Extensible Markup Language

XML

XML

- È un linguaggio di markup sviluppato dallo XML Working Group del W3C a partire dal 1996
- XML 1.0 è una raccomandazione del W3C dal febbraio 1998
- Nasce dall'esigenza di avere un meccanismo per la rappresentazione testuale di informazione strutturata o semi-strutturata

XML

- XML descrive i dati e non la loro rappresentazione
- Ha un formato aperto e leggibile, visivamente simile al linguaggio HTML
- Come HTML, deriva da SGML (Standard Generalized Markup Language, ISO standard dal 1986)

XML

" … XML is one of the most important development in the history of computing. In the last few years it has been adopted in fields as diverse as law, aeronautics, finance, insurance, robotics, multimedia, hospitality, art, software design, physics, literature, …

XML has become the syntax of choice for newly designed document formats across almost all computer applications …"

XML: Document- vs Data-Centric

- Esistono due classi di applicazioni nell'area delle tecnologie XML
- Document-centric
  XML fornisce un meccanismo per rappresentare documenti semi-strutturati (ad esempio, manuali tecnici, documenti legali, cataloghi di prodotti)

XML: Document- vs Data-Centric

- Data-centric
  XML fornisce un meccanismo per rappresentare dati fortemente strutturati: i record di un database relazionale o informazioni relative ad una transazione finanziaria
- I documenti di questo tipo sono "prodotti" e "consumati" - molto spesso on the fly - tramite appositi software

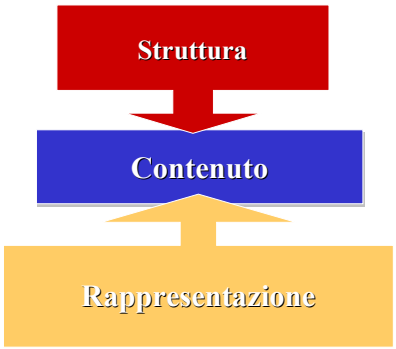
XML: Document- vs Data-Centric

- In entrambi i casi, tra gli obiettivi di queste applicazioni ricordiamo
  - ✓compatibilità con applicazioni diverse (interoperabilità)
  - ✓facilità di creazione ed elaborazione dei documenti
  - ✓necessità di disporre di un linguaggio per descrivere e strutturare i dati

XML

- Un documento XML viene interpretato da un'applicazione formata da due parti
  - ✓una che effettua il controllo sintattico del documento (parser)
  - ✓una che si occupa di visualizzare o trasformare il documento (processor)

XML: componenti di un documento



XML: contenuto

- Un documento XML è un documento di testo che ha un'estensione .xml
- È formato da un prologo, seguito da un elemento radice che contiene il resto del documento
- Il prologo serve per
  - ✓identificare il documento come un documento XML
  - ✓includere eventuali commenti e meta-informazioni sul documento

XML: contenuto

- Prologo: processing instruction

  <?xml ..... ?>

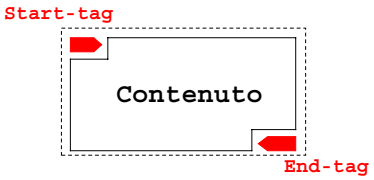
- Esempio:
  <?xml version="1.0" encoding="UTF-8" ?>
- Lo standard UTF-8 genera caratteri "7-bit safe" e rende facile lo scambio di documenti XML usando protocolli standard quali HTTP, SMTP, FTP
- XML supporta altre codifiche, tra cui UNICODE e ISO/IEC 10646 (Universal Multiple-Octet Coded Character Set)

XML: contenuto

- XML permette all'utente di definire il proprio insieme di tag (elementi)
- I tag definiti dall'utente possono avere dei nomi che rispecchiano il contenuto del documento stesso

XML: elementi

- Un elemento è un blocco elementare



Esempio: <author>Mario Rossi</author>

NB: XML è case sensitive

XML: contenuto di un elemento

- Altri elementi (sub-elements)

```
<indirizzo>
     <via>Via Po, 15</via>
     <citta>Torino</citta>
</indirizzo>
```

- Testo (data content)

```
<via>Via Po, 15</via>
```

- Contenuto misto (mixed content)

```
<par>Oggi, <date>09-12-2004</date>
<name>Bill Gates</name> è a Genova per
presentare … </par>
```


XML: esempio

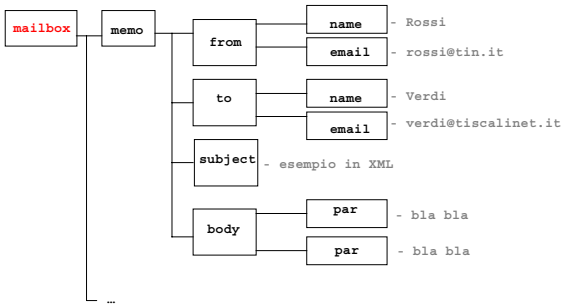
```
<?xml version="1.0" …  ?>
<mailbox>
<memo>
        <from>
                <name>Rossi</name>
                <email>rossi@tin.it</email>
        </from>
        <to>
                <name>Verdi</name>
                <email>verdi@tiscalinet.it</email>
        </to>
        <subject>Esempio in XML</subject>
        <body>
                <par>bla bla</par>
                <par>bla bla</par>
        </body>
</memo>
……
</mailbox>
```

elemento radice

XML: struttura ad albero

- Un documento XML definisce una struttura ad albero che si ottiene guardando le relazioni di annidamento che esistono tra i tag
- Esiste un solo tag che non sta all'interno di nessun altro: l'elemento radice

XML: struttura ad albero



XML: attributi

- Anche i tag XML possono avere degli attributi
- Un attributo è una coppia nome="valore" che viene associata al tag iniziale di un elemento

```
<persona altezza="170" peso="60">
   Mario Rossi
</persona>
```

NB: non esiste un modo univoco per decidere cosa deve essere elemento e cosa deve essere attributo

XML: elementi vs attributi

- **Elemento**, quando:
  - ✓si richiede di recuperare i dati velocemente
  - ✓è rilevante per il significato del documento
- **Attributo**, quando:
  - ✓esprime una scelta
  - ✓non è rilevante per il significato del documento

XML: esempio (spesa CLICK)

XML: esempio (spesa CLICK)

```
<?xml version="1.0" encoding="UTF-8" ?>
<orderonline>
<order num="234" date="2004-12-09">
  <item idprod="1" quantity="10" price="2">
      <nameprod>Post-It Notes and Dispenser</nameprod>
      <description>Available in a wide variety of colours,
      sizes and formats to help get the job done …
      </description>
  </item>
  <item idprod="2" quantity ="10" price ="3">
      <nameprod>Post-it Index Dispensers</nameprod>
      <description>Our repositionable tabs help you mark up
      pages, colour code subject categories …
      </description>
  </item>
  …
</orderonline>
```


XML: esempio (spesa CLICK)

```
<?xml version="1.0" encoding="UTF-8" ?>
<orderonline>
<order num="234" date="2004-12-09">
  <client>
      <name>Mario</name>
      <surname>Rossi</surname>
      <address>
            <street> … </street>
            <city> … </city>
            <postalcode> … </postalcode>
      </address>
  </client>
  <item>
  <!--  info prodotti qui -->
  </item>
</order>
</orderonline>
```

File XML (minimale) per descrivere un ordine

XML: documenti ben formati

- Un documento XML è ben formato (well formed) se
  - ✓tutti i suoi tag sono chiusi, nell'ordine corretto
  - ✓esiste un solo elemento radice
  - ✓i valori degli attributi sono scritti tra virgolette
  - ✓gli elementi non hanno due attributi con lo stesso nome
  - ✓i nomi degli elementi e degli attributi non contengono i caratteri < > &

XML: Namespaces

- Una proprietà importante dei documenti XML è che possono essere composti per creare nuovi documenti
- Purtroppo la composizione crea problemi di riconoscimento e collisione

XML: esempio (spesa CLICK)

Scenario:
spesaClick vuole ricevere gli ordini attraverso un noto XML messaging system

```
<message from="…" to="…" sent="…">
<text><!-- testo del messaggio --></text>

<!-- un messaggio può avere uno o più attachment -->
  <attachment>
     <description> … </description>
     <item> … </item>
  </attachment>
</message>
```

spesaClick, ordine on line

XML: esempio (spesa CLICK)

```
…
…
<attachment>
 <description>
  spesaClick, ordine on line
 </description>
 <item> … </item>
</attachment>
```


XML: esempio (spesa CLICK)

```
…
…
<attachment>
 <description
  spesaClick, or
 </descriptio
 <item> … </i
</attachment>
```

```
<orderonline>
<order num="234" date="2004-12-09">
 <client>
  <name>Mario</name>
  <surname>Rossi</surname>
  <address>
   <street> … </street>
   <city> … </city>
   <postalcode> … </postalcode>
  </address>
 </client>
 <item> … </item>
 <item> … </item>
</order>
…
```

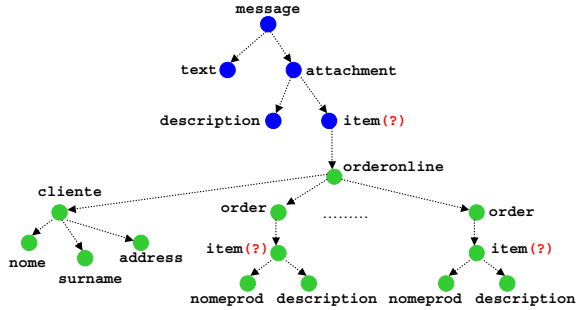

XML: esempio (spesa CLICK)

```
…
…
<attachment>
 <descriptic
  spesaClick,
 </descripti
 <item> … </
</attachment>
```

Problema!

```
<orderonline>
<order num="234" date="2004-12-093">
 <client> … </client>
 <item idprod="1" quantity="10" price="2">
  <nameprod>
     Post-It Notes and Dispenser
  </nameprod>
  <description>
     Available in a wide variety of colours, sizes and formats to help get the job done …
  </description>
 </item>
…
</order>
…
```


XML: esempio (spesa CLICK)



XML: Namespaces

- **Riconoscimento**
  Come fa un'applicazione XML a distinguere tra gli elementi XML che descrivono il messaggio e quelli che sono parte dell'ordine?

- **Collisione**
  Come fa un'applicazione XML a capire a cosa si riferiscono i tag con lo stesso nome?

XML: Namespaces (dal W3C)

We envision applications of Extensible Markup Language (XML) where a single XML document may contain elements and attributes that are defined for and used by multiple software modules. One motivation for this is modularity; if such a markup vocabulary exists which is well-understood and for which there is useful software available, it is better to re-use this markup rather than re-invent it.

Such documents, containing multiple markup vocabularies, pose problems of recognition and collision. Software modules need to be able to recognize the tags and attributes which they are designed to process, even in the face of "collisions" occurring when markup intended for some other software package uses the same element type or attribute name.

These considerations require that document constructs should have universal names, whose scope extends beyond their containing document. This specification describes a mechanism, XML namespaces, which accomplishes this.

XML: Namespaces

- Si introducono i **nomi qualificati**

  Qualified name = Namespace prefix + Local part

- Per costruire un namespace identifier si usano degli **URI** (Uniform Resource Identifier) [RFC 2396]

XML: Namespaces

- Si procede in due passi
  - ✓ Si associa un prefisso (myPrefix) ad ogni namespace identifier
  - ✓ Si definiscono i nomi qualificati che hanno la forma

    myPrefix:myElementName

XML: Namespaces

```
<msg:message from="…" to="…" sent="…"
  xmlns:msg="http://www.msgexchange.com/ns/msg"
  xmlns:ord="http://www.spesaclick.com/ns/order"
>
<msg:text>Ecco il mio ordine, grazie Rossi</text>
<msg:attachment>
      <msg:description>
            spesaClick, ordine on line
      </description>
      <msg:item> … </item>
  </attachment>
</message>
…
```

nome qualificato

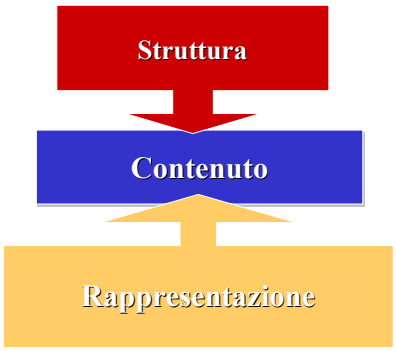
XML: Namespaces

```
<ord:order num="234" date="2004-12-09">
  <ord:item idprod="1" quantity="10" price="2">
      <ord:nameprod>Post-It Notes and Dispenser</nameprod>
      <ord:description>Available in a wide variety of
      colours, sizes and formats …</description>
  </item>

  <ord:item idprod="2" quantity ="10" price ="3">
      …
  </item>
   …
</order>
```

nome qualificato

XML: componenti di un documento



XML: struttura

- Si può specificare in modo formale la struttura di un document XML definendo una Dichiarazione di Tipo di Documento (DTD)

> "DTD offered the basic mechanism for defining a vocabulary specifying the structure of XML documents in attempt to establish a contract between multiple parties working with the same type of XML."

XML: struttura

- Grazie all'uso di una DTD si può fare un "check" sulla corretta strutturazione di un documento XML
- Un documento XML è detto valido se è conforme a quanto specificato nella sua DTD. La validità è opzionale
- Anche per HTML è stata definita in modo formale una DTD cui si "attengono" i produttori di browser

XML: DTD

- Utile per i programmatori: definisce il tipo di documento che andranno a processare
- Utile per definire i fogli di stile
- Utile per creare documenti "corretti"
  La DTD può essere vista come un vincolo sull'informazione da inserire nel documento

XML: DTD

- Le regole per definire una DTD
  - ✓stabiliscono gli elementi che possono essere usati
  - ✓stabiliscono gli attributi da inserire negli elementi
  - ✓impongono vincoli sulle relazioni tra gli elementi (fratelli, elemento-sottoelemento,...)

XML: DTD

- Un esempio di DTD orientata alla narrazione è la Text Encoding Initiative (TEI), un'applicazione XML per il markup della letteratura classica
- DocBook è un'applicazione XML progettata per descrivere documenti di tipo tecnico
- DublinCore ha come scopo la descrizione di libri e altre opere letterarie

XML: DTD

- Element A
- Sequence (A,B,C)
- Choice (A|B|C)
- Multiplicity
  - ✓ ? elemento opzionale
  - ✓ + elemento obbligatorio e ripetibile
  - ✓ * elemento opzionale e ripetibile

DTD: dichiarazione elemento

- La dichiarazione di un elemento inizia con <!ELEMENT seguito dall'identificatore dell'elemento, seguito da una sua specifica

```
<!ELEMENT persone (generalita,professione*)>
<!ELEMENT generalita (nome,cognome)>
<!ELEMENT nome (#PCDATA)>
<!ELEMENT cognome (#PCDATA)>
<!ELEMENT professione (#PCDATA)>
```

NB: PCDATA (Parsed Character DATA)

DTD: elemento vuoto

- Esistono degli elementi particolari che non hanno nessun contenuto
- Possono essere scritti in due modi

  <nometag/>

  <nometag></nometag>
- Vengono dichiarati usando la parola chiave EMPTY

  <!ELEMENT nome_elemento EMPTY>

DTD: attributi

- Consentono di associare informazioni aggiuntive agli elementi
- La dichiarazione di un elenco di attributi inizia con <!ATTLIST

```
<!ELEMENT recapito (#PCDATA)>
<!ATTLIST recapito
      email CDATA #REQUIRED
      tel   CDATA #REQUIRED
      fax   CDATA #IMPLIED>
```

CDATA = character data

DTD: entità

- Sono delle variabili che fanno riferimento a porzioni di testo
- Ne esistono di predefinite per specificare caratteri che non si possono usare come valori di elementi

&lt;	<
&amp;	&
&gt;	>
&quot;	"
&apos;	'

DTD: entità

```
<!ENTITY nome "valore">
```

```
<!ENTITY sede "spesa CLICK,
     sede legale: Torino via Po 1">
```

Nel documento si potrà usare &sede; che verrà sostituito dal valore corrispondente

DTD: dove si scrive?

- All'interno di un file XML oppure in un file esterno
- Se la DTD è inclusa nel file XML, nel prologo si avrà

  <?xml version="1.0" standalone="yes" ?>
- Altrimenti si scrive

  <?xml version="1.0" standalone="no" ?>

DTD: embedded

```
<?xml version="1.0" standalone= "yes"  ?>
  <!DOCTYPE persone [
  <!ELEMENT persone (generalita,professione*)>
      <!ELEMENT generalita (nome,cognome)>
      <!ELEMENT nome (#PCDATA)>
      <!ELEMENT cognome (#PCDATA)>
      <!ELEMENT professione (#PCDATA)>
  ]>
<persone>
  …
</persone>
```

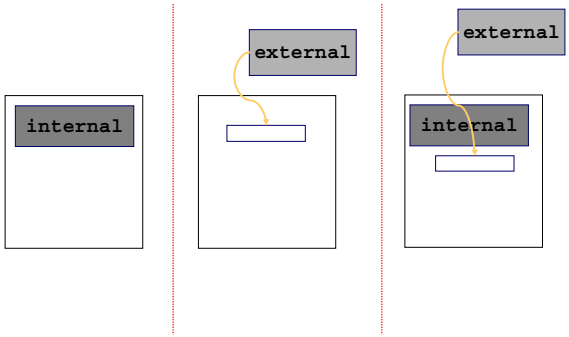
Esempio di file XML con DTD embedded

DTD: external

- Si scrive un file esterno con estensione .dtd che viene incluso nel file .xml

```
<?xml version="1.0" standalone="no" ?>
<!DOCTYPE persone
            SYSTEM "persone.dtd">
```


DTD: dove si scrive?



DTD: problemi

- La sintassi per costruire una DTD non è basata su XML
- Le DTD non facilitano la riusabilità
- Non sono pensate per gestire i namespace
- Non si può imporre un vincolo sul numero di elementi di un certo tipo
- Le DTD non hanno una nozione di tipo di dato e non permettono di esprimere semplici regole come ad esempio "il valore di questo attributo deve essere intero positivo"