

## Introduzione al linguaggio XML, eXtensible Markup Language

Lezione del Corso Interazione Uomo Macchina 1,

Docente Francesco Mele

Corso di Laurea in Informatica Università di Napoli Federico II,  
Anno Accademico 2005-2006

## HTML

HTML: Hyper Text Markup Language, rappresenta il linguaggio dei browser. Il concetto di ipertesto è quello di un documento che contenga, oltre al testo, anche parti di diversa natura, come immagini, suoni, applicazioni, aree interattive, rimandi ad altri documenti (hyperlinks).

La pagina html e' strutturata in due parti principali:

- l'intestazione (head);
- il corpo del documento (body).

Per convenzione, l'intero documento va racchiuso tra i tag:

**<HTML> ..... </HTML>.**

## **SGML (Standard Generalized Markup Language)**

**SGML** fu creato da Tim Berners-Lee come linguaggio di pubblicazione ipertestuale (definisce regole per scrivere markup). Essendo già uno standard SGML fu utilizzato per la pubblicazione di documenti di grandi dimensioni, come manuali di manutenzione di veicoli.

Il perché non si sia utilizzato in partenza SGML per i documenti che circolano sul Web è semplice: SGML è un linguaggio complesso, fin troppo ricco di funzionalità che rendono i parser *pesanti* e di difficile implementazione.

## **XML (eXtensible Markup Language)**

**XML** sviluppato dal W3C, il World Wide Web Consortium, XML è un sottoinsieme di SGML (volutamente non comprende alcune funzionalità complesse di SGML difficilmente implementabili su Web).

**XML** studiato per il Web e per superare i limiti di HTML, viene adesso utilizzato in altri differenti modi.

## Cosa è XML ?

Una risposta semplice lo etichetta come un mezzo per rappresentare e descrivere dati

. . . i suoi modi di utilizzo e le sue applicazioni variano così tanto che c'è molta confusione sulla sua definizione

. . . una preferenza personale: etichettare XML di volta in volta - per il ruolo che svolge nei particolari contesti

## Ad esempio in relazione a HTML

- **XML** è considerato un **metalinguaggio** - contrariamente ad HTML che è un linguaggio predefinito - non ha tag predefiniti ma consente di definire nuovi linguaggi e/o metalinguaggi.

- **XML** è estensibile

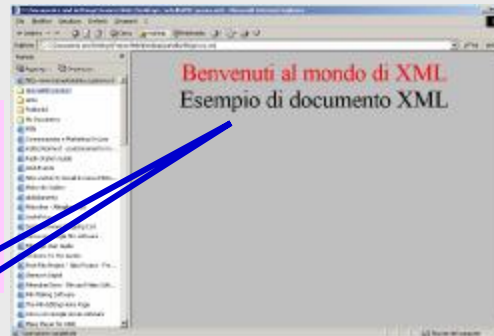
## Un esempio: possibilità di definire linguaggi di style sheet

```
<?xml version="1.0" standalone="yes"?>
<?xml-stylesheet type="text/css" href="greeting.css"?>

<DOCUMENT>
  <benvenuto> Benvenuti al mondo di XML</benvenuto>
  <messaggio> Esempio di documento XML</messaggio>
</DOCUMENT>
```

```
benvenuto {display: block; font-size: 36pt;
            color: #FF0000; text-align:
            center;}
messaggio {display: block; font-size: 36pt;
           color: #000000; text-align:
           center.}
```

Rendering

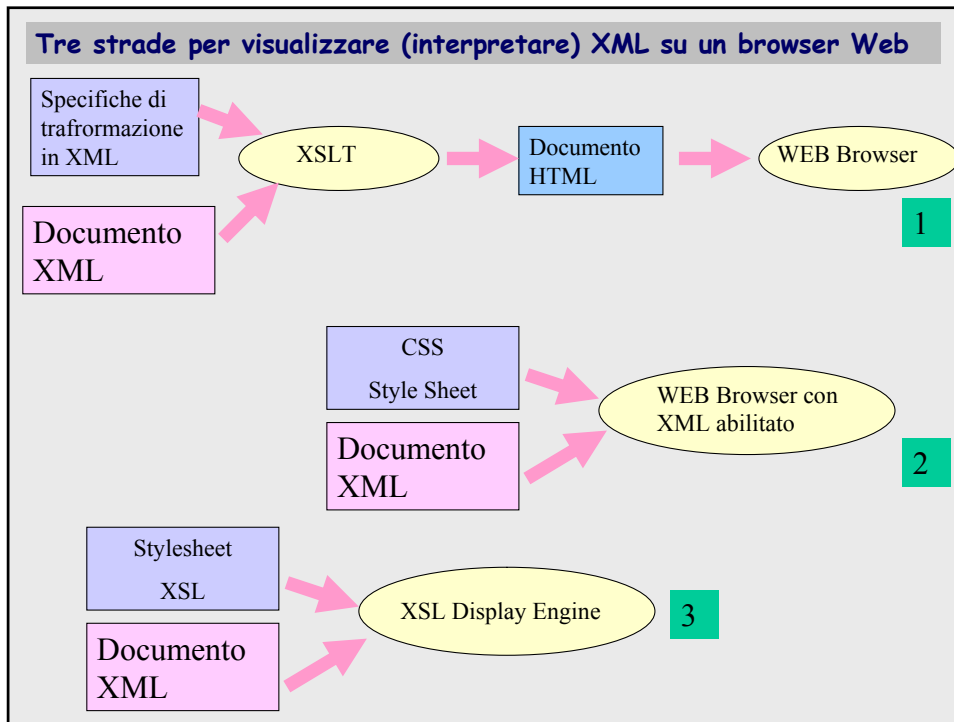


## XML rispetto ad HTML

- **Possibilità di definire linguaggi di style sheet:** con HTML i tag sono fissati dal linguaggio stesso, mentre con XML si possono creare in base al tipo di documento che si sta costruendo.

- **XML a differenza di HTML ha una struttura orientata ai database.**

- **XML richiede maggiore precisione nella scrittura del codice a differenza dell'interprete HTML** (ciò se da un lato richiede maggior impegno per la scrittura del codice, dall'altro mette al riparo da sorprese e effetti collaterali indesiderati).



12

<p><b>XML source</b></p> <pre>&lt;?xml version="1.0"?&gt; &lt;xml:stylesheet type="text/xsl" href="xsl/tutorial.xsl" /&gt; &lt;title&gt;XSL&lt;/title&gt; &lt;author&gt;John Smith&lt;/author&gt; &lt;/xml:stylesheet&gt;</pre>	<p><b>XSL stylesheet 1</b></p> <pre>&lt;xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform" &gt; &lt;xsl:template match="/"&gt; &lt;H1&gt;&lt;xsl:value-of select="//title"/&gt;&lt;/H1&gt; &lt;H2&gt;&lt;xsl:value-of select="//author"/&gt;&lt;/H2&gt; &lt;/xsl:template&gt; &lt;/xsl:stylesheet&gt;</pre>
<p><b>HTML output 1</b></p> <pre>&lt;HTML&gt; &lt;HEAD&gt; &lt;/HEAD&gt; &lt;BODY&gt; &lt;H1&gt;XSL&lt;/H1&gt; &lt;H2&gt;John Smith&lt;/H2&gt; &lt;/BODY&gt; &lt;/HTML&gt;</pre>	
<p><b>XSL</b></p> <p>John Smith</p>	<p>Rendering</p>

### Vantaggi di XML (I)

- consente di utilizzare documenti strutturati;
- offre un ottimo formato di trasmissione di dati;
- è un formato che probabilmente durerà a lungo poiché strutturato, estensibile, non ambiguo e completamente leggibile (non binario);
- la strutturazione e l'utilizzo di un linguaggio estensibile, basato su tag, consente una più semplice interazione con altri programmi, compresi i data base, e quindi un trattamento dei dati più semplice ed efficace;
- è **estensibile**, permette di aggiungere sempre nuovi marcatori;
- **portabilità** (indipendente dalla piattaforma e dal processore);

### Vantaggi di XML (II)

- permette un semplice utilizzo di metadati, come Dublin Core, RDF;
- ricerche più semplici e più efficaci, prendiamo ad esempio una interrogazione effettuata tramite un motore di ricerca: attraverso il controllo sui tag sarà più inerente a ciò che realmente stiamo cercando;
- offre un buon meccanismo di rappresentazione, una ottima capacità di rappresentare dati complessi (notazioni matematiche, interfacce grafiche);
- la creazione di linguaggi "ad hoc" che possono servire a specifiche comunità di utenti (Esempio di questi linguaggi sono SMIL (Synchronized Multimedia Integration Language, Mathematical Markup Language, etc)

## SMIL (Synchronized Multimedia Integration Language)

```
<?xml version="1.0"?>
<!DOCTYPE smil PUBLIC "-//W3C/DTD SMIL 1.0//EN"
"http://www.w3.org/TR/REC-smi/SMIL10.dtd">
<smil>
  <body>
    <seq id=mozart>
      <audio mozart1.wav />
      <video src="amadeus1.mov" />
      <text src"mozart1.htm" />

      <audio mozart2.wav />
      <video src="amadeus2.mov" />
      <text src"mozart2.htm" />
    </seq id=mozart>
  </body>
</smil>
```

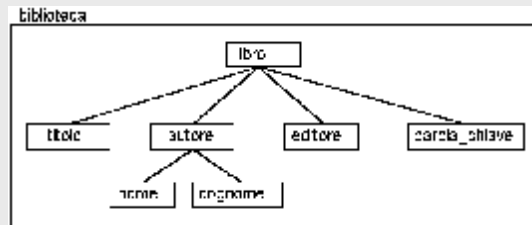
## XML

In un programma **XML** esistono tre parti sempre distinte

- il contenuto;
- le specifiche relative agli elementi, la struttura (DTD);
- le specifiche relative alla visualizzazione, lo stile (XSL)

## Contenuti in un documento XML

```
<?xml version="1.0"?>
<biblioteca>
  <libro codice="R414">
    <titolo>2001: Odissea nello spazio</titolo>
    <autore>
      <cognome>Clarke</cognome>
      <nome>Arthur Charles</nome>
    </autore>
    <editore>Rizzoli</editore>
    <parola_chiave>romanzo</parola_chiave>
    <parola_chiave>fantascienza</parola_chiave>
  </libro>
</biblioteca>
```



## Due nozioni chiave per i documenti XML:

- ben formatezza (documenti ben formati);
- validità (documenti validi)

**(I) Un documento XML è ben formato se rispetta le seguenti regole:**

- La dichiarazione XML deve essere posta all'inizio del documento

```
<?xml version = "1.0" standalone= "yes"?  
<DOCUMENTO>  
  <TAG1> xxxx </TAG1>  
  ...  
</DOCUMENTO>
```

- **Inclusione di uno o più elementi**

Un documento per essere ben formato dovrà includere uno più elementi. Il primo elemento che il documento dovrà includere è l'elemento radice.

- **Uso dei tag d'inizio e fine per elementi non vuoti**

Ad ogni marcatore di apertura (es: <tag1>) deve corrisponderne uno di chiusura (es: </tag1>).

- **Chiusura dei tag vuoti con />**

Gli elementi vuoti non hanno chiusura. Questi tipi di tag hanno un contenuto e non racchiudono perciò alcun dato. La sintassi ben formata per tale marcatore è: <tagx />.

**(II) Un documento XML è ben formato se rispetta le seguenti regole:**

- **L'elemento radice deve contenere tutti gli altri elementi**

```
<LIBRI>  
  <LIBRO>  
    <TITOLO> XML TUTTO& OLTRE </TITOLO>  
    <AUTORE>S. HOLZNER  
  </LIBRO>  
  
  <LIBRO>  
    <TITOLO> xxxx</TITOLO>  
    <AUTORE>xxxx </AUTORE>  
  </LIBRO>  
  ...  
</LIBRI>
```

### (III) Un documento XML è ben formato se rispetta le seguenti regole:

#### • Annidamento corretto

L'idea fondamentale: se un elemento contiene il tag di inizio di un tag non vuoto deve contenere anche il tag di chiusura dell'elemento.

Esempio di annidamento scorretto:

```
<LIBRI>
  <LIBRO>
    <TITOLO> xxxx
    <AUTORE>xxxx
  </TITOLO>
  </AUTORE>
</LIBRO>
....
</LIBRI>
```

#### • Uso di nomi univoci per gli attributi

Nessun nome di attributo può apparire più di una volta nello stesso tag di apertura e nello stesso tag di un elemento non vuoto.

**<PERSONA COGNOME ="Boffi" <Persona Cognome="Banfi"**  
**[errore]**

(XML è case sensitive: i seguenti attributi sono diversi

<PERSONA COGNOME ="Boffi" <persona cognome="Banfi")

### (IV) Un documento XML è ben formato se rispetta le seguenti regole:

#### • Valori degli attributi tra doppi apici

Tutti i valori degli attributi di un elemento devono essere racchiusi tra doppi apici; es: `<libro codice="R414">`

#### • Uso di entità dichiarate

Le **entità** sono strutture fisiche che compongono un documento XML. Sono quindi una sorta di *alias*, un mezzo, per associare a un nome un contenuto più o meno complesso, testuale e di altro tipo. L'utilizzo di entità è consentito solo previa la loro dichiarazione.

## Documenti XML validi

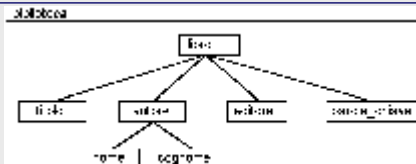
Un documento XML è valido se è ben formato e inoltre rispetta una struttura descritta in un particolare file associato al documento, chiamato DTD (Document Type Definition)

Il DTD contiene le regole di definizione dei tag, indica gli elementi e il loro ordine all'interno del documento XML. L'uso del DTD non è obbligatorio, ne è comunque consigliato l'utilizzo per verificare la validità del documento

Il DTD può essere interno o esterno al documento XML, il suo nome per convenzione corrisponde a quello dell'elemento radice (nel nostro esempio "biblioteca")

## Esempio di documento XML con DTD interno

```
<?XML version = "1.0" encoding = "UTF-8"?>
<!DOCTYPE biblioteca [
<!ELEMENT biblioteca (libro+)>
<!ELEMENT libro (titolo, autore+, editore, parola_chiave+)>
<!ATTLIST libro codice ID #REQUIRED>
<!ELEMENT titolo (#PCDATA)>
<!ELEMENT autore (cognome, nome)>
<!ELEMENT editore (#PCDATA)>
<!ELEMENT parola_chiave (#PCDATA)>
<!ELEMENT cognome (#PCDATA)>
<!ELEMENT nome (#PCDATA)>]>
<biblioteca> ..... </biblioteca>
```



## Strutture Logiche

**Una struttura logica è formata dal materiale da cui è costituito il documento e il modo con cui questo materiale è organizzato**

**Le strutture logiche contengono elementi.**

Un **elemento in XML** è l'insieme composto dal marcatore di apertura, dal marcatore di chiusura e da ciò che è contenuto fra questi due. Se il marcatore è vuoto, l'elemento è costituito solo dal marcatore.

Ogni documento XML deve contenere almeno un elemento

**Es.: <biblioteca>... *contenuto* ...</biblioteca>**

## Strutture Fisiche

Le strutture fisiche contengono entità. Che rappresentano una sorta di *alias*, un mezzo, per associare a un nome un contenuto più o meno complesso, testuale e di altro tipo.

**Esempi di strutture fisiche sono:**

Set di caratteri permessi

Costrutti di buona formazione e di validità dei documenti

Regole per la codifica dei caratteri

Contenuto testuale di un documento

## Entità

E' possibile suddividere un documento XML in una o più entità ognuna delle quali contiene dati.

Le entità sono costituite da:

Un nome che identifica l'entità;

Un valore (contenuto) che rappresenta o il dato dell'entità o un puntatore ai dati

## Entità Interne

Sono le entità che si dichiarano nel documento XML e sono spesso usate per esprimere caratteri speciali o abbreviazioni di parole o frasi lunghe.

Esempio:

```
<!ENTITY xml "eXtensible Markup Language">
```

Esistono entità interne predefinite in XML, che quando sono incontrate dal parser sono così sostituite:

**&lt; --> <**

**&quot; --> "**

**&gt; --> >**

**&apos; --> '**

## Entità Esterne

Sono quelle che referenziano elementi esterni al documento come, ad esempio, un altro file xml o un file binario.

Se la risorsa a cui si riferisce l'entità è un testo, questo rimpiazza l'entità, mentre se è binaria il parser si limita a passarne il contenuto all'applicazione, senza interpretarlo

Esempio:

```
<!ENTITY cap1 SYSTEM "/book/capitolo1.xml">
```

## Entità Parametriche

Differiscono da quelle interne per il fatto che queste ultime vengono semplicemente espanso e poi passate all'applicazione, mentre le parametriche vengono espanso e interpretate come parte della Document Type Definition (DTD).

Vengono dichiarate con il simbolo %.

Esempio:

```
<!ENTITY % indir_corto '<!ELEMENT indirizzo (via, città)>'  
<!ENTITY % indir_lungo '<!ELEMENT indirizzo (via, numero,  
città, cap, provincia)>'
```

Utilizzo

```
<!ELEMENT contatto (nome, indirizzo)>
```

```
%indir_corto;
```

## Sintassi che regola la scrittura del DTD

### Dichiarazione di un elemento non vuoto.

```
<!ELEMENT biblioteca (libro+)>  
<!ELEMENT libro (titolo, autore+, editore, parola_chiave+)>
```

Le righe sopra cominciano la descrizione di un DTD per un ipotetica biblioteca. La rappresentazione XML per l'esempio della biblioteca sarà formata da uno o più (simbolo +) libri, ognuno dei quali a sua volta è composto da:

- un titolo;
- uno o più (simbolo +) autori;
- un editore;
- una o più parole chiave.

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

La riga precedente sta ad indicare che l'elemento titolo potrà essere composto da qualsiasi testo o altro carattere che non sia un markup o ", & oppure ]] (PCDATA = Parsed Character Data).

## Sintassi che regola la scrittura del DTD

### Dichiarazione di attributi.

```
<!ATTLIST libro  
          codice ID #REQUIRED>
```

Il libro ha come attributo: codice, che è sempre necessario (#REQUIRED).

#### Alcuni tipi di attributi

**CDATA:** Stringhe di tipo *parsed*

**ID:** Un nome che identifica un elemento univocamente. Ogni elemento può avere un solo attributo di tipo **ID**

**ENTITY:** Nome di una entità dichiarata nel documento

#### Alcuni valori per gli attributi

**#REQUIRED:** Deve essere sempre specificato un valore per l'attributo

**#IMPLIED:** Non è obbligatorio specificare un valore per l'attributo e non viene assegnato un valore di default

**"valore":** Non è obbligatorio specificare un valore per l'attributo e se non lo si specifica viene assegnato un valore di default

## Tipi di marcatori per un file XML

### Marcatori di Istruzioni

**<?XML version = "1.0" encoding = "UTF-8"?>**

Fornisce il primo insieme di istruzioni al processore XML su come gestire il documento:

- XML Version = "1.0" indica il formato del documento;
- encoding = "UTF-8" indica lo schema di codifica caratteri a 8 bit Unicode da utilizzare durante l'analisi dei caratteri.

### Marcatori di Dichiarazione

**<!DOCTYPE biblioteca SYSTEM "biblioteca.dtd">**

Questa riga indica il DTD di riferimento per il documento XML in questione.

### Marcatori di Descrizione

**<biblioteca>..... </biblioteca>**

Sono i marcatori che descrivono il documento XML. Il primo marcatore, detto radice deve avere il nome assegnato al DTD nell'eventuale sezione di dichiarazione.

## Ancora un esempio: rappresentare una nota in XML

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

```
<!DOCTYPE note [  
<!ELEMENT note (from,to,heading,body)>  
  
<!ELEMENT from (#PCDATA)>  
<!ELEMENT to (#PCDATA)>  
<!ELEMENT heading (#PCDATA)>  
<!ELEMENT body (#PCDATA)>  
]>
```

```
<note>  
  <from> Francesco </from>  
  <to> Studenti </to>  
  <heading> Messaggio </heading>  
  <body> La prossima lezione è su XSLT </body>  
</note>
```

## Ancora un esempio: rappresentare un breakfast menu

```
<?xml version="1.0" encoding="ISO8859-1" ?>
<breakfast-menu>
  <food>
    <name> Belgian Waffles </name>
    <price>$5.95</price>
    <description> two of our famous Belgian Waffles with plenty of
      real maple syrup </description>
    <calories>650</calories>
  </food>
  .....
  <food>
    <name> Homestyle Breakfast </name>
    <price> $6.95 </price>
    <description> two eggs, bacon or sausage, toast, and our ever-
      popular hash browns </description>
    <calories>950</calories>
  </food>
</breakfast-menu>
```

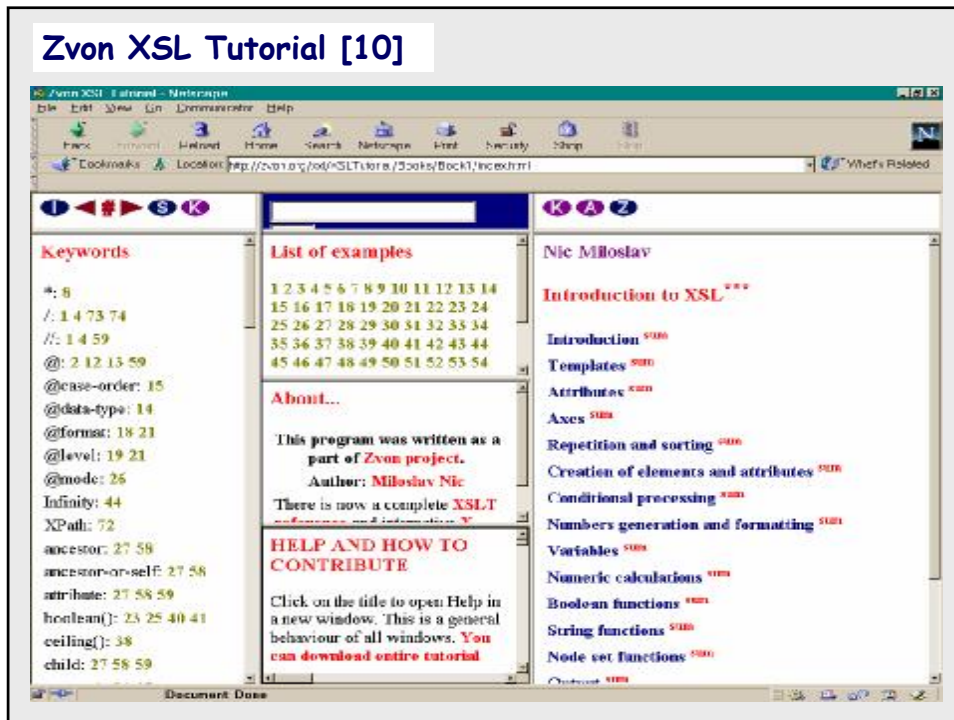
## Il linguaggio XSL

XSL consiste di componenti descritte in documenti della W3C:

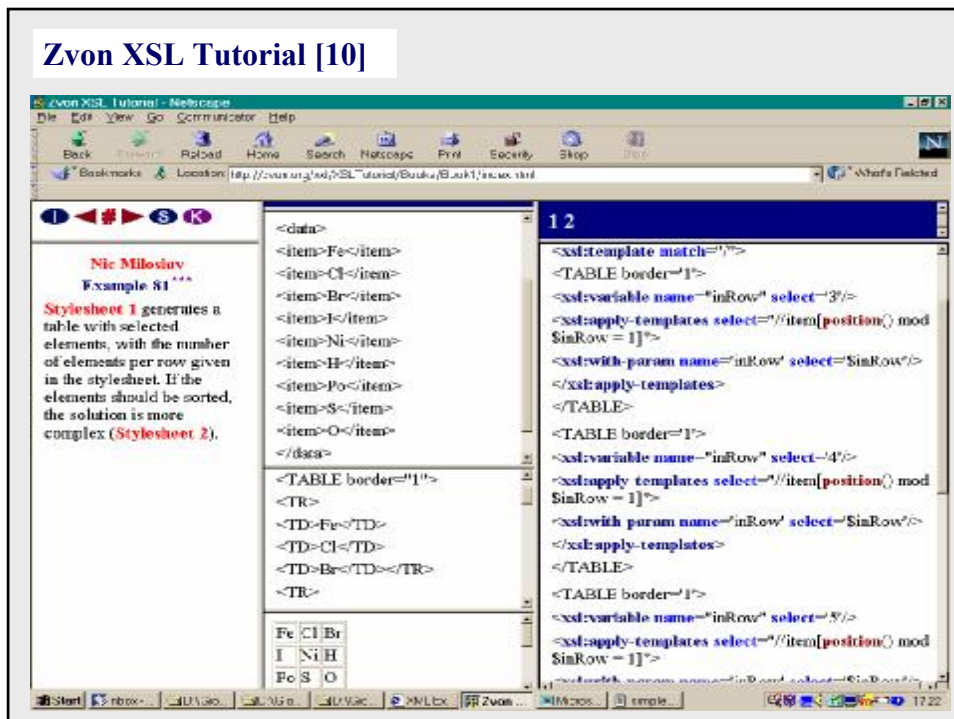
- XSLT: XSL Transformations – un linguaggio per descrivere come trasformare un documento XML (rappresentato come un albero) in un altro documento.
- XSL: Extensible Stylesheet Language – XSLT con in più una descrizione di un insieme di oggetti per la formattazione di un documento.



## Zvon XSL Tutorial [10]



## Zvon XSL Tutorial [10]



## Riferimenti

1. Extensible Markup Language (XML) - <http://www.w3.org/XML>
2. A Technical Introduction to XML - Norman Walsh. <http://www.xml.com>
3. XML Tutorials for Programmers - <http://www.software.ibm.com/xml/education/tutorialprog/abstract.html>
4. XML: Structuring Data for the Web" - Ken Sall <http://www.wdvl.com/Authoring/Languages/XML/Intro/index.html>
5. XML for Dummies - Ed Tittel, Norbert Mikula, Ramesh Chandak. Apogee. IDG Books Worldwide
6. Introduzione a XML 1.0 - Alessandro Ronchi-<http://www.cs.unibo.it/~ronchi/xmlintro-221098.html>
7. Introduzione a HTML - M. Moro - <http://linda.dei.unipd.it/seminar/list1.html>
8. Online XSL Tutorial XSL Concepts and Practical Use - P. Grosso, N. Walsh <http://www.arbortext.com/xsl/>
9. Extensible Stylesheet Language (XSL) - <http://www.w3.org/Style/XSL/>
10. XSL on line tutorial - <http://www.zvon.org/xxl/XSLTutorial/Books/Book1/index.html>

## Riferimenti

11. eXcelon Stylus: Visual Editor for XSL Stylesheets - [http://www.exceloncorp.com/products/excelon\\_stylus.html](http://www.exceloncorp.com/products/excelon_stylus.html)
12. XMLwriter - <http://XMLwriter.net/>
13. XML Spy - <http://www.xmlspy.com/>
14. EPC Edit - <http://www.tksgml.de/> -