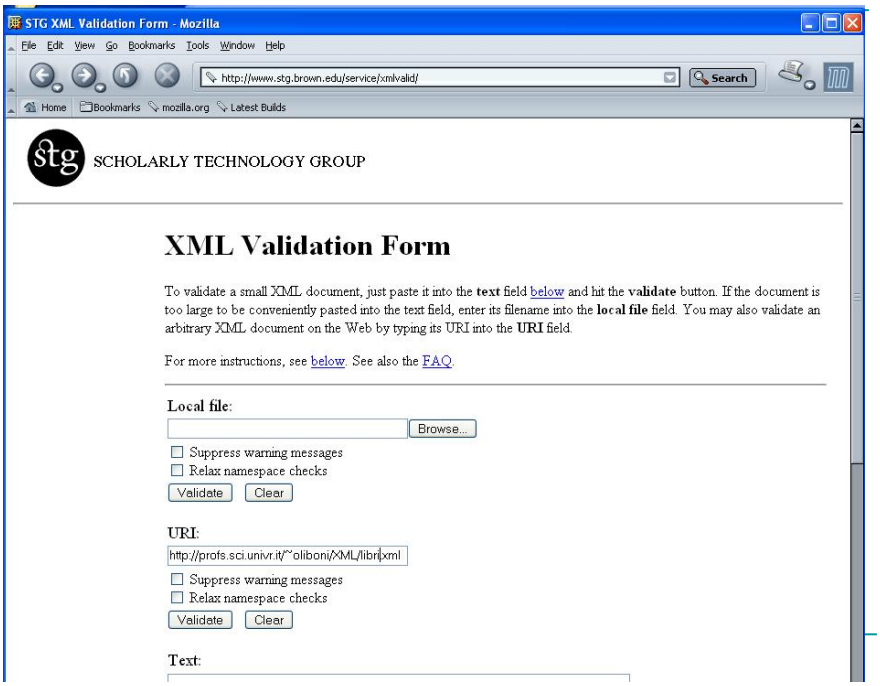
- Validatori online:
  - Brown University Scholarly Technology Group's XML Validation Form

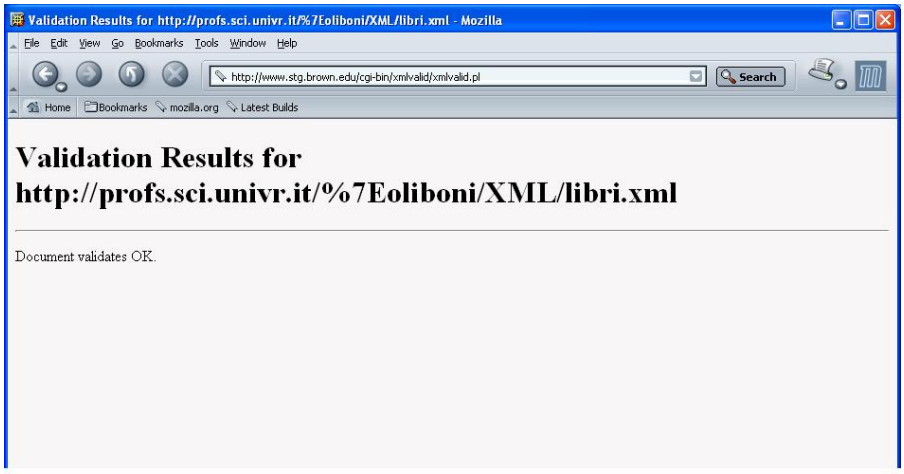
    http://www.stg.brown.edu/service/xmlvalid
  - Verificatore di ben formazione XML e validatore di Richard Tobin

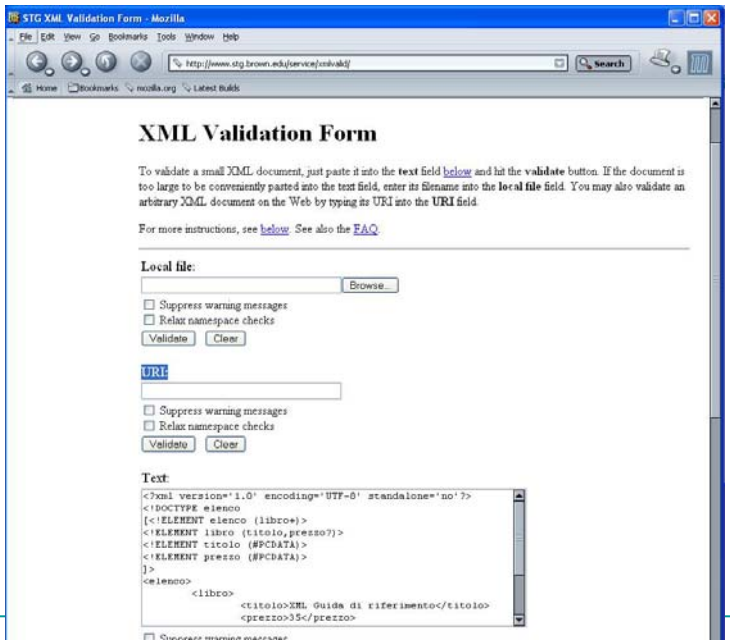
    http://www.cogsci.ed.ac.uk/%7Erichard/xml-check.html
- Per utilizzare questi validatori i documenti e i DTD associati devono essere su un server Web accessibile al pubblico

Esempio: file **libri.xml**
http://www.stg.brown.edu/service/xmlvalid

```
<?xml version='1.0' encoding='UTF-8' standalone='no'?>
<!DOCTYPE elenco
[<!ELEMENT elenco (libro+)>
<!ELEMENT libro (titolo,prezzo?)>
<!ELEMENT titolo (#PCDATA)>
<!ELEMENT prezzo (#PCDATA)>
]>
<elenco>
   <libro>
       <titolo>XML Guida di riferimento</titolo>
       <prezzo>35</prezzo>
   </libro>
   <libro>
       <titolo>Basi di dati</titolo>
   </libro>
</elenco>
```


STG XML Validation Form - Mozilla
File Edit View Go Bookmarks Tools Window Help
http://www.stg.brown.edu/service/xmlvalid/
Search
Home Bookmarks mozilla.org Latest Builds
SCHOLARLY TECHNOLOGY GROUP
XML Validation Form
To validate a small XML document, just paste it into the text field below and hit the validate button. If the document is too large to be conveniently pasted into the text field, enter its filename into the local file field. You may also validate an arbitrary XML document on the Web by typing its URI into the URI field.
For more instructions, see below. See also the FAQ.
Local file:
Browse...
Suppress warning messages
Relax namespace checks
Validate Clear
URI:
http://profs.sci.univr.it/~oliboni/XML/libri.xml
Suppress warning messages
Relax namespace checks
Validate Clear
Text:


Validation Results for http://profs.sci.univr.it/%7Eoliboni/XML/libri.xml - Mozilla
File Edit View Go Bookmarks Tools Window Help
http://www.stg.brown.edu/cgi-bin/xmlvalid/xmlvalid.pl
Search
Home Bookmarks mozilla.org Latest Builds
Validation Results for
http://profs.sci.univr.it/%7Eoliboni/XML/libri.xml
Document validates OK.

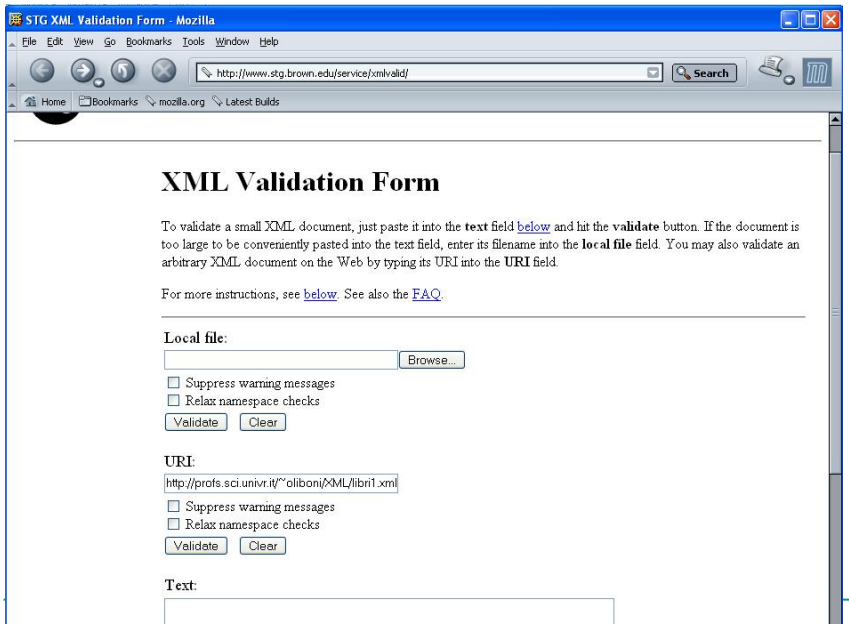

STG XML Validation Form - Mozilla
File Edit View Go Bookmarks Tools Window Help
Search
Home Bookmarks mozilla.org Latest Builds
XML Validation Form
To validate a small XML document, just paste it into the text field below and hit the validate button. If the document is too large to be conveniently pasted into the text field, enter its filename into the local file field. You may also validate an arbitrary XML document on the Web by typing its URI into the URI field.
For more instructions, see below. See also the FAQ.
Local file:
Browse...
Suppress warning messages
Relax namespace checks
Validate Clear
URI:
Suppress warning messages
Relax namespace checks
Validate Clear
Text:
<?xml version='1.0' encoding='UTF-8' standalone='no'?>
<!DOCTYPE elenco
[<!ELEMENT elenco (libro+)>
<!ELEMENT libro (titolo,prezzo?)>
<!ELEMENT titolo (#PCDATA)>
<!ELEMENT prezzo (#PCDATA)>
]>
<elenco>
<libro>
<titolo>XML Guida di riferimento</titolo>
<prezzo>35</prezzo>


Validation Results for [user-supplied text] - Mozilla
File Edit View Go Bookmarks Tools Window Help
http://www.stg.brown.edu/cgi-bin/xmlvalid/xmlvalid.pl
Search
Home Bookmarks mozilla.org Latest Builds
Validation Results for [user-supplied text]
Document validates OK.

Esempio: file **libri1.xml** ed **elenco.dtd**

http://www.stg.brown.edu/service/xmlvalid

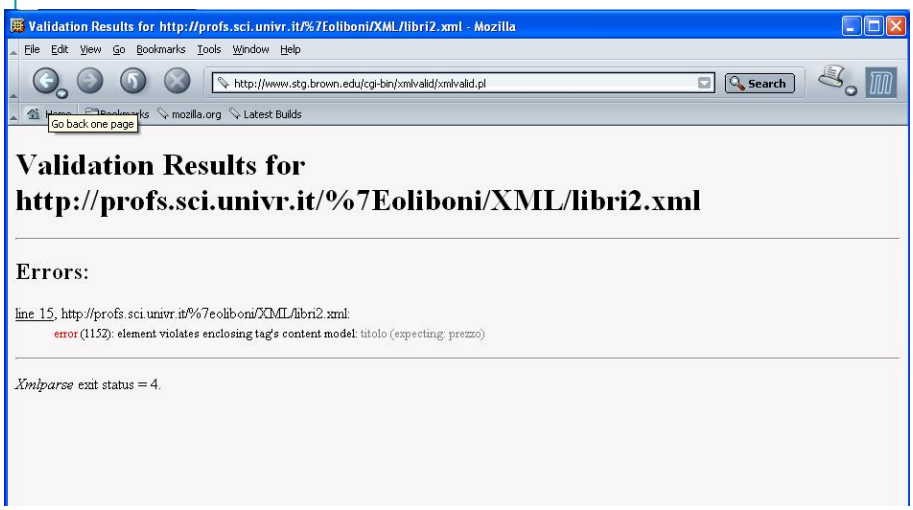
```
<?xml version='1.0' encoding='UTF-8' standalone='no'?>
<!DOCTYPE elenco SYSTEM 'elenco.dtd'>
<elenco>
   <libro>
        <titolo>XML Guida di riferimento</titolo>
        <prezzo>35</prezzo>
   </libro>
   <libro>
        <titolo>Basi di dati</titolo>
   </libro>
</elenco>

<!ELEMENT elenco (libro+)>
<!ELEMENT libro (titolo,prezzo?)>
<!ELEMENT titolo (#PCDATA)>
<!ELEMENT prezzo (#PCDATA)>
```



Esempio: file **libri2.xml**

http://www.stg.brown.edu/service/xmlvalid

```
<?xml version='1.0' encoding='UTF-8' standalone='no'?>
<!DOCTYPE elenco
[<!ELEMENT elenco (libro+)>
<!ELEMENT libro (titolo,prezzo?)>
<!ELEMENT titolo (#PCDATA)>
<!ELEMENT prezzo (#PCDATA)>
]>
<elenco>
   <libro>
       <titolo>XML Guida di riferimento</titolo>
       <prezzo>35</prezzo>
   </libro>
   <libro>
       <titolo>Basi di dati</titolo>
       <titolo>Basi di dati2</titolo>
   </libro>
</elenco>
```



XML e Bioinformatica

XML e BIO: alcune motivazioni

- I dati biologici possono essere rappresentati in formati diversi
  - FASTA (formato basato su testo per rappresentare sequenze)
  - GFF (formato per descrivere geni e altre caratteristiche legate a DNA, RNA e sequenze di proteine)
- Fonti di dati e basi di dati eterogenee per la memorizzazione di dati biologici
  - Grande numero di BD accessibili via web
- Mancanza di standard per la memorizzazione, gestione e interrogazione di dati biologici

XML e BIO: alcune motivazioni

- Assenza di nomenclatura standard per molti dati biologici
- Assenza di formati standard per lo scambio di dati biologici
- Assenza di modelli dei dati standard
- Grandi difficoltà nella gestione (manipolazione/scambio) di dati biologici

DTD per BIO XML

- BIOML: The Biopolymer Markup Language
  http://www.proteome.ca/x-bang/bioml/b_toc.htm
- RNAML: A syntax for exchanging RNA information
  http://www-lbit.iro.umontreal.ca/rnaml/
- GEML: Gene Expression Markup Language
  http://xml.coverpages.org/geml.html

RNAML

http://www-lbit.iro.umontreal.ca/rnaml/

```
...
<!-- SEQUENCE: A description of the rna sequence. -->

<!ELEMENT sequence (numbering-system*,
                    seq-data?,
                    seq annotation?)>
<!ATTLIST sequence
          strand         CDATA        #IMPLIED
          length         CDATA        #IMPLIED
          circular       (true|false) #IMPLIED
          comment        CDATA        #IMPLIED
          reference-ids  IDREFS       #IMPLIED
          analysis-ids   IDREFS       #IMPLIED
          database-ids   IDREFS       #IMPLIED>
...
```