

Gestione dei dati

Parte 7

Introduzione a XML

Riccardo Rosati

Facoltà di Ingegneria dell'Informazione
Sapienza Università di Roma
Anno Accademico 2011/2012

<http://www.dis.uniroma1.it/~rosati/gd/>



SAPIENZA
UNIVERSITÀ DI ROMA

Sommario

- Il linguaggio XML
- Document Type Definition (DTD)

IL LINGUAGGIO XML



Il linguaggio XML

XML = eXtensible *Markup Language*

- linguaggio di marcatura descrittiva
- naturale successore di HTML come linguaggio per la rappresentazione di informazione sul Web

XML si è imposto come standard de facto per lo scambio di informazioni semi-strutturate

Forma di un documento XML

Un documento XML è costituito da:

- un *prologo* (opzionale), costituito da
 - una dichiarazione XML
 - una DTD
- un'*istanza di documento*, contenente
 - testo libero
 - elementi delimitati da etichette
 - attributi associati ad elementi
 - entità
 - sezioni CDATA
 - istruzioni di processamento
 - commenti

Codifica dei caratteri

- XML usa la codifica dei caratteri *Unicode* (UTF-16)
- è un codice a 16 bit (65536 simboli)
- è sufficiente a rappresentare i caratteri di tutte le lingue
- ogni carattere è rappresentato da al più quattro cifre esadecimali
Es. `ꀟ`
- i primi 256 caratteri sono il codice ASCII (UTF-8)
Es. `
` (linefeed), ` ` (blank), `N` (lettera N), ...
- Un processore XML:
 - deve riconoscere almeno i codici Unicode ed ASCII
 - può riconoscere altri codici

Documenti XML ben formati

- Un documento che rispetta la specifica di XML è detto *ben formato*:
- costituito da un *insieme di elementi* annidati
- un *elemento* è costituito da una coppia di etichette di apertura e di chiusura e da tutto quello che racchiudono
<autore>Jeffrey D. Ullman</autore>
<autore><nome>Jeffrey D.</nome> ... </autore>
 - *nome* dell'elemento: autore
 - *etichetta di apertura* dell'elemento: <autore>
 - *etichetta di chiusura* dell'elemento: </autore>
 - *contenuto* dell'elemento: Jeffrey D. Ullman
<nome>Jeffrey D.</nome>

Documenti XML ben formati

- Il *contenuto* di un elemento è costituito da:
 - altri elementi (detti *figli*), delimitati da coppie di etichette
 - testo libero (non contenente marcatura), detto `#PCDATA`
 - riferimenti ad entità
- `<esempio>`
Qui inizia il contenuto di un elemento
`<figlio>` questo e` un elemento figlio `</figlio>`
testo libero con riferimento all'entita` `&`;
`<figlio>` un altro elemento figlio `</figlio>`
`</esempio>`

Documenti XML ben formati

- le etichette devono essere *annidate correttamente*
Esempio di documento non ben formato:

```
<testo>  
  <bold><italic>XML</bold></italic>  
</testo>
```

- ci deve essere *un solo elemento radice*
Esempio di documento con più radici:

```
<bibliografia>  
  <pubbl><autore>Ullman</autore> ...</pubbl>  
  <pubbl><autore>Floyd</autore> ... </pubbl>  
  Adesso e' ben formato  
</bibliografia>
```

Elementi vuoti

- Il contenuto di un elemento può essere vuoto.
- Due modi di denotare un *elemento vuoto*:
 - coppia di etichette di apertura e chiusura:
<vuoto></vuoto>
 - etichetta di elemento vuoto: <vuoto/>

<p></p>

<h1>Bibliografia</h1>

Ullman ...

Floyd ...

<hr/>

Uso di elementi vuoti

- Non sono inutili in quanto aggiungono informazione al documento:
 - attraverso attributi associati all'elemento:

```
<IMG src="colosseo.gif"
  align="left"
  height="50"
  width="30"/>
```
 - attraverso la presenza stessa dell'elemento:

```
<BR/>
```

Nomi di elemento

- XML è *case-sensitive* - esempio non ben formato:

```
<elenco-clienti>  
  <cliente><codice>CC128</codice> ... </CLIENTE>  
</Elenco-Clienti>
```

- Errore comune: dimenticare "/" nell'etichetta di chiusura - es:

```
<elenco-clienti>  
  <cliente><codice>CC128</codice> ... <cliente>  
</elenco-clienti>
```

Nomi di elemento

- Possono contenere solo: lettere, cifre, -, _, ., :.
- Carattere ":" usato solo per separare *namespace*.
- Nomi che iniziano con XML,xml,xML,...sono riservati.

Nomi di elemento

```
<nomiPermessi>  
  <xsl:template/>  
  <Nome_elemento_lungo/>  
  <Altro-nome-lungo/>  
  <nome.con.punti/>  
  <a1233-231-231/>  
  <_12/>  
</nomiPermessi>  
<nomi+Vietati>  
  <questiNO@#$$%^()+?*;$=/>  
  <Un;nome*2/>  
  <XML-riservato/>  
  <8-inizia-con-cifra/>  
  <niente spazi/>  
  <riservati<&>>  
</nomi+Vietati>
```

Caratteri riservati

- Il testo non può contenere i caratteri riservati per la marcatura
 - carattere "<" (denota l'inizio di un'etichetta)
 - carattere "&" (per le entità - più avanti)
- *Si usano*
 - "<" per "<"
 - "&" per "&"
- `<editore>Wiley & sons</editore>`
`<disequazione>x*y < z</disequazione>`
- I caratteri `>`, `"`, e `'` possono essere sostituiti da `>`, `"`, e `'`

Attributi

- Un elemento può avere nessuno, uno o più *attributi*.
- un attributo ha un *nome* ed associa una *proprietà* ad un elemento
- i caratteri ammessi nei nomi sono gli stessi che per gli elementi
- gli attributi vengono specificati nell'etichetta iniziale attraverso coppie *attributo="valore"*
- il valore deve essere delimitato da una coppia di apici singoli o doppi
- un elemento non può avere due attributi con lo stesso nome

Attributi

```
<attributi-ben-formati>  
  <elemento _ok="yes"/>  
  <un attr="il suo valore"/>  
  <molti primo="1" secondo='2' terzo="333"/>  
  <apici-doppi-o-singoli  
    doppi="John's"  
    singoli='Stampa: "Ciao Mondo!" '/>  
</attributi-ben-formati>
```

N.B.:

- In HTML gli apici intorno al valore possono mancare
- In XML gli *apici sono obbligatori*

Attributi

```
<attributi-mal-formati>  
  <carattere-non-ammesso a*b="23432"/>  
  <separatori-diversi valore="12"/>  
  <cambia-separatore valore="aa"aa"/>  
  <cambia-separatore valore='bb'bb'/>  
  <parola-riservata XML-ID="xml234"/>  
</attributi-mal-formati>
```

Uso di attributi ed elementi

Per memorizzare dati possiamo usare sia elementi che attributi.

```
<impiegato>  
  <nome>A. Rossi</nome>  
  <ddn>1/1/1950</ddn>  
</impiegato>
```

```
<impiegato ddn="1/1/1950">  
  <nome>A. Rossi</nome>  
</impiegato>
```

La scelta dipende dall'applicazione:

- non si può forzare #PCDATA a non essere vuoto
- gli attributi non si possono strutturare
- si possono enumerare i valori ammessi per gli attributi

Entità

- un'*entità* è una parte di documento a cui si è dato un *nome* (una specie di macro)
- può essere
 - interna: definita all'interno del documento (o del DTD)
 - esterna: memorizzata su un altro file
- il testo può contenere riferimenti ad entità: &nome-entita;
- quando viene processato il documento, i riferimenti alle entità devono essere espansi
- entità predefinite: <, &, >, ", ';
- Vedremo più avanti come definire nuove entità.

Sezioni CDATA

- servono ad includere testo contenente caratteri riservati
- possono comparire ovunque può comparire testo libero
- tutti i caratteri riservati perdono il loro significato speciale
 - "<" e "&" *non* vanno sostituiti con "<" e "&"
 - le sequenze <elem>, </elem> e <elem/> *non* vengono interpretate come marche
- le sezioni CDATA non possono essere annidate
- racchiuse tra "<![CDATA[" e "]]>"
- una sezione CDATA non può contenere la stringa "]]>"

Esempi

Questo lucido in formato XML

```
<lucido>
  <titolo>Sezioni CDATA - Esempi</titolo>
  Questo lucido in formato XML
  <esempio>
    <![CDATA[
      <lucido>
        <titolo>Sezioni CDATA - Esempi</titolo>
        Questo lucido in formato XML
        ... caratteri riservati OK: &, <
      </lucido>
    ]]>
  </esempio>
</lucido>
```

Commenti

- servono a escludere una parte di documento dal processamento
- possono comparire ovunque all'esterno della marcatura
- un processore XML può o meno rendere disponibili le parti di documento racchiuse tra commenti
- sono delimitati da `<!--` e `-->`
- possono contenere qualsiasi carattere (inclusi "`<`" e "`&`"), tranne "`-`"
- non possono essere annidati

Esempi

```
<bibliografia>
  <!-- voce commentata <pub> ... </pub> -->
  <pub> ... </pub>
  <!-- commento su piu` righe
  <pub> ... </pub>   <- usa "<"
  <pub> ... </pub>   anche "&" e` ammesso
  -->

  <!-- commento non -- valido -->
  <pub <!-- qui no commento --> id="Ullm82"> ... </pub>
  <!-- un commento
  <!-- non puo` contenere commenti annidati -->
  -->
</bibliografia>
```

Commenti in sezioni CDATA

In una sezione CDATA i commenti non vengono riconosciuti come tali

<esempio>

Questo e` un <!-- commento -->

<![CDATA[Sezione CDATA con "commento"

<!-- fa parte del testo --> altro testo

]]>

Questo e` di nuovo un <!-- commento -->

</esempio>

Istruzioni di processamento (PI)

- permettono di includere nel documento istruzioni da passare alle applicazioni
- fanno parte del prologo (non dei caratteri che compongono il documento)
- devono essere passate alle applicazioni
- hanno la forma `<?PITarget ... ?>`, dove
 - PITarget denota l'applicazione a cui è diretta la PI
 - ... denota ulteriori dati da passare all'applicazione

Esempio: processamento di uno stylesheet da parte di XSLT

```
<?xml-stylesheet href="mystyle.xsl" type="text/xsl" ?>
```

Dichiarazione XML iniziale

- Documenti XML possono (e in realtà dovrebbero) iniziare con una *dichiarazione XML* che specifica la versione di XML utilizzata

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

```
<testo>Questo documento e` conforme alla specifica  
di XML 1.0.</testo>
```

La dichiarazione XML può anche specificare la *codifica dei caratteri* usati:

- ```
<?xml version="1.0" encoding="ISO-8859-2"?>
```
- La codifica dei caratteri specifica il mapping tra i byte che costituiscono la rappresentazione fisica del documento (ad es., file, socket, ...) ed i caratteri.

# Documento XML = albero

L'annidamento degli elementi definisce in modo esplicito una struttura ad albero:

<Mail>

<From> <Address> Dante@dsn.fi.it </Address> </From>

<To> <Address> Beatrice@pitti.fi.it </Address>

<Address> Virgilio@spqr.rm.it </Address>

</To>

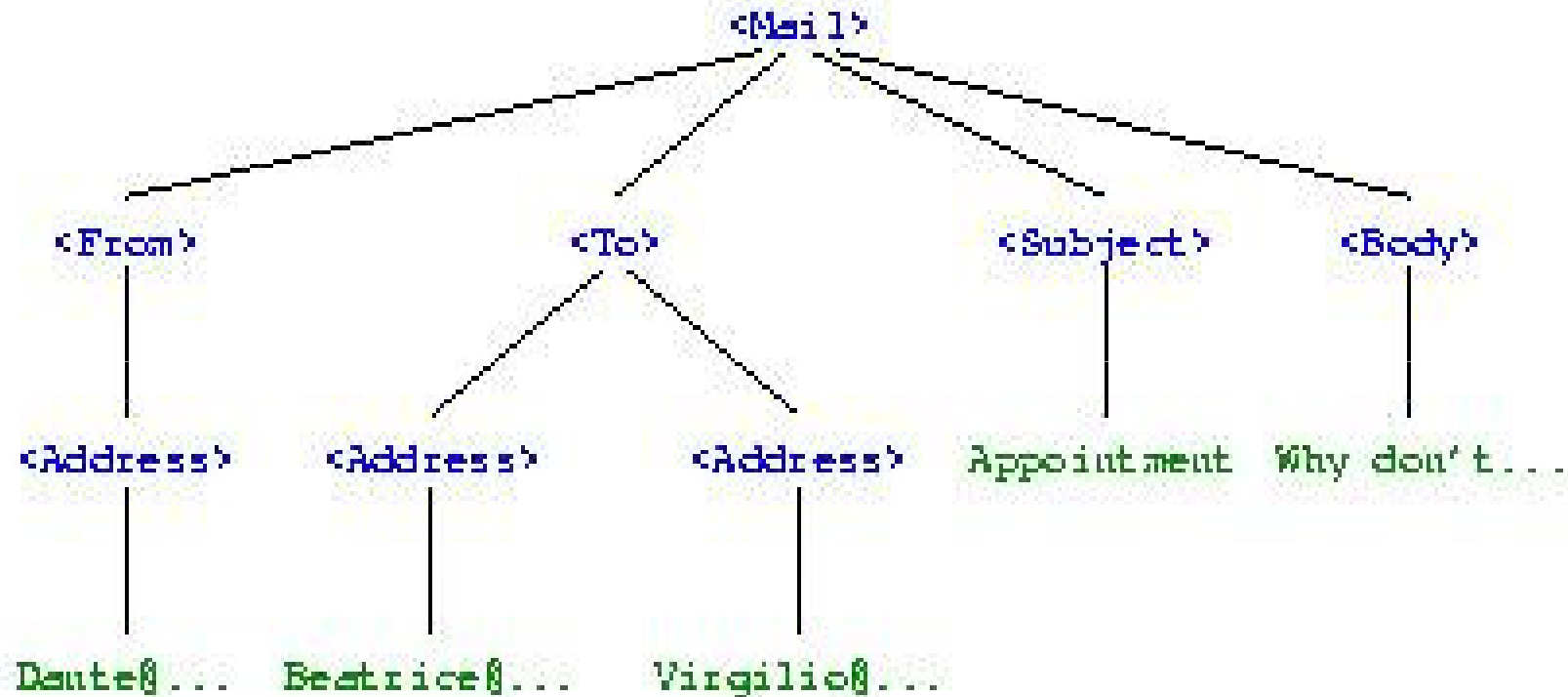
<Subject> Appointment </Subject>

<Body> Why don't we meet at disco.inferno at  
midnight. Tell also Caronte. Cheers,  
- D.A.

</Body>

</Mail>

# Documento XML = albero



# Il Document Object Model

Ogni documento XML corrisponde a una *struttura ad albero*:

- ad ogni elemento corrisponde un nodo interno
- al testo libero corrispondono le foglie
- i figli di un nodo (elemento) sono gli elementi o il testo in esso contenuti
- il modello ad albero può essere esteso anche ad attributi, commenti, istruzioni di processamento, ...

Tale struttura è definita in modo standard dal *Document Object Model*

- fornisce un'API per manipolare i nodi dell'albero
- introduce un livello di astrazione tra il documento XML e l'applicazione che lo deve processare

# HTML e XHTML

*XHTML* è una versione recente di HTML compatibile con XML:

- i documenti HTML non sono necessariamente ben formati
- i documenti XHTML lo sono

Principali *differenze* tra XHTML e XML

- etichette devono essere annidate correttamente
- tutte le etichette devono essere chiuse
- nomi di elementi ed attributi sono minuscoli
- valori di attributi racchiusi tra apici
- altre differenze dovute alle differenze nei DTD di XML ed SGML

È possibile trasformare un documento HTML in un documento XHTML equivalente, se il documento HTML non è ambiguo.

# DOCUMENT TYPE DEFINITION (DTD)

# Funzione della DTD

- Un documento XML *ben formato* può avere una *struttura arbitraria*:
- elementi di nome arbitrario
- attributi arbitrari in qualunque elemento
- elementi organizzati in modo arbitrario
- Sono utili, ma poco più utili di testo non strutturato.
- Serve un meccanismo per *imporre struttura* ad un documento:
- elementi ammessi
- attributi ammessi
- strutturazione degli elementi
- definizione di nuove entità (interne ed esterne)

# Definizione di tipo di documento (DTD)

Una *DTD* specifica quali sono le strutture ammesse per l'istanza di documento:

- specifica il *tipo del documento*, ovvero il tipo dell'elemento radice
- contiene tre insiemi di *dichiarazioni*:
  - di *tipi di elemento*
  - di *attributi*
  - di *entità*

L'insieme di tipi di elemento in un DTD viene detto *vocabolario*

# Dichiarazione di tipo di documento

- Una dichiarazione di tipo di documento (o DOCTYPE) associa una DTD ad un documento XML
- Spesso *definizione* di tipo di documento e *dichiarazione* di tipo di documento vengono confuse
- In realtà la definizione di tipo di documento è una parte della dichiarazione di tipo di documento

# Dichiarazioni in XML

- In XML tutte le *dichiarazioni* hanno la forma:
- `<!OGGETTO-DICHIARATO ... >`
- Esempi:
- `<!DOCTYPE ... >`  
`<!ELEMENT ... >`  
`<!ATTLIST ... >`  
`<!ENTITY ... >`  
`<!NOTATION ... >`

# Dichiarazione di tipo di documento

- Un documento XML *può* contenere una DTD
- La DTD *deve* precedere il primo elemento (l'elemento radice)
- Le dichiarazioni nella DTD possono essere:
- *interne* al documento stesso
  - `<!DOCTYPE esempio [  
    <!ELEMENT esempio (#PCDATA)> ]>`
- *esterne* al documento in un altro file (locale o specificato da un URI)
  - `<!DOCTYPE esempio SYSTEM "esempio.dtd">`
- esterne con un DTD subset interno

# Esempio

```
<!DOCTYPE TechRepDip [
 <!ELEMENT TechRepDip (Intestazione, Sezione+, Bibliografia?)>
 <!ELEMENT Intestazione (Numero, Data, Titolo, Autore+,
Sommaro?)>
 <!ELEMENT Data (Giorno?, Mese, Anno)>
 <!ELEMENT Autore (Cognome, Nome+)>
 <!ELEMENT Sezione (TitoloSezione, Testo?, Sezione*)>
 <!ELEMENT Bibliografia (VoceBiblio)+>
 <!ELEMENT Numero (#PCDATA)>

 ...
 <!ELEMENT VoceBiblio (#PCDATA)>
 <!ATTLIST Sezione id ID #REQUIRED
 num NMTOKEN #IMPLIED
 stato (finale | provvisorio) "finale" >
 <!ENTITY DIS "Dipartimento di Informatica e Sistemistica">
 <!ENTITY SEZIONE1 SYSTEM "sezione1.xml">
 <!ENTITY SEZIONE2 "<Sezione>Ancora da scrivere.</Sezione>">
]>
```

# Documenti XML validi

Un documento XML viene detto *valido* se:

- è *ben formato*, ovvero rispetta tutti vincoli imposti da XML
- contiene una *dichiarazione di tipo di documento*
- è *conforme* alla DTD

# Dichiarazioni di tipo di elemento

- Sintassi di una dichiarazione di tipo di elemento:  
`<!ELEMENT nome-elemento content-model>`
- Il *content model* specifica la struttura degli elementi di nome *nome-elemento*.
- *content model* è un'espressione regolare costruita su:
  - i tipi di elemento
  - #PCDATA (Parsed Character Data), che rappresenta del testo libero (senza etichette)
- Ci sono delle limitazioni nell'uso di #PCDATA nel content model

# Forma del content model

Gli operatori usati nel content model sono quelli delle espressioni regolari:

- *sequenza*, indicata con  $(a,b,\dots,h)$
  - *alternativa*, indicata con  $(a | b | \dots | k)$
  - *opzionalità*, indicata con  $a?$
  - *ripetizione zero o più volte*, indicata con  $a^*$
  - *ripetizione una o più volte*, indicata con  $a^+$
- (dove  $a,b,\dots$  indicano un generico content model)

# Esempio

```
<!ELEMENT ESE (TI, S+)>
<!ELEMENT S (TI?, (S+ | TXT))>
<!ELEMENT TI (#PCDATA)>
<!ELEMENT TXT (#PCDATA)>
```

Esempio di documento valido per questa DTD:

```
<ESE> <TI>esempi xml</TI>
 <S> <TI>tit sez.1</TI> <TXT>testo 1</TXT> </S>
 <S> <TI>tit sez.2</TI>
 <S> <TXT>testo sez.2.1</TXT> </S>
 <S> <TI>titolo sez.2.2</TI>
 <S> <TXT>testo sez.2.2.1</TXT> </S>
 </S>
 </S>
</ESE>
```

# Elementi con contenuto EMPTY

- Un elemento può essere forzato ad avere contenuto vuoto:
- `<!ELEMENT elem EMPTY>`
- Un elemento dichiarato EMPTY può comparire solo con l'etichetta di elemento vuoto.

`<!ELEMENT VUOTO EMPTY>`  
`<!ELEMENT E1 (#PCDATA)>`

- Corretto:  
`<E1></E1> <E1/> <VUOTO/>`
- Scorretti:  
`<VUOTO> deve essere vuoto </VUOTO>`  
`<VUOTO></VUOTO>`

# Uso di elementi con contenuto EMPTY

- Non sono inutili in quanto aggiungono informazione al documento:

- attraverso attributi associati all'elemento:

```
<IMG src="colosseo.gif"
 align="left"
 height="50"
 width="30"/>
```

- attraverso la presenza stessa dell'elemento:

```

 di HTML
```

# Elementi con contenuto ANY

- Si può usare il contenuto ANY per specificare che il contenuto di un elemento è una sequenza di testo ed elementi qualsiasi:

`<!ELEMENT elem ANY>`

- Tutti gli elementi che compaiono in un elemento con contenuto ANY devono essere stati dichiarati.

`<!ELEMENT TUTTO ANY>`

`<!ELEMENT E1 (#PCDATA)>`

`<!ELEMENT E2 (#PCDATA)>`

- Corretto:

`<TUTTO> <E1>xxx</E1> yyy <E2/> <E1/> </TUTTO>`

- Scorretto:

`<TUTTO> <E3>non dichiarato</E3> </TUTTO>`

# Dichiarazioni di attributi

- Un elemento può contenere degli *attributi nell'etichetta iniziale*.
- Se il documento è valido tutti gli attributi devono essere stati dichiarati.
- Sintassi di una *dichiarazione di attributi* per un elemento:
- `<!ATTLIST nome-elemento lista-attributi>`
- *nome-elemento* specifica l'elemento per cui si dichiarano gli attributi.
- *lista-attributi* è una lista di specifiche  
*nome tipo valore-di-default*

# Tipi degli attributi

XML distingue 10 possibili *tipi di attributo*:

- *enumerato*, ovvero una lista di possibili valori
- CDATA: testo senza marcatura (è il tipo più comune); il testo non può contenere "<" e ""
- ID: il valore dell'attributo identifica l'elemento; ci può essere un solo attributo di tipo ID per elemento
- IDREF: il valore dell'attributo è l'ID di un altro elemento
- ENTITY: il nome di un'entità dichiarata nel DTD permette di collegarsi a dati esterni in forma binaria (ad es. un'immagine)

# Tipi degli attributi

XML distingue 10 possibili *tipi di attributo*:

- NMTOKEN: un nome XML corretto in nomi XML coincidono con gli identificatori Java, JavaScript, ...)
- IDREFS, ENTITIES, NMTOKENS: versioni al plurale dei precedenti (valori separati da spazio)
- NOTATION: il nome di una notation XML una notation permette di specificare il formato di dati non-XML e un'applicazione che li processi

# Specifica del valore di default

Il *valore-di-default* può essere

- un *valore esplicito* che viene assunto in mancanza di specifica

```
<!ATTLIST P align (left|center|right) "left" ...>
```

- #REQUIRED, che obbliga a specificare l'attributo
- #IMPLIED, che rende la specifica del valore opzionale
- #FIXED "*valore-fisso*", che fissa l'attributo ad avere un valore fisso (non può essere modificato all'interno del documento)

# Esempio

Elemento img di XHTML:

```
<!ELEMENT img EMPTY>
<!ATTLIST img
 id ID #IMPLIED
 class CDATA #IMPLIED
 title CDATA #IMPLIED
 src CDATA #REQUIRED
 alt CDATA #REQUIRED
 longdesc CDATA #IMPLIED
 height CDATA #IMPLIED
 width CDATA #IMPLIED
 usemap CDATA #IMPLIED
 ismap (ismap) #IMPLIED
 ...
>
```

# Entità

Servono a:

- definire abbreviazioni usate comunemente nel documento
- suddividere il documento in più parti, memorizzate separatamente
- suddividere la DTD in più parti e renderla modulare

# Dichiarazioni e riferimenti ad entità

- *entità interne*: il cui valore è definito nel documento stesso (specie di macro)  
<!ENTITY nome-entita "testo">
- *entità esterne*: il cui valore è accessibile tramite un'URI  
<!ENTITY nome-entita SYSTEM "URI">
- *Riferimento ad un'entità* nel documento: &nome-entita;

# Esempio

- Dichiarazioni nella DTD:

```
<!ENTITY DIS "Dip. di Informatica e Sist.">
<!ENTITY SEZIONE1 SYSTEM "sezione1.xml">
<!ENTITY SEZIONE2 "<Sezione>Da
scrivere.</Sezione">
```

- Riferimenti nell'istanza di documento:

Presso il &DIS; vengono studiati i linguaggi per dati semistrutturati, come illustrato nelle due sezioni seguenti.

&SEZIONE1;

&SEZIONE2;