<?xml?>

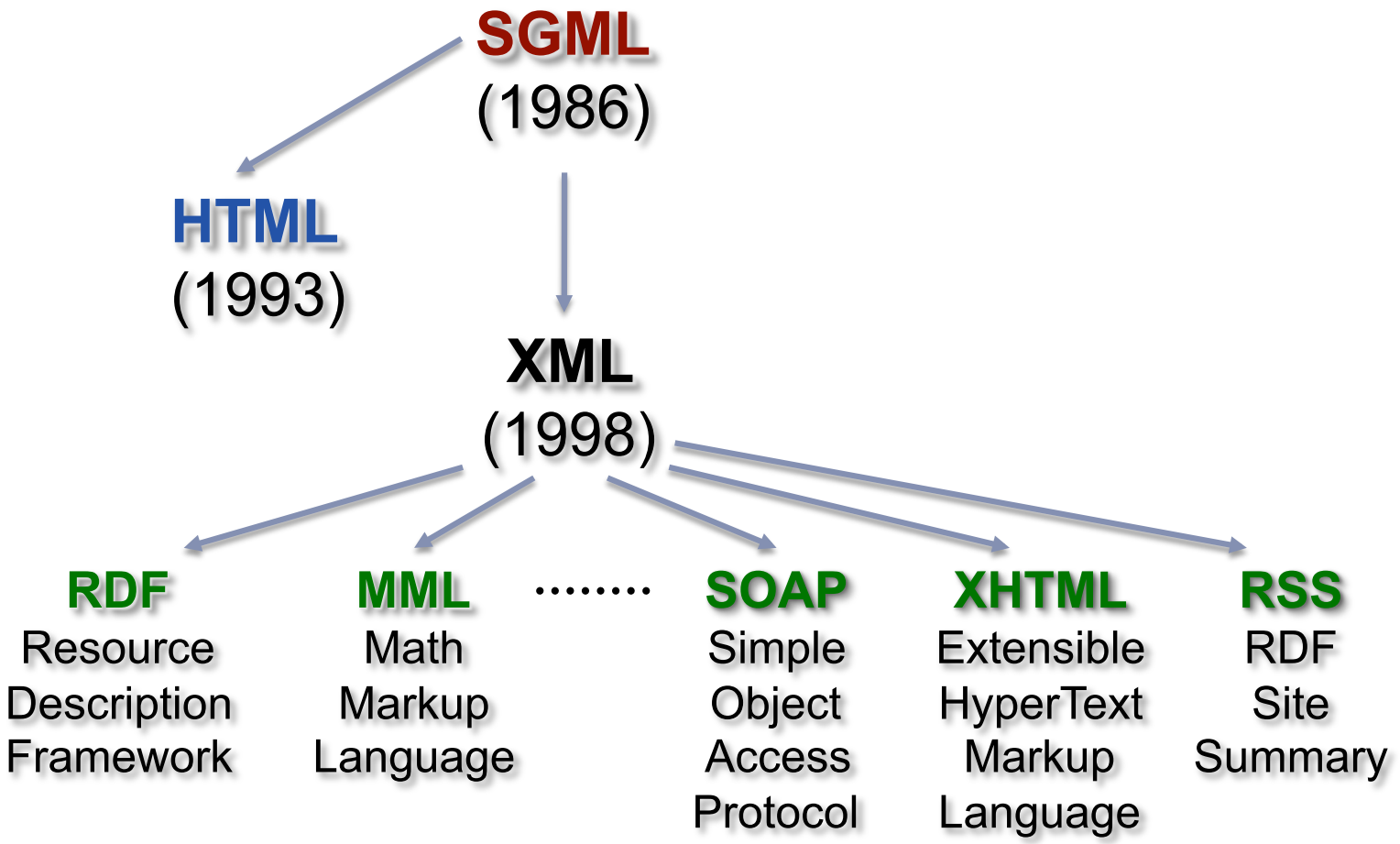
Reti di Calcolatori

XML

XML

- Extensible Markup Language
  - E' un metalinguaggio che permette di creare linguaggi di markup
    - Permette di definire senza limitazioni nuovi elementi (tag) di marcatura
  - E' stato sviluppato da un gruppo di lavoro del W3C
    - E' uno standard aperto di riferimento per tutti i produttori
    - E' una semplificazione di SGML (Standard Generalized Markup Language)
  - XML è basato sul testo (con codifica Unicode) e permette di definire informazioni di tipo strutturale e semantico
    - Rispetto ad HTML non fornisce dettagli relativi alla presentazione
    - Definisce una struttura gerarchica del documento utilizzando la nidificazione dei tag
    - E' utilizzato per la rappresentazione di strutture dati arbitrarie
      - es. formato di salvataggio dei documenti OpenOffice

Sviluppo dei linguaggi di markup



I linguaggi di markup

- Il documento è strutturato per mezzo di TAG
  - Un tag racchiude una porzione di documento con una data caratteristica
  - E' previsto un tag di apertura e uno di chiusura del contesto di validità del tag

```
<tag>.......</tag>
```

  - Alcuni tag prevedono attributi

```
<A href="home.html">...</A>
```

  - Alcuni tag possono comparire solo all'interno del contesto di un altro tag (nesting)

```
<TABLE><TR><TD>..</TD>...</TR>...</TABLE>
```

HTML

- L'insieme dei tag è predefinito
- Per ogni tag è definita una regola di rendering sul browser
  - HTML è strettamente legato alla presentazione
- I tag HTML definiscono
  - Struttura del documento (`<TITLE>`, `<TABLE>`, `<LIST>`,..)
  - Una limitata semantica (`<TITLE>`, `<H1>`,`<A>`,...)
- I browser sono tolleranti sulla correttezza dei documenti
  - Sono ammessi anche documenti non ben formati
  - Il risultato può essere diverso con browser diversi

Caratteristiche di XML

- Prevede la definizione di tag e attributi “custom”
  - Si possono definire linguaggi di markup adatti alla specifica applicazione
  - Definisce la sintassi per la strutturazione dei documenti
  - Può essere utilizzato per rappresentare dati strutturati
    - Tabelle di database
    - Strutture dati
- La presentazione del documento è separata dalla sua strutturazione attraverso l’uso di fogli di stile
- Il documento è un file di testo in Unicode
  - E’ quindi indipendente dalla piattaforma

Tipi di documento XML

- **Document-Centric XML**
  - Rappresentazione di documenti semi-strutturati (manuali tecnici, cataloghi di prodotti, documenti legali…)
  - Pensato per la lettura da parte di un essere umano
  - Definizione della semantica dei tag flessibile
  - Forte uso di attributi degli elementi

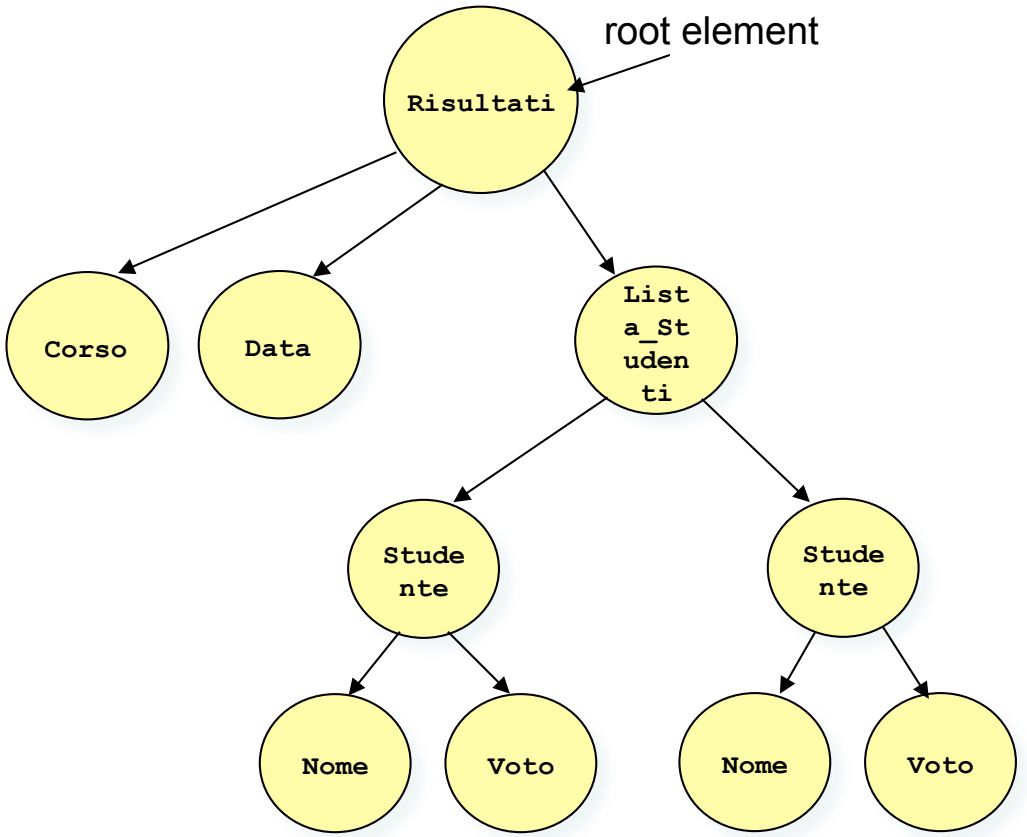
- **Data-Centric XML**
  - Rappresentazione di dati fortemente strutturati (dati relazionali di database, informazioni su transazioni finanziarie, strutture dati di linguaggi di programmazione…)
  - Pensato per l'elaborazione automatica
  - Definizione della semantica dei tag rigorosa
  - Scarso uso di attributi degli elementi

Struttura di XML

- la definizione di un documento XML prevede tre aspetti
  - Il contenuto
  - Specifiche relative alla struttura (DTD – Document Type Definition/XML Schema)
    - Come è organizzata l'informazione (sintassi)
    - Come interpretare la struttura (sintassi/semantica)
  - Specifiche relative alla visualizzazione (XSL – StyleSheet)
    - Come presentare l'informazione

Un esempio di documento XML

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

<Risultati>
  <Corso>Basi di dati</Corso>
  <Data>10 Dicembre 2003</Data>
  <Lista_Studenti>
    <Studente>
    <Nome>Mario Rossi</Nome>
    <Voto>27</Voto>
    </Studente>
    <Studente>
    <Nome>Giuseppe Verdi</Nome>
    <Voto>30L</Voto>
    </Studente>
  </Lista_Studenti>
</Risultati>
```



Tag in XML

- I tag definiscono la struttura del contenuto
  - I tag sono stringhe Unicode case-sensitive
  - I tag devono essere a coppie
    - il tag di chiusura è obbligatorio
    - Una coppia di tag definisce un elemento XML
  - I tag definiscono una struttura del documento gerarchica ad albero
    - Un elemento definisce un nodo dell'albero che può avere elementi figli
  - La coppia di tag più esterni definiscono il root element
    - E' consentita la presenza di un solo root element
  - Non è ammesso di avere coppie di tag che si intersecano
    - `<a>..<b>..</a>..</b>` non è corretto!
  - Se un tag non ha contenuto si può scriverlo in forma abbreviata
    - `<a></a>` può essere scritto come `<a />`

Nidificazione (nesting)

- Gli oggetti possono essere inseriti uno dentro l'altro con l'uso della nidificazione

```
<Studente>
    <Nome>...</Nome>
    <Voto>...</Voto>
</Studente>
```

sono validi solo dentro il contesto del tag **<Studente>**

  - Ogni oggetto secondario (nidificato) deve risiedere interamente all'interno dell'elemento che lo contiene
    - es. **<Voto>** all'esterno dell'elemento **<Studente>** perde il suo significato
  - La nidificazione definisce una relazione fra gli elementi di un documento

Documenti ben formati

- Un documento ben formato
  - contiene almeno un elemento
  - ciascun elemento deve annidare correttamente al suo interno gli elementi figli
  - tutti i tag di apertura e di chiusura corrispondono
  - esiste un solo root element
- Se il documento non è ben formato il parser XML deve generare un errore
- Riguarda le regole generali di costruzione corretta dei documenti XML indipendentemente dai tag usati

Documenti validi

- Un documento valido
  - Ha un DTD/XML Schema che ne specifica la sintassi
  - Il documento è conforme al suo DTD/XML Schema

- La validazione con uno schema permette di verificare
  - quali elementi ed attributi sono permessi nel documento
  - l'aderenza dei valori ai tipi predefiniti
  - l'architettura logica del documento nella sua interezza
  - i vincoli di annidamento degli elementi

Struttura di un documento XML

- Un documento XML prevede due parti
  - Prologo
    - Informazioni sulla versione di XML usata, sulla codifica dei caratteri e sulla necessità di elaborare o meno il DTD/XML Schema

      <? xml version="1.0" encoding="UTF-8" standalone="yes" ?>

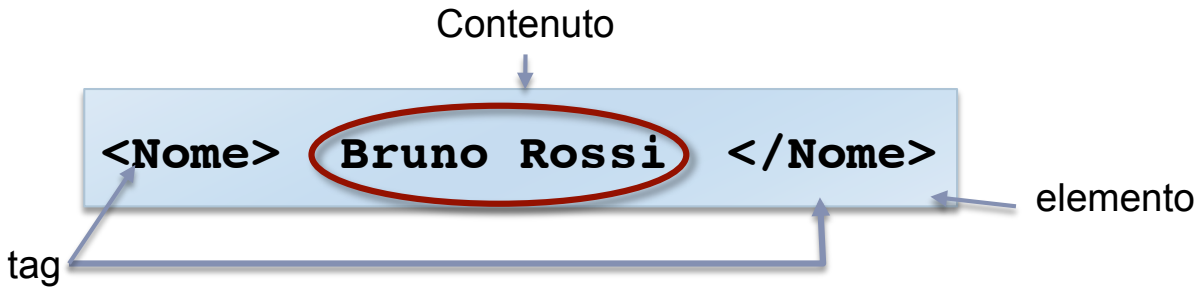
    - Dichiarazioni sul tipo di documento (es. DTD esterno)

      <!DOCTYPE name SYSTEM "http://www.xxx.it/yyy.dtd" >

  - Istanza
    - Elementi
    - Sezioni miste

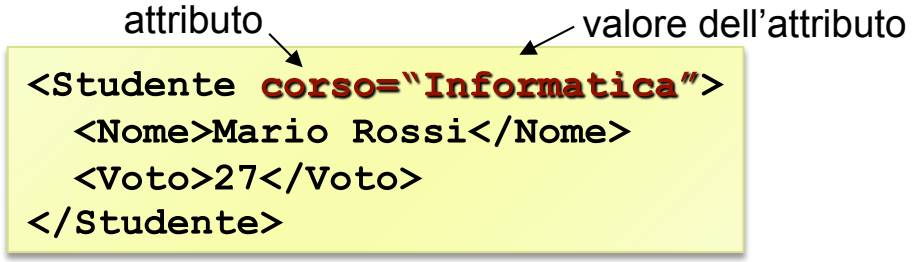
Istanza

- E' un'istanza della classe di documenti definiti dal DTD che è costituita da
  - elementi (element)
    - Ogni elemento ha un contenuto delimitato dai tag di apertura e chiusura corrispondenti

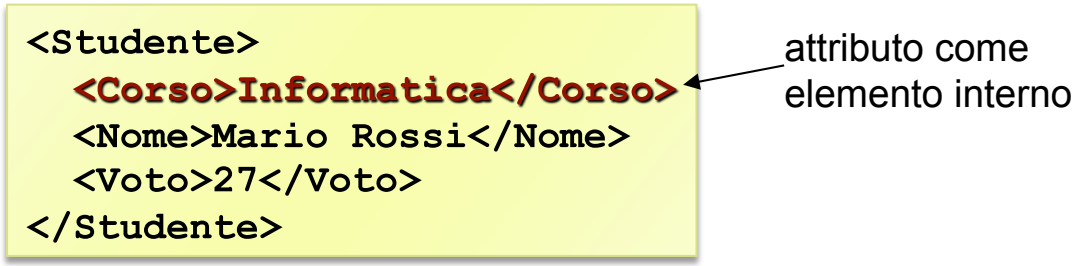


    - Il contenuto può comprendere altri elementi
    - Un elemento può avere attributi
  - parte mista (misc)

Attributi degli elementi



- L'attributo aggiunge una proprietà all'elemento
- Ogni attributo ha un nome e un valore
- Gli attributi ammissibili per un elemento sono specificati nel DTD/ XML Schema
- Un attributo può essere rappresentato da un elemento interno



Parti misc [1]

- Entità
  - Sono dati definiti nel DTD/XML Schema

  - Si può riferirsi all'entità con `&nome;` (es. `&author;`)
  - Un esempio di entità sono le definizioni per i caratteri non usabili nel testo analizzato dal parser XML (es. < è `<`)

- Commenti
  - Sono inclusi fra "`<!--`" e "`-->`"
  - Non si possono mettere prima della dichiarazione <? xml ... ?>

```
<!-- commento -->
```

Parti misc [2]

- Sezioni CDATA
  - Permettono di definire sezioni di dati che non sono analizzate dal parser XML
  - Nelle sezioni CDATA sono ammessi tutti i caratteri
  - Ad esempio può essere usata per inserire codice di script

```
<jscript><![CDATA[
    if(a>b && d==1) {
    document.write("<br />Prova<br />");
    }
]]></jscript>
```

parte non analizzata

Namespaces

- Un namespace aiuta ad assicurare l'unicità della definizione degli elementi XML
  - Si utilizzano nomi "qualificati" ovvero preceduti da un prefisso che indica il namespace di riferimento

```
spazioNomi:nomeLocale
```

  - Un namespace è quindi un insieme di nomi identificati da un URI di riferimento (in genere un URL)
  - In questo modo si evitano collisioni sui nomi di elementi ed attributi quando si uniscono documenti di origine diversa
  - Di solito il namespace è definito nel root element ma può essere definito a livello di qualsiasi elemento

Definizione del namespace

- Si può definire un namespace di default per un elemento

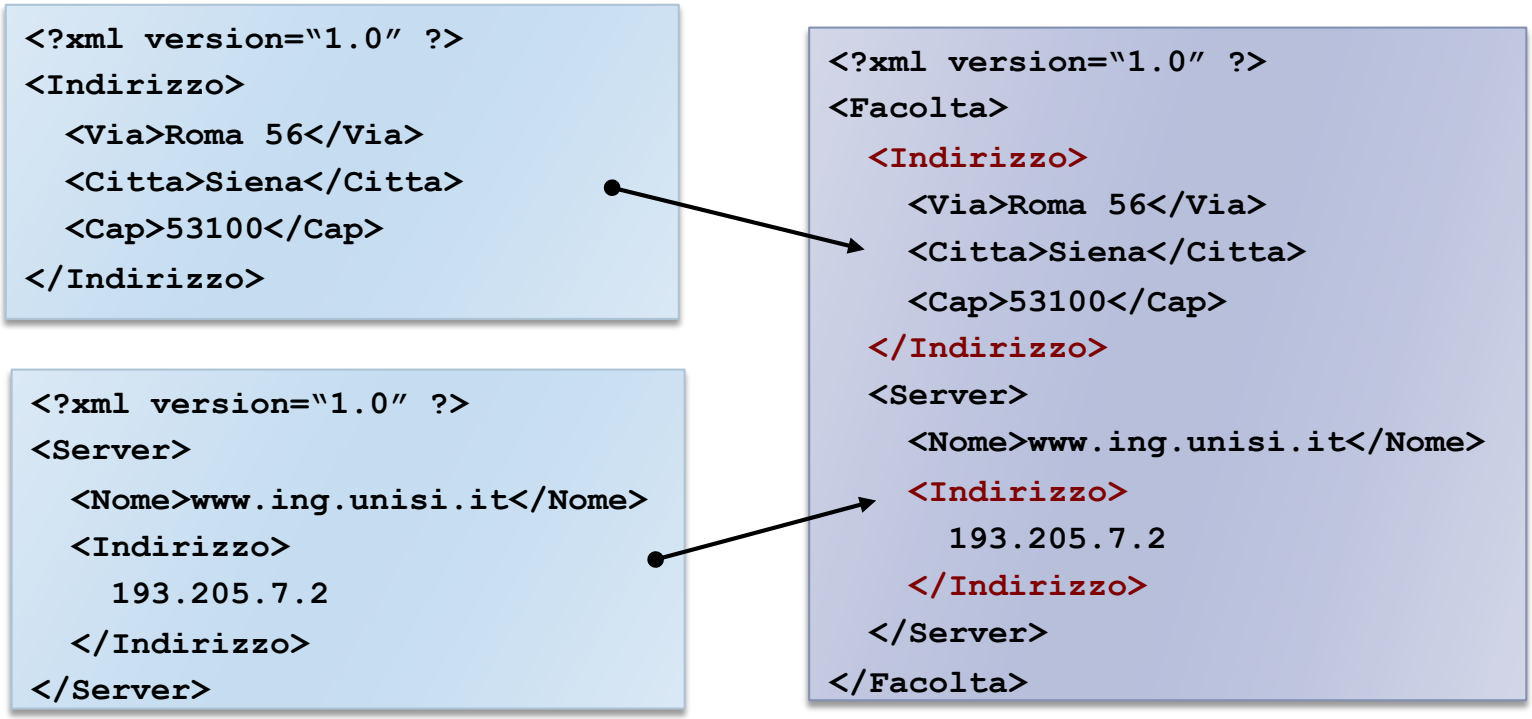
```
<Risultati xmlns="http://www.ing.unisi.it/%7emaggini">
  <Corso>Basi di dati</Corso>
</Risultati>
```

- Si può definire un namespace esplicito

```
<mag:Risultati xmlns:mag="http://www.ing.unisi.it/%7emaggini">
  <mag:Corso>Basi di dati</mag:Corso>
</mag:Risultati>
```

  - **mag** è un alias del nome del namespace completo che è rappresentato dall'URL

Esempio



- C'è un conflitto nell'interpretazione dell'elemento **`Indirizzo`**

Soluzione con namespace

```
<?xml version="1.0" ?>
<Facolta>
  <ind:Indirizzo xmlns:ind="www.ing.unisi.it/ind">
    <ind:Via>Roma 56</ind:Via>
    <ind:Citta>Siena</ind:Citta>
    <ind:Cap>53100</ind:Cap>
  </ind:Indirizzo>
  <srv:Server xmlns:srv="www.ing.unisi.it/srv">
    <srv:Nome>www.ing.unisi.it</srv:Nome>
    <srv:Indirizzo>193.205.7.2</srv:Indirizzo>
  </srv:Server>
</Facolta>
```

- Si utilizzano due namespace distinti (l'unicità è garantita dall'uso di URI)

XML Schema

- Metodo per la definizione della struttura di una classe di documenti XML
  - Una schema XML è di facile lettura perché è in XML
  - Gli schemi XML supportano i tipi dati e i namespace

- Si definiscono due tipi di elementi
  - di tipo semplice
    Non contengono altri elementi ma solo dati semplici come stringhe, numeri, date..
  - di tipo complesso
    Contengono altri elementi o attributi

Uso di XML Schema

- XML Schema definisce
  - quali elementi possono essere presenti in un documento
  - l'ordine e delle relazioni tra elementi
  - Gli attributi di ogni elemento e se sono opzionali, richiesti o godono di qualche altra proprietà particolare
  - Il tipo di dato del contenuto degli attributi

Definizione di elementi

- Gli elementi sono dichiarati usando l'elemento **element**

```
<xsd:element name="nomeElemento" type="nomeTipo" />
```

  - Il namespace **xsd** è usato convenzionalmente per tutti gli elementi di XML schema
  - **nomeElemento** è il nome dell'elemento nel sorgente XML
  - **nomeTipo** definisce un nome di tipo di dato semplice o complesso
  - Si può anche non usare l'attributo type per specificare il tipo di dato, definendo il tipo all'interno dell'elemento element

```
<xsd:element name="nomeElemento">
  - definizione tipo -
</xsd:element>
```

Tipi di dato

- XML Schema supporta la definizione di tipi di dato
  - Permette di specificare meglio i contenuti ammessi nel documento
  - E' più facile validare la correttezza dei dati
  - Si possono definire restrizioni sui dati (facets)
  - E' possibile definire pattern sui dati (formato dei dati)
  - E' più semplice convertire dati da un tipo all'altro
  - Prevede tipi di dato semplici e complessi
- Si possono definire nuovi tipi di dato a partire dalle quelli esistenti
  - E' predefinito un insieme di tipi primitivi

Tipi di dato semplici

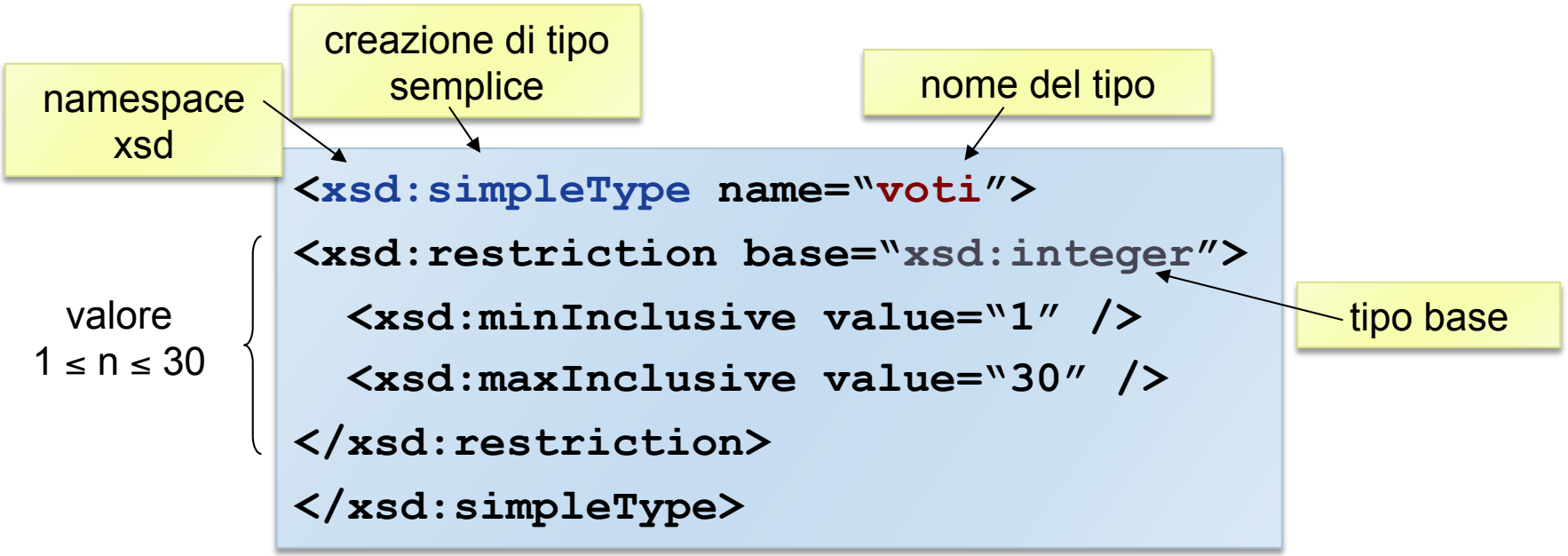
- Possono essere definiti dall'utente con l'elemento **simpleType**

```
<xsd:simpleType name="nomeTipo">
  - definizione del tipo -
</xsd:simpleType>
```

- Sono presenti tipi predefiniti nella specifica di XML schema
  - **string**
  - **boolean**
  - **decimal**
  - **integer**
  - **date**
  - **binary**
  - **uriReference**

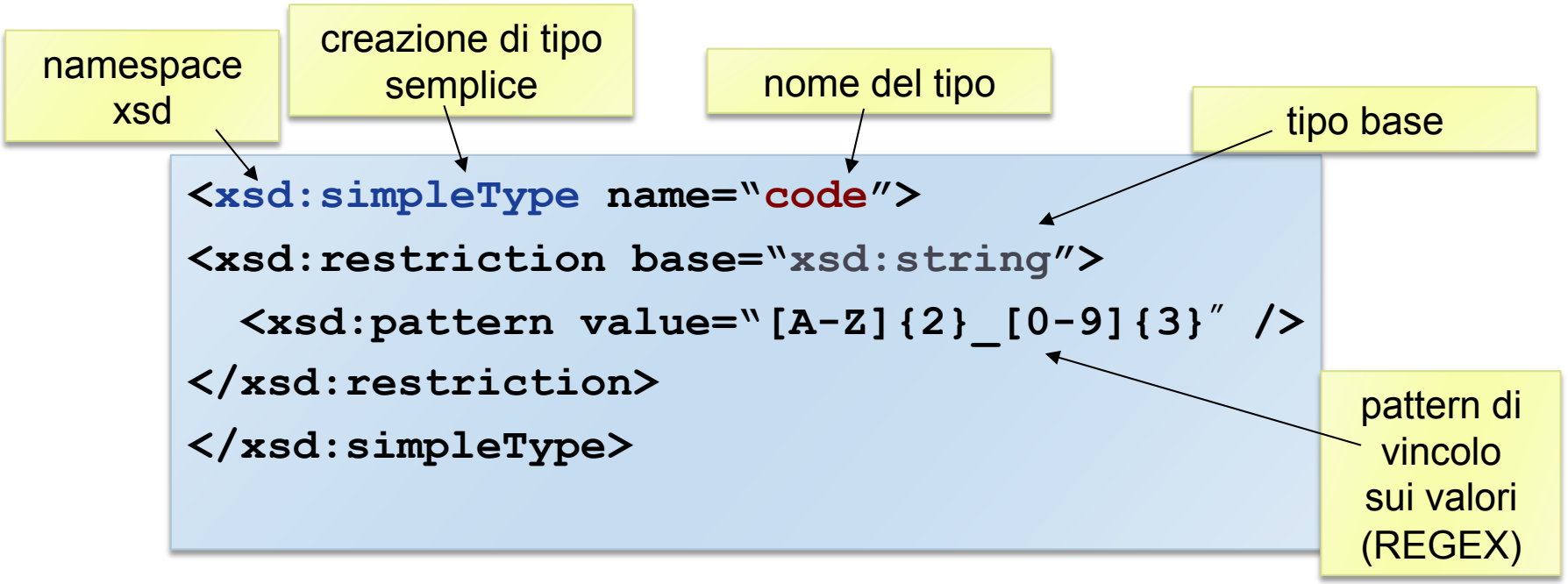
Costruzione di tipi semplici

- Definizione di un nuovo tipo semplice derivato con una restrizione dal tipo intero



Un altro esempio

- Definizione di una stringa che rispetta un pattern specifico
  - 2 caratteri maiuscoli seguiti da underscore e un numero di 3 cifre



Tipi di dato complesso

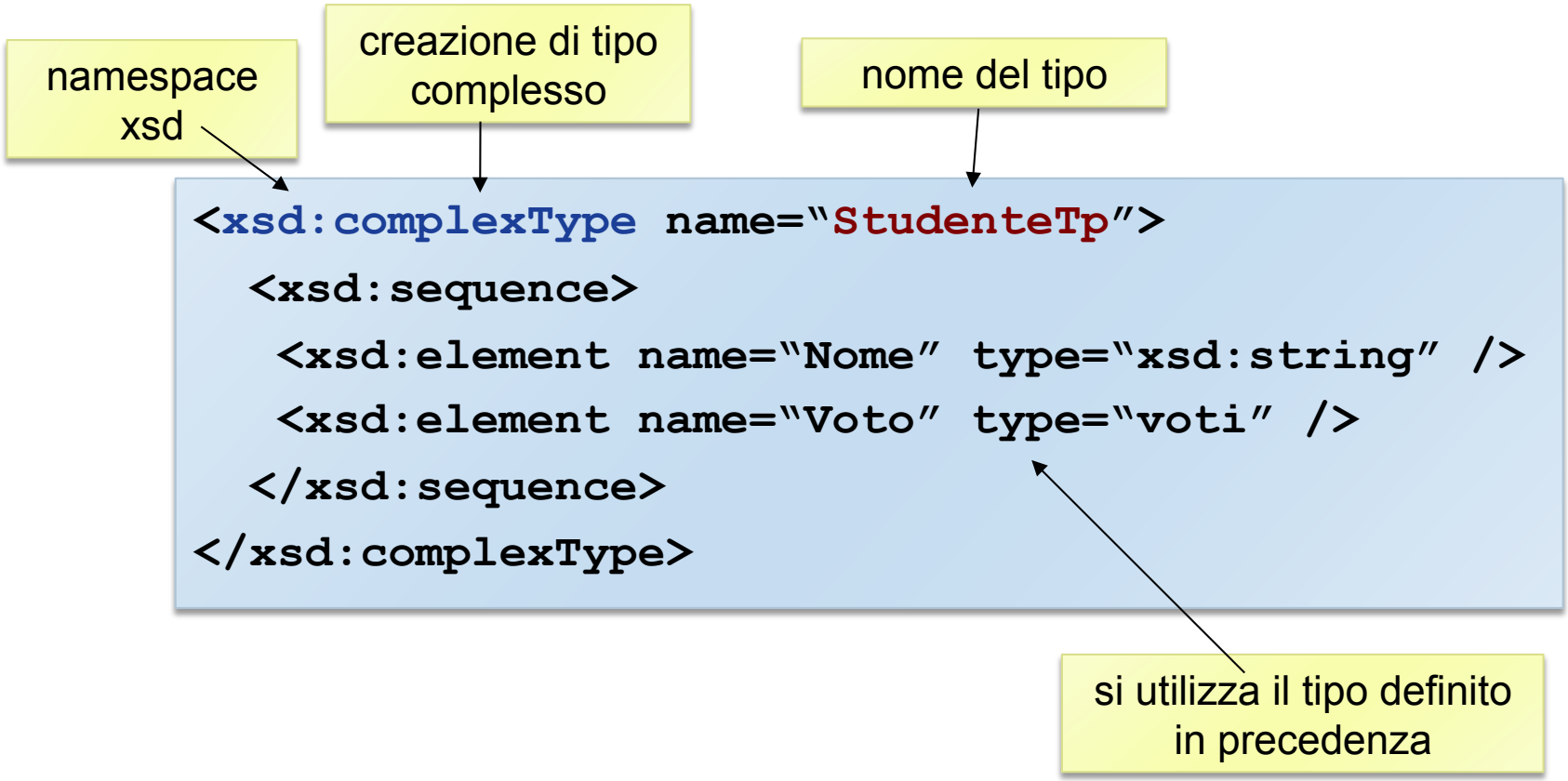
- Si definiscono utilizzando l'elemento **complexType**

```
<xsd:complexType name="nomeTipo">
  - definizione del tipo -
</xsd:complexType>
```

  - La definizione può contenere dichiarazioni di elementi, riferimenti ad elementi
  - La definizione può prevedere un elemento
    - **<sequence>** per specificare una lista ordinata di elementi
    - **<choice>** per indicare un sottoinsieme di tutti gli elementi specificati
    - **<all>** per indicare un insieme di elementi non ordinato

Costruzione di tipi complessi

- Definizione del tipo complesso **`StudenteTp`**



Indicatori di occorrenza

- Sono utilizzati per indicare quante volte un elemento può occorrere
  - **`maxOccurs`** : numero massimo
  - **`minOccurs`** : numero minimo
- Per default **`minOccurs=1`** e **`maxOccurs=minOccurs`**
- **`minOccurs=0`** identifica un elemento opzionale
- se il numero massimo di ripetizioni non è definito si usa "**`unbounded`**"

```
<xsd:complexType name="Lista_StudentiTp">
  <xsd:sequence>
    <xsd:element name="Studente" type="StudenteTp"
                 minOccurs="0" maxOccurs="unbounded" />
  </xsd:sequence>
</xsd:complexType>
```

Definizione degli attributi

- Gli attributi si possono associare solo ad elementi complessi
  - L'attributo è definito con l'elemento attribute

```
<xsd:attribute name="nomeAttr" type="tipoAttr" />
```

  - Gli attributi sono sempre definiti usando un tipo semplice
  - Gli attributi si possono dichiarare opzionali o obbligatori usando la specifica **use**

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

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

Definizione di attributi

- Si possono definire il valore di default o un valore fisso per un dato attributo

```
<xsd:attribute name="language" type="xsd:string" default="EN" />

<xsd:attribute name="disponibile" type="xsd:string" fixed="SI" />
```

- Per convenzione gli attributi si definiscono come ultimo elemento di un tipo complesso

```
<xsd:element name="Studente" type="StudenteTp" />
<xsd:complexType name="StudenteTp">
  <xsd:sequence>
    <xsd:element name="Nome" type="xsd:string" />
    <xsd:element name="Voto" type="voti" />
  </xsd:sequence>
  <xsd:attribute name="matricola"
   type="xsd:string" />
</xsd:complexType>
```

Definizione dello schema

- L'elemento **`schema`** è il root element di ogni definizione di schema

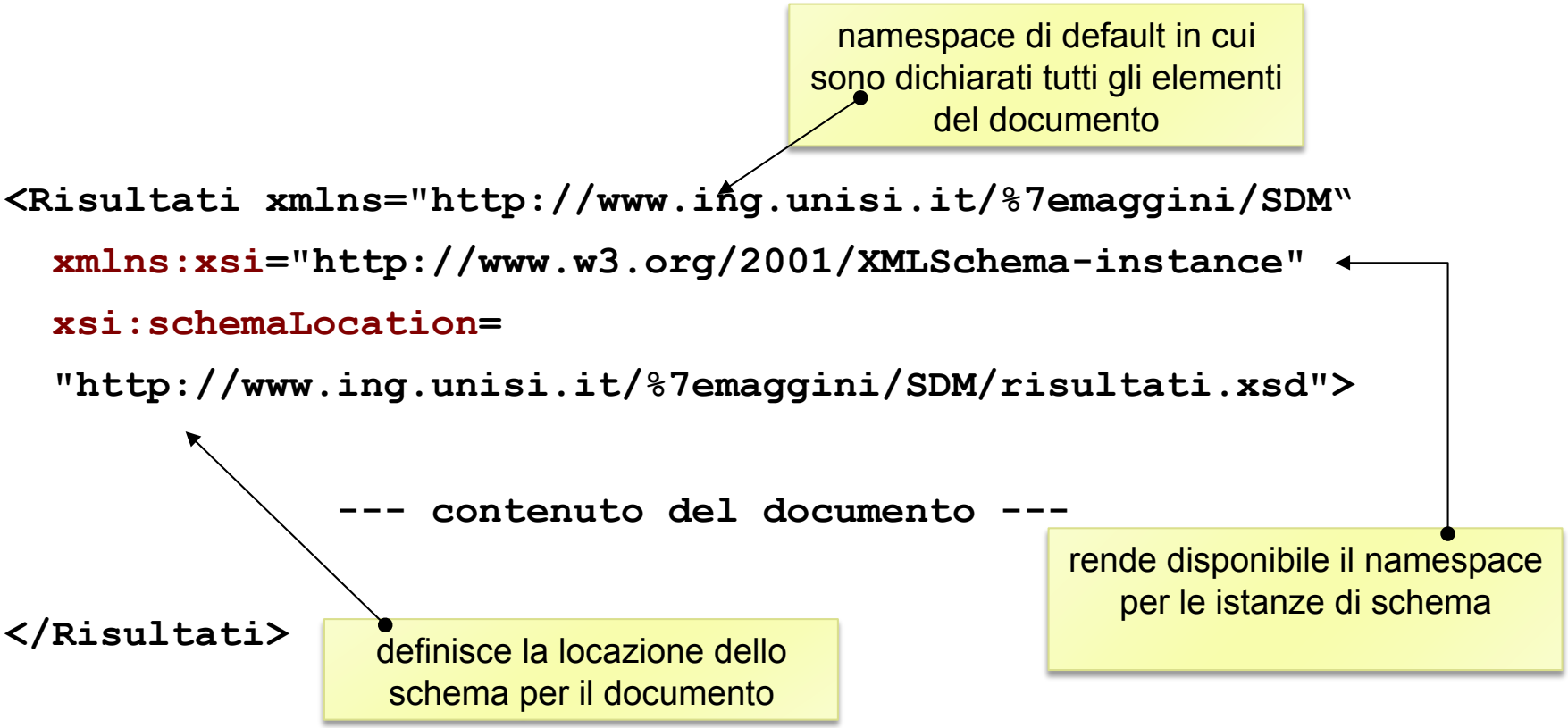
namespace per gli elementi di XML schema

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  targetNamespace="http://www.ing.unisi.it/%7emaggini/SDM"
  xmlns="http://www.ing.unisi.it/%7emaggini/SDM">

                --- Dichiarazioni -


</xsd:schema>
```

namespace di default per gli elementi dell'XML descritto (target namespace)

Come si riferisce uno schema

- Si indica lo schema di riferimento per indicare all'analizzatore dove trovare la definizione dello schema



Ancora in XML schema

- XML Schema definisce molti altri tag e attributi
  - `<xsd:any />` per consentire la presenza di elementi non definiti nello schema
  - `<xsd:anyAttribute />` per estendere uno schema con altri attributi
  - `<xsd:element ref="riferimento" />` per riutilizzare la definizione di un elemento
  - `<xsd:element name="nomeAlternativo" substitutionGroup="nome">` per dare più nomi allo stesso elemento (definire tag equivalenti)
  - definire gruppi di elementi o attributi
  - estendere tipi complessi definiti

Parsing XML (php)

- SAX (Simple API for XML Parsing)
  - Elaborazione come flusso di eventi elemento per elemento (Event-based Parser)
  - Un evento corrisponde all'apertura o chiusura di un tag, alla lettura del contenuto di un elemento, ecc.
  - Si scrive un gestore (funzione handler) per ogni evento

```
bool xml_set_element_handler (resource $parser,
    callable $start_element_handler, callable $end_element_handler)
```

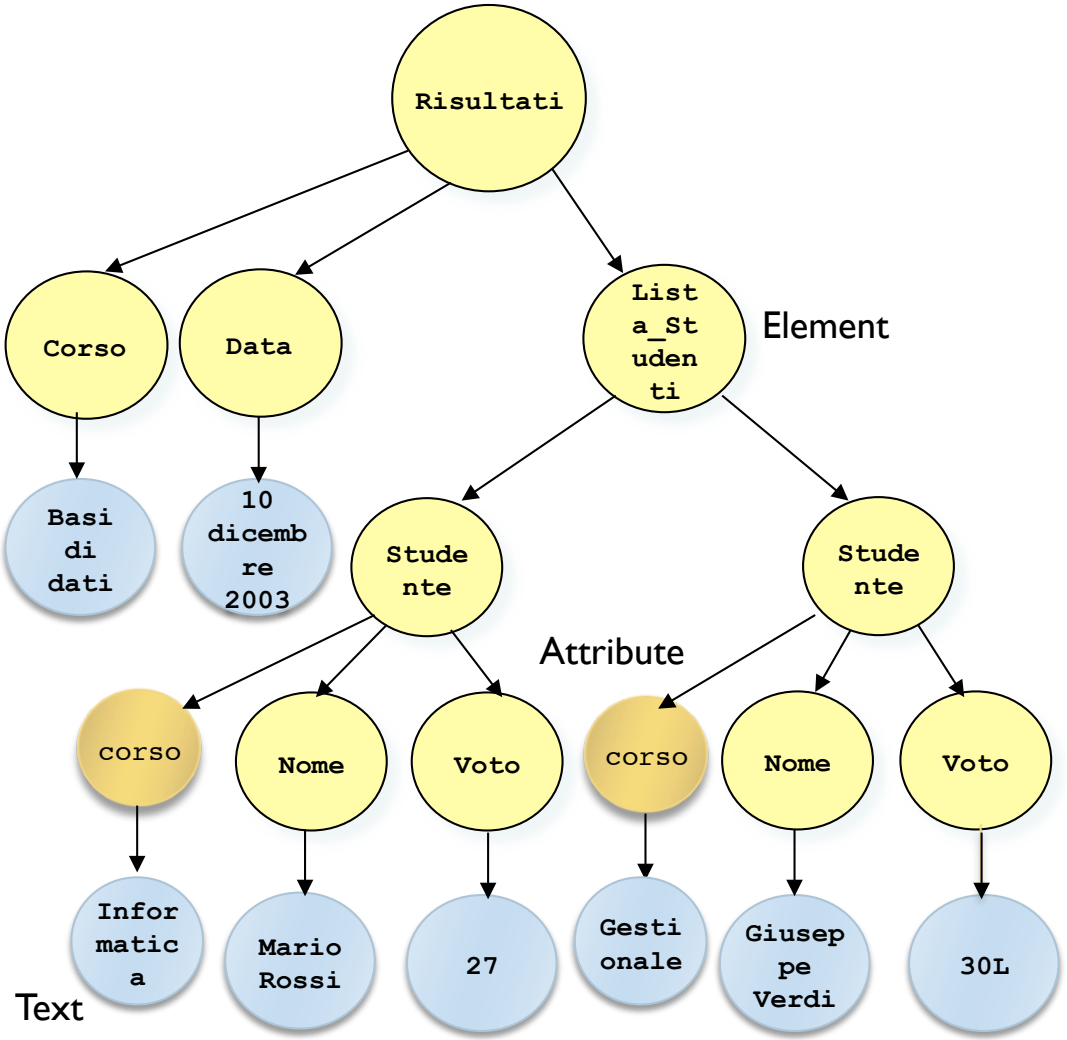
  - Basso uso di risorse e velocità
- DOM (Document Object Model)
  - Costruisce la rappresentazione ad albero (Tree-based parser)
  - Mantiene tutto l'object model in memoria
  - Richiede di leggere tutta la struttura del file XML

Document Object Model

- IL DOM è uno standard del W3C
  - Definisce le modalità di accesso e modifica del contenuto, struttura e stile di un documento strutturato
  - Prevede 3 livelli
    - Core DOM
    - XML DOM
    - HTML DOM
  - Il DOM definisce gli oggetti e le proprietà di tutti gli elementi di un documento e i metodi per accedervi
  - Un documento XML è visto come un insieme di nodi
    - L'intero documento è un *nodo documento*
    - Ogni elemento XML è un *nodo elemento*
    - Il testo contenuto in un elemento è un *nodo testo*
    - Ogni attributo è un *nodo attributo*
    - I commenti sono *nodi commento*

Esempio di DOM XML

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

<Risultati>
  <Corso>Basi di dati</Corso>
  <Data>10 Dicembre 2003</Data>
  <Lista_Studenti>
    <Studente corso="Informatica">
    <Nome>Mario Rossi</Nome>
    <Voto>27</Voto>
    </Studente>
    <Studente corso="gestionale">
    <Nome>Giuseppe Verdi</Nome>
    <Voto>30L</Voto>
    </Studente>
  </Lista_Studenti>
</Risultati>
```



DOM parser (php)

- La libreria DOM implementa un parser tree-based
  - E' una libreria ad oggetti
    - **`DOMDocument`** rappresenta un intero documento XML (o HTML)
      - Si crea un nuovo oggetto DOM con **`$doc = new DOMDocument()`**
      - Si costruisce la rappresentazione DOM di un documento XML leggendolo da un file con **`$doc->load('file.xml')`**
      - La lettura può generare errori, ad esempio se il documento XML non è ben formato
      - Si può validare il documento DOM con un file XML schema con il metodo **`$doc->schemaValidate('file.xsd')`**
      - La gestione dei messaggi di errore può essere quella standard (va attivata la visualizzazione) o utente (**`libxml_use_internal_errors(true);`**)
      - In genere i caratteri di spaziatura (anche \n) fra i tag generano nodi Testo nel DOM; si possono eliminare usando l'opzione **`$xml->preserveWhiteSpace = false;`**
      - E' un caso specifico della classe più generale **`DOMNode`** che rappresenta tutti i possibili nodi che si possono trovare in un DOM

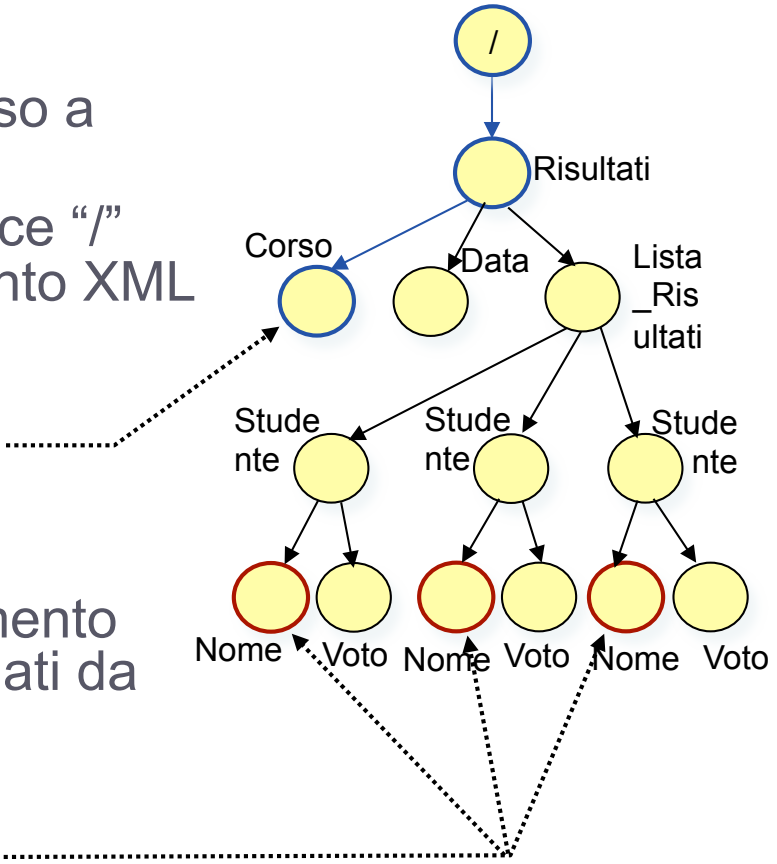
DOMNode

- **`DOMNode`** è un generico nodo del DOM
  - Ha proprietà per accedere ai figli **`DOMNodeList $childNodes`**, agli eventuali attributi **`DOMNamedNodeMap $attributes`**, al nodo padre **`DOMNode $parentNode`**, ecc..
  - Ha memorizzato un codice che ne identifica il tipo **`int $nodeType`** (Element, Text, Attribute, CDATA, Comment,...)
  - Ha metodi per aggiungere **`appendChild(DOMNode $newnode)`**, **`DOMNode insertBefore(DOMNode $newnode [, DOMNode $refnode])`** o eliminare **`removeChild(DOMNode $oldnode)`** figli
  - Le sottoclassi hanno metodi o proprietà specifiche
    - per **`DOMElement`** ad esempio
      - La proprietà **`$node->tagName`** è la stringa del nome del tag
      - Il metodo **`setAttribute(string $name , string $value)`** permette di associare un valore ad un attributo del nodo (lo crea se non esiste)

XPath

- Permette di individuare sezioni di un documento XML
  - I nodi sono individuati col percorso a partire dal nodo corrente
  - Un path assoluto parte dalla radice "/" che rappresenta l'intero documento XML

```
/Risultati/Corso
```



  - Tutti i nodi di un dato tipo di elemento presenti nell'albero sono individuati da

```
//Nome
```

Selezione di nodi

- Selezione in base al tag
  - **`Studente`** – seleziona i nodi figli del nodo Studente (se presente nel contesto corrente)
  - **`/Risultati`** - seleziona dal nodo radice
  - **`//Nome`** – seleziona tutti i nodi del tipo specificato presenti nel sottoalbero a partire dal nodo corrente
  - **`.`** – seleziona il nodo corrente
  - **`..`** – seleziona il padre del nodo corrente
- Selezione con predicati
  - Si può specificare una condizione di selezione fra [*condizione*]
    - **`/Risultati/Lista_Studenti/Studente[1]`** – il primo elemento Studente
    - **`/Risultati/Lista_Studenti/Studente[last()]`** – l'ultimo elemento Studente
    - **`/Risultati/Lista_Studenti/Studente[position()<3]`** – i primi due elementi Studente
    - **`/Risultati/Lista_Studenti/Studente[Voto>25]`** – gli elementi Studente che contengono un elemento Voto con contenuto maggiore di 25
    - Le condizioni atomiche possono essere combinate con **`or`** o **`and`**

XPath: attributi e selezione condizionale

- E' possibile selezionare gli attributi

```
//@colore
```

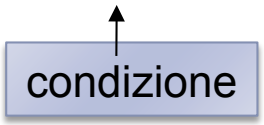
  - seleziona qualsiasi attributo di nome **colore**

```
@colore
```

  - seleziona l'attributo **colore** del nodo corrente

- Si possono definire delle condizioni di match

```
prodotto[@disponibile]
prodotto[@colore='rosso']
```



Operatori e Funzioni XPath

- E' possibile usare le wildcard
  - `*` - ogni elemento
  - `@*` - ogni proprietà
  - `node()` - un qualsiasi nodo
- E' possibile unire più path con l'operatore |
- XPath definisce una serie di funzioni (~100) che si applicano a un nodo o a un insieme di nodi
  - **`name()`** ottiene il nome di un nodo
  - **`text()`** ottiene il testo contenuto in un nodo
  - **`position()`** fornisce la posizione di un nodo
  - **`count("elemento")`** – il numero di elementi col nome specificato
  - **`sum("elemento")`** - somma i valori di tutti gli elementi di un insieme
  - **`number()`** converte un testo in un numero

Espressioni per i path

- Un path può essere
  - Assoluto `/step/step/.../step`
    - inzia con / e specifica il cammino a partire dalla radice
  - Relativo step/step/.../step
    - non ha / all'inizio e specifica il cammino a partire dal nodo corrente

- Ogni elemento del path ha la struttura

```
axis::nodename[condition]
```

  - **`axis`** (opzionale) specifica la relazione dei nodi di interesse rispetto alla posizione corrente (ancestor, child, descendant, parent,...)
  - **`nodename`** specifica i nodi di interesse nell'insieme specificato da axis
  - **`condition`** (opzionale) fornisce un ulteriore criterio di selezione

DOMXpath (php)

- La classe **`DOMXpath`** permette di valutare un'espressione Xpath su un documento DOM
  - Si crea un oggetto Xpath associato a un DOM

```
$xpath = new DOMXPath($doc);
```

  - Si esegue un'interrogazione Xpath (eventualmente relativa a un sottoalbero con radice `$context` del DOM)

```
$result = $xpath->evaluate($query [, $context]);
```

  - Il risultato può essere una lista di nodi o un valore
    - **`$query = "//Studente[Voto>28]"`**
      nodi Studente con l'elemento Voto contenente un valore maggiore di 28
    - **`$query = "count(//Studente[Voto>28])"`**
      il numero di nodi Studente con l'elemento Voto > 28