



Tecnologie Web T

Introduzione a XML

Home Page del corso: <http://www-db.disi.unibo.it/courses/TW/>

Versione elettronica: 2.01.XML.pdf

Versione elettronica: 2.01.XML-2p.pdf

Che cos'è XML?

- XML: **E**xtensible **M**arkup **L**anguage:
 - è un **linguaggio** che consente la rappresentazione di documenti e dati strutturati su supporto digitale
 - è uno strumento potente e versatile per la creazione, memorizzazione e distribuzione di documenti digitali
 - la sua **sintassi rigorosa** e al contempo **flessibile** consente di utilizzarlo nella rappresentazione di dati strutturati anche molto complessi

Le origini

- XML è stato sviluppato dal **World Wide Web Consortium**
- Nel **1996** è stato formato un gruppo di lavoro con l'incarico di definire un linguaggio a markup estensibile di uso generale
- Le **specifiche** sono state rilasciate **come W3C Recommendation** nel **1998** e aggiornate nel **2004**
- **XML deriva da SGML**, che come sappiamo è un linguaggio di mark-up dichiarativo sviluppato dalla International Standardization Organization (**ISO**), e pubblicato ufficialmente nel 1986 con la sigla ISO 8879
- **XML nasce come un sottoinsieme semplificato di SGML orientato all'utilizzo su World Wide Web**
- Ha assunto ormai un ruolo autonomo e una diffusione ben maggiore del suo progenitore

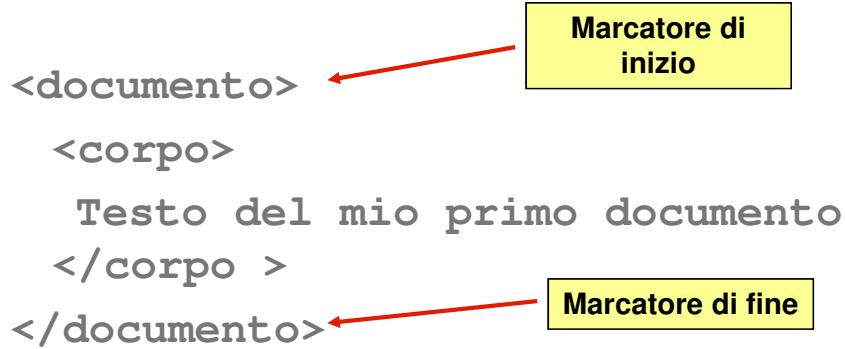
XML come linguaggio di markup

- Come SGML **XML** è un **linguaggio a marcatori** (markup)
- Un **linguaggio di markup** è composto da istruzioni, definite **tag** o **marcatori**, che descrivono la struttura e la forma di un documento
- Ogni marcatore (o coppia di marcatori) identifica un **elemento** o componente del documento
- I marcatori vengono inseriti all'interno del documento
- Un documento XML è "leggibile" da un utente umano senza la mediazione di software specifico 😊

Esempio

- Un documento XML è leggibile ,chiaro, intuibile:

```
<documento>
  <corpo>
    Testo del mio primo documento
  </corpo >
</documento>
```



- **Attenzione:** XML è **case sensitive**
 - nei **nomi dei tag** distingue fra maiuscole e minuscole

Altro esempio...

```
<prenotazione>
  <idVolo>PA321</idVolo>
  <idCliente>PP2305</idCliente>
  <data>22-10-2001</data>
  <prezzo valuta="Euro">245</prezzo>
</prenotazione>
```

XML: caratteristiche

- XML è indipendente dal tipo di piattaforma hardware e software su cui viene utilizzato
- Permette la rappresentazione di qualsiasi tipo di documento (e di struttura) indipendentemente dalle finalità applicative
- È indipendente dai dispositivi di archiviazione e visualizzazione
 - può essere archiviato su qualsiasi tipo di supporto digitale
 - può essere visualizzato su qualsiasi dispositivo di output
 - può essere facilmente «trasMESSO» via Internet tramite i protocolli a noi ben noti HTTP, SMTP, FTP

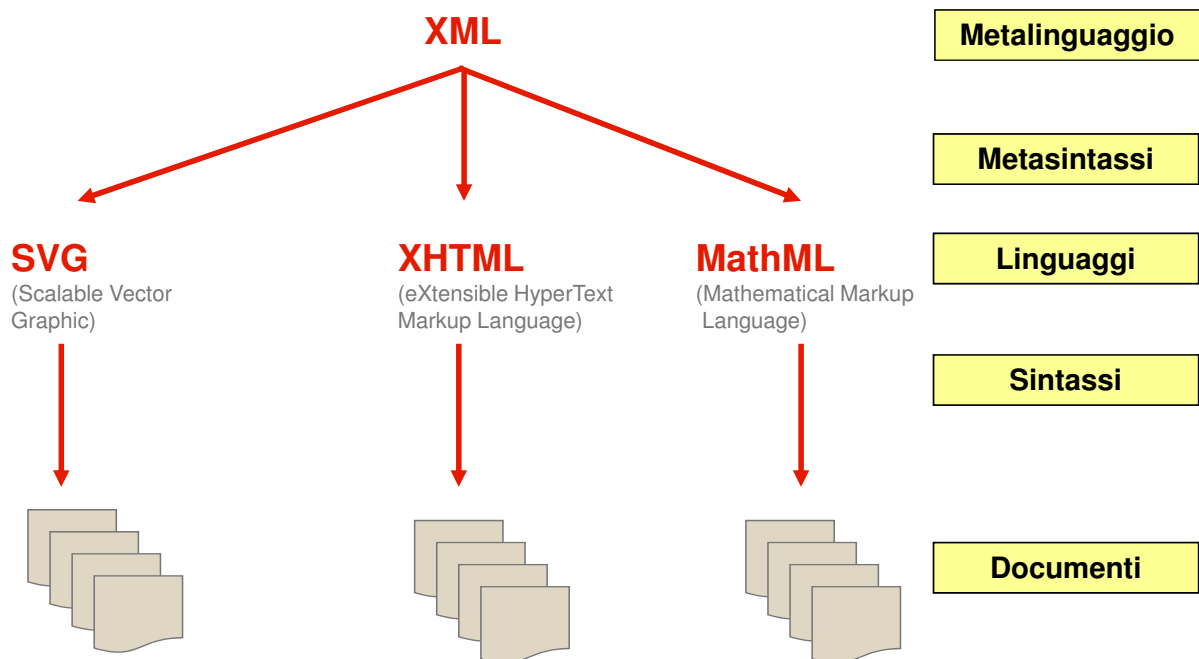
XML: caratteristiche

- XML è uno standard di pubblico dominio
- ogni software “conforme a XML” è in grado di gestire dati in formato XML
- sono disponibili numerose applicazioni e librerie open source per la manipolazione di dati in formato XML basate su diversi linguaggi di programmazione (Java, C, C#, Python, Perl, PHP...)
- una applicazione in grado di elaborare dati in formato XML viene definita **elaboratore XML**

XML come metalinguaggio

- XML è un **metalinguaggio**
- Definisce un **insieme regole (meta)sintattiche**, attraverso le quali è possibile descrivere **formalmente un linguaggio di markup**, detto **applicazione XML**
- Ogni applicazione XML:
 - eredita un insieme di caratteristiche sintattiche comuni
 - definisce una sua sintassi formale
 - è dotata di una semantica

Metalinguaggio e linguaggi



Linguaggi e grammatiche

- Per definire un linguaggio è necessario un meccanismo che vincoli l'utilizzo dei tag all'interno dei documenti
- Si deve poter stabilire **quali tag possono essere utilizzati e come**, secondo una precisa struttura logica
- Abbiamo cioè bisogno di definire una **grammatica**
- Una grammatica è un **insieme di regole** che indica quali vocaboli (elementi) possono essere utilizzati e con che struttura è possibile comporre frasi (documenti)
- Se un **documento XML rispetta le regole** definite da una grammatica è detto **valido** per un particolare linguaggio di markup

Documenti ben formati e documenti validi

- In XML ci sono **regole sintattiche** (o meglio meta-sintattiche)
 - **come** dobbiamo scrivere le informazioni all'interno dei documenti
- Ci possono essere (ma non è obbligatorio) **regole semantiche**
 - **cosa** possiamo scrivere in un documento XML
- Un documento XML che rispetta le **regole sintattiche** si dice **ben formato** (**well-formed**)
- Un documento XML che rispetta le **regole sintattiche** e le **regole semantiche** si dice **valido**
- Un documento ben formato può non essere valido rispetto ad una grammatica, mentre un documento valido è necessariamente ben formato

Struttura logica di un documento XML

- Un documento XML
 - è strutturato in modo gerarchico
 - è composto da **elementi**
- Un **elemento**
 - rappresenta un **componente logico** del documento
 - può contenere un frammento di testo oppure altri elementi (**sotto-elementi**)
- Ad un elemento possono essere associate informazioni descrittive chiamate **attributi**
- Gli elementi sono organizzati ad albero con radice **root**
- Ogni documento XML può essere rappresentato come un albero
 - **document-tree**

Struttura fisica di un documento XML

- Un documento XML è un semplice file di testo (.xml)
- La **struttura del documento** viene rappresentata mediante marcatori (**markup**)
- Gli **elementi** sono rappresentati mediante **tag**
 - coppie di marcatori che racchiudono il contenuto dell'elemento
- I **sottoelementi** sono tag contenuti all'interno di un altro tag
- Gli **attributi** vengono rappresentati sotto forma di coppie **nome-valore** all'interno dei tag
- La **radice** è un tag che racchiude tutto il resto del documento (e quindi tutti gli altri tag)
- Un documento può inoltre contenere **spazi bianchi**, **a capo** e **commenti**

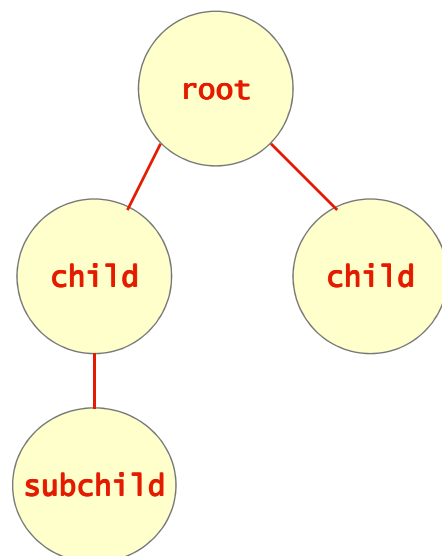
Aspetti di sintassi generale

- Un documento XML è una stringa di caratteri ASCII o Unicode
- Nomi di elementi, attributi e entità sono **case-sensitive**
- Il mark-up è separato dal contenuto testuale mediante caratteri speciali:
 - **< > &** (parentesi angolari e ampersand)
- I caratteri speciali non possono comparire come contenuto testuale e devono essere eventualmente sostituiti mediante i riferimenti a entità
 - **<** (<), **>** (>), **&** (&)

Struttura logica e fisica

- Esiste una corrispondenza diretta fra struttura fisica e struttura logica (tree)

```
<root>  
  <child>  
    <subchild>  
      ...  
    </subchild>  
  </child>  
  <child>  
    ...  
  </child>  
</root>
```



Struttura formale di un documento XML

- Un documento è costituito da due parti
 - **Prologo**: contiene una **dichiarazione XML** ed il **riferimento (opzionale) ad altri documenti** che ne definiscono la struttura o direttive di elaborazione
 - **Corpo**: è il **documento XML vero e proprio**

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/css" href="gree.css"?>

<root>
  <!-- Questo è un commento -->
  <child>
    ...
  </child>
  <child>
    ...
  </child>
</root>
```

Prologo

Corpo

XML

17

Prologo: XML Declaration

- Ogni documento XML inizia con un prologo che contiene una **XML declaration**
- Forme di XML declaration:
 - **<?xml version="1.0"?>**
 - **<?xml version="1.0" encoding="UTF-8"?>**
- Contiene informazioni su:
 - Versione: 1.0
 - Set di caratteri (opzionale)

XML

18

Prologo: riferimenti a documenti esterni

- Il prologo può contenere riferimenti a documenti esterni utili per il trattamento del documento
- **Processing instructions:** istruzioni di elaborazione
 - Esempio 1: rappresentazione mediante CSS:
`<?xml-stylesheet type="text/css" href="gree.css"?>`
- **Doctype declaration:** grammatica da utilizzare per la validazione del documento
 - grammatica contenuta in un file locale
`<!DOCTYPE book SYSTEM "book.dtd">`
 - grammatica accessibile ad un URL pubblico
`<!DOCTYPE book PUBLIC "http://www.books.org/book.dtd">`

Commenti

- I commenti possono apparire ovunque in un documento XML (sia nel prologo che nel corpo)
- I commenti sono utili per
 - spiegare la struttura del documento XML
 - commentare parti del documento durante le fasi di sviluppo e di test del nostro software
- I commenti non vengono mostrati dai browser ma sono visibili da parte di chi guarda il codice sorgente del documento XML

`<!-- Questo è un commento -->`

Elementi e Tag

- Un **elemento** è un frammento di testo racchiuso fra uno **start tag** e un **end tag**
- Uno **start tag** è costituito da un nome più eventuali attributi (v. slide succ.) racchiusi dai simboli '<', '>'

<TagName attribute-list>

- Un **end tag** è costituito da un nome (lo stesso dello start tag) racchiuso da '</','>':

</TagName>

- Un **tag vuoto** è rappresentabile come:

<TagName attribute-list />

- Equivale a

<TagName attribute-list></TagName>

! Attenzione: I tag non possono avere nomi che iniziano per *XML, XMI, Xml, xml...*

Attributi

- A ogni elemento possono essere associati uno o più **attributi** che ne specificano ulteriori caratteristiche o **proprietà** non strutturali
- Ad esempio:
 - la lingua del suo contenuto testuale
 - un identificatore univoco
 - un numero di ordine
 - ...
- Gli attributi XML sono caratterizzati da
 - un **nome** che li identifica
 - un **valore**

Esempio di documento con attributi

```
<?xml version="1.0" ?>
<articolo titolo="Titolo dell'articolo">
  <paragrafo titolo="Titolo del primo paragrafo">
    <testo>Blocco di testo del primo
paragrafo</testo>
    <immagine file="immagine1.jpg"></immagine>
  </paragrafo>
  <paragrafo titolo="Titolo del secondo paragrafo">
    <testo>Blocco di testo del secondo
paragrafo</testo>
    <codice>Esempio di codice</codice>
    <testo>Altro blocco di testo</testo>
  </paragrafo>
  <paragrafo tipo="bibliografia">
    <testo>Riferimento ad un articolo</testo>
  </paragrafo>
</articolo>
```

Elementi o attributi?

- Qualche regola per decidere:
 - Un elemento è estendibile in termini di contenuto (con elementi figli) e di attributi
 - Un attributo non è estendibile: può solo modellare una proprietà di un elemento in termini di valore
 - Un elemento è un'entità a se stante (un oggetto?)
 - Un attributo è strettamente legato ad un elemento
 - Un attributo può solamente contenere un valore "atomico"
- In pratica non c'è una regola valida in assoluto
- La scelta dipende da diversi fattori: leggibilità, semantica, tipo di applicazione, efficienza...

Elementi o attributi: esempio

- Vediamo tre varianti dello stesso pezzo di documento che usano in modo diverso elementi e attributi

```
<libro isbn="1324AX" titolo="On the road" />
```

```
<libro isbn="1324AX">  
  <titolo>On the road</titolo>  
</libro>
```

```
<libro>  
  <isbn>1324AX</isbn>  
  <titolo>On the road</titolo>  
</libro>
```

Riferimenti ad entità

- I **riferimenti ad entità** servono per rappresentare caratteri riservati (per esempio, < > o &)

Nome entità	Riferimento	Carattere
lt	<	<
gt	>	>
amp	&	&
apos	'	'
quot	"	"

- Oppure per rappresentare caratteri UNICODE mediante la notazione **&#XXXX**:
 - ½ → ½
 - è → è

Sezione CDATA

- Per poter inserire brani di testo (porzioni di codice XML o XHTML) senza preoccuparsi di sostituire i caratteri speciali si possono utilizzare le sezioni **CDATA** (**Character Data**)
- Il testo contenuto in una sezione CDATA **NON** viene **analizzato dal parser**
- Una sezione CDATA può contenere caratteri “normalmente” proibiti
- Si utilizza la seguente sintassi:

<![CDATA[*Contenuto della sezione* **]]>**

- L'unica sequenza non ammessa è **]]** (chiusura)
- Esempi:

```
<E1> <![CDATA[ <<"'!] && ]]> </E1>
<E> <![CDATA[<Elemento/><A>Ciao</A>]]> </E>
```

Conflitti sui nomi

- Capita abbastanza comunemente, soprattutto in documenti complessi, di dare nomi uguali ed elementi (o attributi) con significati diversi
- Ad esempio:

```
<libro>
  <autore>
    <titolo>Sir</titolo>
    <nome>William Shakespeare</nome>
  </autore>
  <titolo>Romeo and Juliet</titolo>
</libro>
```

Namespace

- Per risolvere il problema si ricorre al concetto di “spazio dei nomi” (namespace)
- Si usano **prefissi che identificano il vocabolario di appartenenza** di elementi ed attributi
- Ogni **prefisso** è associato ad un **URI** (Uniform Resource Identifier) ed è un alias per l’URI stesso
- L’URI in questione è normalmente un URL: si ha quindi la certezza di univocità
- È un meccanismo simile ai nomi lunghi delle classi in Java (i package definiscono un sistema di namespace):
 - Nome breve: **JButton**
 - Nome lungo: **javax.swing.JButton**

Esempio di uso di namespace

- Riprendiamo l’esempio del libro usando i namespace:

Prefisso **Dichiarazione del prefisso e associazione all’URI** **URI**

```
<lb:libro xmlns:lb="mysite.com/libri">
  <au:autore xmlns:au="mysite.com/autori">
    <au:titolo>Sir</au:titolo>
    <au:nome>William Shakespeare</au:nome>
  </au:autore>
  <lb:titolo>Romeo and Juliet</lb:titolo>
</lb:libro>
```

Definizione di namespace

- Per **definire** un namespace si usa la seguente sintassi:
`xmlns:NamespacePrefix="NamespaceURI"`
- La definizione è un attributo di un elemento e può essere messa ovunque all'interno del documento
- Lo **scope** del namespace è l'elemento all'interno del quale è stato dichiarato
 - Si estende a tutti i sotto-elementi
 - Se si dichiara un namespace nell'elemento radice, il suo scope è l'intero documento
- L'URI può essere qualsiasi (il parser non ne controlla l'univocità) ma dovrebbe essere scelto in modo da essere effettivamente univoco

Esempio

```
<DC:Docenti xmlns:DC="www.unibo.it/docenti">
  <DC:Docente codAteneo="112233">
    <DC:Nome>Ilaria</DC:Nome>
    <DC:Cognome>Bartolini</DC:Cognome>
    <CR:Corso id="123" xmlns:CR="www.unibo.it/corsi">
      <CR:Nome>Tecnologie Web T</CR:Nome>
    </CR:Corso>
    <CO:Corso id="124" xmlns:CO="www.unibo.it/corsi">
      <CO:Nome>Multimedia Data Management M</CO:Nome>
    </CO:Corso>
  </DC:Docente>
</DC:Docenti>
```

1. CR e CO sono prefissi "collegati" allo stesso namespace
2. Nel secondo elemento Corso è necessario ripetere la dichiarazione di namespace poiché ricade fuori dallo scope della prima dichiarazione

Per evitare la seconda dichiarazione basta dichiarare il namespace in un elemento più in alto nella gerarchia

Namespace di default

- È possibile definire un **namespace di default** associato al **prefisso nullo**
- Tutti gli elementi non qualificati da prefisso appartengono al namespace di default
- Attenzione: riduce la leggibilità di un documento

```
<Docenti xmlns="www.unibo.it/docenti">
  <Docente codAteneo="112233">
    <Nome>Ilaria</Nome>
    <Cognome>Bartolini</Cognome>
    <CR:Corso id="123" xmlns:CR="www.unibo.it/corsi">
      <CR:Nome>Tecnologie Web T</CR:Nome>
    </CR:Corso>
  </Docente>
</Docenti>
```

Ridefinizione di prefissi

- Un **prefisso** di namespace (anche quello vuoto di default) può essere associato a diversi namespace all'interno di uno stesso documento
- È però preferibile evitare le ridefinizioni: riducono la leggibilità del documento

```
<PR:Docenti xmlns:PR="www.unibo.it/docenti">
  <PR:Docente codAteneo="112233">
    <PR:Nome>Ilaria</PR:Nome>
    <PR:Cognome>Bartolini</PR:Cognome>
    <PR:Corso id="123" xmlns:PR="www.unibo.it/corsi">
      <PR:Nome>Tecnologie Web T</PR:Nome>
    </PR:Corso>
  </PR:Docente>
</PR:Docenti>
```

Vincoli di buona formazione

- Affinché un documento XML sia **ben formato**:
 - Deve contenere una dichiarazione corretta
 - Il corpo deve avere un unico elemento radice
 - Ogni elemento deve avere un tag di apertura e uno di chiusura
 - se l'elemento è vuoto si può utilizzare la forma abbreviata (<nometag/>)
 - Gli elementi devono essere opportunamente nidificati, cioè i tag di chiusura devono seguire l'ordine inverso dei rispettivi tag di apertura
 - I nomi dei tag di apertura e chiusura devono coincidere
 - anche in termini di maiuscole e minuscole
 - I valori degli attributi devono sempre essere racchiusi tra singoli o doppi apici

Documenti ben formati e documenti validi

- Come già detto, in XML ci sono **regole sintattiche**
 - **come** dobbiamo scrivere le informazioni all'interno dei documenti
- Ci possono essere **regole semantiche** (definite da grammatiche)
 - **cosa** possiamo scrivere in un documento XML
- Un documento XML che rispetta le **regole sintattiche** si dice **ben formato**
- Un documento XML che rispetta le **regole sintattiche** e le **regole semantiche** si dice **valido**

Validazione: Document Type Definition (DTD)

- Un primo strumento per definire **grammatiche** è costituito dalla **Document Type Definition**
- Un DTD è costituito da un elenco di dichiarazioni (**markup declaration**) che descrivono la struttura del documento
- Le dichiarazioni di un DTD definiscono:
 - gli elementi strutturali (**element**) di un documento mediante un identificatore generico
 - il modello di contenuto di ogni elemento (**content model**), ovvero gli elementi che contiene e le loro relazioni (un elemento può essere vuoto)
 - la lista degli **attributi** associati a ciascun elemento e il loro tipo

Limiti e problemi dei DTD

- I DTD sono difficili da comprendere
- Sono scritti in un linguaggio diverso da quello usato per descrivere le informazioni
- Soffrono di alcune limitazioni:
 - Non permettono di definire il tipo dei dati
 - *! ogni dato è di tipo "testo"*
 - Non consentono di specificare il numero minimo o massimo di occorrenze di un tag in un documento

Validazione: XML Schema (XSD)

- Dato che XML può descrivere tutto perchè non usarlo per descrivere anche lo schema di un documento?
- È stato quindi definito lo standard XSD (**XML Schema Definition**)
- XSD nasce dall'idea di utilizzare XML per descrivere la struttura di XML:
 - Descrive le regole di validazione di un documento
 - Permette di tipizzare i dati (intero, stringa, ora, data, ecc.)
 - È estensibile ed aperto alla possibilità di supportare modifiche

Elementi di XSD

- Un documento XML Schema (XSD) comprende:
- **Namespace** di riferimento:
<http://www.w3.org/2001/XMLSchema>
- **Dichiarazione di:**
 - Elementi
 - Attributi
- **Definizione di tipi**
 - Semplici
 - Complessi
 - Estesi

HTML e XML: XHTML

- HTML è una grammatica XML?
- Quasi, però sono ammessi “pasticci” che XML non prevede:
 - Tag non chiusi: `
` (in XML `
</br>` o `
`)
 - Tag “incrociati” `<u>Ciao<u>`
- HTML non è una grammatica XML
- È stata definita una versione di HTML “corretta” in modo da rispettare la sintassi XML: XHTML
- Un documento XHTML è un documento XML ben formato che può essere validato su uno schema definito dal W3C
- Quindi XHTML è una grammatica XML

DOM

- Il DOM (Document Object Model) è un modello ad oggetti definito dal W3C per navigare e creare contenuti XML
- Rappresenta il contenuto di un documento XML tramite un albero in memoria
- Permette di navigare l'albero ragionando per gradi di parentela (nodi figli, nodo padre, ecc.)
- Esistono 3 interfacce base
 - **Node** (è praticamente la base di tutto)
 - **NodeList** (collezione di nodi)
 - **NamedNodeMap** (collezione di attributi)
- Un **parser DOM** è un'applicazione in grado di leggere un file XML e creare un DOM e viceversa

Presentazione di documenti XML

- Un documento XML definisce il contenuto informativo e non come deve essere rappresentato tale contenuto
- La **presentazione di un documento XML viene controllata da uno o più fogli di stile**
- I linguaggi di stile utilizzabili con XML sono
 - *Extensible Stylesheet Language* (**XSL**)
 - *Cascading Style Sheet* (**CSS**) (usati con **XHTML**)

XSL

- **XSL** = e**X**tensible **S**tylesheet **L**anguage
- Si occupa della trasformazione e della impaginazione di contenuti XML
- Si basa principalmente su:
 - **XSLT** (**XSL** for **T**ransformations): gestisce le trasformazioni e non l'impaginazione dei contenuti
 - **XSL-FO** (**XSL** **F**ormating **O**bjects): orientato alla visualizzazione ed impaginazione dei contenuti (es. in PDF)
 - **XPath** (**XML** **P**ath Language): serve per costruire percorsi di ricerca di informazioni all'interno di documenti XML

XSLT

- XSLT è un linguaggio di programmazione a tutti gli effetti
- Permette di gestire **variabili**, **parametri**, **cicli**, **condizioni**, **funzioni**
- È una grammatica XML
- Lavora sulla **struttura** del documento
 - Costruisce l'albero del documento (DOM)
 - Lo attraversa cercando le informazioni indicate
 - Produce un nuovo documento - di solito **XHTML** - applicando le regole definite

“My CD Collection” <http://www.w3schools.com/xsl/tryxslt.asp?xmlfile=catalog&xsltfile=catalog>

XML Code:

Edit and Click Me >>

XSLT Code:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!-- Edited by XMLSpy® -->
<catalog>
  <cd>
    <title>Empire Burlesque</title>
    <artist>Bob Dylan</artist>
    <country>USA</country>
    <company>Columbia</company>
    <price>10.90</price>
    <year>1985</year>
  </cd>
  <cd>
    <title>Hide your heart</title>
    <artist>Bonnie Tyler</artist>
    <country>UK</country>
    <company>CBS Records</company>
    <price>9.90</price>
    <year>1988</year>
  </cd>
  <cd>
    <title>Greatest Hits</title>
    <artist>Dolly Parton</artist>
    <country>USA</country>
    <company>RCA</company>
    <price>9.90</price>
  </cd>
</catalog>
```

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!-- Edited by XMLSpy® -->
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

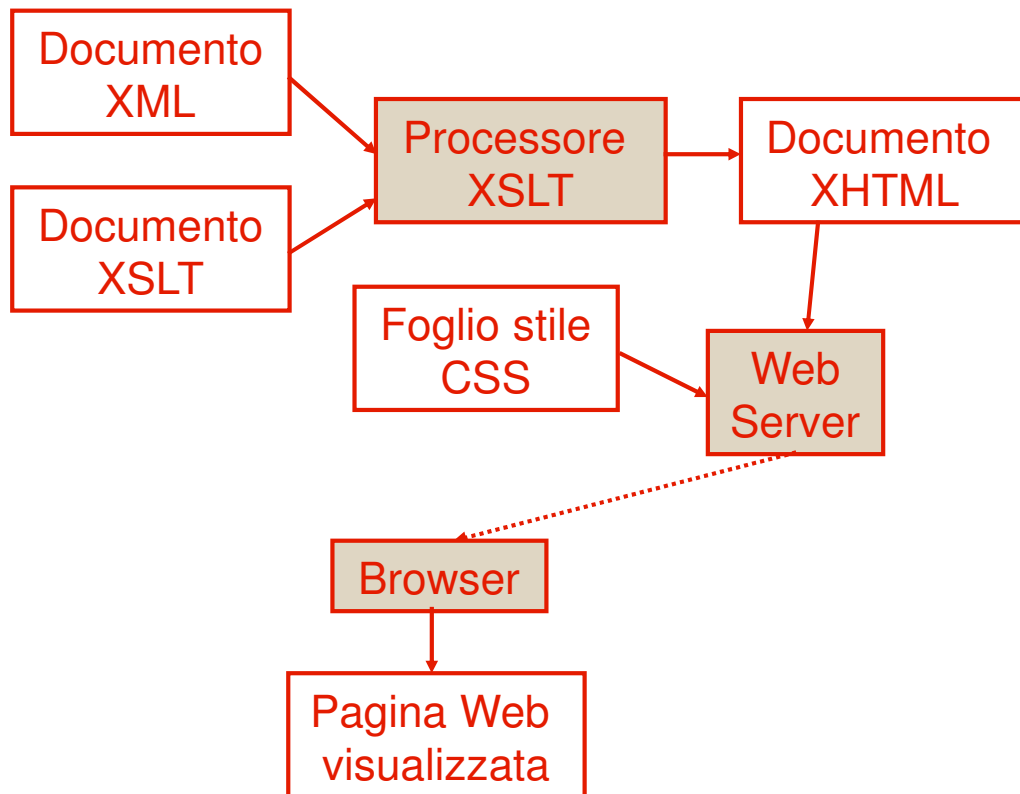
  <xsl:template match="/">
    <html>
    <body>
      <h2>My CD Collection</h2>
      <table border="1">
        <tr bgcolor="#9acd32">
          <th>Title</th>
          <th>Artist</th>
        </tr>
        <xsl:for-each select="catalog/cd">
          <tr>
            <td><xsl:value-of select="title"/></td>
            <td><xsl:value-of select="artist"/></td>
          </tr>
        </xsl:for-each>
      </table>
    </body>
    </html>
  </xsl:template>
</xsl:stylesheet>
```

Your Result:

My CD Collection

Title	Artist
Empire Burlesque	Bob Dylan
Hide your heart	Bonnie Tyler
Greatest Hits	Dolly Parton

Un esempio tipico completo



Riferimenti

- XML Specification: <http://www.w3.org/XML/>
- XSL Specification: <http://www.w3.org/Style/XSL/>
- Guida in inglese (molto completa e ben fatta)
<http://www.w3schools.com/xml/default.asp>