ITTIG - CNR

Standard Documentali XML

Enrico Francesconi

Istituto di Teoria e Tecniche dell'Informazione Giuridica

7 Febbraio 2006

Introduzione

- Testo
  - Una delle principale forma di comunicazione della conoscenza;
- Documento
  - Definito in maniera non rigorosa, denota una singola unità di informazione;
  - Può essere qualunque unità fisica:
    - un file;
    - una email;
    - una pagina Web;

Introduzione

- Un documento è caratterizzato da:
  - Una struttura fisica (definita dall'applicazione o dal creatore del documento;
  - Una struttura logica (semantica) (definita dall'autore dei contenuti);
  - Eventualmente uno stile di presentazione (come deve essere visualizzato o stampato);
  - Informazioni su se stesso (metadati).

Introduzione

- Sintassi di descrizione dei documenti:
  - Implicita o espressa in un linguaggio (es: LaTeX)
  - Alcune modalità di descrizione sono orientate ad un particolare utilizzo (Postscript);
- Se i documenti sono descritti con un linguaggio potente e talvolta proprietario sono più semplici da analizzare, ma è più difficile convertirli in altri formati (es: Word);
- I linguaggi aperti sono migliori perché consentono interoperabilità fra sistemi informativi;

Introduzione

- La semantica dei testi scritti in linguaggio naturale è di difficile interpretazione da parte di un sistema automatico;

- Tendenza: linguaggi che forniscono informazioni su struttura, formato e semantica, leggibili sia dall'uomo che dalla macchina (es: SGML, HTML, XML);

Introduzione

- Le più recenti applicazioni tendono a definire documenti in cui i contenuti siano indipendenti dallo stile;
- In altri casi lo stile può essere contenuto nel documento (LaTeX, RTF)

Introduzione

- Metadati;
- Proprietà del testo;
- Linguaggi di marcatura;

Metadati

- “Dati sui dati”
  - es.: in un DBMS, lo schema specifica nomi delle relazioni, le entità coinvolte, il dominio di ciascuna entità;
- Metadati Descrittivi (Amministrativi):
  - sono per lo più esterni al contenuto e riguardano le modalità della sua creazione (es: Autore, fonte della pubblicazione, lunghezza della pubblicazione);
  - Es: Dublin Core Metadata Element Set (la maggior parte dei 15 metadati definiti sono descrittivi);
- Metadati Semantici:
  - Descrivono il contenuto del documento;
  - Es: *dc:subject* di Dublin Core.

Formati di Metadati

- MARC è un formato (schema) di metadati utilizzato per record bibliografici;
  - E' composto da metadati descrittivi (titolo, autore) e semantici (classificazione).
- Metadata per documenti Web:
  - di catalogazione, sul contenuto, diritti di proprietà, firma digitale;
  - Nuovo standard: Resource Description Framework (RDF);
    - Facilita la gestione automatica delle metainformazioni;
    - Nodi (qualunque risorsa Web) con coppie attributo valore;
  - Descrizione di oggetti non testuali (es: immagini):
    - Keyword utilizzate per indicizzare e ricercare immagini.

Testo

- Testo codificato in bit
  - EBCDIC, ASCII
    - Inizialmente 7 bit. Successivamente, 8 bit
  - Unicode
    - 16 bit, per la gestione delle lingue orientali

Testo

- Formati
  - Non esiste un unico formato (RTF, DOC, ASCII, PDF, PS);
  - I sistemi di IR dovrebbero recuperare informazioni da differenti formati;
  - In passato: i sistemi di IR convertivano i documenti;
  - Attualmente: si usano filtri;

Linguaggi di Markup

- Sintassi extra-testuale che può essere usata per descrivere formattazioni, informazioni strutturali, semantica del testo, attributi;
- SGML, HTML, XML;

SGML

- SGML = Standard Generalized Markup Language;
- E' un metalinguaggio per la marcatura di un testo;
- E' possibile definire una grammatica di marcatura che definisce la struttura di un tipologia (classe) di documenti;
- Tale grammatica è definita in un file DTD (document type definition);
- Una DTD definisce elementi e attributi, assegnando loro caratteristiche di obbligatorietà, opzionalità e cardinalità;
- Inoltre nelle DTD-SGML sono indicati i tag che devono essere chiusi e quelli per i quali tale caratteristica è opzionale;

HTML

- Hypertext Mark-up Language è una istanza di SGML;
- Esiste una DTD per HTML, ma molti documenti HTML non seguono strettamente la DTD;
- I tag HTML seguono le convenzioni SGML e includono anche direttive di formattazione;
- La formattazione può essere introdotta mediante CSS;
- Tre DTD per HTML:
  - Strict = evita le direttive di formattazione;
  - Transitional = usa le caratteristiche di presentazione compatibili con i vecchi browser (che non supportano CSS);
  - Frameset = consente il partizionamento del browser in frame;

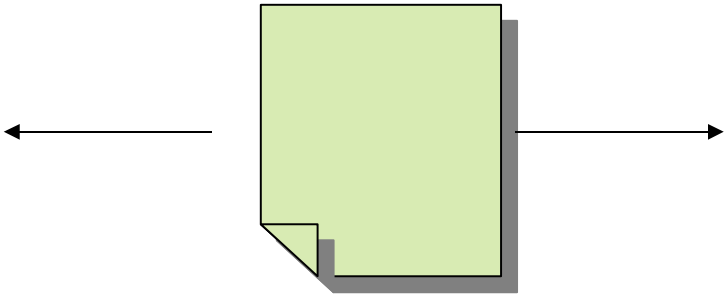
XML

- E' una versione semplificata di SGML;
- E' un meta-linguaggio, capace di definire linguaggi di marcatura per specifiche tipologie di documenti;
- Rende più rigorosa la sintassi di descrizione per una tipologia di documenti;
- Due modalità di validazione:
  - “Well formed” = i tag sono inseriti secondo la sintassi corretta (es: ogni tag aperto ne ha un corrispondente chiuso);
  - “Valid” = la struttura dei tag segue quella suggerita dalla DTD associata;

XML

- Possono essere introdotti marcatori con lo stesso nome purché se ne specifichi il *namespace* (dominio di validità del nome);
- La tendenza è quella di rappresentare in XML i contenuti (struttura e semantica);
- XSL rappresenta il linguaggio per le direttive di formattazione;

Perché XML



Doc Elettronico

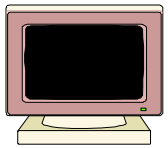
Wysiwyg <?xml?> DB

XML è uno standard comprensibile agli uomini e alle macchine

Perché XML

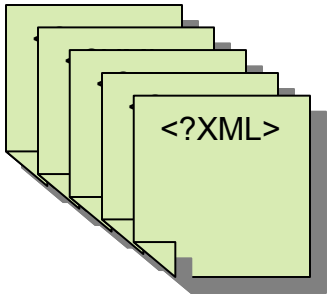
Rendering diverso per device diversi

Palmtop (WML)

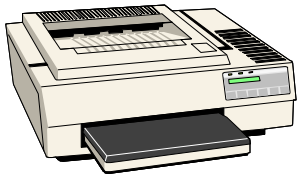


Web Browser (Xhtml)

Voice Browser (Voice XML)



CD-DVD (Xhtml|PDF)

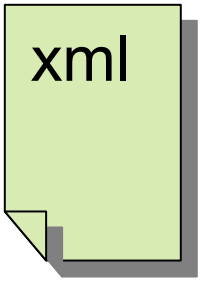


Stampa (PDF)

Scrivi una volta usa molte

Perché XML

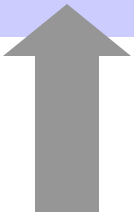
Ricerche e statistiche



- Dammi il numero delle leggi in cui Rossi è stato primo firmatario nell'ultima legislatura
- Dammi il testo della legge 675 vigente al 1 gennaio 2001

Porzione di un documento XML

<titoloDoc>Disposizioni concernenti l'elezione diretta del Presidente della Giunta regionale e l'autonomia statutaria delle Regioni.</titoloDoc>



elemento titoloDOC

Porzione di un documento XML

(tag di apertura)

<titoloDoc>Disposizioni concernenti l'elezione diretta del Presidente della Giunta regionale e l'autonomia statutaria delle Regioni.**</titoloDoc>** (tag di chiusura)

Markup + Contenuto = Documento XML

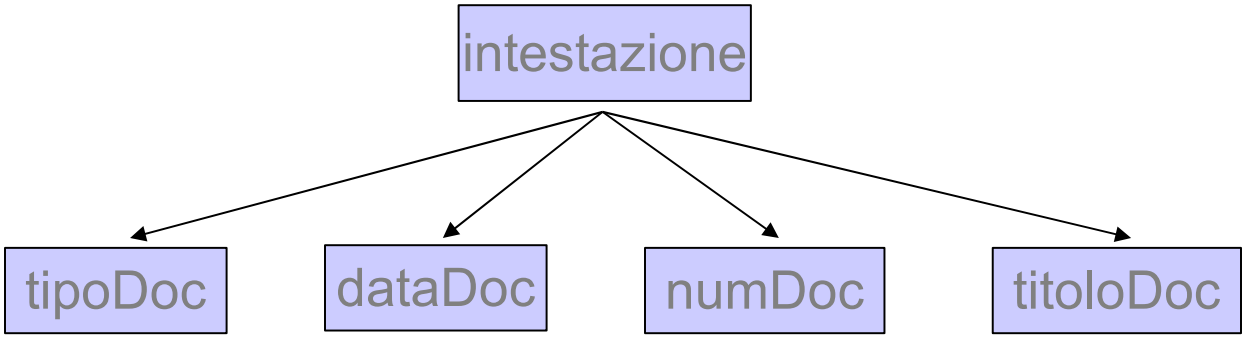
Gerarchie di elementi

```
<intestazione>
    <tipoDoc>LEGGE COSTITUZIONALE</tipoDoc>
    <dataDoc norm="19991122">22 novembre 1999</dataDoc>
    <numDoc>1</numDoc>
    <titoloDoc>Disposizioni concernenti l'elezione diretta del Presidente
    della Giunta regionale e l'autonomia statutaria delle Regioni.
    </titoloDoc>
</intestazione>
```

5 elementi

(intestazione,tipoDoc,dataDoc, numDoc, titoloDoc)

Gerarchie di elementi



Document Type Definition (DTD)

- Elemento facoltativo di un documento XML
- Tramite un DTD riesco a **definire** vincoli di correttezza
  - LESSICALE quali elementi sono validi
    - Ex intestazione, tipoDoc, dataDoc, numDoc, titoloDoc
  - SINTATTICI regole di composizione tra gli elementi (SINTASSI)
    - Ex intestazione contiene nell'ordine tipoDoc, dataDoc, numDoc e titoloDoc

DTD

- Una DTD definisce un linguaggio con cui rappresentare documenti.
- Tramite opportuni programmi (parser) si può verificare se un documento è conforme a questo linguaggio;
- Le DTD esprimono vincoli solo strutturali non sul tipo dei dati immessi (es: una data);
- XMLSchema nuovo standard;

DTD e Grammatiche

- XML è un metalinguaggio: sintassi che consente di definire dei linguaggi specifici;
- Una DTD consente di definire:
  - Il lessico (vocabolario T dei simboli terminali ammessi);
  - Le categorie funzionali del linguaggio (simboli N non terminali);
  - Le regole sintattiche e semantiche (vincoli grammaticali di opzionalità, precedenza, dipendenza, in termini di produzioni P);
  - Lo scopo S della grammatica (definizione di una classe di documenti);

DTD

- Le DTD consentono di specificare:
  - l'insieme degli elementi e degli attributi che si possono usare nel documento XML;
  - le relazioni gerarchiche fra gli elementi;
  - l'ordine in cui gli elementi appariranno nel documento XML;
  - gli elementi e gli attributi opzionali;
- Quando un documento XML è ben formato e rispetta le regole del DTD a cui si riferisce si dice che è un **documento XML valido**.

Elementi Base delle DTD

- Dal punto di vista di una DTD, tutti i documenti XML sono costituiti da:
  - Elements = costituenti base di un documento XML rappresentati mediante tag;
  - Attributes = informazione aggiuntiva per un elemento;
  - Entities = alias per la definizione di parti di un documento XML o sottostrutture (entity predefinite sono &lt = <, &gt = >, ecc);
  - PCDATA = testo analizzato da un parser XML (elementi tag interni ad elementi e entità vengono espansi);
  - CDATA = testo non analizzato da un parser XML

Elementi

`<!ELEMENT element-name category>`

- Categorie di Element
  - Vuoti (`<!ELEMENT element-name EMPTY>`)
  - Contenenti “parsed characters”
    `(<!ELEMENT element-name (#PCDATA)>)`
  - Contenenti qualunque dato
    `<!ELEMENT element-name ANY>`
- Con figli

```
<!ELEMENT element-name (child-element-name)>
<!ELEMENT element-name
     (child-element-name,child-element-name,.....)>
```

`<!ELEMENT element-name (child-name+)>` (1 o più)

`<!ELEMENT element-name (child-name*)>` (0 o più)

`<!ELEMENT element-name (child-name?)>` (0 o 1)

`<!ELEMENT element-name (child-name1|child-name2)>` (alternativi)

Sintassi Elementi DTD

- I caratteri *, +, ? indicano il numero delle occorrenze di un elemento;
- Il carattere “,” indica che i due elementi, precedente e successivo, devono comparire esattamente in quell'ordine;
- Il carattere “|” significa che solo uno degli elementi a destra o a sinistra deve apparire nel documento;
- La combinazione (elem1 | elem2)* indica che i due elementi possono apparire in qualsiasi ordine;

Attributi

```
<!ATTLIST element-name attribute-name
 attribute-type default-value>
```

- Esempio:

  ```
  <!ATTLIST payment type CDATA "check">
  ```

  XML valido:

  ```
  <payment type="check"/>
  ```

- Esempio:

  ```
  <!ATTLIST payment type (check|cash) "cash">
  ```

  XML valido:

  ```
  <payment type="check"/>
  ```

  oppure

  ```
  <payment type="cash"/>
  ```

Attributo `attribute-type`

- `attribute-type` può avere i seguenti valori:
  - CDATA = The value is character data
  - (en1|en2|..) = The value must be one from an enumerated list
  - ID = The value is a unique id
  - IDREF = The value is the id of another element
  - IDREFS = The value is a list of other ids
  - NMTOKEN = The value is a valid XML name
  - NMTOKENS = The value is a list of valid XML names
  - ENTITY = The value is an entity
  - ENTITIES = The value is a list of entities
  - NOTATION = The value is a name of a notation
  - Xml = The value is a predefined xml value

Attributo `default-value`

- Possibili valori del `default-value`
  - value = il valore di default dell'attributo
  - #REQUIRED = valore attributo obbligatorio
  - #IMPLIED = attributo opzionale
  - #FIXED value = l'attributo ha un valore fisso
- Esempi:

```
<!ELEMENT square EMPTY>
<!ATTLIST square width CDATA "0">
    Valid XML: <square width="100" />

<!ATTLIST contact fax CDATA #IMPLIED>
    Valid XML: <contact fax="555-667788" />  <contact/>

<!ATTLIST person number CDATA #REQUIRED>
    Valid XML: <person number="5677"/>  (Invalid: <person/>)

<!ATTLIST sender company CDATA #FIXED "Microsoft">
    Valid XML: <sender company="Microsoft"/>
   (Invalid:   <sender company="Sun"/>)
```

Entity

- Dichiarazione di Entità Interna

```
<!ENTITY entity-name "entity-value">
```

  - Esempio di entità generale:

```
<!ENTITY writer "Dante Alighieri">
<!ENTITY copyright "Copyright Unifi.">
```

XML: `<author>&writer;&copyright;</author>`

  - Entità a parametro (contenuta solo in DTD):

```
<!ENTITY %idreq   "id   ID    #REQUIRED">
```

- Dichiarazione di Entità Esterna

```
<!ENTITY entity-name SYSTEM "URI/URL">
```

  - Esempio:

```
<!ENTITY writer
 SYSTEM "http://www.ing.unifi.it/dtd/entities.dtd">
<!ENTITY copyright
 SYSTEM "http://www.ing.unifi.it/dtd/entities.dtd">
```

XML: `<author>&writer;&copyright;</author>`

Esempio di DTD

```
<!-- definizione elemento radice: un "address book" è una lista di "entries" -->
<!ELEMENT address-book (entry+)>

<!--una "entry" è un nome seguito da 0 o più indirizzi, n. tel, fax e email -->
<!ELEMENT entry (name,address*, tel*, fax*, email*)>

<!-- un nome "name" è composto da stringhe, nome e cognome-->
<!ELEMENT name (fname | lname)*>
<!ELEMENT fname (#PCDATA)>
<!ELEMENT lname (#PCDATA)>

<!--definizione della struttura di un indirizzo -->
<!ELEMENT address (street,region?,postal-code,locality,country)>
<!ATTLIST address preferred (true | false) "false">
<!ELEMENT street (#PCDATA)>
<!ELEMENT region (#PCDATA)>
<!ELEMENT postal-code (#PCDATA)>

...
<!ELEMENT tel (#PCDATA)>
<!ATTLIST tel preferred (true | false) "false">

<!--elemento "email" vuoto, attributo "href" di tipo stringa, "preferred" indica
   il default (esempio di raggruppamento di definizione di attributi per uno
   stesso elemento-->
<!ELEMENT email EMPTY>
<!ATTLIST email href CDATA #REQUIRED
          preferred (true | false) "false">
```

Esempio di Documento XML

```
<?xml version="1.0"?>
<!DOCTYPE address-book SYSTEM "address-book.dtd">
<address-book>
    <entry>
        <name>John Doe</name>
        <address>
            <street>34 Fountain Square Plaza</street>
            <region>OH</region>
            <postal-code>45202</postal-code>
            <locality>Cincinnati</locality>
            <country>US</country>
        </address>
        <tel preferred="true">513-555-8889</tel>
        <tel>513-555-7098</tel>
        <email href="mailto:jdoe@emailaholic.com"/>
    </entry>
    <entry>
        <name><fname>Jack</fname><lname>Smith</lname></name>
        <tel>513-555-3465</tel>
        <email href="mailto:jsmith@emailaholic.com"/>
    </entry>
</address-book>
```

Riferimento alla DTD

```
<?xml version="1.0"?>
<!DOCTYPE address-book
          SYSTEM "address-book.dtd">
```

- DOCTYPE contiene la dichiarazione del tipo del documento corrente e la DTD che lo definisce
  - SYSTEM indica un system identifier una URI (Universal Resource Identifier) che punta alla DTD. Una URI è una generalizzazione di una URL;
  - PUBLIC è un modo indiretto per riferirsi alla posizione (remota o locale) di una DTD tramite un catalogo che include la DTD in maniera indiretta;

XML Schema

- XML Schema è una alternativa XML all'uso della DTD;
- Un XML Schema descrive la struttura di un documento XML;
- Il linguaggio di descrizione di un XML Schema si chiama anche XML Schema Definition (XSD);
- Un XML Schema è scritto in XML;
- Originariamente proposto da Microsoft, dal Maggio 2001 è una W3C recommendation;

XML Schema

- Come una DTD, un XML Schema definisce gli oggetti ammessi per un documento XML di un certo tipo;
- Un XML Schema definisce:
  - Gli elementi e gli attributi ammessi in un documento;
  - Le relazioni gerarchiche e l'ordine fra elementi;
  - Il numero possibile dei figli di un elemento;
  - Se un elemento è vuoto o meno;
  - I valori di default per gli attributi;
- In più rispetto ad una DTD definisce:
  - Eventuali valori di default per gli elementi;
  - I tipi di dato per gli elementi;

XML Schema vs DTD

- Vantaggi di XML Schema rispetto a DTD:
  - Supporta i tipi di dati;
  - Usa una sintassi XML;
  - Maggior sicurezza delle comunicazioni mediante l'uso dei tipi (es: un tipo “date” consente una corretta interpretazione di una stringa come una data);
  - Consente la validazione sul contenuto degli elementi;

Esempio di documento XML con Riferimento a DTD

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE note SYSTEM "note.dtd">
<note>
    <to>Tove</to>
    <from>Jani</from>
    <heading>Reminder</heading>
    <body>Don't forget me this
  weekend!</body>
</note>
```

Esempio di documento XML con Riferimento a XMLSchema

```
<?xml version="1.0" encoding="iso-8859-1"?>
<note
  xmlns="http://www.ing.unifi.it"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-
  instance"
  xsi:schemaLocation="http://www.ing.unifi.it/
  note.xsd">
  <to>Tove</to>
  <from>Jani</from>
  <heading>Reminder</heading>
  <body>Don't forget me this weekend!</body>
</note>
```

Classe di documenti relativa definita tramite DTD

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

Classe relativa definita tramite XMLSchema

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
           targetNamespace="http://www.ing.unifi.it"
           xmlns="http://www.ing.unifi.it"
           elementFormDefault="qualified">
  <xs:element name="note">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="to" type="xs:string"/>
        <xs:element name="from" type="xs:string"/>
        <xs:element name="heading" type="xs:string"/>
        <xs:element name="body" type="xs:string"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Sintassi di XMLSchema

- `xs:schema`
  - Radice di ogni dichiarazione dello Schema
- `xmlns:xs="http://www.w3.org/2001/XMLSchema"`
  - Namespace da cui provengono gli oggetti necessari per la definizione di un particolare Schema (element, type, complexType, targetNamespace, ecc.)
- `targetNamespace="http://www.ing.unifi.it"`
  - Indica il namespace cui appartengono gli elementi definiti dallo Schema corrente;
- `xmlns="http://www.ing.unifi.it"`
  - Indica il namespace di default degli elementi per i quali non è specificato il namespace di appartenenza;
- `elementFormDefault="qualified"`
  - Indica che tutti gli elementi dichiarati in una istanza di documento XML relativo allo Schema corrente devono avere un namespace di riferimento;

Simple Elements

- Elemento che contiene solo testo e sono privi di attributi;
- Dichiarazione di Simple Element

```
<xs:element name="to" type="xs:string"/>
<xs:element name="age" type="xs:integer"/>
<xs:element name="dateborn" type="xs:date"/>
```

  - Possibili tipi sono (xs:string, xs:decimal, xs:integer, xs:boolean, xs:date, xs:time);
  - Un default value è assegnato automaticamente in caso di mancanza di un altro valore;

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

  - Un fixed value è assegnato automaticamente e nessun altro valore può essere assegnato;

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

Attributi

- Se un elemento ha attributi esso è un Complex Type;
- Un attributo è definito come Simple Type;

es:

```
<xs:attribute name="lang" type="xs:string"
  default="EN"/>
```

  - Attributo lang di tipo stringa, con valore di default “EN”;
  - Altre possibili specificazioni:
    - `fixed= “EN”, use = “optional”, use = “required”`

XSD Restrictions/Facets

- Utilizzate per controllare valori accettabili sia per elementi che per attributi;
- Restrictions su valori
  - Elemento “age” con 0<=value<=100:

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

XSD Restrictions/Facets

- Restrictions su insiemi di valori;
- Elemento “car” con vincolo enumerativo (il contenuto di “car” può essere solo Audi, Golf, BMW):

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

XSD Restrictions/Facets

- L'esempio precedente può anche essere scritto nel modo seguente:

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

- In questo modo è stato definito il tipo enumerativo `carType` che può essere riutilizzato per altri elementi;

XSD Restrictions/Facets

- Restrizioni su serie di valori;
- Elemento “letter” che può contenere solo 1 lettera minuscola [a-z];

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

- Elemento “letter” che può contenere solo 3 lettera maiuscole [A-Z]:

```
<xs:pattern value="[A-Z][A-Z][A-Z]"/>
```

- Elemento “letter” che può contenere solo 1 delle 3 lettere [xyz]:

```
<xs:pattern value="[xyz]"/>
```

- Elemento “letter” che può contenere

```
<xs:pattern value="([a-z])*"/>
```
0 o più lettere [a-z]

```
<xs:pattern value="([a-z][A-Z])+"/>
```
1 o più lettere [a-z] [A-Z]

XSD Restrictions/Facets

- Elemento "gender" con valori accettabili “male” o “female”

```
<xs:element name="gender">
<xs:simpleType>
  <xs:restriction base="xs:string">
    <xs:pattern value="male|female"/>
  </xs:restriction>
</xs:simpleType>
</xs:element>
```

- Elemento “password” che può contenere solo 8 caratteri maiuscoli o minuscoli da 'a' a 'z' e da '0' a '9':

```
<xs:element name="password">
<xs:simpleType>
  <xs:restriction base="xs:string">
    <xs:pattern value="[a-zA-Z0-9]{8}"/>
  </xs:restriction>
</xs:simpleType>
</xs:element>
```

XSD Restrictions/Facets

- Restrizioni sui separatori:
- Mantenimento di tutti i separatori presenti

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

- Sostituzione di tutti i separatori con spazi

```
<xs:whiteSpace value="replace"/>
```

- Sostituzione di tutti i separatori con uno spazio

```
<xs:whiteSpace value="collapse"/>
```

XSD Restrictions/Facets

- Restrizioni sulla lunghezza
- Elemento “password” di esattamente 8 caratteri:

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

- Elemento “password” di 5<=n.caratteri<=8:

```
<xs:minLength value="5"/>
<xs:maxLength value="8"/>
```

Complex Element

- Elemento che contiene altri elementi o attributi;
  - Elemento "product" vuoto:

```
<product pid="1345"/>
```

  - Elemento "employee", contenente altri:

```
<employee>
 <firstname>John</firstname>
 <lastname>Smith</lastname>
</employee>
```

  - Elemento "food", contenente solo testo:

```
<food type="dessert">Ice cream</food>
```

  - Elemento "description", che contiene elementi e testo

```
<description>It happened on
<date lang="norwegian">03.03.99</date>....</description>
```

Definizione di Complex Element

- Dichiarazione di “employee” contenente gli elementi “firstname” e “lastname”;

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

- <xs:sequence> indica che gli elementi contenuti devono comparire in quello stesso ordine;

```
<xs:complexType name="personinfo">
  <xs:sequence> ... </xs:sequence>
</xs:complexType>
```

Definizione di Complex Element

- E' possibile definire un complex element come un nuovo tipo di dato e dichiarare il content model di nuovi elementi attraverso il nuovo tipo;

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

<xs:element name="employee" type="personinfo"/>
<xs:element name="student" type="personinfo"/>
<xs:element name="member" type="personinfo"/>
```

Definizione di Complex Element

- E' possibile definire tipi sulla base di tipi esistenti (meccanismo di ereditarietà per gli elementi);

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

Complex Element di solo testo

- Esempio di ComplexType contenente solo testo;

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

- E' anche possibile assegnare al nuovo elemento il rango di type e riferirsi ad esso per definire elementi con lo stesso content model o sue estensioni

```
<xs:element name="shoesize" type="shoetype"/>
```

Complex Element con Content Model Misto

- Occorre assegnare alla definizione di ComplexType l'attributo mixed=”true”;

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

- Esempio di porzione di documento:

```
<letter>
Dear Mr.<name>John Smith</name>.
Your order <orderid>1032</orderid>
will be shipped on <shipdate>2001-07-13</shipdate>.
</letter>
```

ComplexType con Order Indicators

- <xs:all>: gli elementi possono comparire in qualunque ordine una e una sola volta;

```
<xs:element name="person">
 <xs:complexType>
  <xs:all>
     <xs:element name="employee" type="employee"/>
     <xs:element name="member" type="member"/>
  </xs:all>
 </xs:complexType>
</xs:element>
```

- <xs:choice>: può comparire solo uno degli elementi;
- <xs:sequence>: gli elementi devono comparire in quello specifico ordine;

ComplexType con Occurrence Indicators

- maxOccurs, minOccurs: indicano il numero minimo e massimo delle occorrenze di un elemento (il default è = 1);

```
<xs:element name="person">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="full_name" type="xs:string"/>
      <xs:element name="child_name" type="xs:string"
      maxOccurs="10" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

ComplexType con Group Indicators (Element Groups)

- <xs:group> è utilizzato per definire insiemi di elementi;

```
<xs:group name="persongroup">
  <xs:sequence>
    <xs:element name="firstname" type="xs:string"/>
    <xs:element name="lastname" type="xs:string"/>
    <xs:element name="birthday" type="xs:date"/>
  </xs:sequence>
</xs:group>

<xs:element name="person" type="personinfo"/>

<xs:complexType name="personinfo">
  <xs:sequence>
    <xs:group ref="persongroup"/>
    <xs:element name="country" type="xs:string"/>
  </xs:sequence>
</xs:complexType>
```

ComplexType con Group Indicators (Attribute Groups)

- <xs:attributeGroup> è utilizzato per definire insiemi di attributi;

```
<xs:attributeGroup name="personattrgroup">
  <xs:attribute name="firstname" type="xs:string"/>
  <xs:attribute name="lastname" type="xs:string"/>
  <xs:attribute name="birthday" type="xs:date"/>
</xs:attributeGroup>

<xs:element name="person">
  <xs:complexType>
    <xs:attributeGroup ref="personattrgroup"/>
  </xs:complexType>
</xs:element>
```

Meccanismo di estensione di uno Schema: Elementi

- <xs:any> consente di estendere lo standard con elementi non previsti;
- family.xsd:

```
<xs:element name="person">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="firstname" type="xs:string"/>
      <xs:element name="lastname" type="xs:string"/>
      <xs:any minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

- Estensione dello schema “family.xsd” con lo schema “children.xsd” che definisce elementi `childern` successivi a `lastname`;
- children.xsd:

```
<xs:element name="children">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="childname" type="xs:string"
      maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
</xs:schema>
```

Meccanismo di estensione di uno Schema: Elementi

- "Myfamily.xml" usa elementi da due diversi schema "family.xsd" e "children.xsd":

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<persons xmlns="http://www.unifi.it"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:SchemaLocation="http://www.unifi.it family.xsd
http://www.ing.unifi.it children.xsd">
<person>
   <firstname>Hege</firstname>
   <lastname>Refsnes</lastname>
   <children>
      <childname>Cecilie</childname>
   </children>
</person>
<person>
<firstname>Stale</firstname>
<lastname>Refsnes</lastname>
</person>
</persons>
```

Meccanismo di estensione di uno Schema: Attributi

- <xs:anyAttribute> per estendere lo standard con attributi non previsti;
- “family.xsd”

```
<xs:element name="person">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="firstname" type="xs:string"/>
      <xs:element name="lastname" type="xs:string"/>
    </xs:sequence>
    <xs:anyAttribute/>
  </xs:complexType>
</xs:element>
```

- Estensione di "person" con l'attributo "gender"
- “attribute.xsd”:

```
<xs:attribute name="gender">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:pattern value="male|female"/>
    </xs:restriction>
  </xs:simpleType>
</xs:attribute>
</xs:schema>
```

Meccanismo di estensione di uno Schema: Attributi

- “MyFamily.xml”

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<persons xmlns="http://www.unifi.it"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:SchemaLocation="http://www.unifi.it family.xsd
  http://www.ing.unifi.it attribute.xsd">
  <person gender="female">
     <firstname>Hege</firstname>
     <lastname>Refsnes</lastname>
  </person>
  <person gender="male">
     <firstname>Stale</firstname>
     <lastname>Refsnes</lastname>
  </person>
</persons>
```

substituionGroup

- Attributo dello Schema che indica il sostituto di un elemento;
- Es. di dichiarazione di un elemento principale e di un suo sostituto in una lingua diversa:

```
<xs:element name="name" type="xs:string"/>
<xs:element name="nome" substitutionGroup="name"/>
<xs:complexType name="custinfo">
  <xs:sequence>
    <xs:element ref="name"/>
  </xs:sequence>
</xs:complexType>
<xs:element name="customer" type="custinfo"/>
<xs:element name="cliente" substitutionGroup="customer"/>
```

- Per non consentire la sostituibilità di un elemento si usa l'attributo `block= "substitution"`

```
<xs:element name="name" type="xs:string"
                        block="substitution"/>
<xs:element name="nome" substitutionGroup="name"/>
```

substituionGroup

- I due contenuti seguenti sono entrambi validi:

```
<customer>
  <name>John Smith</name>
</customer>


<cliente>
  <nome>John Smith</nome>
</cliente>
```

- substitutionGroup consente per esempio di localizzare uno schema rispetto a lingue diverse, in modo da poter scegliere di scrivere un documento con gli elementi XML in una lingua o in un'altra;
- Attenzione: gli elementi del substitutionGroup devono essere dichiarati globali (figli diretti di <xs:schema>)

Tipi predefiniti

- Stringhe
  - xs:string
  - xs:normalizedString (stringa priva separatori a inizio/fine);
  - xs:token (stringa priva di separatori a inizio/fine e di spazi multipli);
- Date
  - xs:date Defines a date value
  - dateTime Defines a date and time value
  - duration Defines a time interval
  - gDay Defines a part of a date - the day (DD)
  - gMonth Defines a part of a date - the month (MM)
  - gMonthDay Defines a part of a date - the month and day (MM-DD)
  - gYear Defines a part of a date - the year (YYYY)
  - gYearMonth Defines a part of a date - the year and month (YYYY-MM)
  - time Defines a time value

Tipi predefiniti

- Tipi Numerici
  - byte — A signed 8-bit integer
  - decimal — A decimal value
  - int — A signed 32-bit integer
  - integer — An integer value
  - long — A signed 64-bit integer
  - negativeInteger — An integer containing only negative values (.., -2, -1.)
  - nonNegativeInteger — An integer containing only non-negative values (0, 1, 2, ..)
  - nonPositiveInteger — An integer containing only non-positive values (.., -2, -1, 0)
  - positiveInteger — An integer containing only positive values (1, 2, ..)
  - short — A signed 16-bit integer
  - unsignedLong — An unsigned 64-bit integer
  - unsignedInt — An unsigned 32-bit integer
  - unsignedShort — An unsigned 16-bit integer
  - unsignedByte — An unsigned 8-bit integer

Tipi predefiniti

- Tipi misti
  - anyURI
  - base64Binary
  - boolean
  - double
  - float
  - hexBinary
  - NOTATION = tipo notation definito da W3C: il contenuto non è validato dal parser xml, fornisce al parser un meccanismo per localizzare programmi esterni o processing istructions;
  - QName = qualified name (prefix:name)

Stylesheet

- Un documento XML
  - Contiene solo informazione sulla struttura
  - Nessuna informazione su come visualizzarlo

```
<intestazione>
    <tipoDoc>LEGGE COSTITUZIONALE</tipoDoc>
    <dataDoc norm="19991122">22 novembre 1999</dataDoc>
    <numDoc>1</numDoc>
    <titoloDoc>Disposizioni concernenti l'elezione diretta del Presidente
    della Giunta regionale e l'autonomia statutaria delle Regioni.
    </titoloDoc>
</intestazione>
```

Stylesheet

- Per visualizzare un documento XML occorre associare delle informazione di stile
  - Fogli di stile (stylesheet)

```
titoloDoc {
        font-family: “New York”;
        font-size: 12pt;
        color: red;
}
```

Stylesheet

- Due linguaggi per scrivere stylesheet
  - CSS (Cascading Style Sheet)
  - XSL (XML Stylesheet Language)
- CSS
  - Originalmente creato per HTML;
  - Fornisce ai browser le indicazioni sullo stile (font, formattazione, margini, colori, ecc.) con cui visualizzare certe informazioni;
- XSL
  - Trasforma un documento XML prima di visualizzarlo;
  - E' organizzato in due parti:
    - XSLT: XSL Transformation
    - XSL-FO: XSL Formatting Objects

XSLT

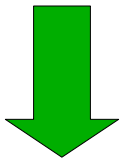
- Linguaggio per descrivere trasformazioni di un documento XML;
- Non fornisce solo direttive di stile ma consente la trasformazione di un documento XML anche per altri scopi:
  - Inserimento di un logo per la visualizzazione;
  - Crea nuovi contenuti (es: Sommario);
  - Evidenzia porzioni del documento piuttosto di altre a seconda del lettore cui è destinato;
  - Converte un documento per aderire a DTD diverse (es: versioni successive di una DTD);
  - Trasforma un XML verso un HTML per compatibilità con i vecchi browser;

XSLT

- Il nucleo di uno stylesheet è formato da una lista di template;
- Es:

```
<?xml version=”1.0”?>
<article fname=”19990101_xsl”>
  <title>XML Style Sheets</title>
  <date>January 1999</date>
  <abstract>...</abstract>
<section>
   <title>Styling</title>
</section>
...
</article>
```

```
<xsl:template match=“section/title”>
<P><I><xsl:apply-templates/></I></P>
</xsl:template>
```



<P><I>Styling</I></P>

XSLT

```
<xsl:template match="section/title">
  <P><I><xsl:apply-templates/></I></P>
</xsl:template>
```

- Il contenuto del parametro `match` è il path dell'elemento a cui si applica il template;
- Il contenuto di template elenca gli elementi da inserire nell'albero XML risultante;
- Il path per identificare gli elementi è descritto nella sintassi XPath;

XSLT

```
<p>Send comments and suggestions to
<url protocol="mailto">nome@ing.unifi.it</url>.</p>
```

```
<xsl:template match="url[@protocol='mailto']">
<A>
 <xsl:attribute name="href">mailto:<xsl:apply-templates/>
 </xsl:attribute>
 <xsl:apply-templates/>
</A>
</xsl:template>
```

<A href="mailto:nome@ing.unifi.it">
nome@ing.unifi.it</A>

XSLT + CSS

- XSLT trasforma gli elementi di un file XML;
- CSS esprime direttive di formattazione per gli elementi originari XML;

- XSL

```
<xsl:template match=”section/title”>
        <P><I><xsl:apply-templates/></I></P>
  </xsl:template>
```

```
<xsl:processing-instruction name="xml-stylesheet">
      <xsl:text>href="mystyle.css"
type="text/css"</xsl:text>
</xsl:processing-instruction>
```

L'output generato è

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

che include nel nuovo documento il foglio di stile css

CSS (mystyle.css)

```
section>title
{
  display:block;
  font-style:italic;
}
```

Stylesheet

```
intestazione { display: block; margin: 20pt; padding:10pt; font-size: 10pt;
               text-align: center; background-color: ccccff; }
titoloDoc    { font-size: 12pt; font-weight: bold ; color: red; }
```

```
<intestazione>
    <tipoDoc>LEGGE COSTITUZIONALE</tipoDoc>
    <dataDoc norm="19991122">22 novembre 1999</dataDoc>
    <numDoc>1</numDoc>
    <titoloDoc>Disposizioni concernenti l'elezione diretta del Presidente
    della Giunta regionale e l'autonomia statutaria delle Regioni.
    </titoloDoc>
</intestazione>
```

LEGGE COSTITUZIONALE del 22 novembre 1999, n. 1

Disposizioni concernenti l'elezione diretta del Presidente della Giunta regionale e l'autonomia statutaria delle Regioni.

XSLT- XSLFO

- XSLFO inserisce le caratteristiche di CSS dentro un XSLT;

```
<xsl:template match=”title”>
  <fo:block
      font-size=”13pt”
      font-weight=”bold”>
      <xsl:apply-templates/>
  </fo:block>
</xsl:template>
```

Gestione Software di un Documento XML

- Ogni applicazione che gestisce un documento XML si basa su un parser;
- Un parser consente di mappare un file XML in una struttura dati in memoria;
- Esistono 2 tipi di parser:
  - Parser DOM: mappa, ed eventualmente valida secondo una DTD, un documento XML in una struttura dati detta DOM (Document Object Model);
    - DOM è un interfaccia W3C ad oggetti, di cui esistono varie implementazioni (Sun, Apache,...);
  - Parser SAX: interfaccia event-based che gestisce eventi del tipo: apertura/chiusura elemento, element opening tags, entità, parsing errors;
    - SAX non è W3C standard;

Gestione Software di un Documento XML basato su un XML Schema

- Un elemento definito da XML Schema standard può essere visto come una classe Java (tipo di dato con le sue proprietà);

- Tecnologie “XML&Java data binding”:
  - Consentono di mappare un tipo di dato definito da un XMLSchema in una classe Java (in generale un XMLSchema in una gerarchia di classi Java);
  - L'accesso ad un documento XML e ai suoi elementi avviene attraverso i metodi (`[NomeElemento].set...` `[NomeElemento].get...`) delle classi corrispondenti;

Esempi di tecnologie XML&Java Data Binding

- Apache XML Beans
  - Una modalità s*chema-oriented* per vedere documenti XML attraverso Java types basati su quello schema (generazione di una gerarchia di classi corrispondenti agli elementi definiti da un XML schema);
  - Una modalità *schema-agnostic* per attraversare un intero documento XML.
  - Uno schema object model attraverso cui è possibile esaminare e compilare uno schema;
- Castor
  - Castor XML: Java object model to and from XML
  - Castor JDO: Java object persistence to RDBMS
  - Castor DAX: Java object persistence to LDAP

URI

- Stringa di caratteri utilizzata per identificare in modo compatto e uniforme una risorsa indipendentemente dalla sua natura (fisica o astratta).
- **Uniform**
  - **Modalità uniformi per identificare fisicamente (istanza) o astrattamente una risorsa;**
- **Resource**
  - **Una risorsa è qualsiasi cosa abbia un'identità. Esempi: Documenti elettronici, immagini, servizi, insiemi di altre risorse**
- **Identifier**
  - **Un identificatore è un oggetto che costituisce un riferimento per un ente che abbia un'identità. Nel caso di URI, ha anche una sintassi ben definita.**

URI, URN, URL

- Il W3C ha contribuito a generare una certa confusione (non ancora del tutto risolta) nel fornire le definizioni di URI, URN e URL (http://www.w3.org/TR/uri-clarification)
- URN = Uniform Resource Name
  - Identificatore univoco (un tipo di URI) basato sulle proprietà di un documento, indipendente dalla sua localizzazione fisica;
- URL = Uniform Resource Locator
  - Identificatore della istanza di un documento; è un tipo di URI che identifica una risorsa per mezzo della rappresentazione del suo primario meccanismo d'accesso (es. la sua localizzazione sulla rete), piuttosto che attraverso le sue proprietà;