eXtensible Markup Language (XML)
Elementi di base

Giuseppe Della Penna
Università degli Studi di L'Aquila

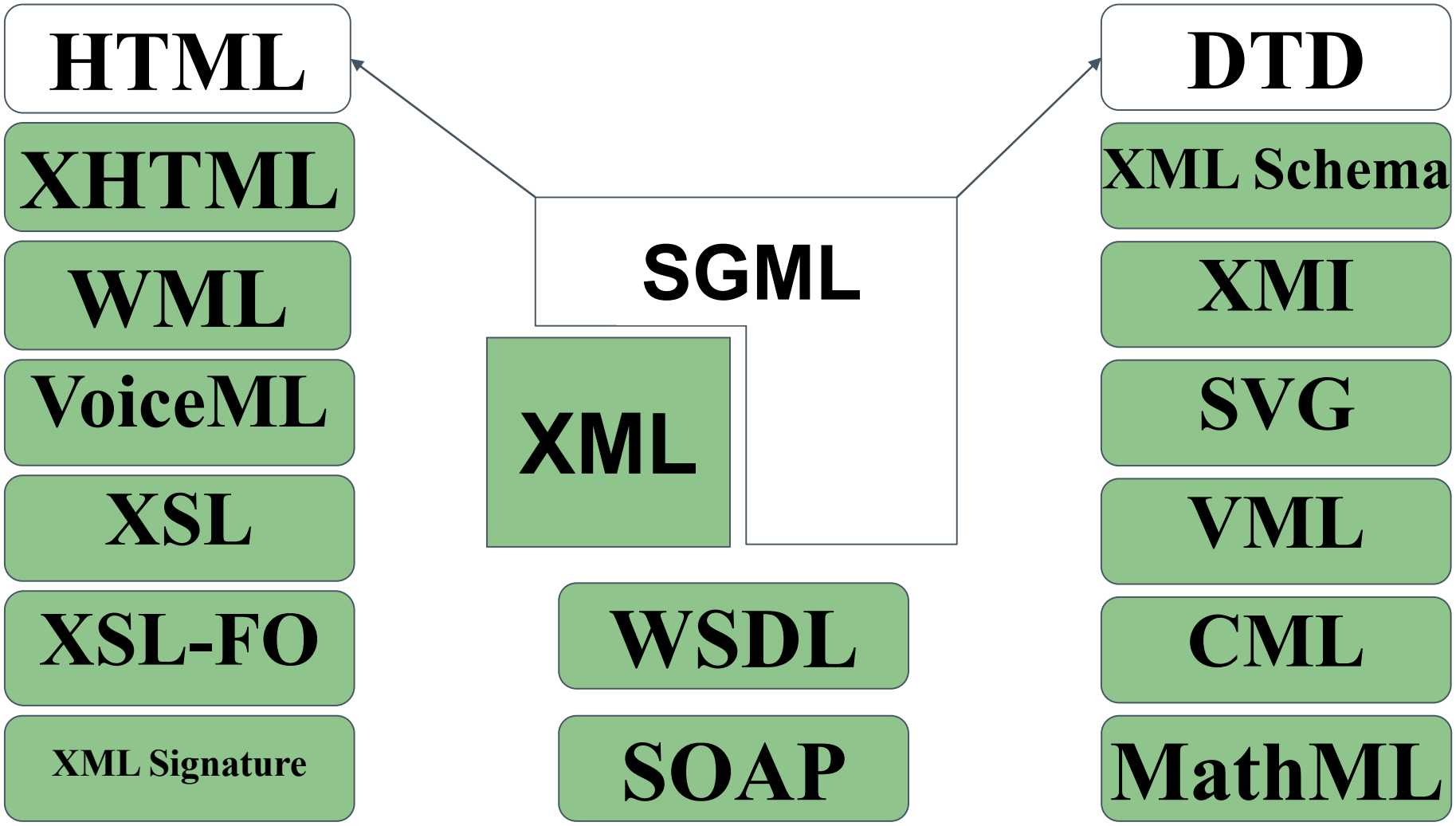
giuseppe.dellapenna@univaq.it
http://people.disim.univaq.it/dellapenna

Versione documento: 230302

Origini di XML

- XML è un metalinguaggio, cioè un linguaggio sulla cui base vengono creati nuovi linguaggi.
- In particolare, XML definisce le regole di base per la creazione **di linguaggi markup**, cioè linguaggi il cui contenuto (testuale) è strutturato tramite particolari delimitatori detti tag.
- XML deriva da SGML, un altro metalinguaggio noto ma diffuso solo in ambienti professionali (es. editoria).
- Rispetto a SGML, XML è stato notevolmente **semplificato** e sono state aggiunte piccole **estensioni** per renderne l'uso più agevole.

La Famiglia di XML



XML: Vantaggi

- XML permette agli sviluppatori di creare facilmente **linguaggi ad-hoc** per contenere informazione strutturata.
- XML è completamente **text-based**, quindi leggibile anche dagli esseri umani e facilmente editabile anche a mano. Supporta UNICODE, quindi è adatto a ogni tipo di scrittura.
- Le strutture definite con XML sono utili anche per creare **strutture dati** indipendenti dalla piattaforma ed auto-descrittive.
- **L'elaborazione automatica** di un linguaggio XML è particolarmente semplice ed efficiente. Le rigide regole di formato e di identificazione dei linguaggi basati su XML ne rendono il trattamento automatico molto conveniente.
- Essendo puro testo (strutturati esattamente come HTML), i dati XML possono essere trasportati usando il **protocollo HTTP** anche attraverso firewall (SOAP, servizi web).

XML: Svantaggi

- I documenti XML, a causa della loro struttura testuale e dei tag, tendono ad essere molto più ingombranti dei corrispondenti in formato binario.
- Le librerie di manipolazione XML non sono veloci come i parser scritti ad-hoc per formati specifici, soprattutto se binari.
- In generale, quindi, l'uso di XML si presenta più oneroso in termini di risorse necessarie (tempi di trasmissione, memoria e tempo CPU necessari alla sua decodifica, ecc.)

XML: Applicazioni

- Nonostante i (pochi) svantaggi visti, l'uso di XML è diffusissimo e in continua espansione:
  - **Servizi Web**
    - SOAP, WSDL, …
  - **Scienza**
    - MathML, CML,…
  - **Web ed Editoria**
    - XHTML, VoiceML, XSL, XSL-FO, …
  - **Multimedia**
    - SMIL, SVG,…
  - **Definizione di strutture formali**
    - XMLSchema, XMI,…
  - **Sicurezza**
    - XML Encryption, XML Signature

Un Documento XML

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="C:\Documenti\Collection.xsl"?>
<!DOCTYPE collection SYSTEM "C:\DocumentiCollection.dtd">
<collection xmlns="www.univaq.it/~gdellape/esempio/">
    <CD owner="giuseppe">
        <song album="darkside" track="13">
            <title>Eclipse</title>
            <length>67</length>
        </song>
        <song album="darkside" track="6">
            <artist>Gilmour, Wright</artist>
            <title>Time</title>
            <length>120</length>
            <comment>  <![CDATA[Time has gone…]]></comment>

        </song>
        <album ID="darkside">
                <artist>Pink Floyd</artist>
                <title>The Dark Side of the Moon</title>
                <year>1963</year>
        </album>
    </CD>
</collection>
```

La Struttura di un Documento XML

- Un documento XML è composto da un prologo e da un corpo
- Il corpo del documento può contenere:
  - **testo**,
  - **tag** (delimitatori della struttura),
  - annotazioni (**commenti**),
  - **processing instructions** (indicazioni per l'elaborazione automatica),
  - **entità** (simili da macro testuali)
  - Inoltre, i tag possono contenere **attributi** e **namespaces**.

Prologo: Dichiarazione XML

<?xml version="1.0" encoding="ISO-8859-1"?>

- La prima riga del prologo è la **dichiarazione XML**, che è obbligatoria e deve essere posta all'immediato inizio del documento.
- L'espressione "<?xml" è detta tag di apertura della dichiarazione XML. La dichiarazione è chiusa dal simbolo "?>".
- All'interno della dichiarazione, troviamo due espressioni della forma nome="valore". Questo tipo di notazione è usata per definire un attributo del tag in cui è contenuta. Un attributo modifica o completa il significato di un tag, ed è un concetto largamente usato in XML.
- Gli attributi della dichiarazione XML sono:
  - **version**: (obbligatorio) indica la versione di XML usata.
  - **encoding**: (opzionale) è nome della codifica dei caratteri usata nel documento (default: UTF-8 o 16, cioè UNICODE a 8 o 16 bit, ISO-8859-1 è quella più adatta ai nostri caratteri nazionali)
  - **standalone**: (opzionale) se vale yes indica che il file non fa riferimento ad altri file esterni. (default: no)

Prologo: Dichiarazione DOCTYPE

- Ai documenti XML possono (e dovrebbero) essere associate delle specifiche che **definiscono formalmente il linguaggio** utilizzato nel documento e le sue regole sintattiche.
- Il sistema predefinito in XML per creare queste specifiche è la document type definition (DTD)
- Se a un documento è associata una **DTD**, è necessario inserire nel prologo una dichiarazione DOCTYPE che dichiari l'associazione. Questa dichiarazione eredita la sintassi del corrispondente SGML.
- Esistono tuttavia altri sistemi di definizione per le specifiche dei linguaggi XML, come gli **Schemi**, che usano metodi di associazione diversi.

Prologo: Dichiarazione DOCTYPE

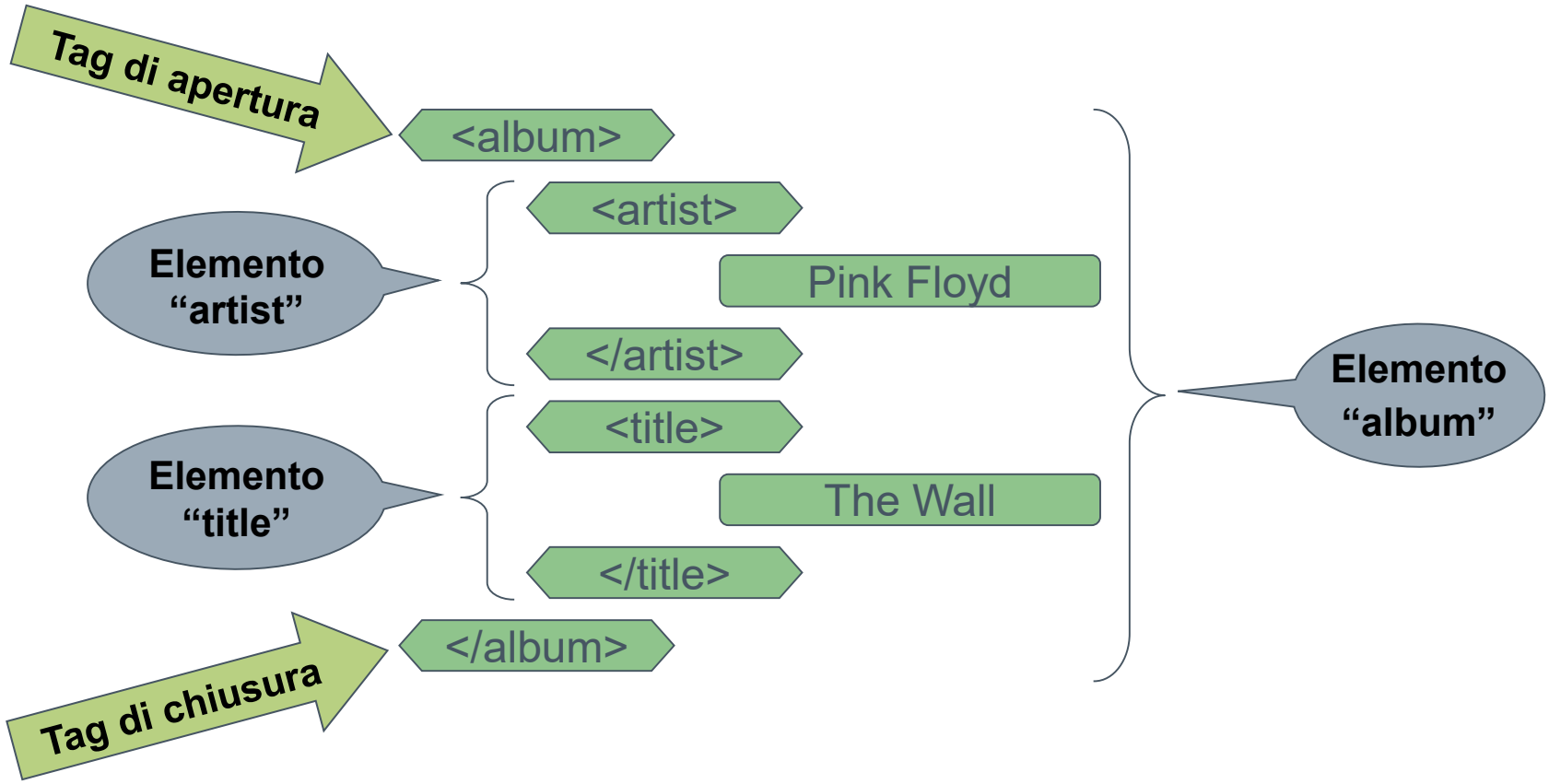
<!DOCTYPE *RootElement ExternalDTDReference* [*InternalDTDSubset*]>

- La dichiarazione si apre con il tag “<!DOCTYPE” ed è chiusa dal simbolo “>”. Al suo interno appaiono i seguenti elementi.
- **RootElement** (obbligatorio) è il nome dell’elemento radice del documento, cioè il nome del tag che conterrà l’intero documento.
- **ExternalDTDReference** (opzionale) punta a un file che contiene la DTD vera e propria, e può valere:
  - **SYSTEM "uri",** dove uri identifica un file esterno.
  - **PUBLIC "pubid" "uri"**, dove pubid è un identificatore univoco per la DTD e uri punta a un file di riferimento che la contiene.
- **InternalDTDSubset** (opzionale) è un DTD, o un suo frammento, che può essere specificato direttamente all’interno del documento.

Elementi

- Gli **elementi** sono alla base della struttura dei documenti XML.
- Un elemento è un **frammento di dati**, *limitato* ed *indentificato* (tramite un nome) da un tag.
- Il contenuto di un elemento è tutto ciò che appare tra il suo tag di apertura e il suo tag di chiusura.
- Gli elementi possono essere nidificati, cioè degli elementi possono far parte del contenuto di un elemento più esterno.

Elementi


Tag di apertura
<album>
Elemento "artist"
<artist>
Pink Floyd
</artist>
Elemento "title"
<title>
The Wall
</title>
Elemento "album"
Tag di chiusura
</album>

Elementi: Regole di Base

- I nomi degli elementi sono **case-sensitive**.
- **Ogni elemento deve essere chiuso**, cioè il suo tag di chiusura deve apparire prima della fine del documento.
- Nel caso di elementi nidificati, **i tag di chiusura devono apparire in ordine inverso a quello di apertura**, cioè i contenuti degli elementi non si possono "accavallare".
- Ogni documento XML deve avere un unico elemento "**radice**", in cui tutti gli altri sono nidificati.

Elementi: Sintassi

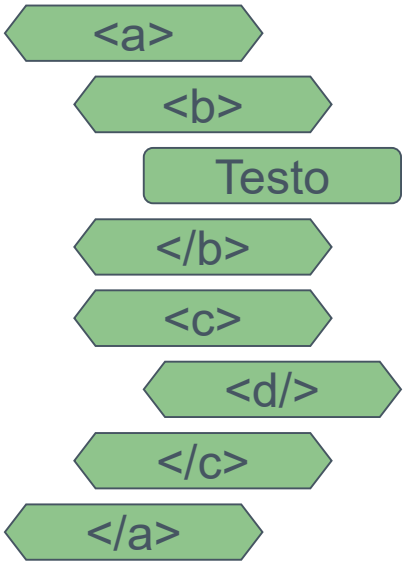
```
1 <nome>
      …
2 </nome>

3 <nome/>

<a>
  <b/>
  <c>
    Testo
    <d>
      <e/>
    </d>
  </c>
</a>
```

- Il tag di apertura di un elemento ha la forma mostrata in (1), dove nome è il nome dell'elemento. Il corrispondente tag di chiusura è mostrato in (2)
- Infine, alcuni elementi possono essere privi di contenuto: in questo caso è possibile omettere il tag di chiusura scrivendo quello di apertura nella forma abbreviata mostrata in (3).

Gerarchia degli Elementi



- Gli elementi, nidificandosi, creano la struttura ad albero tipica dei documenti XML.
- All'interno di questa struttura si definiscono alcuni "rapporti di parentela" utili per individuare gli elementi:
  - a è il nodo radice
  - b e c sono figli di a, il testo è figlio di b, d è figlio di c
  - c è il padre di d, b è il padre del testo, a è il padre di b e c
  - b e c sono fratelli
  - b, c, d e il testo sono discendenti di a, d è un discendente di c, il testo è un discendente di b
  - a è un predecessore di b, c, d e del testo, b è un predecessore del testo, c è un predecessore di d.

Attributi

- Gli attributi permettono di specificare **proprietà degli elementi**, modificandone o meglio definendone il significato.
- Gli attributi vengono **inseriti all'interno dei tag di apertura** degli elementi.
- **L'ordine** con cui gli attributi appaiono nel tag di apertura non è considerato significativo.
- Il valore di un attributo deve essere **semplice**: in caso contrario è meglio usare un elemento nidificato per contenerlo.

Attributi: Regole di Base

- I nomi degli attributi sono **case-sensitive**.
- Lo stesso elemento non può contenere due attributi con lo stesso nome.
- Non sono ammessi attributi senza valore (solo nome).
- Il valore degli attributi deve essere specificato **tra virgolette semplici o doppie**.
- Il valore può contenere **riferimenti ad entità** ma nessun altra struttura XML (elementi, processing instructions, ecc...).

Attributi: Sintassi

1. `<nome attributo="valore">`
2. `<nome attr1="val1" attr2="val2">`
3. `<nome attributo=' "valore" '>`

```
<a x="txt" y="2">
  <c> Testo
    <d>
      <e z="abc123"/>
    </d>
  </c>
</a>
```

- La sintassi di base per un attributo inserito nel tag di apertura di un elemento è mostrata in (1)
- Per specificare più attributi è sufficiente elencarli separandone la definizione con uno o più spazi come mostrato in (2)
- Per includere virgolette nel valore, è necessario usare un tipo diverso da quello usato per delimitare il valore stesso (3)

Namespaces

- I namespaces servono a dichiarare **l'appartenenza di elementi e attributi a un particolare linguaggio** XML, fornendone una semantica.
- Sono particolarmente utili se **più linguaggi vengono mescolati nello stesso documento**, con possibili di collisioni tra nomi.
- Le **dichiarazioni di namespace** sono inserite nei tag di apertura, in modo simile a un attributo, e sono valide per l'elemento e il suo contenuto.

Namespaces: Sintassi

```
1 <name xmlns:prf="uri">
2 <name xmlns="uri">
3 <nome xmlns="uri" xmlns:prf="uri">
```

- Le dichiarazione di namespace esplicito (1), inserita in un tag di apertura, indica che tutti gli elementi il cui nome è prefissato da “prf:” (prefisso di namespace) andranno considerati appartenenti al namespace identificato da uri.
- La speciale dichiarazione di namespace standard (2) indica il namespace di appartenenza per tutti gli elementi privi di un prefisso esplicito di namespace.
- In ogni elemento si possono dichiarare più prefissi di namespace espliciti, ma solo un namespace di default (3)
- Gli uri usati nelle dichiarazioni sono solo identificatori convenzionali per associati ai diversi namespaces, e non puntano ad alcuna informazione particolare.

Namespaces: Esempi

```
<a xmlns="ns1" xmlns:html="ns2">
  <b/>
  <html:p><html:b>testo</html:b>
    <c xmlns="ns3"><d/></c>
    <d/>
    <e xmlns:xsl="ns4" xsl:attr="val">
      <xsl:f>testo</xsl:f>
    </e>
  </html:p>
</a>
```

- Saper manipolare e comprendere i namespace è importante per poter gestire i documenti XML complessi e la loro semantica.
- In questo esempio:
- Il namespace “ns1” contiene gli elementi a,b,d,e.
- Il namespace “ns2” contiene gli elementi html:p,html:b.
- Il namespace “ns3” contiene gli elementi c,d.
- Il namespace “ns4” contiene l’elemento xsl:f e l’attributo xsl:attr.
- *Notare che esistono due elementi d nel documento, appartenenti a namespace diversi.*

Entità

- Nel gergo XML, i documenti sono costituiti da una serie di entità.
  - Ogni carattere è una character entity, ogni tag è un'entità e il documento stesso è un'entità.
- Ogni entità, tranne il documento e il DTD esterno, ha un nome.
- Le entità si distinguono in **parsed** e **unparsed**:
  - Ogni entità parsed ha un suo corrispondente valore testuale. Il parser XML sostituisce l'entità col suo valore quando analizza il documento.
  - Una entità unparsed, invece, non viene sostituita dal parser, e può avere un valore anche non testuale, accessibile tramite le notazioni.

Entità (parsed): Sintassi

1. `&nome;`
2. `&#numero;`
3. `&#xnumero;`

```
&gt;    →  >
&lt;    → <
&quot;  →  "
&amp;   →  &
&#32;   →  [spazio]
&#x20;  →  [spazio]
```

- Le **entità generali**, che possono rappresentare stringhe qualsiasi, sono definite nel DTD e si richiamano nel documento XML con la sintassi (1), dove nome è il nome dell'entità.
- Le **entità carattere**, che rappresentano singoli caratteri UNICODE, si richiamano con la sintassi (2), dove numero è il codice decimale UNICODE per il carattere, oppure con la sintassi (3), dove numero è il codice esadecimale UNICODE per il carattere.

Entità: Uso

- Le entità **parsed** sono un modo pratico per inserire stringhe nel documento facendo riferimento a una definizione esterna, invece di scriverle esplicitamente.
- Sono utili nel caso ci siano **caratteri non digitabili** direttamente, o per **espandere stringhe** usate di frequente, oppure per scrivere **caratteri che non sono ammessi in maniera esplicita in un contesto**, perché riservati (come le virgolette o i segni ‘<’ e ‘>’).

Testo

- Il testo inseribile nei documenti XML comprende tutti i caratteri **definiti in UNICODE**.
- È possibile inserire caratteri speciali o riservati tramite **entità carattere**.
- È possibile inserire stringhe predefinite tramite **entità generali**.
- **Non è possibile usare esplicitamente i caratteri ‘>’, ‘<’ e ‘&’,** per i quali è sempre necessario usare le corrispondenti entità carattere.

Sezioni CDATA

```
<![CDATA[
<< &pippo;
solo testo!<
>>
]]>
```

- Le sezioni CDATA di definire esplicitamente **aree in cui si trova solo testo**.
- All'interno delle sezioni CDATA il parser non effettua alcuna operazione di riconoscimento ed espansione per elementi, attributi, entità e altre strutture XML
- Il tag di apertura di una sezione CDATA è la stringa "<![CDATA[", mentre il tag d chiusura è "]]>", che ovviamente non può comparire nel contenuto.

Processing Instructions

<?target data ?>

- Le Processing Instructions (PI) vengono usate per **passare informazioni extra ai programmi che manipoleranno il file XML** e possono apparire ovunque nel documento.
- La forma generale di una PI prevede un tag di apertura del tipo “<?target”, dove target identifica quale applicazione dovrà elaborare la processing instruction, e un tag di chiusura “?>”.
  - *Notare che la dichiarazione XML non è altro che una processing instruction.*
- All’interno del tag è possibile scrivere qualsiasi tipo di dati testuali. L’unica regola è che i dati non possono contenere la sequenza “?>”. I due esempi riportati sotto sono rispettivamente (1) la PI che associa a un documento il suo foglio di stile XSL e (2) uno script PHP.

<?xml-stylesheet type="text/xsl" href="sms_pdf.xslt"?>
<?php echo “hello” ?>

Commenti

<!-- Questo è un commento XML (e SGML) -->

- I commenti sono utili agli esseri umani, e vengono ignorati dai programmi di manipolazione XML.
- I commenti possono apparire ovunque, tranne che all'interno del valore di un attributo.
- I commenti XML seguono la sintassi SGML, e sono quindi identici a quelli usati, ad esempio, in HTML.
- Il tag di apertura di un commento è la sequenza "<!--", mentre il tag di chiusura è la sequenza "-->"
- Il contenuto del commento è testo generico, che non deve però contenere la sequenza di chiusura.

Validazione di Documenti XML

- **Un documento XML è ben formato** se rispetta le regole generali di sintassi viste nella parte precedente.
- **Un documento XML è valido** se è ben formato e rispetta le regole sintattiche e semantiche contenute del DTD associato. *Un documento senza DTD non è mai valido*.
- Esistono **parser validanti e non validanti**. Questi ultimi possono ignorare tutto l'eventuale DTD, tranne le dichiarazioni di entità generali.

Riferimenti

- **Specifica di XML dal W3C**
  http://www.w3c.org/TR/XML/