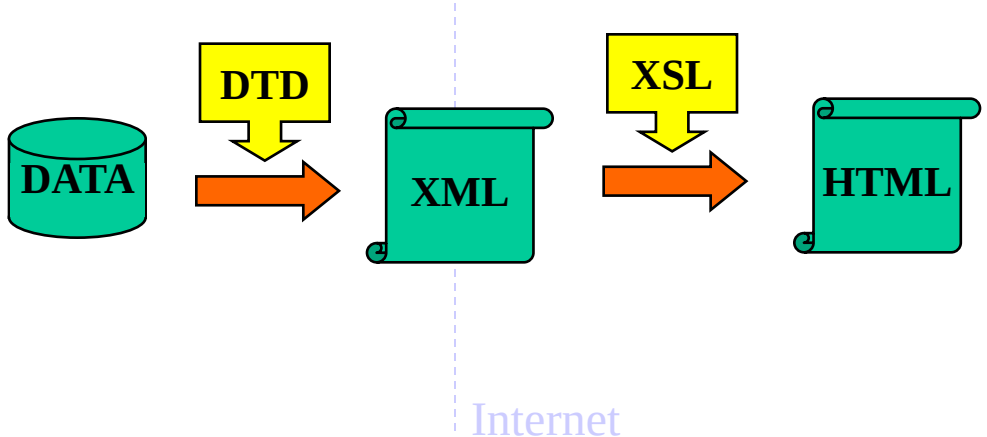
XML

(eXtensible Markup Language)

XML: eXtensible Markup Language

L' **eXtensible Markup Language** è lo standard adottato dal W3C per lo ***scambio di dati sul web*** e con lo scopo di fungere da complemento a HTML come linguaggio per la pubblicazione di dati sul web.

XML è stato progettato per descrivere il contenuto e non lo stile di presentazione dei dati.



HTML vs XML

Lo standard definito per i linguaggi di markup è ***SGML*** (***S***tandard ***G***eneralized ***M***arkup ***L***anguage). SGML è estensibile, permette cioè di generare linguaggi più specifici per lo scambio informazioni.

HTML è un'applicazione di SGML ottimizzata per la presentazione di dati sul web

- Non permette la creazione di nuovi tag (non è estensibile).
- Contiene tag per che definiscono la modalità di presentazione dei dati.
- Non definisce una struttura dei dati che contiene.

XML è un "dialetto" di SGML, e può quindi essere "personalizzato" per descrivere particolari classi di informazioni.

- Consente di descrivere la grammatica che definisce i dati.
- Permette la creazione di nuovi tag (è estensibile).

XML concetti di base I

Il concetto fondamentale di XML è l'***elemento***: il testo compreso tra una coppia di ***tag*** corrispondenti. All'interno di un elemento si può trovare del testo e/o altri elementi.

```
<person>
    Ogni persona è descritta da:
    <name>Alan</name>
    <age>42</age>
    <email>alan@abc.com</email>
</person>
```

Ogni elemento deve essere delimitato da un ***tag di apertura*** e da un ***tag di chiusura***. L'unica abbreviazione possibile è quella che identifica un ***tag vuoto***: `<married></married>` può essere scritto come: `<married/>`

Un documento XML è ***ben formato*** se c'è un'esatta corrispondenza tra ogni coppia di tag.

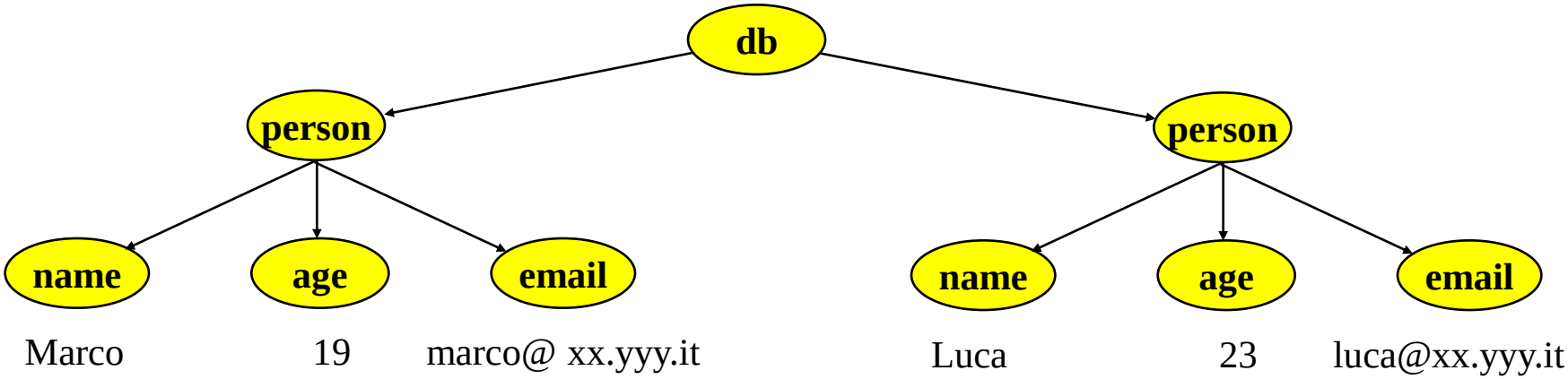
Si noti che in XML:

- L'unico tipo di base è il testo (PCDATA).
- I dati si considerano ordinati.

Il modello dei dati XML

Nella loro forma elementare i dati XML sono rappresentabili mediante rappresentazione ad albero in cui:

- Ogni nodo rappresenta un elemento.
- Ogni arco indica un rapporto di inclusione.
- A ogni foglia è associato un dato vero e proprio.



N.B. XML permette, per mezzo degli attributi, la rappresentazione di stutture a grafo.

XML concetti di base II

XML permette di associare ***attributi*** agli elementi. Un attributo rappresenta una proprietà associata all'elemento definita da un'accoppiata nome=valore.

```
<product>
   <name language="French">Croissant</name>
   <price euro="1.15" />
   <address format="XLB56" language="French">
      <street>31, rue Croix-Bosset</street> <zip>92310</zip>
      <phone>001/997755</phone>
      <phone>333/224466</phone>
   </address>
</product>
```

- Come per gli elementi anche gli attributi sono definibili dall'utente.
- La differenza tra elementi e attributi è che questi ultimi possono presentarsi solo una volta all'interno di ogni tag, mentre un sub-elemento può essere ripetuto più volte all'interno di un elemento.

XML concetti di base III

ATTENZIONE!! Gli attributi introducono ambiguità nella rappresentazione dei dati

```
<person>
    <name>Alan</name>
    <age>42</age>
    <email>alan@abc.com</email>
</person>
```

```
<person name”Alan” age=“42”
        email=“alan@abc.com”/>
```

```
<person age=“42”>
    <name>Alan</name>
    <email>alan@abc.com</email>
</person>
```

Gli attributi permettono di rappresentare strutture dati a grafo mediante attributi particolari che hanno il ruolo di identificatori (**id**) e di referenze (**idref**, **idrefs**).

```
<state id=“s2”>
    <scode>NE</scode>
    <sname>Nevada</sname>
</state>
```

```
<city id=“c2”>
    <ccode>CCN</ccode>
    <cname>Carson city</cname>
    <state-of idref=“s2”/>
</city>
```

XML concetti di base IV

Un documento XML può contenere un **Prologo**, che contiene informazioni utili per l'interpretazione del documento.

In particolare, esso può contenere:

- Una dichiarazione che il documento è in formato xml (opzionale).
- Un riferimento a una *grammatica* (es. DTD o XML-Schema) che permette di validare il contenuto del documento (opzionale).
- Commenti e informazioni per applicazioni software che utilizzeranno il documento (Processing Instructions, o PI) (zero o più).

```
<?xml version="1.0"?>
<!DOCTYPE doc SYSTEM "myDoc.dtd">
<!-- Istruzione per applicare un foglio di stile -->
<?xml-stylesheet type="text/css" href="style.css"?>
```

La grammatica di un documento XML I

Il **Document Type Definition** specifica la grammatica/schema utilizzato all'interno di un documento XML. Nel DTD viene specificata la struttura del documento:

```
<!DOCTYPE db [
   <!ELEMENT db (person*)>
   <!ELEMENT person (name,age,email)>
   <!ELEMENT name (#PCDATA)>
   <!ELEMENT age (#PCDATA)>
   <!ELEMENT email (#PCDATA)>
]>
```

Un'espressione regolare definisce il possibile contenuto di ogni elemento. I concetti esprimibili sono i seguenti:

- 0 o più istanze: `e*`
- 0 o 1 istanza: `e?`
- Sequenza di istanze: `e,e'`
- 1 o più istanze: `e+`
- Istanze alternative: `e|e'`
- Solo testo: `#PCDATA`
- Elemento vuoto: `#EMPTY`

La grammatica di un documento XML II

Nel **DTD** vengono definiti anche gli attributi:

```
....
<!ELEMENT name (#PCDATA)>
<!ATTLIST name  language CDATA  #REQUIRED
                department CDATA #IMPLIED>
```

Un ruolo particolare hanno gli attributi definiti come:

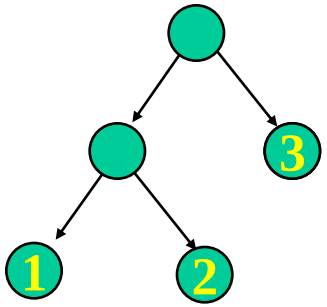
- ID: identificatore
- IDREF: referenza
- IDREFS: referenza multipla

```
....
<!ELEMENT family (person)*>
<!ELEMENT person (name)>
<!ELEMENT name   (#PCDATA)>
<!ATTLIST person id       ID      #REQUIRED
                 mother   IDREF   #IMPLIED
                 father   IDREF   #IMPLIED
                 children IDREFS  #IMPLIED>
```

La grammatica di un documento XML III

Un DTD è una grammatica context-free che definisce il documento. La grammatica può anche essere ricorsiva come nel caso del seguente in cui viene descritto un albero binario:

```
<!DOCTYPE tree [
   <!ELEMENT tree (node?)>
   <!ELEMENT node (leaf | (node,node))>
   <!ELEMENT leaf (#PCDATA)>
]>

<tree>

   <node>
       <node>
           <node><leaf> 1 </leaf></node>
           <node><leaf> 2 </leaf></node>
       </node>
       <node>
           <leaf> 3 </leaf>
       </node>
   </node>

</tree>
```



Un documento XML è ***valido*** se è conforme al relativo DTD, ossia se nel documento:

- appaiono solo e tutti gli elementi richiesti dal DTD.
- l'ordine specificato è rispettato.

Interrogazioni nei documenti XML

I dati XML possono essere considerati una fonte di informazione interrogabile (dati semi-strutturati).

Porzioni di contenuto possono essere individuate all'interno di un documento XML tramite un cammino (path).

Il linguaggio XPath (standard W3C), permette a tale scopo la scrittura di path expressions.

XPath è utilizzato come "sottolinguaggio" dei linguaggi di interrogazione e manipolazione per XML veri e propri (quali XSL e XQuery, DOM)

Il linguaggio XPath

Una path expression XPath specifica il tipo e la posizione degli elementi ricercati:

`*`	qualsiasi elemento
`/`	la root (un cammino che inizia con `/` è detto assoluto)
`//`	un qualunque punto di partenza nel documento
`el1`	un elemento di tipo `el1`
`/el1`	un elemento di tipo `el1` che segua la radice
`//el2`	un elemento di tipo `el2` in qualsiasi punto
`el1/el2`	un elemento di tipo `el2` che segua elemento di tipo `el1`
`el1//el2`	un elemento di tipo `el2` posto in ogni punto all'interno di un elemento di tipo `el1`
`el1 | el2`	un elemento di tipo `el1` o un elemento di tipo `el2`
`el1[el2]`	un elemento di tipo `el1` che contenga un elemento di tipo `el2`
`@at1`	un attributo di tipo `at1`
`el1[@at1]`	un elemento di tipo `el1` che abbia un attributo di tipo `at1`

`el1[n]`	l'n-simo elemento di tipo `el1` (0 per il primo)
`el1[last()]`	l'ultimo elemento di tipo `el1`
`el1[@at1="val"]`	un elemento di tipo `el1` che abbia un attributo di tipo `at1` il cui valore sia "val"
`el1[text()="testo"]`	un elemento di tipo `el1` il cui contenuto sia "testo"

Esempi di espressioni XPath

`/libro/parte/capitolo`
seleziona tutti gli elementi `capitolo` che si raggiungono dalla radice seguendo un percorso `libro` e poi `parte`

`//capitolo`
seleziona tutti gli elementi `capitolo` ovunque si trovino nel documento

`/libro/*`
seleziona tutti gli elementi contenuti nell'elemento `libro` (figlio della radice)

`/libro/*/capitolo`
seleziona tutti gli elementi `capitolo` figli di un qualunque figlio di `libro`

`//parte[capitolo]`
seleziona (ovunque si trovino) gli elementi `parte`
che hanno un elemento `capitolo` come figlio (ossia contengono un tale elemento)

`//parte[@numero='2']`
seleziona (ovunque si trovino) gli elementi `parte`
che hanno `2` come valore dell'attributo `numero`

`//parte[@numero='2']/capitolo`
seleziona gli elementi `capitolo` che sono figli di elementi `parte` (ovunque si trovino) che hanno `2` come valore dell'attributo `numero`

Interrogazioni nei documenti XML

XSL (**eXtensible Stylesheet Language**) è stato proposto da W3C come linguaggio estendibile per le trasformazioni di formato di documenti XSL. L'utilizzo più immediato (ma non l'unico possibile) è nella trasformazione di documenti XML in pagine HTML.

Un programma XSL è formato da un insieme di regole ricorsive da applicare a un documento XML:

- Ogni regola è composta da un ***pattern*** e da un ***modello***. Il modello definisce la struttura dei documenti generati.
- Il sistema, partendo dalla radice del documento XML ricerca la regola il cui pattern coincide col nodo corrente.
- Se viene trovata una corrispondenza il sistema esegue il modello corrispondente.

I programmi XSL possono estrarre anche sotto-porzioni di documenti XML e per questo motivo XSL può essere considerato un linguaggio di interrogazione per XML.

Pattern e modelli XSL

Un ***pattern*** XSL è una path expression (XPath) che specifica il tipo e la posizione degli elementi ricercati.

Un ***modello*** XSL è composto dalle porzioni di testo che devono essere generate e da istruzioni XSL (XSLT):

`<xsl:template match="//el">`	prova a determinare il match tra il nodo corrente e il pattern "//el"
`<xsl:apply-templates/>`	applica i pattern al nodo corrente
`<xsl:value-of/>`	inserisce nel risultato il valore dell'elemento corrente
`<xsl:value-of select="el"/>`	inserisce nel risultato il valore dell'elemento "el" contenuto nell'elemento corrente
`<xsl:for-each select="el"/>`	seleziona tutti gli element "el" contenuti nell'elemento corrente
`<xsl:if test="cond"/>`	applica il proprio contenuto se l'espressione "cond" applicata nell'elemento corrente risulta vera
`<xsl:choose>` con `<xsl:when test="cond">` e `<xsl:otherwise>`	permette di esprimere test condizionali multipli

Interrogazioni nei documenti XML (1)

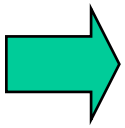
```
<bib>
   <book>
     <title> t1 </title>
     <author> a1 </author>
     <author> a2 </author>
   </book>
   <paper>
     <title> t2 </title>
     <author> a3 </author>
     <author> a4 </author>
   </paper>
   <book>
     <title> t3 </title>
     <author> a5 </author>
     <author> a6 </author>
     <author> a7 </author>
   </book>
</bib>
```

```
<xsl:template match="/">
   <xsl:apply-templates/>
</xsl: template>

<xsl: template match="bib/*">
   <result>
     <xsl:value-of select="title"/>
   </result>
</xsl:template>
```

1) Il programma viene applicato alla root / e tenta di trovare una corrispondenza coi pattern

2) Il modello del pattern trovato bib/* indica di applicare il programma al contenuto del nodo di tutti gli elementi <book>... </book> e tutti gli elementi <paper>... </paper>

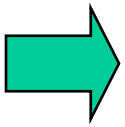
5) Il relativo modello indica di creare un nuovo elemento <result> e di inserirvi il valore contenuto in <title>...</title>.



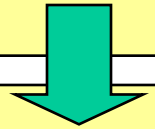
```
<result> t1 </result>
<result> t2 </result>
<result> t3 </result>
```

Interrogazioni nei documenti XML (2)

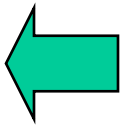
```
<bib>
   <book>
     <title> t1 </title>
     <author> a1 </author>
     <author> a2 </author>
   </book>
   <paper>
     <title> t2 </title>
     <author> a3 </author>
     <author> a4 </author>
   </paper>
   <book>
     <title> t3 </title>
     <author> a5 </author>
     <author> a6 </author>
     <author> a7 </author>
   </book>
</bib>
```



```
<xsl:template match="/">
<html>
 <body>
   <h2>BOOKS</h2>
   <table border="1">
     <tr bgcolor="red">
       <th>Title</th> <th>1st Author</th>
     </tr>
     <xsl:for-each select="bib/book">
       <tr>
         <td><xsl:value-of select="title"/></td>
         <td><xsl:value-of select="author"/></td>
       </tr>
     </xsl:for-each>
     </table>
 </body>
</html>
</xsl:template>
```



```
<tr>
   <td>t1</td>
   <td>a1</td>
</tr>
<tr>
   <td>t3</td>
   <td>a5</td>
</tr>
```

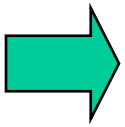


BOOKS

Title	1st Author
t1	a1
t3	a5

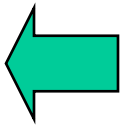
Interrogazioni nei documenti XML (3)

```
<bib>
   <book>
     <title> t1 </title>
     <author> a1 </author>
     <author> a2 </author>
   </book>
   <paper>
     <title> t2 </title>
     <author> a3 </author>
     <author> a4 </author>
   </paper>
   <book>
     <title> t3 </title>
     <author> a5 </author>
     <author> a6 </author>
     <author> a7 </author>
   </book>
</bib>
```



```
<tr bgcolor="red">
   <th>Title</th> <th>Authors</th>
</tr>
<xsl:for-each select="bib/book">
   <tr>
      <td><xsl:value-of select="title"/></td>
      <td><xsl:value-of select="author"/>
            <xsl:for-each select="author">
                , <xsl:value-of select="."/>
            </xsl:for-each>
      </td>
   </tr>
</xsl:for-each>
```

```
<tr>
   <td>t1</td>
   <td>a1, a2</td>
</tr>
<tr>
   <td>t3</td>
   <td>a5, a6, a7</td>
</tr>
```



BOOKS

Title	Authors
t1	a1, a2
t3	a5, a6, a7

Interrogazioni nei documenti XML (4)

```
<xsl:template match="/">
  <HTML>
    <HEAD>
      <TITLE>Biblio Entries</TITLE>
    </HEAD>
    <BODY>
      <xsl:apply-templates select="bib"/>
    </BODY>
  </HTML>
</xsl:template>

<xsl:template match="title">
  <TD>
    <xsl:value-of select="."/>
  </TD>
</xsl: template>
```

```
<xsl:template match="book">
  <TR>
    <xsl:apply-templates select="title"/>
    <xsl:apply-templates select="author"/>
  </TR>
</xsl:template>

<xsl: template match="bib">
  <TABLE>
    <TBODY>
      <xsl:apply-templates select="book"/>
    </TBODY>
  </TABLE>
</xsl: template>

.....
```

```
<HTML>
  <HEAD>
    <TITLE>Biblio Entries</TITLE>
  </HEAD>
  <BODY>
    <TABLE>
      <TBODY>
        <TR> <TD> t1 </TD> <TD> a1 </TD> <TD> a2 </TD> </TR>
        <TR> <TD> t2 </TD> <TD> a3 </TD> <TD> a4 </TD> </TR>
        <TR> <TD> t3 </TD> <TD> a5 </TD> <TD> a6 </TD> <TD> a7 </TD> </TR>
      </TBODY>
    </TABLE>
  </BODY>
</HTML>
```

Il modello dei dati XSL

I dati in XSL sono rappresentabili mediante una struttura ad albero in cui la radice rappresenta l'intero documento.

Il modello dei dati di XSL è diverso da quello di XML poichè contiene un nodo addizionale che è posizionato al di sopra dell'elemento più generale di XML.

Il nodo addizionale permette di processare commenti e istruzioni che devono far parte del risultato ma che possono presentarsi prima dell'elemento più esterno di XML.

Il nodo radice XSL è in corrispondenza con la sola /.

Il modello dei dati XSL è utilizzato anche da altri linguaggi e formalismi (es. DOM) che si appoggiano su XSL per la selezione di porzioni di documento XML

Un documento XML con foglio di stile “allegato”

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="bib.xsl"?>

<bib>
    <book>
        <title> t1 </title>
        <author> a1 </author>
        <author> a2 </author>
    </book>
    <paper>
        <title> t2 </title>
        <author> a3 </author>
        <author> a4 </author>
    </paper>
    <book>
        <title> t3 </title>
        <author> a5 </author>
        <author> a6 </author>
        <author> a7 </author>
    </book>
</bib>
```

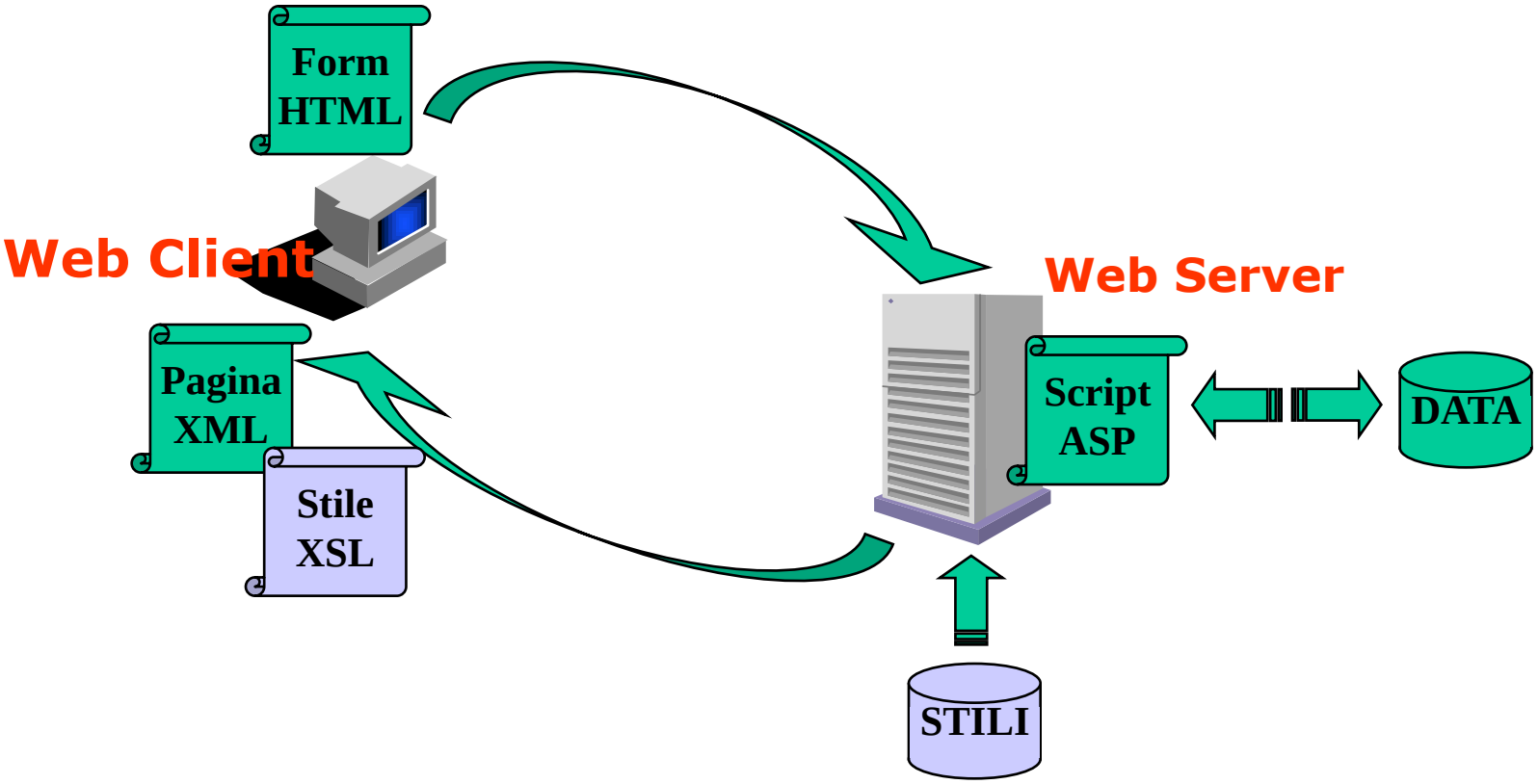
bib.xml

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

<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match="/">
<html>
  <body>
    <h2>BOOKS</h2>
    <table border="1">
      <tr bgcolor="red">
        <th>Title</th> <th>1st Author</th>
      </tr>
      <xsl:for-each select=“bib/book">
        <tr>
          <td><xsl:value-of select="title"/></td>
          <td><xsl:value-of select="author"/></td>
        </tr>
      </xsl:for-each>
      </table>
  </body>
</html>
</xsl:template>

</xsl:stylesheet>
```

bib.xsl

XML+ASP+Database: un esempio

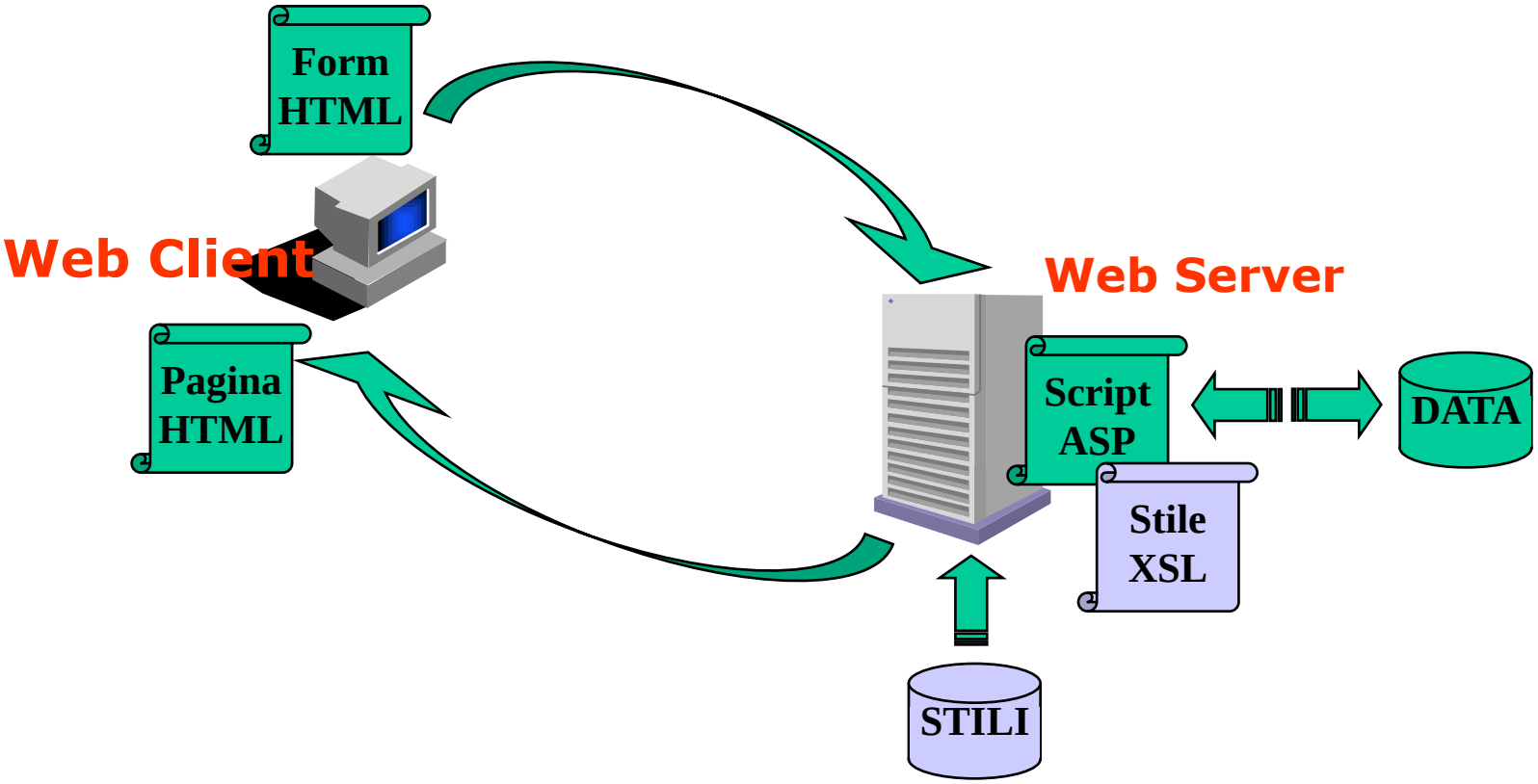
L'esempio riportato di seguito permette di visualizzare su un web browser una vista dinamica sui dati presenti in un DB.



I dati restituiti in formato XML vengono trasformati sul lato client in un documento HTML visualizzabile dal Browser

XML+ASP+Database: un esempio

L'esempio riportato di seguito permette di visualizzare su un web browser una vista dinamica sui dati presenti in un DB.



I dati in formato XML possono essere trasformati anche sul lato server

Il futuro di XML

XML non deve essere pensato solo come un linguaggio per la trasmissione di dati da pubblicare su pagine web ma, più in generale, come un linguaggio per lo scambio di dati.

L'espansione di XML è ora subordinata alla:

- Nascita di linguaggi di interrogazione commerciali per documenti XML (lo stesso XSL può essere visto come un linguaggio di interrogazione).
- Ampio inserimento nelle applicazioni (DBMS, tool di office automation, ecc.) di funzioni di esportazione/importazione di dati XML.

- Un linguaggio di interrogazione ("simile" a SQL) in via di affermazione come standard W3C è XQuery (XML Query)
- I principali DBMS commerciali (relazionali o object-relational) offrono supporto per memorizzare, indicizzare, interrogare dati in formato XML
- Si stanno diffondendo anche DBMS XML "nativi"