Università degli Studi di Torino
Dip. di Informatica

Andres Baravalle (andres@di.unito.it)
Vitaveska Lanfranchi (vita@di.unito.it)

Goals

- Conoscere XML come linguaggio di markup:
  - Struttura
  - Sintassi
- Interpretare un documento XML
- Esplorare le possibilità applicative.
- Conoscere DTD e XML Schema
- XML DOM

XML: breve introduzione

- XML è un nuovo linguaggio creato a partire dallo standard SGML al fine di garantire maggior flessibilità e generalità di utilizzo per implementare diversi tipi di dati.
- E' stato sviluppato dallo XML Working Group (originariamente noto come SGML Editorial Review Board) costituitosi all'interno del W3C nel 1996.
- XML risponde alla necessità di mettere in comunicazione sistemi eterogenei.

XML: breve introduzione (2)

- Attualmente esiste una notevole quantità di formati per rappresentare informazioni testuali o strutturate, proprietari o meno, che sono difficilmente gestibili all'infuori dell'applicativo originario (ex Microsoft Word, Adobe PDF).

XML: obiettivi

- Compatibilità con applicazioni differenti.
- Creare un linguaggio utilizzabile con facilità in Internet ma più flessibile di HTML.
- Facilità di progettazione ed elaborazione dei documenti.
- Necessità di disporre di un linguaggio che permettesse di descrivere e strutturare i dati.

XML: obiettivi (2)

- Comprensibilità elevata dei documenti grazie all'utilizzo di elementi aventi nomi non arbitrari o convenzionali ma che rispecchino il loro contenuto (ad esempio si potrebbe definire un elemento author in cui vada inserito il nome dell'autore di un libro, nel contesto di un sito creato per la vendita o la consultazione di libri), rendendo anche più facile la catalogazione dei documenti per eventuali ricerche.

XML: obiettivi (3)

- Permettere la pubblicazione online di documenti indipendenti dal tipo di dispositivo che vi avrà accesso.
- Permettere alle industrie di sviluppare protocolli indipendenti dalle piattaforme per lo scambio dei dati.
- Permettere agli sviluppatori di visualizzare l'informazione nel modo desiderato, grazie al supporto dei fogli di stile.

XML: definizione della sintassi

- La sintassi di XML può essere definita dall'utente a seconda del tipo di documento da creare.
- E' possibile definire delle classi di documenti significative (chiamate DTD – Document Type Definition ossia definizione di tipo del documento) oppure degli schemi di documento (XMLSchema) e associarvi particolari proprietà mediante un foglio di stile esterno, che può essere realizzato in formato CSS o XSL (XSLT per trasformazione, XSL-FO per formattazione a stampa).

XML: definizione della sintassi (2)

- L'utente può scegliere se utilizzare delle DTD già create (ad esempio sono state create delle DTD per uniformare i documenti creati per particolari settori, come MathML - Mathematical Markup Language).

XML: struttura di un documento

- Un documento XML deve innanzitutto essere ben formato, ossia rispettare tutte le regole di sintassi previste dal linguaggio:
  - Presenza di un unico elemento radice che contiene tutti gli altri.
  - Chiusura di tutti gli elementi aperti (a meno che siano elementi vuoti, che vanno comunque chiusi con sintassi del tipo
)
  - Differenza tra maiuscolo e minuscolo.
  - Controllo dell'indentamento dei tag.

XML: struttura di un documento (2)

- Un documento XML è valido se possiede una DTD a cui corrisponde.
- Dichiarazione iniziale:
  - <?XML version="1.0" encoding="UTF.8"?>
- Encoding permette di definire il tipo di codifica utilizzato per presentare i dati. Sono ammesse la maggior parte delle codifiche esistenti.

XML: struttura di un documento (3)

- Un documento XML:
  - E' basato su una struttura gerarchica.
  - Inizia con una "radice", ossia un elemento padre che contiene e delimita tutti gli altri e ricorre una sola volta all'interno del documento.
  - E' composto da entità, elementi e commenti.
- Tutti gli elementi vanno annidati in modo preciso perché il documento sia accettabile.

Analisi e Visualizzazione del documento

- Un documento XML viene interpretato da una specifica applicazione, costituita da due parti fondamentali:
  - Parser
  - Processore

Analisi e Visualizzazione del documento (2)

- Il parser esegue il controllo sintattico del documento e si occupa della gestione degli errori.
- Il controllo avviene su due livelli:
  - Sulla validità del documento, se esiste una DTD.
  - Sulla forma del documento (se è ben formato oppure no).

Analisi e Visualizzazione del documento (3)

- Il processore si occupa di visualizzare il documento utilizzando un apposito foglio di stile.
- Per ottenere una visualizzazione della pagina web è necessario aggiungere valore semantico agli elementi dichiarati, ossia applicare al documento un foglio di stile (CSS o XSL) tramite una dichiarazione posta subito dopo l'iniziale dichiarazione del linguaggio:

```
<?xml:stylesheet href="xml.css" title="Stile“
type="text/css"?>
```

Analisi e Visualizzazione del documento (4)

XML: namespace

- Lo spazio dei nomi è il nome attribuito alla lista degli elementi contenuti nel documento.
- Modo per eliminare ambiguità nell'uso di elementi, marcatori e attributi.
- Ognuno degli elementi è preceduto da un prefisso, separato dal nome dal carattere ":". es fo:, utilizzato da XSL.

  curriculum.fo

XML: namespace (2)

- XML utilizza l'attributo predefinito "xmlns" (eXtensible Markup Language Name Space) per introdurre i prefissi utilizzati. Il valore di quest'attributo è un URI (Uniforme Resource Identifier), ad esempio http://www.w3.org/1999/XSL/Transform
- Ogni spazio dei nomi deve essere dichiarato prima di poter essere usato.

XML: namespace (3)

- La sintassi per la dichiarazione è:
  xmlns:[prefisso] = "[URI]"
- Esempio di dichiarazione per xsl:fo
  <fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">

XML: sintassi

- Forma degli elementi:
  <nome
  (attributo:valore)></nome>
- Elementi vuoti :
  <nome />

XML: sintassi (2)

- I tag possono iniziare con una lettera, un underscore (_), oppure ":". Non è possibile iniziare nessun tag con la combinazione di lettere "xml", né maiuscole né minuscole.
  curriculum_vita.xml

Microsoft e XML

- Spazio dei nomi XSLT:
  - Explorer richiede lo spazio dei nomi relativo alla versione precedente di XSL (http://www.w3.org/TR/WD-xsl) mentre le specifiche W3C richiedono http://www.w3.org/1999/XSL/Tranform.
- Tipo MIME XSLT: "test/xsl" invece che "text/xml".
- Template di default: non supportato.

XML e Internet

- In un documento web dinamico i componenti vengono elaborati e composti solo nel momento in cui arriva una richiesta esplicita. Questo tipo di documento è utilizzato nei casi in cui sia necessario generare dei contenuti in modo automatico o in risposta ad un'operazione interattiva effettuata dall'utente.
- XML è una ottima soluzione per la gestione dei contenuti dinamici.

XML e Internet (2)

- L'approccio ai contenuti dinamici più in voga attualmente è quello multimodale:
  - Una risorsa web è disponibile a più media differenti.
  - Ogni medium riceve la risorsa web secondo modalità differenti.
  - Non è necessario scrivere diverse versioni dello stesso testo per medium diversi.

XML e Internet (3)

- Le tecnologie disponibili possono essere divise in due gruppi principali:
  - Tecnologie dal lato client.
  - Tecnologie dal lato server.

XML e Internet (4)

- La scelta dell'una o dell'altra avverrà in base a:
  - Considerazioni sul carico di lavoro da affidare a client e server.
  - Considerazione sull'affidabilità della tecnologia.
  - In alcuni casi, inoltre, si sceglierà una tecnologia client side, con una sorta di "soluzione di riserva".
  - Le tecnologie client side (JavaScript, CSS o Java) sono soggette alle impostazioni dell'utente, e possono quindi avere limitazioni o essere addirittura disattivate.
- Comunemente, non si utilizzano tecnologie client side per compiti critici.

XML e Internet (5)

- E' possibile effettuare la trasformazione del codice XML sul lato server mediante:
  - Codice ASP.
  - PHP.
  - Java servlet.
  - CGI (ex Perl, Python etc.).
  - Appositi processori (ex. Sablotron, Xalax, Saxon).
- Per un utilizzo in Internet è sempre preferibile trasformare XML dal lato server.

XML e Internet (6)

- Prima di realizzare un applicativo basato su XML valutare:
  - Il codice disponibile (se applicabile).
  - Le prestazioni necessarie.
  - Il grado di conoscenza della piattaforma di sviluppo.
  - Le risorse hardware e software.
  - Le risorse economiche.

XML e Internet (7)

- In generale per soluzioni complesse di e-commerce si preferisce usare Java su piattaforma *NIX.
- Per la maggior parte dei casi non è necessaria una piattaforma, un linguaggio specifico o un particolare hardware.
- Si possono realizzare siti web basati su XML anche con Perl!

DTD

- Una DTD descrive lo "schema" del documento.
- In particolare descrive quali sono i costituenti “legali” di un documento XML.
- Definizione dell'elemento principale, dei figli, degli attributi (con il dominio di valori assumibili).
- E' possibile anche dichiarare entità, ossia riferimenti a blocchi di dati, interni o esterni al documento.
- Un elemento XML può avere una struttura predefinita tramite la DTD in modo da vincolarne il contenuto.

DTD: dichiarazione

- Una DTD può essere dichiarata internamente o esternamente al documento XML:
- Se viene dichhiarata internamente si deve utilizzare, all' interno del file XML, la seguente sintassi:

<!DOCTYPE root-element [element-declarations]>

DTD: dichiarazione esterna

- Una DTD può essere esterna al file XML.
- In questo caso la DTD viene salvata in un file avente estensione .dtd e richiamata all' interno del documento XML con la sintassi:

<!DOCTYPE root-element SYSTEM "filename">

DTD: elementi

- Gli elementi possono essere dichiarati con la sintassi:
  - <!ELEMENT element-name category>, nel caso di elemento che non ha figli
  - <!ELEMENT element-name (child-element-name)>, per un elemento con figli: inomi dei figli vengono dichiarati all' interno delle parentesi.
- Le virgole, a separare gli elementi (ad esempio i figli), indicano un ordine di successione obbligatorio.

DTD: elementi (2)

- Gli elementi vuoti vengono dichiarati con la sintassi:
  <!ELEMENT element-name EMPTY>
- Gli elementi di tipo carattere sono dichiarati con la sintassi:
  <!ELEMENT element-name (#PCDATA)>
- Gli elementi di tipo misto sono dichiarati con la sintassi:
  <!ELEMENT element-name ANY>

DTD: elementi (3)

- Gli elementi dichiarati possono essere:
  - Obbligatori, ma presenti una sola volta <!ELEMENT element-name (child-name)>
  - Obbligatori, e presenti almeno una volta <!ELEMENT element-name (child-name+)>
  - Facoltativi e presenti n volte: <!ELEMENT element-name (child-name*)>
  - Facoltativi e, se presenti, una volta sola: <!ELEMENT element-name (child-name?)>
- Per dichiarare due elementi presenti in alternativa si usa la sintassi:

<!ELEMENT note (to,from|header,(message|body))>

DTD: attributi

- Per ogni elemento possono essere indicati una serie di attributi (ATTLIST), definiti in base al tipo di carattere (PCDATA o CDATA) e alla obbligatorietà o meno della loro presenza.
- 4 casi base:
  - #DEFAULT, valore di default dell' attributo
  - #REQUIRED, la presenza dell'attributo è obbligatoria.
  - #FIXED, il valore dell'attributo è fissa
  - #IMPLIED, l'attributo è previsto ma non obbligatorio e non ha un valore di default.

DTD: attributi (2)

Esempi di valori degli attributi:

- PCDATA (valore di tipo carattere)
- ID (valore di tipo ID)
- IDREF (valore che indica riferimento a un altro ID)
- ENTITY (entità)

DTD: attributi – default

- Sintassi:
  <!ATTLIST element-name attribute-name attribute-type "default-value">
- DTD:
  <!ATTLIST payment type CDATA "check">
- XML:
  <payment type="check" />

DTD: attributi – implied

- Sintassi:
  <!ATTLIST element-name attribute-name attribute-type #IMPLIED>
- DTD:
  <!ATTLIST contact fax CDATA #IMPLIED>
- XML:
  <contact fax="555-667788" />

DTD: attributi – required

- Syntassi:
  <!ATTLIST element-name attribute_name attribute-type #REQUIRED>
- DTD:
  <!ATTLIST person number CDATA #REQUIRED>
- XML:
  <person number="5677" />

DTD: attributi – fixed

- Syntassi:
  <!ATTLIST element-name attribute-name attribute-type #FIXED "value">
- DTD:
  <!ATTLIST sender company CDATA #FIXED "Microsoft">
- XML :
  <sender company="Microsoft" />

DTD: lista attributi

- Sintassi:
  <!ATTLIST element-name attribute-name (en1|en2|..) default-value>
- DTD:
  <!ATTLIST payment type (check|cash) "cash">
- XML:
  <payment type="check" /> or <payment type="cash" />

DTD: entità

- Le entità servono per dichiarare dei testi o delle iscrizioni fisse che vengono inserite ripetutamente all' interno dei documenti.
- Possono essere interne o esterne.

DTD: entità interne

- Sintassi:
  <!ENTITY entity-name "entity-value">
- DTD:
  <!ENTITY writer "Donald Duck.">
  <!ENTITY copyright "Copyright XXXX.">
- XML:
  <author>&writer;&copyright;</ author>

DTD: entità esterne

- Sintassi:
  <!ENTITY entity-name SYSTEM "URI/URL">
- DTD:
  <!ENTITY writer SYSTEM
  "http://www.xxxx.com/entities/entities.xml">
  <!ENTITY copyright SYSTEM
  "http://www.xxxx.com/entities/entities.dtd">
- XML:
  <author>&writer;&copyright;</ author>

Esercizio

- Creare una DTD che descriva un curriculum con:
  - Dati personali
  - Esperienze formative
  - Esperienze Professionali
- Creare un documento XML contenente il proprio curriculum e validarlo rispetto alla DTD (utilizzare XMLSpy)

XML Schema

- XML Schema è un'alternativa alla DTD, basata su XML, per la descrizione della struttura di un documento XML.
- Il linguaggio utilizzato nell'XML Schema è descritto nell'XML Schema Definition (XSD).
- Il prefisso per qualsiasi elemento di un XML Schema è xsd: (oppure xs:)

XML Schema: caratteristiche

- Definisce gli elementi che appaiono in un documento
- Definisce gli attributi possibili
- Definisce le relazioni di parentela tra gli elementi
- Definisce l' ordine degli elementi figli
- Definisce il numero degli elementi figli
- Definisce se un elemento è vuoto o contiene testo
- Definisce i tipi di dati per gli elementi e gli attributi
- Definisce I valori di default e fissi per gli elementi e gli attributi

XML Schema vs. DTD

- XML Schema è facilmente estendibile
- XML Schema è più espressivo
- XML Schema è scritto in XML
- XML Schema supporta la descrizione di tipi di dati
- XML Schema supporta i namespace

XML Schema: dichiarazione

- Un XML Schema è contenuto in un file, avente estenzione .xsd, che viene richiamato all' interno del file XMl con la sintassi:

```
<note xmlns="http://www.w3schools.com"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation=
    "http://www.w3schools.com/schema/note.xsd">
```

XML Schema: elemento root

- L'elemento di base di un documento XML Schema è:
  `<xsd:schema>`
- Esempio:

```
<xsd:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
    targetNamespace="http://www.xxxx.com"
    xmlns="http://www.xxxx.com" >
```

- Può contenere attributi che fanno riferimento alla localizzazione del namespace :
- xmlns:xs=http://www.w3.org/2001/XMLSchema, fa riferimento al suo namespace
- targetNamespace=http://www. xxxx.com, fa riferimento al namespace utilizzato nel documento XML
- xmlns=http://www. xxxx.com, indica il namespace di default

XML Schema: elementi semplici

- Un elemento semplice può contenere solo testo.
- Il testo può essere però di differenti tipi: booleano, stringa, data etc.
- Tipi di dati:
  - xsd:string
  - xsd:decimal
  - xsd:integer
  - xsd:boolean
  - xsd:date
  - xsd:time

XML Schema: elementi semplici

- Codice XML

```
<lastname>Refsnes</lastname>
<age>34</age>
<dateborn>1968-03-27</dateborn>
```

- XML Schema

```
<xsd:element name="lastname" type="xsd:string"/>
<xsd:element name="age" type="xsd:integer"/>
<xsd:element name="dateborn" type="xsd:date"/>
```


XML Schema: attributi

- Qualsiasi elemento possieda un attributo è considerato un elemento complesso.
- La sintassi per definire gli attributi è:
  `<xsd:attribute name="xxx" type="yyy"/>`
- Esempio di codice XML:
  `<lastname lang="EN">Smith</lastname>`
- Corrispondente dichiarazione di attributo:
  `<xsd:attribute name="lang" type="xsd:string"/>`

XML Schema: attributi (2)

- XML Schema permette di specificare un valore fisso oppure un valore di default per gli attributi:

```
<xsd:element name="color" type="xsd:string" default="red"/>
<xsd:element name="color" type="xsd:string" fixed="red"/>
```

- Tutti gli attributi sono opzionali ma si può comunque forzarne la dichiarazione:

```
<xsd:attribute name="lang" type="xsd:string" use="optional"/>
```

- E' anche possibile dichiarare un attributo obbligatorio:

```
<xsd:attribute name="lang" type="xsd:string" use="required"/>
```


XML Schema: vincoli

- XML Schema permette di stabilire dei vincoli per i valori assumibili da un attributo o da un elemento.
- Il vincolo di stabilisce tramite l'elemento xsd:restriction
- All'interno di questo elemento vengono inseriti I valori assumibili dall'attributo o da un elemento.
- Tramite il tag <xsd:sequence> si indica la sequenza obbligatoria degli elementi. Questo elemento deve sempre essere presente anche se

XML Schema: vincoli (2)

- Si possono avere vincoli di:
  - Enumerazione (xsd:enumeration)
  - Range di valori (xsd:pattern)
  - Range numerico (maxExclusive, maxInclusive, minExclusive, minInclusive)
  - Rispetto degli spazi (xsd:whiteSpace)
  - Lunghezza (xsd:length, xsd:minLength e xsd:maxLength)

Vincolo di enumerazione

```
<xsd:element name="car">
  <xsd:simpleType>
    <xsd:restriction base="xsd:string">
      <xsd:enumeration value="Audi"/>
      <xsd:enumeration value="Golf"/>
      <xsd:enumeration value="BMW"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:element>
```


Range di valori

```
<xsd:element name="letter">
  <xsd:simpleType>
    <xsd:restriction base="xsd:string">
      <xsd:pattern value="[a-z]"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:element>
```


Range numerico

```
<xsd:element name="age">
  <xsd:simpleType>
    <xsd:restriction base="xsd:integer">
      <xsd:minInclusive value="0"/>
      <xsd:maxInclusive value="100"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:element>
```

Rispetto degli spazi

```
<xsd:element name="address">
    <xsd:simpleType>
        <xsd:restriction base="xsd:string">
            <xsd:whiteSpace value="preserve"/>
        </xsd:restriction>
    </xsd:simpleType>
</xsd:element>
```

- Possibili valori dell'attributp xsd:whiteSpace:
  - Preserve - mantiene tutti gli spazi così come scritti
  - Replace – rimpiazza tutti gli spazi con uno spazi singolo
  - Collapse – elimina tutti gli spazi bianchi

Vincoli di lunghezza

- xsd:length definisce la lunghezza di un attributo:

```
<xsd:element name="password">
    <xsd:simpleType>
        <xsd:restriction base="xsd:string">
            <xsd:length value="8"/>
        </xsd:restriction>
    </xsd:simpleType>
</xsd:element>
```

- xsd:minLength e xsd:maxLength defniscono la lunghezza minima e massima che l'attributo deve avere.

XML Schema: elementi complessi

- Un elemento complesso è un elemento che può contenere altri elementi e/o attributi.
- Può:
  - Essere vuoto
  - Contenere altri elementi
  - Contenere solo testo
  - Contenere sia altri elementi che testo
- E' sempre introdotto dal tag <xsd:element>, che conterrà questa volta un tag <xsd:complexType>

XML Schema: elementi complessi (2)

- E' possibile definire un elemento complesso creando direttamente un elemento:

```
<xsd:element name="employee">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element name="firstname" type="xsd:string"/>
            <xsd:element name="lastname" type ="xsd: string"/>
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>
```


XML Schema: elementi complessi (3)

- Tramite il tag <xsd:sequence> si indica la sequenza obbligatoria degli elementi.
- Questo elemento deve sempre essere presente anche se l'elemento complesso è composto da un solo elemento figlio.
- <xsd:element> può avere un attributo maxOccurs che indica quante volte l'elemento può ricorrere all'interno del documento.
- maxOccurs="unbounded" indica un numero infinito di ricorrenze.

XML Schema: elementi complessi (4)

- E' anche possibile crearer un elemento complesso a cui possano fare riferimento più elementi, tramite un attributo type.

```
<xsd:element name="employee" type="personinfo"/>
<xsd:element name="student" type="personinfo"/>
<xsd:element name="member" type="personinfo"/>
<xsd:complexType name="personinfo">
    <xsd:sequence>
        <xsd:element name="firstname" type="xsd:string"/>
        <xsd:element name="lastname" type="xsd:string"/>
    </xsd:sequence>
</xsd:complexType>
```

XML Schema: estensioni di elementi complessi

- E' anche possibile estendere degli elementi complessi, ereditando le loro proprietà e aggiungendone altre, tramite il tag <xsd:extension> e l' attributo base, che indica l'elemento che si vuole estendere:

```
<xsd:element name="employee" type="fullpersoninfo"/>
<xsd:complexType name="personinfo">
    <xsd:sequence>
        <xsd:element name="firstname" type="xsd:string"/>
        <xsd:element name="lastname" type="xsd:string"/>
    </xsd:sequence>
</xsd:complexType>
```


XML Schema: estensioni di elementi complessi (2)

```
<xsd:complexType name="fullpersoninfo">
    <xsd:complexContent>
        <xsd:extension base="personinfo">
            <xsd:sequence>
                <xsd:element name="address" type="xsd:string"/>
                <xsd:element name="city" type="xsd:string"/>
                <xsd:element name="country" type="xsd:string"/>
            </xsd:sequence>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>
```


XML Schema: elementi complessi vuoti

- Per definire un elemento vuoto, ossia senza contenuto, è necessario definire un elemento di tipo complesso che possa contenere solo elementi al suo interno (e non testo!) ma non dichiarare poi alcun elemento contenuto.

```
<xsd:element name="product">
    <xsd:complexType>
        <xsd:complexContent>
            <xsd:restriction base="xsd:integer">
                <xsd:attribute name="prodid" type="xsd:positiveInteger"/>
            </xsd:restriction>
        </xsd:complexContent>
    </xsd:complexType>
</xsd:element>
```


XML Schema: elementi complessi solo testo

- In XML Schema è possibile definire degli elementi complessi che contengano solo testo.

```
<xsd:element name="shoesize">
    <xsd:complexType>
        <xsd:simpleContent>
            <xsd:extension base="xsd:integer">
                <xsd:attribute name="country" type="xsd:string" />
            </xsd:extension>
        </xsd:simpleContent>
    </xsd:complexType>
</xsd:element>
```


XML Schema: elementi complessi misti

- Tramite l'attributo mixed="true" è possibile indicare che il valore di un elemento è sia testo che altri elementi figli.

```
<xsd:element name="letter">
    <xsd:complexType mixed="true">
        <xsd:sequence>
            <xsd:element name="name" type="xsd:string"/>
            <xsd:element name="orderid" type="xsd:positiveInteger"/>
            <xsd:element name="shipdate" type="xsd:date"/>
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>
```


Esercizio

- Creare un XML Schema che descriva la struttura di un documento "reminder", contenente:
  - To
  - From
  - Heading
  - Body
- Cerare un documento XML basato sullo schema, che sia valido.

Esercizio (nuovo)

- Creare un XML schema per il documento di tipo curriculum (di cui si possiede una DTD).

XML DOM

- **DOM XML (Document Object Model) è un'interfaccia standard per la programmazione dei documenti XML.**
- **Definisce il modo in cui si può avere accesso ai documenti XML e manipolarli.**
- E' possibile creare documenti XML, navigare al loro struttura ad albero, aggiungere, eliminare, modificare gli elementi.

XML DOM (2)

- Per accedere al modello a oggetti di un documento è necessario utilizzare un XML parser.
- E' possibile utilizzare svariati linguaggi di programmazione/scripting
- Si basa sull'esistenza di:
  - Proprietà, leggibili e modificabili
  - Metodi, invocabili

DOM: nodi – proprietà

- E' possibile accedere ai nodi di un documento XML per leggerne il nome, il valore, modificarne i parametri etc.

attributes	Restituisce tutti gli attributi del nodo
ChildNodes	Restituisce la lista dei figli del nodo
firstChild	Restituisce il primo figlio
lastChild	Restituisce l'ultimo figlio
nextSibling	Restituisce il fratello

DOM: nodi – proprietà

nodeName	Restituisce il nome del nodo
nodeType	Restituisce il tipo del nodo
nodeValue	Restituisce il valore del nodo
ownerDocument	Restituisce il nodo root
parentNode	Restituisce il nodo padre
previousSibling	Restituisce il precedente nodo con lo stesso padre

DOM:nodi – metodi

appendChild(newChild)	Appende il nodo
cloneNode(boolean)	Restituisce una copia del nodo
hasChildNodes()	Restituisce true se il nodo ha figli
insertBefore(newNode,refNode)	Inserisce un nuovo nodo prima di un determinato nodo esistente
replaceChild(newNode,oldNode)	Sostituisce un nodo esistente con uno nuovo
removeChild(nodeName)	Rimuove un nodo

DOM: lista nodi – proprietà e metodi

Proprietà

length	Restituisce il numero di nodi di una lista

Metodi

item	Restituisce uno specifico nodo all'interno di una lista.

DOM: elementi – proprietà

documentElement	Restituisce l'elemento root
doctype	Restituisce la DTD o lo Schema del documento

DOM: elementi – metodi

createAttribute(attributeName)	Crea un attributo
createCDATASection(text)	Crea una sezione CDATA
createComment(text)	Crea un nodo commento
createElement(tagName)	Crea un elemento
createProcessingInstruction(target,text)	Crea una processing-instruction
getElementsByTagName(tagName)	Ritorna una lista di nodi a partire dal nodo selezionato
createTextNode(text)	Crea un nodo testuale

DOM: attributi - proprietà

name	Specifica o restituisce il nome dell'attributo
value	Specifica o restituisce il valore dell'attributo
specified	Restituisce true se il valore di un attributo è setttao nel documento, false se è settato nella DTD o Schema.

Dom: testo - metodi

splitText(number)	Restituisce il testo a partire dal carattere inserito.

Modifica di un documento XML

- Per modificare un documento XML è innanzitutto necessario creare un'instanza del parser e caricare il documento:

```
<script type="text/javascript">
//istanza parser
var xmlDoc = new ActiveXObject("Microsoft.XMLDOM")
//istruzione di attesa
xmlDoc.async="false"
// caricamento documento
XmlDoc.load("note. xml")
</script>
```

Creazione di un documento XML

- Per creare un documento XML utilizzando Dom è necessario innanzitutto creare un nuovo oggetto di tipo documento XML
- Es. JavaScript

```
var xmlDoc = new ActiveXObject("Microsoft.XMLDOM")
```

- Es. VBScript in ASP:

```
set xmlDoc = Server.CreateObject("Microsoft.XMLDOM")
```


Esercizio 1 (nome nodi)

- Dato un codice XML (note.xml) che descrive un reminder:

```
<?xml version="1.0" ?>
<note time="12:03:46">
 <to>Vitaveska</to>
 <from>Lanfranchi</from>
 <heading>Reminder</heading>
 <body>Lezione ore 9</body>
 </note>
```

- Estrarre il nome di tutti i nodi

Esercizio 1 – svolgimento

```
<script type="text/vbscript">

set xmlDoc=CreateObject("Microsoft.XMLDOM")
xmlDoc.async="false"
xmlDoc.load("note.xml")

for each x in xmlDoc.documentElement.childNodes
   document.write(x.nodeName & "<br />")
next
</script>
```


Esercizio 1 – cont.

- Trovare il contenuto di tutti I nodi
- Stampare il contenuto in una tabella (X)HTML
- Trovare e stampare il nome dell'elemento radice
- Stampare solo il testo di tutto il documento
- Stampare il codice XML originale e stamparne poi una versione con un nodo aggiunto

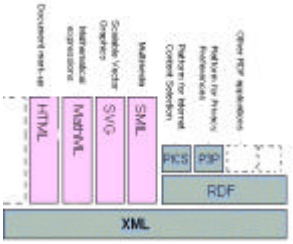
Esercizio 2

- Creare un documento XML con JavaScript avente come radice il tag <curriculum> e due elementi figli <Nome>, <Cognome>.
- Stampare il contenuto del documento XML formattandolo in (X)HTML.

XML e dialetti

- XML può essere considerato come la base di nuovi linguaggi, tra cui MathML, SMIL, RDF che hanno una sintassi che si fonda sulle regole e i costrutti base stabiliti da XML.

XML e dialetti (2)



XML Docbook

- Docbook è una DTD nata nel 1991 per la creazione di documenti di tipo giornalistico, per libri e per articoli, per documentazione tecnica.
  - http://www.docbook.org/oasis/intro.html

Docbook: esempio

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE Article SYSTEM
   "C:\Utility\dev\xmetal\Rules\journalist.dtd">
<Article>
   <Title>Dispense XML</Title>
   <PubDate>07-06-2001</PubDate>
   <Author>
        <FirstName>Vitaveska</FirstName>
        <Surname>Lanfranchi</Surname>
```


Docbook: esempio (2)

```
<Address>
                <City>Via Cavalli 30 Bergamo</City>
                <Country>Italy</Country>
                <Phone>+39035401643</Phone>
                <Fax>+39035401242</Fax>
                <Email>vita@di.unito.it</Email>
        </Address>
   </Author>
```


Docbook: esempio (3)

```
<Author>
        <FirstName>Andres</FirstName>
        <Surname>Baravalle</Surname>
        <Address>
                <City>Viale C.Battisti 22 Biella</City>
                <Country>Italy</Country>
                <Phone>+391520511</Phone>
                <Fax>+39035401242</Fax>
                <Email>andres@di.unito.it</Email>
        </Address>
</Author>
```


Docbook: esempio (4)

```
<Sect1>
        <Title>Breve introduzione a XML</Title>
        <LiteralLayout/>
        <Para>XML è un nuovo linguaggio creato a partire dallo
   standard SGML al fine di garantire maggior flessibilitá  e
generalitá  di utilizzo per implementare diversi tipi di dati.
E' stato sviluppato dallo XML Working Group (originariamente noto
   come SGML Editorial Review Board)
costituitosi all'interno del W3C nel 1996.
Obiettivi:
        </Para>
```

Docbook: esempio (5)

```
        <ItemizedList>
                <ListItem>Creare un linguaggio utilizzabile con
    facilitá in Internet ma piú flessibile di HTML.
                </ListItem>
                <ListItem>Compatibilitá con applicazioni differenti.
                        </ListItem>
                <ListItem>Facilitá di progettazione ed elaborazione dei
    documenti XML.
                </ListItem>
                <ListItem>Necessitá di disporre di un linguaggio che
    permettesse di descrivere e strutturare i dati.
                </ListItem>
        </ItemizedList>
    </Sect1>
</Article>
```


SVG

- SVG è un dialetto di XML creato per descrivere grafica bidimensionale.
- Permette di rappresentare grafica vettoriale, immagini, testo.

SVG: esempio

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 20001102//EN"
 "http://www.w3.org/TR/2000/CR-SVG-20001102/DTD/svg-
    20001102.dtd">
<svg width="100%" height="100%">
    <g id="source" style="fill:white; stroke:black; stroke-width:0.05cm">
        <circle r="10" cx="20" cy="40" />
    </g>
```


SVG: esempio (2)

```
<g id="destination" style="fill:white; stroke:black; stroke-
width:0.05cm">
        <circle r="10" cx="100" cy="40"/>
</g>
<g id="token" style="fill:white; stroke:black; stroke-width:0.05cm">
        <circle r="1" cx="20" cy="40"/>
</g>
<g id="transition" style="fill:white; stroke:black; stroke-
width:0.05cm">
        <rect width="0.25cm" height="1cm" x="55" y="20"/>
</g>
```


SVG: esempio (3)

```
    <g id="arc" style="fill:none; stroke:black; stroke-width:1">
        <line x1="30" y1="40" x2="55" y2="40"/>
        <line x1="65" y1="40" x2="90" y2="40"/>
    </g>
    <g id="tri" style="fill:black; stroke:black; stroke-width:0.05cm" >
        <polygon points="50,50 55,40 50,30" />
        <polygon points="85,50 90,40 85,30" />
    </g>
</svg>
```


RSS

- RSS (RDF Site Summary) è un formato per la descrizione di dati e metadati e per la loro diffusione.
- Permette di creare dei canali (es. Netscape Channels) composti da titolo, link, informazione.
- Si usa generalmente per la condivisione di notizie.

RSS: esempio

```
<?xml version="1.0" ?>
<!DOCTYPE rss (View Source for full doctype...)> - <rss
    version="0.91">-
    <channel>
        <title>Meerkat: An Open Wire Service
        </title>
        <link>http://meerkat.oreillynet.com
        </link>
        <description>Meerkat is a Web-based syndicated content
    reader providing a simple interface to RSS stories.
        </description>
        <language>en-us
        </language>
```


RSS: esempio (2)

```
        <image>
            <title>Meerkat Powered!
            </title>
            <url>http://meerkat.oreillynet.com/icons/...jpg
            </url>
            <link>http://meerkat.oreillynet.com
            </link>
            <width>88
            </width>
            <height>31
            </height>
            <description>Visit Meerkat
            </description>
        </image>
```


RSS: esempio (3)

```
        <item>
            <title>Linux browser wars
            </title>
            <link>http://c.moreover.com/click/here.pl?r25029473
            </link>
        </item>
        <item>
            <title>ELECTRONIC BUSINESS XML: Revolutionizing
    the Engines of Business
            </title>
            <link>http://xml.com/pub/r/1227
            </link>
        </item>
```


RSS: esempio (4)

```
        <item>
            <title>Sun Releases New Version of J2EE
            </title>
            <link>http://www.internetnews.com/dev-
    news/article/0,,10_890091,00.html
            </link>
        </item>
    </channel>
</rss>
```