6 Il linguaggio XSL e la trasformazione dei documenti

Abbiamo visto che, utilizzando i fogli di stile CSS, è possibile stabilire il layout di una pagina a un livello di base per la presentazione dei dati. Per poter operare a un livello più avanzato è stato introdotto il linguaggio **XSL** (*eXtensible Stylesheet Language*), che si chiama così perché descrive le modalità di visualizzazione dei dati, e quindi il foglio di stile, secondo la sintassi XML.
In realtà XSL permette, oltre alla definizione dello stile, di operare sui dati e sulle strutture, per esempio filtrare e ordinare i dati o addirittura modificare la struttura dei dati stessi.

Le parti principali del linguaggio XSL sono:

- **XSLT** (*XSL Transformations*) per trasformare il documento XML in altri formati, per esempio HTML, RTF o PDF.
- **XPath** per accedere e navigare nel documento, di cui si parlerà nel prossimo paragrafo.

I **documenti XSLT** possono essere considerati fogli di stile, anche se si differenziano in modo notevole dai CSS: il CSS definisce gli stili da applicare a una pagina Web, mentre XSLT trasforma un documento XML in un altro documento.

Le trasformazioni possono avvenire in due modi differenti:

- lato server, utilizzando *script* scritti in un linguaggio come Php o ASP.NET;
- lato client, se il browser supporta questa funzionalità.

Progetto 4 **Visualizzare solo il nome dei siti presenti nel documento XML.**

Il file XSLT, che consente di visualizzare il nome dei siti presenti nel documento XML, si presenta in forma essenziale come mostrato di seguito. Esso è salvato su disco con il nome *siti.xsl*.

(siti.xsl)

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/
Transform">
<xsl:template match="/">
<html>
<xsl:for-each select="lista/sito">
<h2><xsl:value-of select="nome"/></h2>
</xsl:for-each>
</html>
</xsl:template>
</xsl:stylesheet>
```

Come si vede, in un file XSLT è possibile inserire qualsiasi tag HTML (nell'esempio il tag *<h2>*), facendo attenzione alle regole XML che richiedono, per esempio, la chiusura di tutti i tag.

Esaminiamo ora in dettaglio il codice XSLT precedente.
Le prime due righe sono occupate dall'intestazione e dalla versione del linguaggio XSL con il percorso dello spazio dei nomi (*xmlns*, XML *namespace*).

Nella terza riga

```
<xsl:template match="/">
```

il tag **xsl:template** viene utilizzato per associare un *template* (modello) a un foglio XML.

L'attributo **match** permette di ottenere tutte le corrispondenze di un'espressione all'interno del file XML.

In questo caso

```
match="/"
```

equivale alla ricerca in tutto il documento XML: la barra "*/*" indica l'elemento *root* (radice) del documento XML.
Il tag *<html>* indica l'inizio della parte dove si possono inserire i normali tag del linguaggio HTML.

La riga seguente

```
<xsl:for-each select="lista/sito">
```

viene utilizzata per selezionare uno specifico nodo all'interno del file XML. In questo caso, attraverso il tag **xsl:for-each**, vengono selezionati tutti i siti presenti nel documento XML: per ciascuno dei siti selezionati, si possono estrarre gli elementi.

Per esempio, per estrarre il *nome* di un sito si usa l'attributo **select** del tag **xsl:value-of**:

```
<xsl:value-of select="nome"/>
```

Questi elementi possono essere poi racchiusi tra i tag HTML, per esempio il tag *<h2>*, per visualizzare il nome in caratteri più grandi.

```
<h2><xsl:value-of select="nome"/></h2>
```

Poiché la regola del linguaggio XML obbliga a chiudere tutti i tag aperti, nella parte finale del codice occorre inserire i tag di chiusura.

```
</xsl:for-each>
</html>
</xsl:template>
</xsl:stylesheet>
```

Dopo aver creato il file XSLT, occorre collegarlo al file XML, inserendo come seconda riga del file XML il seguente codice:

```
<?xml-stylesheet type="text/xsl" href="siti.xsl"?>
```

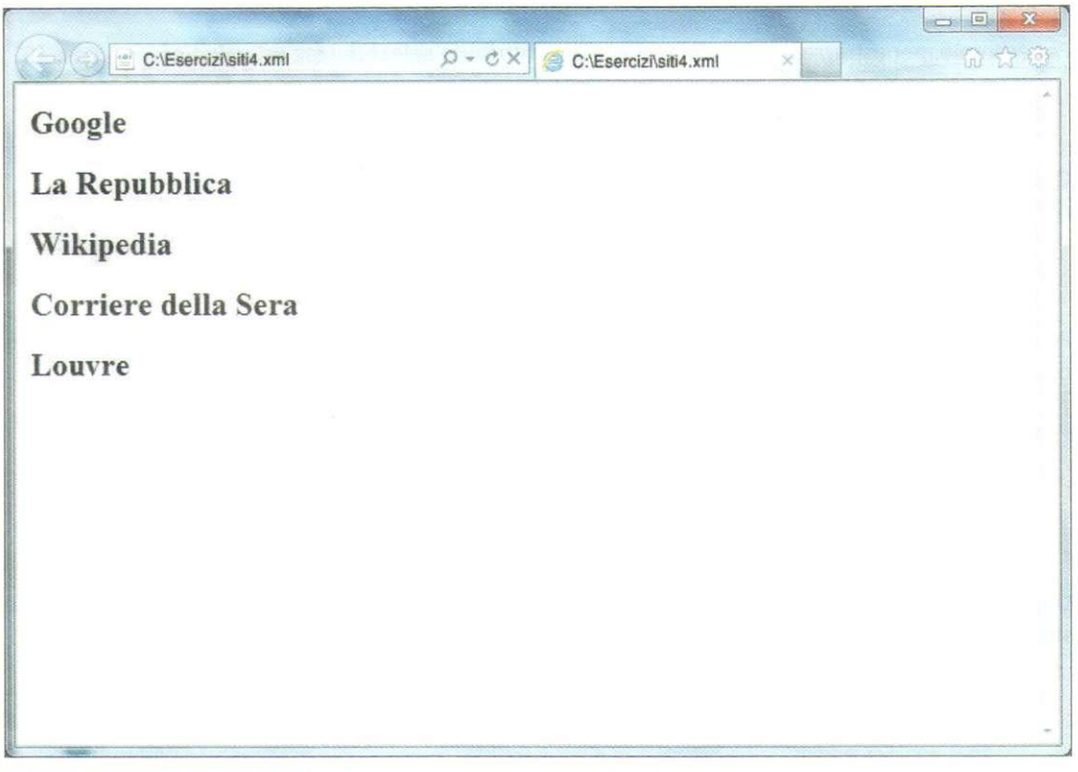
(siti4.xml)

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="siti.xsl"?>

<lista
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="siti.xsd">

<sito IDSito="s001">
<nome>Google</nome>
<URL>http://www.google.it</URL>
<descrizione>Il piu' famoso motore di ricerca</descrizione>
<categoria>motori di ricerca</categoria>
</sito>
<sito IDSito="s002">
.  .  .
.  .  .
</sito>
</lista>
```

Aprendo il file XML con il browser si ottiene l'output illustrato in figura.



Progetto 5 **Organizzare le informazioni di ciascun sito in una tabella con tre righe che contengano il nome del sito scritto in grassetto, l'indirizzo con caratteri normali e la descrizione in corsivo.**

Il file XSLT del progetto precedente viene modificato in modo da presentare le informazioni con l'utilizzo dei tag HTML per l'evidenziazione del testo e per la visualizzazione delle tabelle.

Nome del sito
Indirizzo
Descrizione

Il nuovo file *.xls* è il seguente:

(siti2.xls)

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/
Transform">

<xsl:template match="/">
<html>
<xsl:for-each select="lista/sito">
<table border="1">
<tr><td align="center">
<b><xsl:value-of select="nome"/></b>
</td></tr>
<tr><td>
<xsl:value-of select="URL"/>
</td></tr>
<tr><td>
<i><xsl:value-of select="descrizione"/></i>
</td></tr>
</table>
<br />
</xsl:for-each>
</html>
</xsl:template>
</xsl:stylesheet>
```

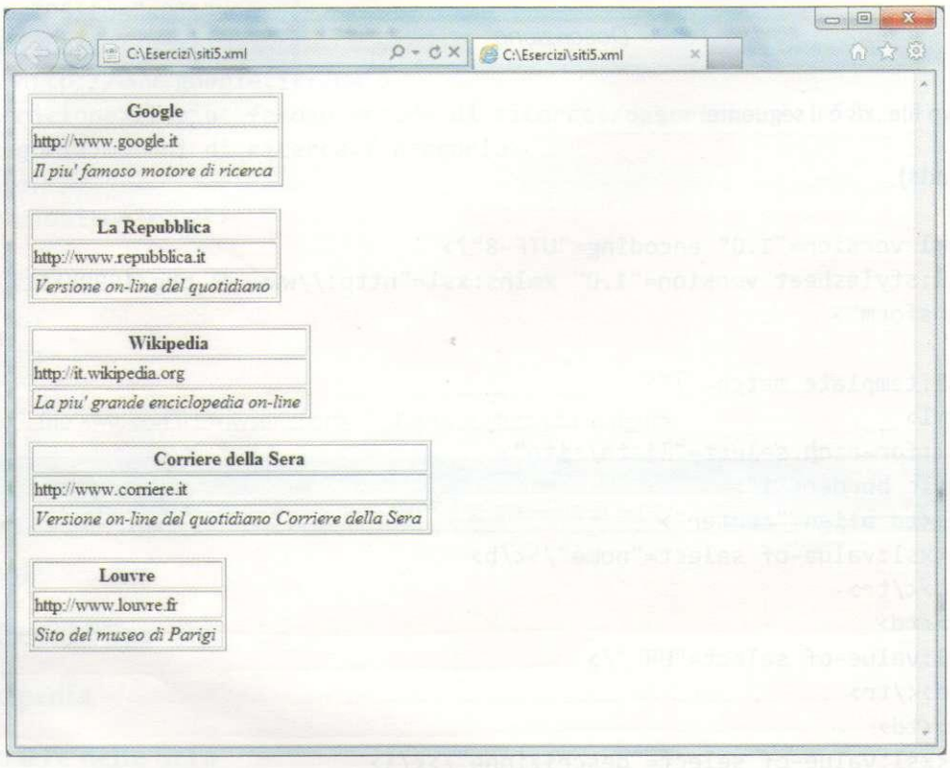
Come si vede, il file contiene tag XSLT e tag HTML per determinare il layout di presentazione dei dati. In particolare i tag HTML *<b>* e *<i>*, per grassetto e corsivo, e *<table>*, *<tr>* e *<td>* per la visualizzazione dei dati in forma tabellare. Il tag *
* inserisce una riga di separazione tra la tabella di un sito e quella successiva.

Inseriamo nella seconda riga del documento XML il riferimento al file *siti2.xls*, come nel progetto precedente:

(siti5.xml)

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="siti2.xsl"?>
. . .
. . .
```

Visualizzando il file XML con il browser si ottiene il seguente output.



Progetto 6 **Visualizzare le informazioni sui siti Web rendendo l'elemento URL di ciascun sito un link al sito stesso, in modo da attivarlo facendo clic su di esso.**

Operiamo ora un ulteriore miglioramento alla presentazione dei dati del progetto precedente, visualizzando l'elemento URL come link attivabile direttamente con un clic del mouse.
All'interno del file XSLT deve essere inserita la seguente porzione di codice che utilizza il tag **<xsl:attribute>** per definire un link:

```
<a target="_blank">
<xsl:attribute name="href">
<xsl:value-of select="URL"/>
</xsl:attribute>
<xsl:value-of select="URL"/>
</a>
```

Il link HTML è rappresentato dal tag *<a>*. Il tag *<xsl:value-of select="URL"/>* compare due volte: la prima per definire il link e la seconda per visualizzare la parola, sottolineata e in colore azzurro (oppure in colore magenta se il sito è già stato visitato in precedenza), che indica il link sopra il quale si può fare clic con il mouse per accedere al sito.
Il sito viene aperto in una nuova pagina del browser avendo indicato l'attributo *target="_blank"* per il tag *<a>*.

Il file XSLT completo, con queste modifiche, è il seguente:

(siti3.xsl)

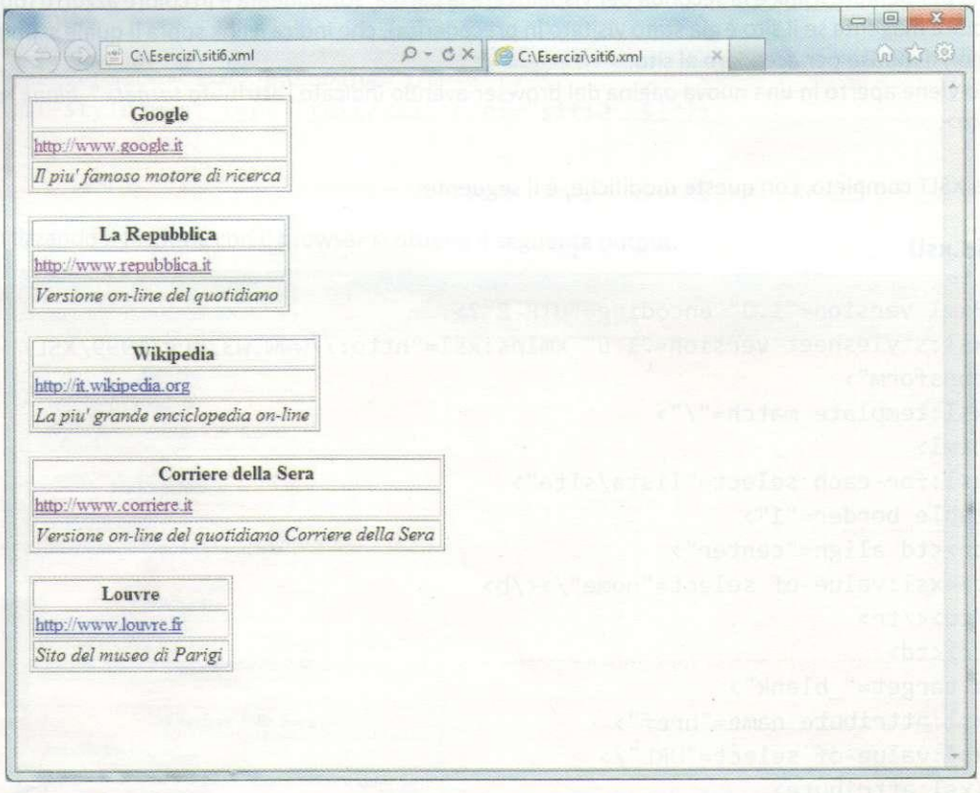
```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/
Transform">
<xsl:template match="/">
<html>
<xsl:for-each select="lista/sito">
<table border="1">
<tr><td align="center">
<b><xsl:value-of select="nome"/></b>
</td></tr>
<tr><td>
<a target="_blank">
<xsl:attribute name="href">
<xsl:value-of select="URL"/>
</xsl:attribute>
<xsl:value-of select="URL"/>
</a>
</td></tr>
<tr><td>
<i><xsl:value-of select="descrizione"/></i>
</td></tr>
</table>
<br />
</xsl:for-each>
</html>
</xsl:template>
</xsl:stylesheet>
```

Inseriamo nella seconda riga del documento XML il riferimento al file *siti3.xls*.

(siti6.xml)

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="siti3.xsl"?>
.  .  .
.  .  .
```

In output si ottiene la visualizzazione dei dati come illustrato in figura.



Riassumendo, i principali tag utilizzati negli esempi precedenti sono:

- **<xsl:template>** permette di associare un template a un elemento XML. Utilizza l'attributo **match** che permette di selezionare una parte del documento.
- **<xsl:value-of>** permette di estrarre un valore di un elemento XML e utilizzarlo in output. Viene utilizzato con l'attributo **select** che permette di specificare qual è il valore interessato.
- **<xsl:for-each>** permette di selezionare con una ripetizione ogni elemento di uno o più nodi XML. Anche questo tag utilizza l'attributo **select**.

7 Controlli XSL sulla trasformazione del documento

I seguenti tag XSL sono solo alcuni tra i numerosi tag del linguaggio che consentono di definire con più precisione la trasformazione dei dati:

- **<xsl:if>** permette di controllare se gli elementi verificano una condizione e applica il codice scritto di seguito se la condizione è verificata.

```
<xsl:if test=". . . . . ">
...
...
...
</xsl:if>
```

- **<xsl:choose>** utilizzato insieme a **<xsl:when>** e **<xsl:otherwise>** permette di creare una struttura di tipo *if-else*, cioè applica il codice contenuto in *<xsl:when>* quando la condizione è vera, altrimenti applica il codice contenuto in *<xsl:otherwise>*.

```
<xsl:choose>
<xsl:when>
...
...
</xsl:when>
<xsl:otherwise>
...
...
</xsl:otherwise>
</xsl:choose>
```

- **<xsl:sort>** permette di ordinare gli elementi selezionati.

Progetto 7 **Visualizzare solo i siti appartenenti alla categoria *'news'*.**

Si tratta di inserire nel file XSLT il tag per impostare una selezione dei siti con il criterio *categoria = 'news'*.

Il tag **<xsl:if>** è utilizzato con la seguente sintassi:

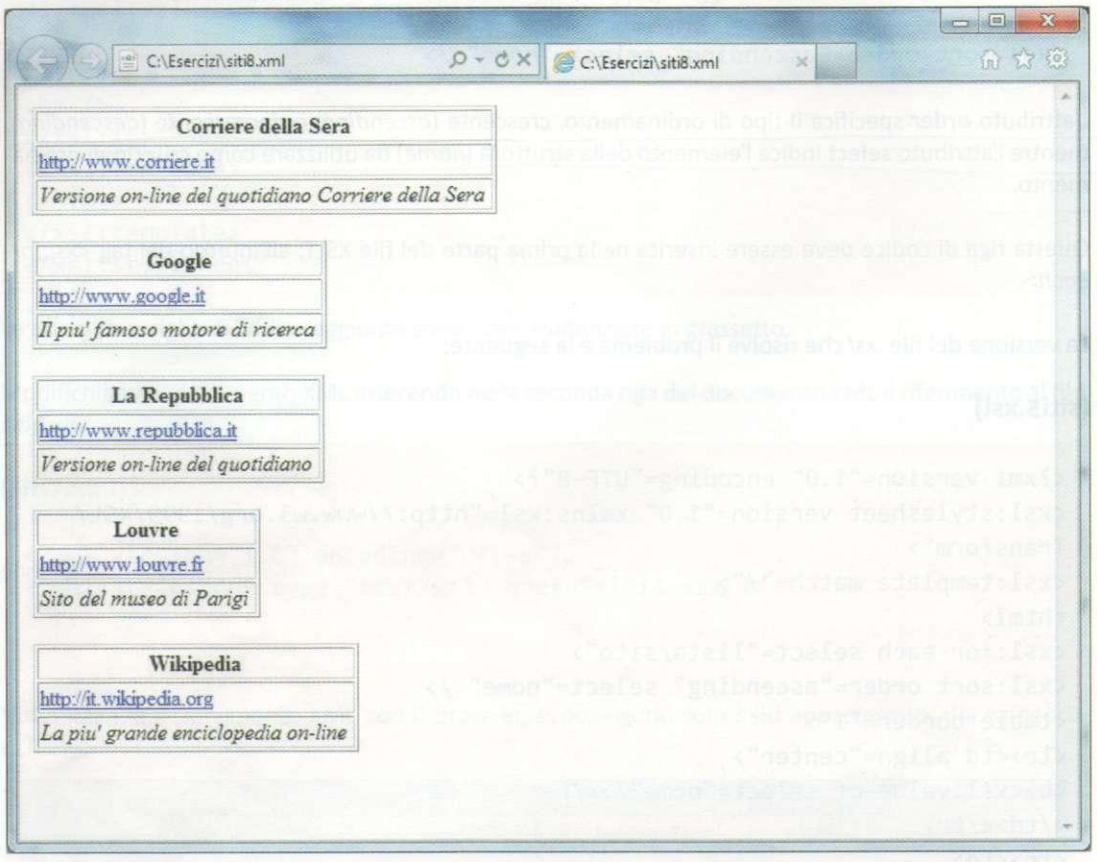
```
<xsl:if test="categoria='news'">
```

La versione modificata del file di trasformazione è la seguente:

(siti4.xsl)

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/
Transform">
<xsl:template match="/">
<html>
<xsl:for-each select="lista/sito">
<xsl:if test="categoria='news'">
<table border="1">
<tr><td align="center">
<b><xsl:value-of select="nome"/></b>
</td></tr>
<tr><td>
<a target="_blank">
<xsl:attribute name="href">
<xsl:value-of select="URL"/>
</xsl:attribute>
<xsl:value-of select="URL"/>
</a>
```

La figura mostra il risultato ottenuto visualizzando il documento XML (*siti8.xml*), modificato con il riferimento al file *siti5.xsl*.



AUTOVERIFICA • Domande da 6 a 9 pag. 239-240 - Problemi da 4 a 12 pag. 241

8 Le espressioni XPath

Nei progetti precedenti si è visto che molti tag selezionano solo una parte dei dati XML oppure determinati nodi, attraverso l'attributo *select*.

Il tag

```
<xsl:template match="/">
```

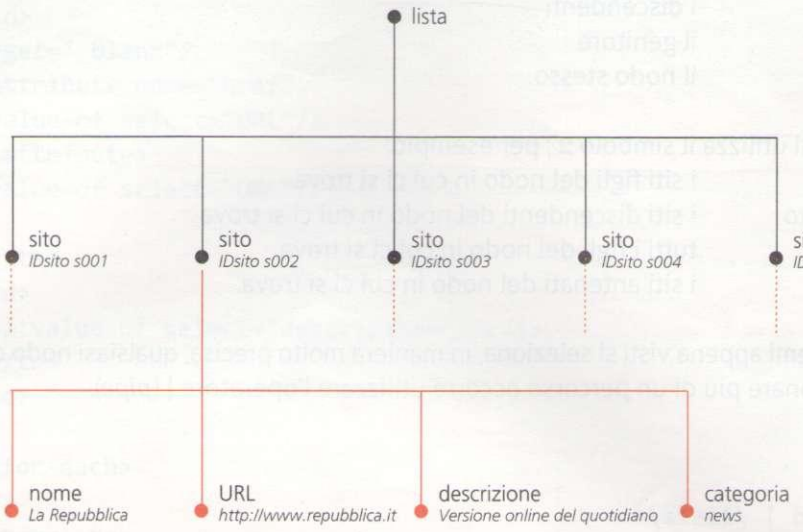
seleziona tutti i dati presenti nel foglio XML, mentre il tag

```
<xsl:for-each select="lista/sito">
```

esamina ciascun sito presente nel documento XML.

Sia "/" che "*lista/sito*" vengono dette **espressioni XPath**.

Il concetto fondamentale di XPath è il **nodo**. Un nodo può essere un elemento, un attributo, un testo, un commento o altro e viene utilizzato per navigare all'interno dei dati XML. Il documento XML può quindi essere visto come un albero.



Tra i nodi esistono delle relazioni di **parentela** (genitore, figlio, ...) che soddisfano determinate proprietà:

- ogni nodo ha un **genitore**: nell'esempio *nome* ha come genitore *sito*;
- un nodo può avere uno o più **figli** (nell'esempio *nome* e *URL* sono figli di *sito*), ma può anche non averne, per esempio *descrizione* non ha figli;
- due nodi con lo stesso genitore sono detti **fratelli**: nell'esempio *nome* e *URL* sono fratelli;
- un nodo ha **antenati**: i genitori, i genitori dei genitori, e così via;
- un nodo può avere **discendenti**: i figli, i figli dei figli, e così via.

Si crea quindi un albero gerarchico all'interno del quale è possibile spostarsi utilizzando i nomi (come nel caso *lista/sito*, che significa "*sito figlio di lista*") oppure i legami di parentela. Le espressioni sono simili a quelle utilizzate per la navigazione tra le directory dei sistemi operativi:

- **/** indica il nodo radice (per esempio */lista*)
- **//** indica qualsiasi nodo nel documento indipendentemente da dove si trovi
- **.** indica il nodo in cui ci si trova
- **..** indica il genitore del nodo in cui ci si trova
- **@** indica gli attributi.

Attraverso l'uso di questi simboli si può navigare attraverso i dati indicando i percorsi (**path**) all'interno dell'albero.
Inoltre i **predicati** permettono di ricercare uno o più nodi particolari.
Per esempio:

- /lista/sito — qualsiasi sito figlio di *lista*
- //sito — qualsiasi *sito*
- /lista/sito[2] — il secondo *sito*
- /lista/sito[last()] — l'ultimo *sito*
- /lista/sito[@IDSito] — il *sito* con un attributo chiamato *IDSito*
- /lista/sito[@IDSito='s002'] — il *sito* con l'attributo *IDSito* uguale a *s002*
- /lista/* — un qualsiasi figlio di *lista*.

Partendo dal nodo in cui ci si trova, è possibile muoversi utilizzando le parentele definite sopra:

- ancestor gli antenati
- child i figli
- descendant i discendenti
- parent il genitore
- self il nodo stesso.

Per fare questo si utilizza il simbolo :: ; per esempio:

- child::sito i siti figli del nodo in cui ci si trova
- descendant::sito i siti discendenti del nodo in cui ci si trova
- child::* tutti i figli del nodo in cui ci si trova
- ancestor::sito i siti antenati del nodo in cui ci si trova.

Utilizzando i sistemi appena visti si seleziona, in maniera molto precisa, qualsiasi nodo o percorso. Per poter selezionare più di un percorso occorre utilizzare l'operatore | (*pipe*).

Per esempio

```
child::sito | /lista/*
```

indica il *sito* figlio del nodo in cui ci si trova e qualsiasi figlio di *lista*.

XPath consente di utilizzare anche altri tipi di operatori:

- le quattro operazioni (+, -, *, *div*)
- operatori di confronto (=, !=, <, <=, >, >=)
- operatori booleani (*or, and*).

I seguenti progetti mostrano l'utilizzo pratico di *XPath*. Le *espressioni XPath* vengono inserite nel tag *<xsl:for-each select="....."*> come valore di *select* nella parte iniziale del file XSL.

Progetto 9 **Selezionare i siti appartenenti alla categoria *"musei"*.**

La condizione per controllare la categoria è racchiusa tra parentesi quadre.

```
<xsl:for-each select="lista/sito[categoria='musei']">
```

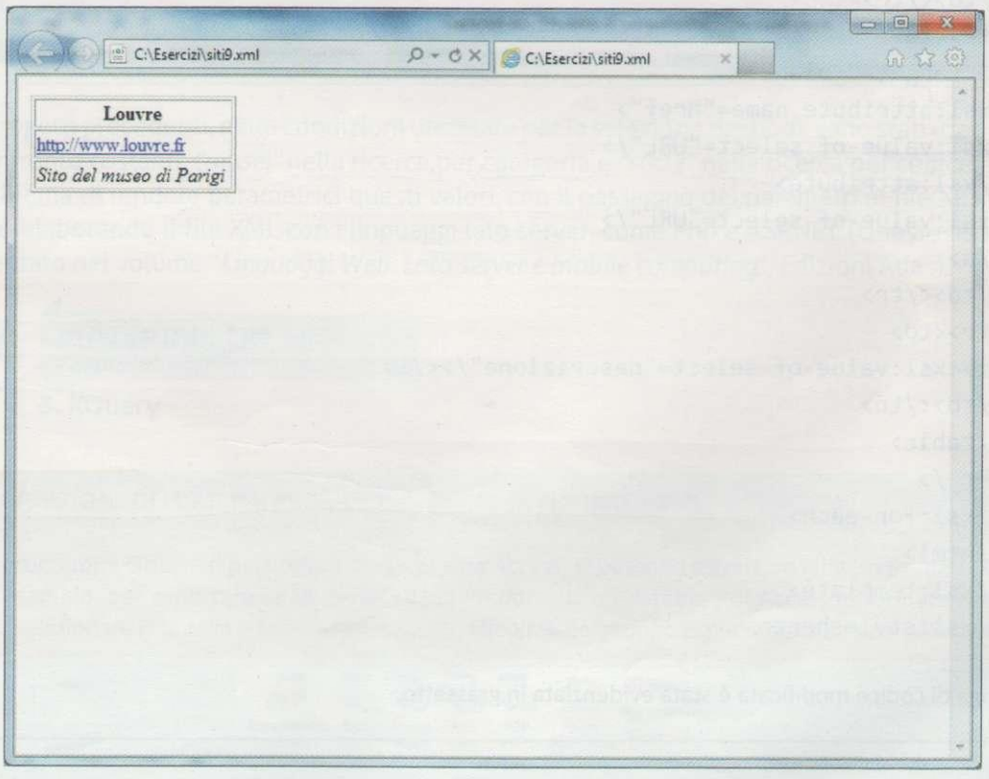
Il file XSL che risolve il problema è riportato di seguito.

(siti6.xsl)

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/
Transform">
<xsl:template match="/">
<html>
<xsl:for-each select="lista/sito[categoria='musei']">
<table border="1">
<tr><td align="center">
```

```
<b><xsl:value-of select"nome"/></b>
</td></tr>
<tr><td>
<a target="_blank">
<xsl:attribute name="href">
<xsl:value-of select="URL"/>
</xsl:attribute>
<xsl:value-of select="URL"/>
</a>
</td></tr>
<tr><td>
<i><xsl:value-of select="descrizione"/></i>
</td></tr>
</table>
<br />
</xsl:for-each>
</html>
</xsl:template>
</xsl:stylesheet>
```

La riga di codice modificata è stata evidenziata in grassetto.
Inserendo nel documento XML (*siti9.xml*) il riferimento al file *siti6.xsl* e aprendo il file nel browser, si ottiene l'output illustrato in figura.



Progetto 10 **Selezionare il sito avente codice uguale a *"s002"*.**

In questo progetto viene effettuata la selezione sui siti, controllando il valore dell'attributo *IDSito*. Nella sintassi *XPath* il nome dell'attributo è preceduto dal carattere **@**.

L'espressione XPath è inserita nel tag *<xsl:for-each select="....">*, come valore di *select* nella parte iniziale del file XSL:

```
<xsl:for-each select="/lista/sito[@IDSito='s002']">
```

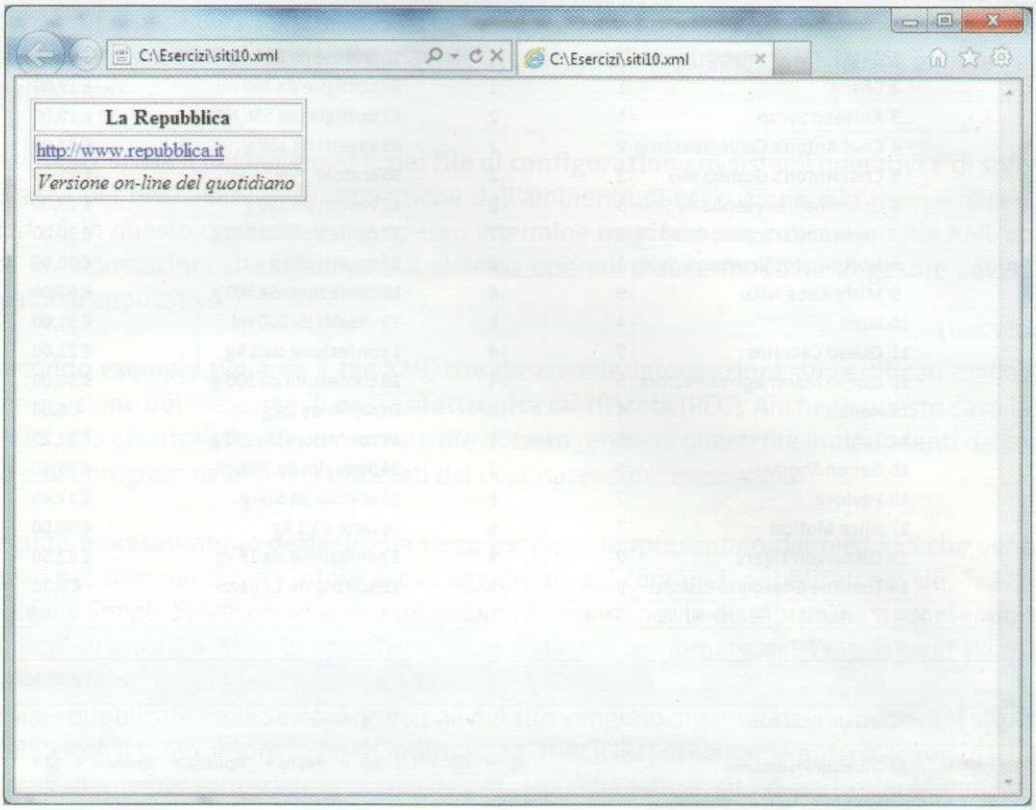
Il file XSL che risolve il problema è riportato di seguito.

(siti7.xsl)

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
<html>
<xsl:for-each select="/lista/sito[@IDSito='s002']">
<table border="1">
<tr><td align="center">
<b><xsl:value-of select="nome"/></b>
</td></tr>
<tr><td>
<a target="_blank">
<xsl:attribute name="href">
<xsl:value-of select="URL"/>
</xsl:attribute>
<xsl:value-of select="URL"/>
</a>
</td></tr>
<tr><td>
<i><xsl:value-of select="descrizione"/></i>
</td></tr>
</table>
<br />
</xsl:for-each>
</html>
</xsl:template>
</xsl:stylesheet>
```

La riga di codice modificata è stata evidenziata in grassetto.

Inserendo nel documento XML (*siti10.xml*) il riferimento al file *siti7.xsl* e aprendo il file nel browser, si ottiene l'output illustrato in figura.



Nei progetti precedenti, nelle condizioni utilizzate per la selezione dei nodi, sono stati usati valori di confronto costanti: *"musei"* nella ricerca per categoria e *"s002"* nella ricerca per codice.
Il problema di rendere parametrici questi valori, con il passaggio dei parametri ai file XLS, viene risolto elaborando il file XML con i linguaggi lato server, come Php e ASP.NET. (Questo aspetto è presentato nel volume "*Linguaggi Web. Lato server e mobile computing*", Edizioni Atlas).

MATERIALI ON LINE
libreriaweb.edatlas.it

3. XQuery

ESPORTAZIONE DEI DATI DAI PROGRAMMI OFFICE IN FORMATO XML

Dai programmi Office, in particolare da *Excel* e da *Access*, si possono esportare i dati in formato XML. Per esempio, per esportare i dati della tabella *Prodotti* da un database di Access in formato XML, si deve selezionare la tabella e fare clic sul pulsante **File XML** nel gruppo **Esporta** della scheda **Dati esterni**.