XML

Ombretta Gaggi
Università di Padova

XML

- Il linguaggio Extensible Markup Language, è un framework per la definizione di linguaggi di markup. Permette di creare un qualsiasi insieme di marcatori ed attributi che identifica un linguaggio (con le regole di composizione)
- Ogni linguaggio si adatta ad un particolare dominio applicativo, ma tutti condividono la stessa sintassi di base e gli strumenti generici per l'elaborazione di documenti
- XML
  - non dice nulla riguardo la semantica dei tag
  - è intrinsecamente internazionale e multipiattaforma (utilizza il formato Unicode)

PS	11.274 B
PDF	4915 B
MS Word	19456 B
HTML	28 B

Applicazioni di XML

- In generale i linguaggi XML possono essere divisi in 4 gruppi:

1. Linguaggi orientati ai dati: usati per descrivere informazioni che tradizionalmente sarebbero memorizzate in basi di dati
   - Ex. Catalogo o inventario
2. Linguaggi orientati ai documenti: linguaggi per descrivere documenti in linguaggio naturale a cui è stata aggiunta una struttura a tag
   - Ex. XHTML
3. Protocolli e linguaggi di programmazione: linguaggi più specializzati per un determinato task
   - Ex. XMLSchema, XSLT, WSDL, linguaggi di configurazione, etc
4. Linguaggi ibridi: contengono le caratteristiche dei primi due gruppi, quindi sia dati altamente strutturati, sia dati con struttura lasca o in linguaggio naturale
   - Ex. Cartelle cliniche

Alcuni esempi

- XHTML
- RSS
- SMIL, Synchronized Multimedia Integration Language
- SVG, Scalable Vector Graphics
- WML, Wireless Markup Language
- CML, Chemical Markup Language
- SOAP, WSDL
- …

Rappresentazione testuale dei documenti XML - 1

- La rappresentazione testuale dei dati XML è un testo Unicode con tag di markup
- La sintassi segue alcune regole
  - i tag e gli attributi sono case sensitive (tutto in minuscolo)
  - i tag devono sempre essere chiusi (anche se sono vuoti)
    -
 e non
, <banana /> è equivalente a <banana></banana>
  - i tag devono essere aperti e chiusi nell'ordine corretto
  - un documento deve avere esattamente un elemento radice
  - l'ordine con cui si inseriscono gli attributi è irrilevante
  - i valori degli attributi vanno riportati tra "virgolette doppie" o 'apici singoli'
  - tutti gli attributi devono avere un valore

Rappresentazione testuale dei documenti XML - 2

- La sintassi segue alcune regole
  - I caratteri speciali come >,<, &, apici (virgolette) dentro gli attributi racchiusi da apici (virgolette), etc, devono essere indicati per mezzo di riferimento a carattere Unicode (&#N; dove N è, in notazione decimale, il code point del carattere voluto in Unicode)
  - I riferimenti Unicode possono anche essere usati per caratteri non presenti nella tastiera

Un documento che segue queste regole si dice *ben formato*

Simbolo	Entità
<, >	< >
&	&
'	'
"	"

Il formato UNICODE

- Un file XML è essenzialmente una sequenza di caratteri.
- Lo standard UNICODE permette di rappresentare tutti i caratteri esistenti, anche quelli non rappresentabili in ASCII
- Un carattere UNICODE è un simbolo che può comparire come parte di un testo
- Sono caratteri le lettere, ma anche simboli (ex. copyright), accenti, etc.
- Ogni carattere ha un nome (ex. lettera latina a minuscola), e un glifo (la sua rappresentazione grafica), ma non c'e' una corrispondenza biunivoca tra nomi e glifi
- Prevede più di un milione di codici, attualmente ne sono stati assegnati circa 100.000, la maggior parte a caratteri cinesi

Esempio

```
<?xml version="1.1" encoding="UTF-8" ?>
<?xml-stylesheet type="text/xsl" href="fogliodistile.xsl"?>
<!DOCTYPE  iscritti SYSTEM "iscritti.dtd">
<iscritti>
   <!- iscritti al corso di tecnologie web-->
   <studente integrazione="no" >
        <nome>Mario Rossi</nome>
        <matricola>111111</matricola>
        <email> mrossi@studenti.math.unidp.it</email>
        <laurea>informatica</laurea>
        <note> <![CDATA[ <questo non è un tag> ]]></note>
   </studente>
</iscritti>
```

prologo XML

processing instruction

collegamento ad uno schema

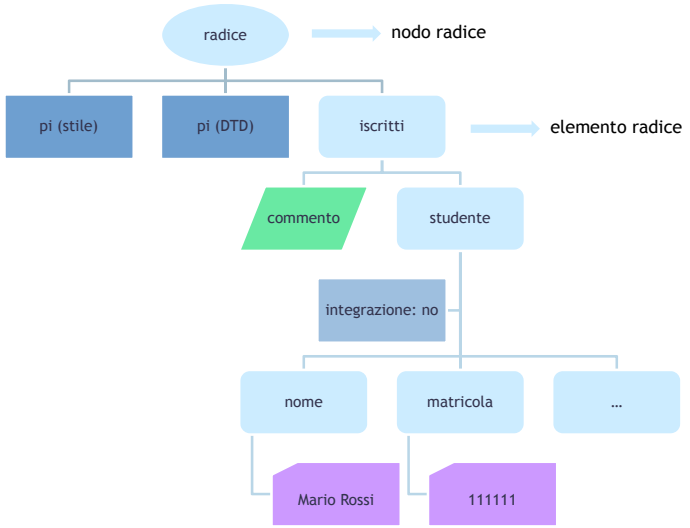
commento

sezione CDATA

Rappresentazione ad albero dei dati XML

- Concettualmente un documento XML è costituito da una struttura gerarchica ad albero in cui sono inseriti i dati
- Esistono diverse terminologie di riferimento per descrivere gli alberi XML. Qui presentiamo l'*XPath Data Model*
- Un albero XML è un albero ordinato, in quanto l'ordine in cui appaiono i figli, diversamente dagli attributi, è rilevante
- Gli attributi invece sono organizzati in insiemi
- Esistono 6 tipi di nodi:
  - Nodi di testo
  - Nodi elemento
  - Nodi attributo
  - Nodi commento
  - Nodi di istruzioni per l'elaborazione
  - Nodi radice

Esempio



URL, URI, URN

- Un *Uniform Resource Locator* è un modo per denotare una risorsa sulla rete

  http://server.com/percorso/
- È costituito da uno schema (che identifica il protocollo di comunicazione), un server e un percorso (che tipicamente corrisponde ad un singolo file su quel server)
- Gli *Uniform Resource Identifier* (URI) sono un concetto più generale

  schema:parte-variabile-a-seconda-dello-schema
- Lo scopo degli URI è descrivere tutto lo spazio dell'informazione anche i punti che non corrispondono ad una risorsa fisica
- Un *Uniform Resource Names* (URN) è un puntatore ad una risorsa che però non fa riferimento ad una posizione
  - urn:isbn:0-471-94128-X

Namespace

- I namespace nascono come soluzione per far coesistere diversi linguaggi XML in un unico documento
  - Ex. Se voglio aggiungere tag SVG all'interno di un file XHTML devo avere un meccanismo per identificare quali tag appartengono ad un linguaggio e quali all'altro
- Risolvono il problema della collisione di nomi definiti in moduli diversi
- Un namespace è identificato da un URI: è sufficiente qualificare i tag con l'URI opportuna per sapere a che linguaggio appartiene
  - Ex. {http://www.w3.org/1999/xhtml} head

Dichiarazione di namespace

```
<elemento xmlns:prefisso="URInamespace" >
   …
</elemento>
```

- Una dichiarazione di namespace influenza l'elemento che la contiene e tutti i suoi discendenti
- Se prefisso manca, si parla di namespace *non qualificato*, altrimenti si tratta di un namespace *qualificato*
- *prefisso* non può cominciare con "xml" o "XML"

DTD, Document Type Definition

XML ben formato e documenti validi

- La buona forma di un documento XML è una proprietà puramente sintattica:
  - tutti i tag sono chiusi, propriamente innestati ed esiste un'unica radice
- La validazione è invece già "semantica", nel senso che ha a che fare con il significato dei dati e l'utilizzo del documento
  - Ex: un documento è formato da capitoli, ciascuno con un titolo e vari paragrafi con determinate proprietà
- Un documento è valido *rispetto ad uno schema*: nasce quindi la necessità di definire linguaggi per la rappresentazione di schemi

Schemi e linguaggi per uno schema

- Uno schema è la definizione formale della sintassi di un linguaggio XML .
- Un documento è valido se è sintatticamente corretto rispetto ad uno schema dato
- Un linguaggio per uno schema (schema language) è un linguaggio formale che permette di definire gli schemi, ovvero le regole sintattiche per la creazione di documenti
  - DTD, XMLSchema, DSD, RELAX NG
- Gli schemi sono quindi delle grammatiche per descrivere la struttura di un documento XML
- Specificano quali attributi ed elementi sono validi per una classe di documenti XML, quali attributi ed elementi sono opzionali e quali sono richiesti
  - specifica del contenuto degli attributi ed elementi

Perché utilizzare uno schema

- Perché la presenza di regole formali precise facilita l'interscambio di dati
- Perché alcuni software sono in grado di lavorare solo con documenti XML validi
- Perché molti strumenti XML (ex. XQuery, etc) in alcuni casi richiedono che i documenti su cui lavorano facciano riferimento ad uno schema
- Per essere utile un linguaggio per la definizione di schemi deve
  - essere sufficientemente espressivo
  - permettere l'implementazione di processori di schemi efficienti
  - essere comprensibile

Espressioni regolari

- Le espressioni regolari servono per descrivere sequenze di elementi o caratteri
- Un'espressione regolare definita su un alfabeto Σ è costruita secondo le seguenti regole:
  - ogni atomo in Σ in è un'espressione regolare;
  - se α e β sono due espressioni regolari, allora anche α?, α*, α+, αβ, α|β, e (α) sono espressioni regolari.
- Gli operatori ? (0 o 1 occorrenze), * (0 o più occorrenze) e + (1 o più occorrenze) hanno precedenza rispetto alla concatenazione, che a sua volta ha precedenza rispetto all'operatore | (unione).
  - Ex. L'espressione regolare che indica una cifra è:

    0|1|2|3|4|5|6|7|8|9

Abbreviazioni per le espressioni regolari

- Intervalli:
  - [0-9] è una abbreviazione per 0|1|2|3|4|5|6|7|8|9
- Ripetizioni:
  - α {n,m} da n a m ripetizioni di α con n ed m interi non negativi

DTD, Document Type Definition

- Il linguaggio DTD è un linguaggio per schemi XML progettato come sottoinsieme di XML
- NON è un linguaggio XML
- È un linguaggio relativamente semplice, con un potere espressivo limitato. È stato il primo linguaggio di schemi definito
- DTD non supporta i namespace perché è stato definito prima della loro definizione

Dichiarazione di tipo di documento

- Un documento può contenere una dichiarazione di tipo di documento, cioè un riferimento ad uno schema DTD
- Le dichiarazioni possono riferirsi a documenti esterni:

  <!DOCTYPE radice SYSTEM “URI” >

  oppure essere interne al documento stesso.
- *radice* mi indica il nome dell’elemento radice e URI è l’identificatore di sistema
- Un documento è valido se il suo elemento radice è *radice* e se rispetta tutti i vincoli descritti nella DTD identificata da URI
- La dichiarazione di tipo va prima dell’elemento radice e (se presente) dopo la dichiarazione XML

```
<?xml version=“1.1” encoding=“ISO-8859-1” ?>
<!DOCTYPE  iscritti  SYSTEM “iscritti.dtd”>
<iscritti>
```


Identificatore pubblico o schemi incorporati

- È possibile usare un identificatore pubblico:

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

- Oppure incorporare la definizione dello schema:

  <!DOCTYPE radice […] >

  dove al posto dei punti segue la dichiarazione degli elementi
- I commenti, definiti come i commenti XML, possono apparire dovunque e spesso vengono utilizzati per esprimere in linguaggio naturale delle proprietà che non possono essere descritte dal linguaggio DTD

Dichiarazione di elemento

- Una dichiarazione di elemento è:

  <!ELEMENT nome modello>

  dove:
  - nome è il nome dell’elemento e
  - modello di contenuto definisce la sua struttura
- Il contenuto di un elemento è la sequenza dei suoi figli. Esistono 4 tipi di modelli di contenuto
  - Vuoto (EMPTY)
  - Qualsiasi (ANY)
  - Contenuto misto: (#PCDATA | e_1 | e_2 | … | e_n)*

    dove e_1 … e_n sono nomi di elementi
  - Contenuto elemento (una sequenza di elementi descritti tramite un’espressione regolare sull’alfabeto composto dai nomi di elementi e la concatenazione è espressa dalla virgola)

Considerazioni

- I commenti e le istruzioni di elaborazione possono comparire ovunque, tranne che negli elementi a contenuto EMPTY
- Un elemento che contenga testo deve necessariamente avere contenuto misto
- Non è possibile imporre vincoli sui dati testuali

Costrutto	Significato
EMPTY	contenuto vuoto
ANY	qualsiasi contenuto
#PCDATA	caratteri
nome elemento	elemento
,	concatenazione
|	unione/scelta
?	opzionale
* (+)	0 (1) o più ripetizioni

Esempio

```
<!DOCTYPE  iscritti SYSTEM “iscritti.dtd”>
<iscritti>
   <!- iscritti al corso di tecnologie web-->
   <studente integrazione=“no” >
        <nome>Mario Rossi</nome>
        <matricola>111111</matricola>
        <email> mrossi@studenti.math.unidp.it</email>
        <laurea>informatica</laurea>
        <note> <![CDATA[ <questo non è un tag> ]]></note>
   </studente>
   ….
</iscritti>
```


Esempi - DTD

```
<!ELEMENT iscritti (studente)+ >
<!ELEMENT studente (nome, matricola, email, laurea, note) >
<!ELEMENT matricola (#PCDATA) >
<!ELEMENT email (#PCDATA) >
<!ELEMENT laurea (#PCDATA) >
<!ELEMENT nome (#PCDATA) >
<!ELEMENT note (#PCDATA) >


<!ELEMENT table
        (caption?, (col*|colgroup*), thead?, tfoot?, (tbody+|tr+)) >
```


Dichiarazione di liste di attributi

- Una dichiarazione di una lista di attributi avviene in questo modo:

  <!ATTLIST *nome-elem definizioni* >

  dove *definizioni* è una lista di attributi nella seguente forma

  nome-attributo tipo default
- Il tipo di un attributo può essere:
  - tipo stringa (CDATA)
  - enumerazione: (s_1 | s_2 | … | s_n) dove s_i è una stringa
  - token di nome (NMTOKEN), la variante NMTOKENS denota una lista di token di nome separati da spazi
  - identificativo (ID)
  - tipo riferimento (IDREF), la variante IDREFS denota una lista di riferimenti a ID separati da spazi

Valori di default per gli attributi

- La dichiarazione di default specifica se un attributo è opzionale o richiesto e il valore di default:
  - #REQUIRED: richiesto. Se l’attributo non è presente, non c’è un valore di default il documento non è valido
  - #IMPLIED: opzionale, nessun valore di default
  - “*valore*”: opzionale con default, dove *valore* è un valore legale per l’attributo; se è assente, il valore specificato viene usato come default
  - #FIXED “*valore*”: prefissato. L’attributo è opzionale, se presente deve essere uguale a *valore* e se assente viene fissato uguale a *valore*

  - Ex:
```
<!ATTLIST input maxlenght CDATA #IMPLIED
               tabindex CDATA #IMPLIED>
<!- una o più cifre -->
```


Esempio

```
<iscritti>
   <studente integrazione="no" >
        <nome>Mario Rossi</nome><matricola>1111</matricola>
        <email> mrossi@studenti.math.unidp.it</email>
        <laurea>informatica</laurea>
        <note> <![CDATA[ <questo non è un tag> ]]>
   </studente>
....
<!ELEMENT iscritti (studente)+ >
<!ELEMENT studente (nome, matricola, email, laurea, note) >
<!ATTLIST studente integrazione CDATA "no" >
<!ATTLIST studente integrazione (si|no) "no" >
```


Altro esempio

```
<!ATTLIST form
        action      CDATA       #REQUIRED
        onsubmit    CDATA       #IMPLIED
        method      (get|post)  "get"
        enctype     CDATA       "application/x-www-form-
                                        urlencoded">
```


DTD e Namespace

- DTD non supporta i namespace ma può coesistere con loro tramite la definizione di un attributo di nome xmlns con valore l'URI del namespace
- Tecnicamente questa è la definizione di un attributo e non una dichiarazione di namespace
- Non è possibile l'uso di prefissi

```
<!ATTLIST html xmlns CDATA #FIXED
                        "http://www.w3.org/1999/xhtml" >
```


Entità interne

- Il linguaggio DTD offre un meccanismo di macroespansione molto limitato: permette di definire delle entità interne, ovvero delle abbreviazioni che vengono rimpiazzate in fase di normalizzazione del documento istanza

```
<!ENTITY copyright "Copyright ACME 2009">
```

Questo documento è soggetto a &copyright;

diventa

Questo documento è soggetto a Copyright ACME 2009

- Le entità interne possono contenere anche markup

Entità parametro interne

- Un'entità parametro interna è una macro che si può applicare solo allo schema e non al documento istanza

<!ENTITY % Forma "(rett|cerchio|poligono|default)">

<!ATTLIST area forma %Forma; "rett" >

diventa

<!ATTLIST area forma (rett|cerchio|poligono|default) "rett" >

Entità esterne

- Una dichiarazione di entità esterna è un riferimento ad un'altra risorsa (ex. un file) che può contenere dati XML oppure no
- Riferimento a dati XML:
  - <!ENTITY altro SYSTEM "http://server/altro.xml">
  - L'occorrenza di &altro; viene rimpiazzata con il contenuto del file altro.xml
- Risorse non XML (*entità unparsed*)
  - <!ENTITY immagine SYSTEM "http://server/immagine.gif" NDATA gif>
    <!NOTATION gif SYSTEM "http://www.iana.org/assignments/media-types/image/gif">
- Le notazioni servono per descrivere il formato dell'entità unparsed

Sezioni condizionali

- Permettono di includere selettivamente alcune parti di uno schema, in modo da poterne scegliere una da applicare al documento istanza corrente

```
<![%persona.simple; [ <!ELEMENT persona(nome,cognome)>]]>

<![%persona.full; [
        <!ELEMENT persona (nome,cognome,email+,tel?)>
        <!ELEMENT email (#PCDATA)>
        <!ELEMENT tel(#PCDATA)>
]]>
<!ELEMENT nome (#PCDATA)>
<!ELEMENT cognome (#PCDATA)>
```

```
<!ENTITY % persona.simple "INCLUDE">
<!ENTITY % persona.full "IGNORE">
```


La validazione

- Il processore DTD verifica:
  - che il nome dell'elemento radice sia corretto
  - per ogni elemento nodo:
    - verifica che corrisponda al modello contenuto nella sua dichiarazione
    - normalizza i suoi attributi sulla base della dichiarazione della lista di attributi
    - verifica che tutti gli attributi #REQUIRED siano presenti
    - verifica che i valori degli attributi corrispondano ai loro tipi
  - l'unicità dei valori ID
  - la validità degli attributi ID e IDREF

Riferimenti Bibliografici

- Siti web W3C
  - http://www.w3.org/XML/
- W3C Tutorial
  - http://www.w3schools.com/xml/
  - http://www.w3schools.com/dtd/default.asp
- Libro
  - Anders Moller, Micheal Schwartzbach, *Introduzione a XML*, Pearson, 2007.