XML

XML è un linguaggio derivato dall'SGML, da intendersi come un sottoinsieme «compatibile» con questo; in particolare, il nome rappresenta l'acronimo di *Extensible markup language*. Il motivo per il quale è stata introdotta questa variante dell'SGML è dovuto all'esigenza di trovare un compromesso tra l'SGML originale e l'HTML, che è solo un'applicazione di SGML troppo limitata per la documentazione multimediale. In pratica, l'intento è stato ed è quello di semplificare leggermente l'SGML rendendo disponibili molte qualità dell'SGML che un'applicazione rigida come l'HTML non è in grado di offrire.

In generale, un documento XML è un'***applicazione*** di XML; nello stesso modo, l'HTML (come linguaggio) è un'applicazione SGML.

52.1 Transizione da SGML a XML

XML nasce come sottoinsieme di SGML, ma nel tempo si evolve e si allontana da SGML. In particolare, la prima caratteristica che comincia il distacco tra i due è costituita dai domini applicativi (*namespace*), i quali rendono difficile l'abbinamento di un DTD a un documento XML che ne fa uso.

52.1.1 Codifica

La novità più importante di XML è l'utilizzo predefinito dell'insieme di caratteri universale, prevalentemente attraverso la forma UTF-8 e UTF-16. Questo fatto ha delle implicazioni importanti, in quanto i riferimenti a macro del tipo '`&`***`n`***`;`' e '`&x`***`n`***`;`' si fanno ai punti di codifica dello standard ISO 10646 (nel primo caso il numero è espresso in decimale, mentre nel secondo si tratta di un numero esadecimale).

XML non esclude a priori l'utilizzo di altri tipi di codifica; tuttavia, se non è possibile usare le codifiche UTF-***n***, per evitare ambiguità potrebbe essere conveniente limitarsi all'uso dell'ASCII tradizionale, dal momento che è perfettamente compatibile con la forma UTF-8. Eventualmente è possibile anche specificare il tipo di codifica attraverso un'istruzione apposita, che viene mostrata in seguito.

52.1.2 Commenti

I commenti si indicano in linea di massima come in SGML, attraverso la forma:

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

Come nell'SGML si deve evitare l'uso di due trattini in sequenza, '`--`', ma in XML non è ammissibile il commento nullo nella forma '`<!>`'.

52.1.3 Marcatori ed elementi vuoti

In XML, gli elementi devono essere aperti e chiusi correttamente attraverso i marcatori relativi; in pratica non è possibile più lasciare all'analizzatore XML il compito di determinare da solo la cosa in base al contesto. Questa limitazione è importante per facilitare il compito dei programmi che devono interpretare un documento XML e comunque si riflette positivamente nella struttura del sorgente del documento stesso.

Gli elementi vuoti vanno indicati regolarmente con il marcatore di chiusura, oppure con un solo marcatore speciale, che ha la forma seguente:

```
<nome_elemento/>
```

In pratica, alla fine del marcatore appare una barra obliqua prima del simbolo '`>`'.

Di fatto, per problemi di compatibilità, si lascia uno spazio prima della barra finale. Per esempio: '`<hr />`'.

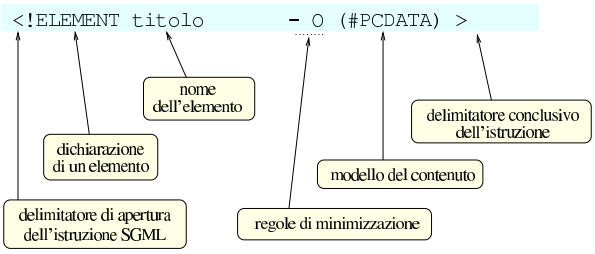
L'assenza della possibilità di definire dei marcatori di apertura o di chiusura opzionali, fa sì che si semplifichi la dichiarazione di questi nel DTD:

```
<ELEMENT nome_elemento modello_del_contenuto >
```

Nella figura 52.1 si vede un confronto tra la dichiarazione SGML e quella XML. Si vede chiaramente che in XML mancano le regole di minimizzazione.

Figura 52.1. Scomposizione delle varie parti della dichiarazione di un elemento SGML e XML.

SGML:



XML:

```
<!ELEMENT titolo (#PCDATA) >
```

In XML, i nomi che si attribuiscono agli elementi e agli attributi sono sensibili alla differenza tra lettere maiuscole e minuscole; per esempio, l'elemento '**testo**' è diverso dall'elemento '**Testo**' e da tutte le altre varianti possibili. Per la precisione, i nomi devono sottostare alle regole seguenti:

- devono iniziare con una lettera alfabetica, oppure con un trattino basso ('_', ovvero #x5F), e possono contenere anche cifre numeriche, il punto, il trattino basso e il trattino normale (#x2D);
- non possono contenere spazi;
- possono contenere i due punti ('**:**'), ma questa possibilità è comunque riservata a situazioni particolari (serve a definire un prefisso che rappresenta un contesto);
- non possono iniziare con la sigla '**xml**', o con qualunque altra variazione delle lettere minuscole e maiuscole, dal momento che questi potrebbero avere in seguito dei significati speciali.

52.1.4 Entità predefinite

Alcune entità standard essenziali sono predefinite e teoricamente non è necessario specificarle nel DTD. Si tratta di '**amp**', '**lt**', '**gt**', '**apos**' e '**quot**'. Le macro relative sono '**&**', '**<**', '**>**', '**'**' e '**"**'.

Si può osservare questo particolare nella dichiarazione SGML di XML:

```
SYNTAX
    ...
    ENTITIES
        "amp"  38
        "lt"   60
        "gt"   62
        "quot" 34
        "apos" 39
```

52.1.5 Entità parametriche

In XML, le entità parametriche possono essere utilizzate solo all'interno del DTD. Da ciò consegue logicamente che le sezioni marcate con le quali si può includere o escludere del testo in base al contenuto di un'entità parametrica, possono esistere solo nel DTD.


```
<!ENTITY % bozza 'INCLUDE' >
<!ENTITY % finale 'IGNORE' >

<![%bozza;[
<!ELEMENT libro (commento*, titolo, corpo)>
]]>
<![%finale;[
<!ELEMENT libro (titolo, corpo)>
]]>
```

L'esempio mostra un pezzo di un DTD ipotetico, in cui vengono dichiarate due entità parametriche, '**bozza**' e '**finale**'. In questo caso, la macro '**%bozza;**' si traduce nella parola '**INCLUDE**', mentre la macro '**%finale;**' si traduce nella parola '**IGNORE**'. In questo modo, viene dichiarato l'elemento '**libro**' nella prima modalità: quella che ammette la presenza dell'elemento '**commento**'.

52.1.6 Altre sezioni marcate

XML ammette l'uso di un'altra sezione marcata soltanto, la sezione '**CDATA**' per delimitare del testo letterale.

```
<![CDATA[Il marcatore <ciao> serve per...]]>
```

L'esempio mostra in che modo sia possibile utilizzare letteralmente i simboli '**<**' e '**>**' in una sezione '**CDATA**'.

52.1.7 Istruzioni di elaborazione

Le istruzioni di elaborazione sono una novità in XML. Servono in qualche modo per passare delle informazioni alle applicazioni. Si distinguono per avere la forma seguente:

```
<?istruzione_di_elaborazione?>
```

Il testo che compone l'istruzione dipende dall'applicazione a cui è diretto. È importante tenere presente che tutto ciò che inizia con la stringa '**xml**', assieme a tutte le sue variazioni di lettere maiuscole e minuscole, è riservato.

In generale, in base al significato che può avere l'istruzione di elaborazione, queste possono trovarsi in qualunque parte del sorgente XML.

Normalmente si inizia sempre un sorgente XML con un'istruzione di elaborazione che dichiara la versione di XML a cui si fa riferimento, assieme alla codifica utilizzata:

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

52.1.8 Convenzioni dell'XML

Nella descrizione fatta fino a questo punto sono già state presentate alcune convenzioni di XML che non sono esprimibili nella dichiarazione SGML relativa. In pratica, si tratta di regole che vanno tenute in considerazione quando si scrive un DTD per un documento XML. Vale la pena di raccogliere le convenzioni più importanti.

- I nomi di elementi e degli attributi che iniziano per '**xml**', con qualsiasi altra variante delle lettere maiuscole e minuscole, sono riservati.
- Gli elementi che ne possono avere bisogno, devono poter disporre di un attributo denominato '**xml:space**', a cui possano essere assegnate le parole chiave '**default**' o '**preserve**'. Il suo scopo è quello di definire il comportamento nei confronti degli spazi (di tutti i caratteri assimilabili a questo concetto). Assegnando la parola chiave '**default**' si intende lasciare che gli spazi vengano gestiti come al solito, eliminando quelli superflui; con la parola chiave '**preserve**' si vuole richiedere di mantenere gli spazi come sono. La dichiarazione di questo attributo può avvenire nel DTD come nell'esempio seguente:

  ```
  <!ATTLIST esempio xml:space (default|preserve) 'preserve'>
  ```

  In particolare, un elemento che per sua natura deve rispettare le spaziature originali, potrebbe essere definito nel modo seguente, dove si vede il caso dell'elemento '**pre**' di XHTML:

```
<!ELEMENT pre %pre.content;>
<!ATTLIST pre
  %attrs;
  xml:space (preserve) #FIXED 'preserve'
>
```

- Gli elementi che ne possono avere bisogno, devono poter disporre di un attributo denominato '**`xml:lang`**', a cui poter assegnare un codice identificativo del linguaggio contenuto. Si prevede l'uso di diversi tipi di codice:
  - un codice di linguaggio composto da due lettere, secondo lo standard ISO 639 (tabella 13.4);
  - un codice di linguaggio registrato dall'autorità IANA (*Internet assigned numbers authority*), a cui va aggiunto comunque il prefisso '**`i-`**', oppure '**`I-`**';
  - un codice stabilito dall'utente o concordato tra le parti, a cui va aggiunto il prefisso '**`x-`**', oppure '**`X-`**'.

  La dichiarazione di questo attributo può avvenire nel DTD come nell'esempio seguente:

```
<!ATTLIST esempio xml:lang NMTOKEN #IMPLIED >
```

  Eventualmente si può anche specificare un linguaggio predefinito, come si vede nell'esempio seguente:

```
<!ATTLIST testo xml:lang NMTOKEN 'it' >
```

52.1.9 Correttezza formale e validità

Possono esistere due livelli di approccio all'XML da parte dei programmi che lo utilizzano: il primo si limita a leggere il documento senza sapere nulla della sua struttura stabilita nel DTD; il secondo invece richiede la conoscenza di questa struttura. Nel primo caso è sufficiente che il documento XML sia stato scritto correttamente dal punto di vista formale, in senso generale; in questo modo si parla di *well formed document*. Nel secondo caso è importante che il documento, oltre che essere corretto dal punto di vista formale, sia anche valido in base alla definizione stabilita nel DTD.

Il documento XML corretto dal punto di vista formale, ha le caratteristiche seguenti:

- contiene un elemento principale unico, all'interno del quale vanno collocati tutti gli altri (si parla comunemente dell'elemento *root*, ovvero della radice);
- tutti i marcatori degli elementi devono essere indicati in modo corretto, attraverso degli annidamenti ordinati;
- tutti gli elementi devono essere delimitati correttamente, senza saltare dei marcatori, inoltre gli elementi vuoti vanno chiusi oppure vanno indicati con il marcatore speciale già mostrato;
- devono essere rispettate le regole stabilite per i nomi degli elementi;
- i valori associati agli attributi vanno delimitati sempre attraverso apici doppi oppure apici singoli;

Il documento XML valido, oltre a essere corretto formalmente, deve anche essere conforme al DTD. Come nell'SGML normale, il DTD può essere indicato attraverso un riferimento, oppure può essere incorporato all'inizio del documento.

52.1.10 Verifica della validità con SP

Il pacchetto SP di James Clark può essere utilizzato anche per convalidare un documento XML, a partire dal suo DTD (il procedimento è analogo a quanto già mostrato nella sezione 51.2); tuttavia, è necessario procurarsi la dichiarazione XML, che si può trovare nell'archivio dei sorgenti di SP stesso: '`pubtext/xml.dcl`'.

Supponendo di disporre del file '`xml.dcl`' nella directory corrente, si può realizzare un catalogo molto semplice come quello seguente:

```
SGMLDECL "xml.dcl"
```


Naturalmente, nel catalogo si possono aggiungere anche altre cose, in base alla necessità o meno di indicare il DTD e le entità generali. Per verificare il funzionamento della cosa, si può provare a eseguire la convalida dell'esempio seguente, che include il DTD nel preambolo e non ha bisogno di entità generali:

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE esempio [
    <!ELEMENT esempio (#PCDATA)>
]>
<esempio>Ciao a tutti!</esempio>
```

Si può osservare che si tratta di un documento elementare, in cui esiste solo l'elemento principale, denominato '**`esempio`**'.

Per la convalida, si può usare l'eseguibile '**`nsgmls`**' nel modo seguente:

```
$ nsgmls -c catalogo.xml -s esempio.xml [Invio]
```

Qui si sottintende che il file del catalogo sia '`catalogo.xml`' e che il sorgente XML sia contenuto nel file '`esempio.xml`'. Se oltre alla convalida si vuole avere il risultato pre-elaborato, si toglie l'opzione '**`-s`**', ottenendo quanto segue:

```
?xml version="1.0" encoding="ISO-8859-1"
(esempio
-Ciao a tutti!
)esempio
C
```

52.1.11 Domini applicativi: «namespace»

Con il termine ***dominio applicativo*** si vuole qui fare riferimento a quello che è noto come *namespace* a proposito di XML. Si definisce un dominio applicativo associando elementi, ma eventualmente anche attributi, a qualcosa che viene identificato tramite un indirizzo URI (sia URL, sia URN; si veda anche la sezione 54.1 a proposito dell'estensione del termine URI). Questa associazione ha lo scopo di evitare ambiguità, quando per qualche ragione si utilizzerebbero elementi o attributi con lo stesso nome, ma con significati differenti.

Dal momento che non è possibile utilizzare nomi di elementi e di attributi che contengano direttamente un URI, si associa questo URI attraverso un attributo particolare:

```
1  <?xml version="1.0"?>
2  <html xmlns="http://www.w3.org/1999/xhtml">
3    <head>...</head>
4    <body>
5      <h1>Bla bla bla</h1>
6      ...
7      <math xmlns="http://www.w3.org/1998/Math/MathML">
8        <mi>x</mi><mo>+</mo><mn>3</mn>
9      </math>
10   </body>
11 </html>
```

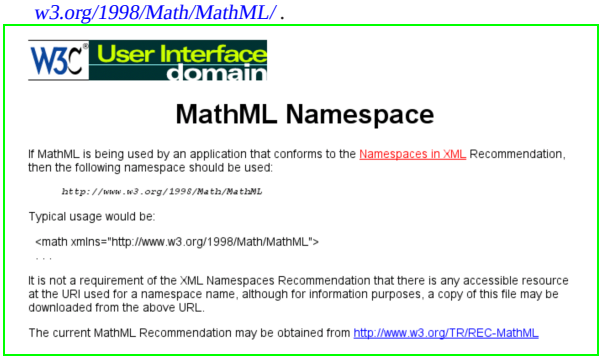
Questo esempio mostra l'utilizzo di MathML all'interno di un documento XHTML (le righe dalla settima alla nona). Si può osservare che l'elemento '**`html`**', alla riga numero due dell'esempio, contiene l'attributo '**`xmlns`**' a cui è associata la stringa '**`http://www.w3.org/1999/xhtml`**'. Questa dichiarazione specifica che all'elemento '**`html`**' e a tutti i suoi discendenti si associa il dominio applicativo *http://www.w3.org/1999/xhtml* . Questo indirizzo (*http://www.w3.org/1999/xhtml*) è solo un punto di riferimento univoco; se si vuole, lo si può anche visitare (figura 52.14), ma ha soltanto lo scopo di dichiarare che gli elementi a cui è associato si riferiscono a XHTML; inoltre, non è nemmeno necessario che esista veramente.

Con i linguaggi SGML, si usa normalmente la dichiarazione del DTD a cui si fa riferimento. Anche con XML, volendo, è corretto usare un riferimento al DTD con l'istruzione '**`<DOCTYPE…>`**', ma si inserisce questo concetto nuovo del dominio applicativo.

Figura 52.14. Come si presenta alla vista l'indirizzo *http://www.w3.org/1999/xhtml* .

Continuando a leggere l'esempio mostrato, alla settima riga appare l'elemento '**`math`**', che, come dichiarato dall'attributo '**`xmlns`**' appartiene al dominio applicativo *http://www.w3.org/1998/Math/MathML/* ; in pratica, dichiara che l'elemento in questione e ciò che contiene riguarda MathML.

Figura 52.15. Come si presenta alla vista l'indirizzo *http://www.w3.org/1998/Math/MathML/* .



Quanto visto in questo esempio, rappresenta un uso «predefinito» del dominio applicativo. Infatti, esiste anche la possibilità di associare gli elementi (e gli attributi) a un dominio applicativo, specificando un prefisso:

```
1  <xsl:stylesheet
2       xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
3    <xsl:template name="/">
4      ...
5    </xsl:template>
6  </xsl:stylesheet>
```

In questo caso si può osservare, alla riga numero uno, che l'elemento '**`xsl:stylesheet`**' contiene l'attributo '**`xmlns:xsl`**', a cui è associato un URI. Ciò significa che, nell'ambito del controllo dell'elemento '**`xsl:stylesheet`**', elementi e attributi il cui nome inizia con il prefisso '**`xsl:`**' appartengono al dominio applicativo *http://www.w3.org/1999/XSL/Transform* ; per la precisione, anche lo stesso elemento '**`xsl:stylesheet`**' appartiene a questo dominio applicativo, proprio perché anche il suo nome inizia con il prefisso '**`xsl:`**'.


```
1  <?xml version="1.0" encoding="utf-8"?>
2  <xsl:stylesheet version="1.0"
3          xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
4          xmlns:fo="http://www.w3.org/1999/XSL/Format">
5
6  <xsl:output method="xml" indent="yes"/>
7  ...
8  <xsl:template match="studio">
9  <fo:root>
10     <fo:layout-master-set>
11         ...
12     </fo:layout-master-set>
13     <fo:page-sequence master-reference="cover">
14         ...
15     </fo:page-sequence>
16     ...
17 </fo:root>
18 </xsl:template>
19 ...
20 </xsl:stylesheet>
```

Questo nuovo esempio mostra l'uso di due domini applicativi, associati rispettivamente ai prefissi '**`xsl:`**' e '**`fo:`**' (le righe tre e quattro). La dichiarazione dei domini applicativi avviene nell'elemento principale, ovvero '**`xsl:stylesheet`**' (che appartiene al dominio applicativo associato al prefisso '**`xsl:`**'), in modo che possa essere valida per tutto il documento, salva la possibilità di modificarla all'interno di elementi ben precisi.

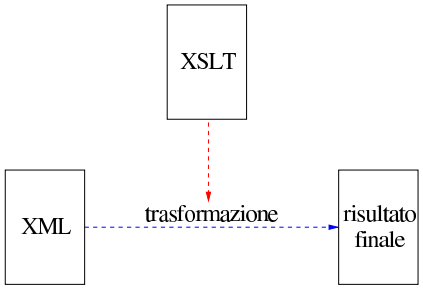
Oltre ai modi mostrati in questi esempi, esistono altre combinazioni; per esempio è possibile affiancare un dominio applicativo predefinito a domini specificati attraverso l'uso di prefissi. Tuttavia si preferisce evitare di creare troppa confusione su un concetto che invece dovrebbe essere semplice, per consentire la comprensione di altri concetti legati a XML. Eventualmente si possono consultare i documenti annotati nella bibliografia al termine del capitolo, per una visione più precisa e dettagliata a proposito dei domini applicativi (ovvero *namespace*).

La presenza dei domini applicativi rende difficile l'abbinamento di un DTD a un documento XML, perché i prefissi devono essere previsti esattamente nel DTD, come parte dei nomi degli elementi e degli attributi, mentre la definizione di un dominio applicativo consentirebbe l'utilizzo di un prefisso libero, deciso nel momento in cui si usa l'attributo '**`xmlns:x`**'.

52.2 Introduzione ai fogli di stile XSLT

XSLT è un linguaggio realizzato in forma di file XML, con il quale si definisce una trasformazione di un file XML in un altro file. Generalmente, il file di destinazione è un altro file XML, anche se può comunque essere qualcosa di diverso. La sigla sta precisamente per *XSL transformations* (ovvero *Extensible stylesheet language transformations*), a indicare che il linguaggio è scritto in quello che viene chiamato «foglio di stile XSL» e serve a trasformare i dati originali in qualcosa di conveniente ai propri scopi. In altri termini, si tratta di un linguaggio che consente di estrarre le informazioni contenute in un file XML, per generare con queste ciò che serve.

Figura 52.18. Utilizzo del linguaggio XSLT per la trasformazione di un file XML.



Naturalmente, l'elaborazione di un file XML secondo il linguaggio XSLT richiede un programma apposito. Qui viene mostrato l'uso di

Xalan (sezione 52.2.6).

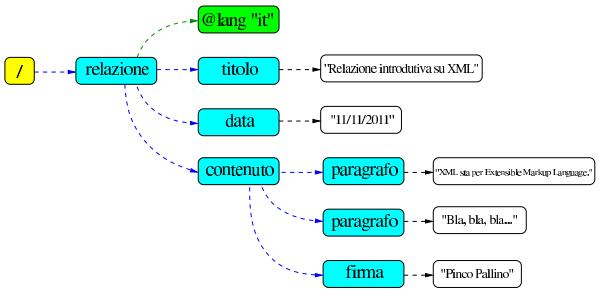
La trattazione che qui viene fatta a proposito dei fogli di stile XSLT è limitata alle funzionalità principali. Per un approfondimento si può consultare la documentazione elencata nella bibliografia alla fine del capitolo.

52.2.1 Struttura ad albero, percorsi e modelli di confronto

Un documento XML, ai fini della sua elaborazione, viene visto come una struttura ad albero, dove ogni componente rappresenta un nodo di questo. Si osservi l'esempio seguente, che rappresenta un file XML molto semplice:

```
<?xml version="1.0" encoding="UTF-8" ?>
<relazione lang="it">
<titolo>Relazione introduttiva su XML</titolo>
<data>11/11/2011</data>
<contenuto>
<paragrafo>XML sta per Extensible Markup Language.</paragrafo>
<paragrafo>Bla, bla, bla...</paragrafo>
<firma>Pinco Pallino</firma>
</contenuto>
</relazione>
```

Si potrebbe rappresentare schematicamente l'albero di questo documento come si vede nella figura successiva:



Nello schema mostrato si deve osservare che elementi, attributi e testo contenuto negli elementi, costituiscono nodi separati.

Per identificare un nodo dell'albero, si usa una notazione che assomiglia ai percorsi dei file system Unix. A titolo di esempio vengono mostrati tutti gli elementi e gli attributi XML degli schemi già apparsi, secondo la sequenza originale, con percorsi assoluti:

```
/
/relazione
/relazione/@lang
/relazione/titolo
/relazione/data
/relazione/contenuto
/relazione/contenuto/paragrafo
/relazione/contenuto/paragrafo
/relazione/contenuto/firma
```

Naturalmente esistono anche percorsi relativi, quando manca la barra obliqua iniziale che rappresenta la radice: questi fanno riferimento al nodo corrente nell'ambito del contesto a cui ci si riferisce.

La sintassi con cui si possono definire questi percorsi è stabilita dal linguaggio XPath, ovvero *XML path language*. Si tratta di un sistema abbastanza complesso che non viene mostrato qui nel dettaglio. Per quanto riguarda il linguaggio XSLT i percorsi vengono usati per definire un modello di confronto con i nodi di un documento XML; nell'ambito di questi modelli di confronto si utilizzano delle notazioni particolari rispetto alla convenzione generale costituita da XPath. La tabella 52.22 riporta alcuni esempi di questi modelli.

Tabella 52.22. Alcuni modelli di confronto per individuare i nodi di un documento XML.

Modello	Descrizione
`/`	Una barra obliqua da sola individua il nodo radice.
`text()`	Individua qualunque nodo di testo.
`node()`	Individua qualunque nodo, esclusi gli attributi e la radice.
`.` `self::node()`	Una di queste due notazioni, indifferentemente, individua il nodo corrente.
`..` `parent::node()`	Una di queste due notazioni, indifferentemente, individua il nodo genitore di quello corrente.
elemento `child::`***elemento***	Un nome scritto senza l'inserzione di simboli speciali rappresenta un elemento. La seconda notazione è completa.
***elemento*`[`*n*`]`	Individua l'***n***-esimo elemento con quel nome, che si trova all'interno dell'elemento che lo contiene. Osservando gli esempi mostrati in precedenza, '**`paragrafo[1]`**' può rappresentare il primo elemento '**`paragrafo`**' che si trova all'interno dell'elemento '**`contenuto`**'.
@attributo `attribute::`***attributo***	Un nome preceduto da una chiocciolina rappresenta un attributo. La seconda notazione è completa.
`*`	L'asterisco corrisponde a qualunque elemento.
`@*`	L'asterisco preceduto da una chiocciolina corrisponde a qualunque attributo.
***elemento_1*`	`*elemento_2***	I nomi di due elementi, separati da una barra verticale, indicano la corrispondenza con l'uno o con l'altro.
***@attributo_1*`	`*@attributo_2***	I nomi di due attributi, separati da una barra verticale, indicano la corrispondenza con l'uno o con l'altro.
elemento_1*`/`*elemento_2	La barra obliqua separa i nodi di un percorso; in questo caso, si vuole individuare l'elemento ***elemento_2*** che è contenuto da ***elemento_1***.
elemento*`/`*@attributo	In questo caso si vuole fare riferimento a un attributo di un certo elemento.
elemento_1*`//`*elemento_2 ***elemento_1***`↩` `↪/descendant-or-self::node()/`↩ `↪`***elemento_2***	Due barre oblique indicano una discendenza non meglio precisata: qui, ***elemento_2*** è contenuto all'interno di ***elemento_1***, direttamente oppure all'interno di annidamenti ulteriori.

52.2.2 Struttura generale del foglio di stile XSLT

Un foglio di stile XSLT è un documento XML con una struttura particolare, abbinato al dominio applicativo *http://www.w3.org/1999/XSL/Transform* (sezione 52.1.11). Normalmente, il prefisso associato a questo dominio applicativo è '**`xsl`**', pertanto, l'elemento principale è '**`xsl:stylesheet`**', oppure '**`xsl:transform`**', indifferentemente:

```
<xsl:stylesheet
    version="1.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    [altri_attributi]>
    [contenuto]
</xsl:stylesheet>
```

```
<xsl:transform
    version="1.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    [altri_attributi]>
    [contenuto]
</xsl:transform>
```

Anche senza definire esplicitamente alcun tipo di trasformazione, si ottiene ugualmente un risultato elaborando un file XML, attraverso delle regole di trasformazione predefinite, con le quali, in buona sostanza, si ottiene il testo del file XML, senza i marcatori che delimitano gli elementi. Per esempio, si può prendere il file XML già presentato all'inizio del capitolo, associandogli il foglio di stile seguente:

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

Supponendo che il file XML si chiami '`prova.xml`' e che il foglio di stile sia '`prova.xsl`', si potrebbe usare Xalan nel modo seguente:

```
$ xalan -IN prova.xml -XSL prova.xsl -OUT prova.txt [Invio]
```

Ecco cosa si ottiene nel file '`prova.txt`':

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

Relazione introduttiva su XML
11/11/2011

XML sta per Extensible Markup Language.
Bla, bla, bla...
Pinco Pallino
```

52.2.3 Modelli di confronto e trasformazione

La definizione di un criterio di trasformazione dei nodi del documento XML di origine in quello che si vuole ottenere, avviene principalmente per mezzo di modelli di confronto, attraverso un elemento del foglio di stile denominato '**`xsl:template`**'. La traduzione del termine *template*, ovvero «mascherina», rende bene l'idea del concetto: gli elementi '**`xsl:template`**' del foglio di stile definiscono un modello di confronto con cui selezionano alcuni nodi del documento XML di origine; su questi nodi applicano delle trasformazioni.

```
<xsl:template
    match="modello_di_confronto"
    [altri_attributi]>
    trasformazione
</xsl:template>
```

Il modello di confronto viene definito secondo le regole che in parte sono state descritte nella tabella 52.22; per esempio, il blocco seguente individua il nodo radice:

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

Per comprendere meglio cosa accade, si prenda il solito esempio di file XML già considerato in precedenza e si applichi il foglio di stile seguente:


```
<xsl:stylesheet
    version="1.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
    <xsl:template match="/relazione/contenuto">
        ciao ciao
    </xsl:template>
</xsl:stylesheet>
```

Come si vede, è stato inserito un solo elemento '**`xml:template`**' che seleziona l'elemento '**`contenuto`**', che si trova all'interno di '**`relazione`**' (che a sua volta è l'elemento principale del documento XML). Quando si individua questo elemento, viene inserito il testo «ciao ciao». Valgono le stesse convenzioni dei nomi dei file già viste in precedenza:

```
$ xalan -IN prova.xml -XSL prova.xsl -OUT prova.txt [Invio]
```

Ecco cosa si ottiene nel file '`prova.txt`':

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

Relazione introduttiva su XML
11/11/2011

        ciao ciao
```

In pratica, le regole di trasformazione predefinite hanno inserito il contenuto degli elementi '**`titolo`**' e '**`data`**'. Quindi, l'elemento '**`contenuto`**', con tutto quello che esiste al suo interno, è stato sostituito con la stringa «ciao ciao».

Per fare in modo che vengano presi in considerazione anche gli elementi contenuti all'interno di ciò che viene individuato, si utilizza l'elemento '**`xsl:apply-templates`**', che di solito è vuoto:

```
<xsl:template
    match="modello_di_confronto"
    [altri_attributi]>
    …
    <xsl:apply-templates
        [select="modello_di_selezione"]
        [altri_attributi]>
        …
    </xsl:apply-templates>
    …
</xsl:template>
```

Per esempio, se il foglio di stile di prova viene modificato nel modo seguente:

```
<xsl:stylesheet
    version="1.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
    <xsl:template match="/relazione/contenuto">
        prima
        <xsl:apply-templates />
        dopo
    </xsl:template>
</xsl:stylesheet>
```

La trasformazione genera questo file:

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

Relazione introduttiva su XML
11/11/2011

        prima

XML sta per Extensible Markup Language.
Bla, bla, bla...
Pinco Pallino

        dopo
```

In pratica, si consente alle regole di trasformazione predefinite (dal momento che non ce ne sono altre nell'esempio) di occuparsi degli elementi contenuti all'interno dell'elemento '**`contenuto`**', cosa che produce semplicemente l'estrazione del testo che questi circoscrivono. Tutto questo avviene perché «l'istruzione» '**`apply-templates`**'

serve proprio a richiamare gli altri modelli di confronto (e trasformazione) per quanto è contenuto in ciò che è stato individuato.

Dal momento che nell'esempio non ci sono altri modelli di confronto e trasformazione, è evidente che si tratta soltanto di quelli predefiniti.

L'elemento '**xsl:apply-templates**' può avere un attributo molto importante, '**select**', che consente di specificare su cosa continuare il confronto con altri modelli; in pratica consente di limitare, o controllare, la ricorsione. Si osservi la variante seguente del foglio di stile:

```
<xsl:stylesheet
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    version="1.0">
    <xsl:template match="/relazione/contenuto">
        prima
        <xsl:apply-templates select="paragrafo" />
        dopo
    </xsl:template>
</xsl:stylesheet>
```

Per prima cosa è bene osservare che il modello indicato con l'attributo '**select**' rappresenta un percorso relativo, che secondo il contesto è riferito esattamente a '`/relazione/contenuto/paragrafo`'. In base a questa selezione, si vuole che nell'ambito del contenuto dell'elemento (o degli elementi) '`/relazione/contenuto`', la ricorsione successiva prenda in considerazione soltanto gli elementi '**paragrafo**'. Ecco cosa si ottiene; come si vede, il testo dell'elemento '**firma**' è stato ignorato:

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

Relazione introduttiva su XML
11/11/2011

        prima
        XML sta per Extensible Markup Language.Bla, bla, bla...
        dopo
```

Naturalmente, è possibile utilizzare '**xsl:apply-templates**' più volte all'interno dello stesso modello di confronto e trasformazione, anche così:

```
<xsl:stylesheet
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    version="1.0">
    <xsl:template match="/relazione/contenuto">
        Io sottoscritto, <xsl:apply-templates select="firma" />,
        affermo che:
        <xsl:apply-templates select="paragrafo" />

        Firmato:
        <xsl:apply-templates select="firma" />
    </xsl:template>
</xsl:stylesheet>
```

Ecco il risultato che si può ottenere:

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

Relazione introduttiva su XML
11/11/2011

        Io sottoscritto, Pinco Pallino,
        affermo che:
        XML sta per Extensible Markup Language.Bla, bla, bla...

        Firmato:
        Pinco Pallino
```

I modelli di confronto e trasformazione possono basarsi su riferimenti relativi ai nodi, se non ha importanza la collocazione esatta di questi nell'albero del documento XML di origine. Si osservi l'esempio seguente:


```
<xsl:stylesheet
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    version="1.0">
    <xsl:template match="paragrafo">
        "<xsl:apply-templates />"
    </xsl:template>
</xsl:stylesheet>
```

In questo caso, si evidenziano gli elementi '**paragrafo**', attorno ai quali si vuole appaiano gli apici doppi; tutto il resto viene gestito in modo predefinito:

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

Relazione introduttiva su XML
11/11/2011


        "XML sta per Extensible Markup Language."


        "Bla, bla, bla..."

Pinco Pallino
```

Come accennato, l'elemento '**xsl:apply-templates**' del foglio di stile XSLT, normalmente è vuoto. Quando contiene qualcosa, ciò serve per elaborare il risultato della scansione che lo riguarda. In pratica, '**xsl:apply-templates**' serve a richiamare i modelli di confronto e trasformazione successivi, in modo da inserire il risultato di queste elaborazioni nel punto in cui si trova, ma se questo elemento non è vuoto, le «istruzioni» che contiene servono a interferire con le elaborazioni successive, per esempio per riordinare i dati ottenuti. Questo tipo di utilizzo di '**xsl:apply-templates**' non viene descritto.

52.2.4 Testo

Negli esempi delle sezioni precedenti, in varie occasioni è stato inserito del testo aggiuntivo nella trasformazione del documento XML di origine. In questo modo è possibile anche aggiungere marcatori e altre componenti, in base agli scopi che ci si prefigge con la trasformazione.

Quando un elemento contiene solo spazi bianchi (spazi veri e propri, tabulazioni orizzontali e interruzioni di riga), il nodo relativo può essere eliminato dalla struttura. In generale, salvo che sia specificato diversamente, nel documento XML di origine vengono preservati tutti gli spazi, mentre nel foglio di stile sono preservati solo quelli contenuti negli elementi '**xsl:text**'.

52.2.4.1 Formato generale del risultato della trasformazione

Esistono diverse modalità di funzionamento a cui si adegua la trasformazione del documento XML, in base al tipo di file che si vuole ottenere. Questa modalità di funzionamento si seleziona con l'elemento '**xsl:output**':

```
<xsl:output
    method="metodo"
    version="versione"
    encoding="codifica"
    omit-xml-declaration="yes"|"no"
    doctype-public="dichiarazione_dtd_public"
    doctype-system="dichiarazione_dtd_system"
    [altri_attributi] />
```

Come si vede dal modello sintattico, si tratta di un elemento del foglio di stile che non ha lo scopo di contenere qualcosa e tutto è indicato attraverso attributi.

L'attributo più importante è '**method**', al quale si associano normalmente le parole chiave '**xml**', '**html**' e '**text**', per indicare rispettivamente che si vuole ottenere un risultato in formato XML, HTML o un file di testo senza una conformazione particolare conosciuta. In

generale, se non si specifica il formato del file che si va a generare, si intende XML, cosa che dovrebbe spiegare il motivo per cui negli esempi mostrati in precedenza appare la dichiarazione XML anche se il file ottenuto è semplicemente un testo puro e semplice.

Come si può intuire, a seconda del tipo di «metodo» prescelto, gli altri attributi possono acquisire o perdere significato.

Tabella 52.36. Alcuni attributi di '**xsl:output**' utili quando il risultato dell'elaborazione deve essere un file XML.

Attributo	Descrizione
`method="xml"`	Dichiara il metodo di trasformazione in un documento XML. Alcuni caratteri (come '**&**' e '**<**') vengono trasformati nelle rispettive entità generali standard.
`version="`***versione***`"`	Dichiara il numero di versione del formato XML. Il valore predefinito è '**1.0**'.
`encoding="`***codifica***`"`	Dichiara il tipo di codifica del file da generare. Sono valide sempre le sigle '**UTF-8**' e '**UTF-16**', ma possono essere disponibili altre sigle, come per esempio '**ISO-8859-*n***'.
`omit-xml-declaration=` `"yes"|"no"`	Consente di omettere (se si assegna il valore '**yes**') la dichiarazione XML iniziale.
`indent="yes"|"no"`	Permette di richiedere un incolonnamento degli elementi nel risultato della trasformazione.
`doctype-public="`***dtd_public***`"`	Permette di specificare la dichiarazione del DTD, utilizzando un identificatore pubblico, come potrebbe essere: '**-//W3C//DTD XHTML 1.1//EN**'.
`doctype-system="`***dtd_system***`"`	Permette di specificare la dichiarazione del DTD, indicando come riferimento un file.

Tabella 52.37. Alcuni attributi di '**xsl:output**' utili quando il risultato dell'elaborazione deve essere un file HTML.

Attributo	Descrizione
`method="html"`	Dichiara il metodo di trasformazione in un documento HTML. Alcuni caratteri vengono trasformati usando le entità generali standard.
`version="`***versione***`"`	Dichiara il numero di versione del formato HTML. Il conoscere la versione permette di verificare la validità del foglio di stile anche in base alla validità del codice HTML che si va a generare. Il valore predefinito è '**4.0**'.
`encoding="`***codifica***`"`	Dichiara il tipo di codifica del file da generare. Sono valide sempre le sigle '**UTF-8**' e '**UTF-16**', ma possono essere disponibili altre sigle, come per esempio '**ISO-8859-*n***'.
`doctype-public="`***dtd_public***`"`	Permette di specificare la dichiarazione del DTD, utilizzando un identificatore pubblico, come potrebbe essere: '**ISO/IEC 15445:2000//DTD HTML//EN**'.
`doctype-system="`***dtd_system***`"`	Permette di specificare la dichiarazione del DTD, indicando come riferimento un file.

Tabella 52.38. Alcuni attributi di '**xsl:output**' utili quando il risultato dell'elaborazione deve essere un file di testo non meglio identificabile.

Attributo	Descrizione
`method="text"`	Dichiara il metodo di trasformazione in un file di testo. In questo caso non avviene alcuna trasformazione dei caratteri, come invece può avvenire con le trasformazioni in XML o in HTML.
`encoding="`***codifica***`"`	Dichiara il tipo di codifica del file da generare. Sono valide sempre le sigle '**UTF-8**' e '**UTF-16**', ma possono essere disponibili altre sigle, come per esempio '**ISO-8859-*n***'.

52.2.4.2 Inserimento di testo aggiuntivo

Generalmente è sufficiente aggiungere del testo estraneo all'interno del foglio di stile XSLT, nell'ambito degli elementi '**xsl:template**', per fare in modo che questo venga inserito nel risultato finale. Per avere un controllo maggiore si può usare l'elemento '**xsl:text**', che tra le altre cose permette di inserire blocchi di spazi quando diversamente verrebbero eliminati:

```
<xsl:text
    [disable-output-escaping="yes"|"no"]>
    testo
</xsl:text>
```

L'attributo '**disable-output-escaping**' permette, se si assegna il valore '**yes**', di disabilitare la sostituzione di alcuni caratteri in entità generali standard (questo vale solo per le trasformazioni che prevedono nella destinazione un formato XML o HTML, perché nel caso di trasformazioni in formato testo, questa sostituzione non viene mai eseguita). Come si può intendere, la funzionalità è disabilitata in modo predefinito, pertanto, normalmente si ottiene la sostituzione di questi caratteri.

In alcune circostanze è necessario inserire del testo nel foglio di stile XSLT che non deve essere interpretato prima della trasformazione, lasciandolo in modo letterale. Per questo si usa normalmente una sezione marcata di tipo CDATA, come nell'esempio seguente, dove si vede l'intenzione di inserire uno stile CSS nel documento finale che è un file HTML:

```
    <xsl:template match="/">
        <HTML>
        <HEAD>
            <TITLE>senza titolo</TITLE>
        <xsl:text disable-output-escaping="yes">
<![CDATA[<STYLE TYPE="text/css">
<!--
BODY {
    background-color: rgb(255, 255, 255)
}
H1 {
    text-align:       right;
    color:            rgb(230, 100, 180)
}
-->
</STYLE>]]>
        </HEAD>
        <BODY>
        <xsl:apply-templates />
        </BODY>
    </xsl:template>
```

52.2.4.3 Estrazione del valore contenuto in un nodo

Il testo contenuto in un nodo viene inserito nel documento finale attraverso l'elemento '**xsl:value-of**' del foglio di stile. Esiste anche un modello di confronto e trasformazione predefinito che esegue questa operazione per tutti i nodi di testo, pertanto, negli esempi di fogli di stile non si è ancora presentata la necessità di mostrarne l'uso:

```
<xsl:value-of
    select="modello_di_selezione"
    [disable-output-escaping="yes"|"no"] />
```

Come si vede dal modello sintattico, l'elemento '**xsl:value-of**' si usa vuoto e, per ottenere il «valore» di qualcosa, occorre specificar-

lo attraverso un modello di selezione, come avviene con l'elemento '**xsl:apply-templates**'.

Anche questo elemento prevede l'attributo '**disable-output-escaping**' come descritto a proposito di '**xsl:text**'.

L'elemento '**xsl:value-of**' si può usare per estrarre il testo contenuto in un elemento del documento di origine, oppure per fare altrettanto da un attributo; in generale, è più frequente l'uso di '**xsl:value-of**' per estrarre il testo di un attributo, come nell'esempio seguente:

```
<xsl:template match="relazione">
    Questo documento utilizza il linguaggio <xsl:value-of
    select="@lang" />.
</xsl:template>
```

In questo caso, quando si raggiunge un elemento denominato '**relazione**', nel documento XML di origine, viene estratto il valore dell'attributo '**lang**' di questo e inserito in una frase, ignorando qualunque altra cosa che possa riguardare l'elemento in questione.

Quando il documento di destinazione è di tipo XML o HTML, è probabile che si vadano a descrivere dei marcatori che contengono la dichiarazione di attributi. Quando negli attributi si vuole inserire un valore estratto da un nodo del documento XML di origine, non si può usare l'elemento '**xsl:value-of**'; al suo posto si usano delle parentesi graffe, come nell'esempio seguente:

```
<xsl:template match="relazione">
    <HTML LANG="{@lang}">
    <BODY>
    <xsl:apply-templates />
    </BODY>
    </HTML>
</xsl:template>
```

Questa volta, come si può vedere, si va a costruire un documento HTML, dove serve il valore dell'attributo '**lang**' dell'elemento '**relazione**' del file di partenza.

52.2.5 Creazione di componenti XML e HTML

Quando il formato di destinazione della trasformazione è un file XML o HTML, è possibile creare alcune componenti tipiche di questi file con l'ausilio di elementi appositi nel foglio di stile XSLT.

52.2.5.1 Creazione di elementi e di attributi

Quando per qualche ragione è difficile inserire letteralmente il testo che rappresenta i marcatori di un elemento nel documento finale, si può usare nel foglio di stile XSLT l'elemento '**xsl:element**':

```
<xsl:element
    name="nome_elemento"
    [altri_attributi]>
    contenuto
</xsl:element>
```

Per dichiarare un attributo si può usare l'elemento '**xsl:attribute**':

```
<xsl:attribute
    name="nome_attributo"
    [altri_attributi]>
    contenuto
</xsl:element>
```

L'esempio seguente mostra l'uso di questi due elementi, per la costruzione della prima parte di un documento HTML, dove in particolare si vede anche l'uso di '**xsl:value-of**':

```
<xsl:template match="/">
    <xsl:element name="HTML">
        <xsl:attribute name="LANG">
            <xsl:value-of select="relazione/@lang" />
        </xsl:attribute>
        <HEAD>
        <TITLE><xsl:value-of select="relazione/titolo" /></TITLE>
        </HEAD>
        <BODY>
        <xsl:apply-templates />
        </BODY>
    </xsl:element>
</xsl:template>
```

Osservando l'esempio si intende che il documento XML di origine contiene l'elemento '**relazione**' all'inizio della gerarchia; inoltre al suo interno si trova l'elemento '**titolo**' che viene usato per completare l'intestazione del file HTML.

52.2.5.2 Creazione di commenti e di istruzioni di elaborazione

Quando il formato di destinazione è di tipo XML o HTML è possibile inserire dei commenti attraverso l'elemento '**xsl:comment**' nel foglio di stile XSLT:

```
<xsl:comment>
    contenuto
</xsl:comment>
```

In modo analogo, è possibile inserire istruzioni di elaborazione quando il formato di destinazione è di tipo XML, con l'elemento '**xsl:processing-instruction**':

```
<xsl:processing-instruction
    name="nome_istruzione">
    contenuto
</xsl:processing-instruction>
```

Si osservi l'esempio seguente:

```
<xsl:template match="/">
    <xsl:processing-instruction name="xml-stylesheet">href="esempio.css"
    type="text/css"</xsl:processing-instruction>
    <xsl:comment>Ecco un commento</xsl:comment>
    <xsl:apply-templates />
</xsl:template>
```

In questo modo, all'inizio del documento di destinazione si ottiene il testo seguente:

```
<?xml-stylesheet href="esempio.css"
    type="text/css"?>
<!--Ecco un commento-->
```

52.2.6 Xalan

Xalan[1] è un elaboratore XSLT disponibile sia in Java, sia in C++. Qui si fa riferimento all'uso di Xalan-C++, ovvero a un programma compilato in modo da avere un eseguibile che non richiede altre forme di interpretazione:

```
xalan -IN documento_xml_originale -XSL foglio_di_stile ↩
↪     -OUT file_da_generare [altre_opzioni]
```

Il modello sintattico mostrato è più che sufficiente per usare bene Xalan; per le altre opzioni disponibili si può consultare la pagina di manuale *xalan(1)*.

Durante l'elaborazione, Xalan emette alcune informazioni ed eventualmente delle segnalazioni di errore, che dovrebbero tornare utili per correggere il foglio di stile XSLT. Si osservi che quando si procede a una trasformazione che deve generare un documento XML o HTML, Xalan si sofferma anche su errori relativi al formato finale. Per esempio, l'estratto seguente, riferito proprio alla generazione di codice HTML genera un errore a causa della mancata chiusura dell'elemento '**HEAD**' nel file che si ottiene:

```
5   <xsl:template match="/">
6       <HTML>
7       <HEAD>
8       <TITLE><xsl:value-of select="relazione/titolo" /></TITLE>
9       <BODY>
10      <xsl:apply-templates />
11      </BODY>
12  </xsl:element>
```

Supponendo che il file XML originale sia ‘`prova.xml`’, che il foglio di stile XSLT sia contenuto nel file ‘`prova.xsl`’ e che si voglia generare il file ‘`prova.html`’, si dovrebbe procedere con il comando seguente:

```
$ xalan -IN prova.xml -XSL prova.xsl -OUT prova.html [Invio]
```

In questo caso, si ottiene l’errore già descritto:

```
========= Parsing prova.xsl ==========

Fatal Error at (file prova.xsl, line 12, column 18): ↩
↪Expected end of tag 'HEAD'

SAXParseException Message is: Expected end of tag 'HEAD' ↩
↪(prova.xsl, line 12, column 18)
```

I riferimenti ai numeri di riga dell’esempio sono corretti, pertanto si può osservare che gli errori vengono segnalati in posizioni abbastanza lontane rispetto alla loro collocazione effettiva.

52.2.7 Esempio completo

Nelle sezioni seguenti vengono mostrati diversi fogli di stile XSLT per ottenere altrettante trasformazioni a partire da un file XML già mostrato in precedenza. Qui viene riportato nuovamente, con qualche piccola modifica e con l’aggiunta della dichiarazione del DTD incorporata:

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE relazione [
    <!ELEMENT relazione     (titolo?, data, contenuto)>
    <!ATTLIST relazione
        xml:lang   CDATA    "">
    <!ELEMENT titolo        (#PCDATA)>
    <!ELEMENT data          (#PCDATA)>
    <!ELEMENT contenuto     (paragrafo+, firma+)>
    <!ELEMENT paragrafo     (#PCDATA)>
    <!ELEMENT firma         (#PCDATA)>
]>

<relazione lang="it">
<titolo>Relazione introduttiva su XML</titolo>

<data>11/11/2011</data>

<contenuto>

<paragrafo>XML sta per Extensible Markup Language.    bla
bla bla... Perché,... così,... perciò,...
sarà...</paragrafo>

<paragrafo>Bla, bla, bla....</paragrafo>

<firma>Pinco Pallino</firma>

</contenuto>
</relazione>
```

Si osservi che questo file, come si vede dalla dichiarazione iniziale, deve usare la codifica UTF-8; di conseguenza, le lettere accentate utilizzano più di un byte per essere rappresentate.

52.2.7.1 Trasformazione in un sorgente LaTeX

Per la trasformazione in un sorgente LaTeX si deve utilizzare un foglio di stile XSLT che elabora il risultato in modalità testo:


```
<xsl:stylesheet
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    version="1.0">
    <xsl:output method="text" encoding="UTF-8" />
    <xsl:template match="/">
\documentclass{article}
\usepackage[utf8]{inputenc}
\usepackage[italian,english]{babel}
\begin{document}
            <xsl:apply-templates />
\end{document}
        </xsl:template>
        <xsl:template match="titolo">
\section{<xsl:apply-templates />}
    </xsl:template>
</xsl:stylesheet>
```

Si può osservare che non sono state stabilite delle regole di trasformazione per gli elementi ‘**`data`**’, ‘**`paragrafo`**’ e ‘**`firma`**’, perché allo scopo risultano sufficienti le regole predefinite. Inoltre, le parentesi graffe sono usate fuori dal contesto in cui servono per ottenere il valore di qualcosa, pertanto hanno soltanto un significato letterale nell’ambito della trasformazione. Ecco cosa si ottiene:

```
\documentclass{article}
\usepackage[utf8]{inputenc}
\usepackage[italian,english]{babel}
\begin{document}


\section{Relazione introduttiva su XML}


11/11/2011



XML sta per Extensible Markup Language.    bla
bla bla... Perché,... così,... perciò,...
sarà...

Bla, bla, bla....

Pinco Pallino



\end{document}
```

Il file che si ottiene utilizza la codifica UTF-8, cosa che consente di evitare l’uso di comandi particolari per rappresentare le lettere accentate, pertanto si è reso necessario l’utilizzo di un «pacchetto» adatto allo scopo (‘**`\usepackage[utf8]{inputenc}`**’). Tuttavia rimangono da risolvere altri problemi legati a caratteri che non possono essere inseriti letteralmente, come per esempio nel caso di ‘**`%`**’, che per LaTeX costituisce l’inizio di un commento. Per questo occorrerebbe inserire nel DTD la dichiarazione di una serie di entità generali, che poi devono essere usate nel sorgente XML.

Infine, si noti che, per semplicità, nella trasformazione viene ignorato completamente il linguaggio, ovvero l’attributo ‘**`lang`**’ dell’elemento ‘**`relazione`**’.

52.2.7.2 Trasformazione in un sorgente HTML

Per la trasformazione in un sorgente HTML si deve utilizzare un foglio di stile XSLT che elabora il risultato in modalità HTML, avendo cura, possibilmente, di predisporre anche la dichiarazione del DTD:

```
<xsl:stylesheet
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    version="1.0">
    <xsl:output
        method="html"
        doctype-public="ISO/IEC 15445:2000//DTD HTML//EN"
        encoding="ISO-8859-1" />
    <xsl:template match="/">
        <xsl:element name="HTML">
            <xsl:attribute name="LANG">
                <xsl:value-of select="relazione/@lang" />
            </xsl:attribute>
```

```
            <HEAD>
            <TITLE><xsl:value-of select="relazione/titolo" /></TITLE>
            <xsl:text disable-output-escaping="yes">
<![CDATA[<STYLE TYPE="text/css">
<!--
BODY {
    background-color: rgb(255, 255, 255)
}
H1 {
    text-align:       right;
    color:            rgb(230, 100, 180)
}

P.paragrafo {
    width:            auto;
    text-align:       left
}
P.data {
    text-align:       left;
    font-weight:      bold
}
P.firma {
    width:            auto;
    text-align:       left
-->
</STYLE>]]>
            </xsl:text>
            </HEAD>
            <BODY>
            <xsl:apply-templates />
            </BODY>
        </xsl:element>
    </xsl:template>
    <xsl:template match="titolo">
        <H1><xsl:apply-templates /></H1>
    </xsl:template>
    <xsl:template match="data">
        <P CLASS="data"><xsl:apply-templates /></P>
    </xsl:template>
    <xsl:template match="paragrafo">
        <P CLASS="paragrafo"><xsl:apply-templates /></P>
    </xsl:template>
    <xsl:template match="firma">
        <P CLASS="firma"><xsl:apply-templates /></P>
    </xsl:template>
</xsl:stylesheet>
```

Questa volta il foglio di stile XSLT è molto più articolato, anche perché incorpora la dichiarazione di alcuni stili CSS. Ecco il risultato che si ottiene; si osservi che il file viene generato usando la codifica ISO-8859-1:

```
<!DOCTYPE HTML PUBLIC "ISO/IEC 15445:2000//DTD HTML//EN">
<HTML LANG="it">
<HEAD>
<META http-equiv="Content-Type" content="text/html;
      charset=ISO-8859-1">
<TITLE>Relazione introduttiva su XML</TITLE>
<STYLE TYPE="text/css">
<!--
BODY {
    background-color: rgb(255, 255, 255)
}
H1 {
    text-align:       right;
    color:            rgb(230, 100, 180)
}

P.paragrafo {
    width:            auto;
    text-align:       left
}
P.data {
    text-align:       left;
    font-weight:      bold
}
P.firma {
    width:            auto;
    text-align:       left
-->
</STYLE>
            </HEAD>
<BODY>
<H1>Relazione introduttiva su XML</H1>

<P CLASS="data">11/11/2011</P>



<P CLASS="paragrafo">XML sta per Extensible Markup Language.    bla
bla bla... Perch&eacute;,... cos&igrave;,... perci&ograve;,...
```



```
sar&agrave;...</P>

<P CLASS="paragrafo">Bla, bla, bla....</P>

<P CLASS="firma">Pinco Pallino</P>


</BODY>
</HTML>
```

52.2.7.3 Trasformazione in un sorgente XHTML

Per la trasformazione in un sorgente XHTML si deve utilizzare un foglio di stile XSLT che elabora il risultato in modalità XML:

```
<xsl:stylesheet
    version="1.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns="http://www.w3.org/1999/xhtml">
    <xsl:output
        method="xml"
        encoding="UTF-8" />
    <xsl:template match="/">
        <xsl:element name="html">
            <xsl:attribute name="xml:lang">
                <xsl:value-of select="relazione/@lang" />
            </xsl:attribute>
            <head>
            <title><xsl:value-of select="relazione/titolo" /></title>
            <xsl:text disable-output-escaping="yes">
<![CDATA[<style type="text/css">
<!--
body {
    background-color: rgb(255, 255, 255)
}
h1 {
    text-align:       right;
    color:            rgb(230, 100, 180)
}

p.paragrafo {
    width:            auto;
    text-align:       left
}
p.data {
    text-align:       left;
    font-weight:      bold
}
p.firma {
    width:            auto;
    text-align:       left
-->
</style>]]>
            </xsl:text>
            </head>
            <body>
            <xsl:apply-templates />
            </body>
        </xsl:element>
    </xsl:template>
    <xsl:template match="titolo">
        <h1><xsl:apply-templates /></h1>
    </xsl:template>
    <xsl:template match="data">
        <p class="data"><xsl:apply-templates /></p>
    </xsl:template>
    <xsl:template match="paragrafo">
        <p class="paragrafo"><xsl:apply-templates /></p>
    </xsl:template>
    <xsl:template match="firma">
        <p class="firma"><xsl:apply-templates /></p>
    </xsl:template>
</xsl:stylesheet>
```

Il risultato che si ottiene manca di alcuni incolonnamenti che nelle altre situazioni venivano mantenuti. Per ovviare all'inconveniente, si può provare ad aggiungere l'attributo '**`indent="yes"`**' nell'elemento '**`xsl:output`**':

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

Ecco il risultato, incolonnato, che si può ottenere:

```
<?xml version="1.0" encoding="UTF-8"?>
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="it">
<head>
<title>Relazione introduttiva su XML</title>
<style type="text/css">
<!--
body {
    background-color: rgb(255, 255, 255)
}
h1 {
    text-align:       right;
    color:            rgb(230, 100, 180)
}

p.paragrafo {
    width:            auto;
    text-align:       left
}
p.data {
    text-align:       left;
    font-weight:      bold
}
p.firma {
    width:            auto;
    text-align:       left
-->
</style>
            </head>
<body>
<h1>Relazione introduttiva su XML</h1>

<p class="data">11/11/2011</p>



<p class="paragrafo">XML sta per Extensible Markup Language.    bla
bla bla... Perché,... così,... perciò,...
sarà...</p>

<p class="paragrafo">Bla, bla, bla....</p>

<p class="firma">Pinco Pallino</p>


</body>
</html>
```

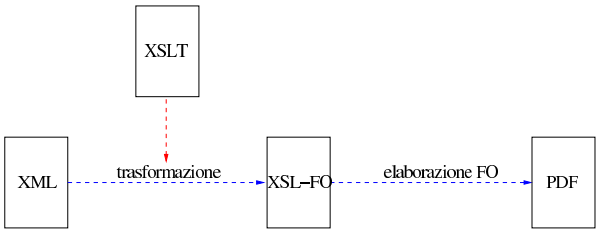
Si ricorda che in questo caso la codifica del file generato è UTF-8, inoltre, si fa notare che nel foglio di stile XSLT appare un dominio applicativo predefinito, allo scopo di riportarlo nel file XHTML di destinazione.

52.3 Introduzione a XSL-FO

XSL-FO (*XSL formatting object*) è un linguaggio realizzato in forma di file XML, con il quale si definisce l'aspetto finale di un documento. In pratica assomiglia concettualmente a HTML o a XHTML, con la differenza che vengono specificati in modo abbastanza preciso i dettagli sulla sua impaginazione per la stampa e generalmente sul suo aspetto finale.

XSL-FO nasce per essere utilizzato prevalentemente nell'ambito di un foglio di stile XSLT, con il quale si definisce la trasformazione di un documento XML in un file XSL-FO. Teoricamente, un navigatore dovrebbe essere in grado di utilizzare direttamente un foglio di stile XSLT che definisce questo tipo di trasformazione; in pratica, inizialmente si esegue la trasformazione per ottenere un file XSL-FO, il quale poi viene convertito in un formato finale comune (di solito è il formato PDF).

Figura 52.55. Passaggi normali per l'utilizzo di XSL-FO.



Nelle sezioni successive vengono mostrati prevalentemente esempi di file scritti secondo il formato XSL-FO, senza usare un foglio di stile XSLT, utilizzando FOP come programma per l'elaborazione, allo scopo di produrre il formato finale per la stampa.

52.3.1 Preparazione degli strumenti

Per poter ottenere la composizione finale di un file in formato XSL-FO occorre un elaboratore FO che generi un altro file più adatto alla consultazione e alla stampa. Qui viene mostrato l'uso di FOP,[2] che assieme a Xalan fa parte del progetto XML di Apache.

A differenza di Xalan, FOP è disponibile solo come programma Java, da interpretare anche una volta compilato. Gli esempi mostrati in questo capitolo sono stati verificati con FOP, eseguito attraverso Kaffe[3].

Una volta installato FOP in un sistema GNU, dovrebbe essere disponibile uno script che consente di avviare il programma senza difficoltà:

```
fop [opzioni] [-fo|-xml] file_ingresso [-xsl file_xsl] ↩
↪ [-pdf|-txt|-altro_formato] file_uscita
```

Se non fosse disponibile questo script, lo si potrebbe riprodurre facilmente utilizzando un contenuto simile a quello seguente:

```
#!/bin/sh
FOPPATH="/usr/share/java/fop.jar"
FOPPATH="$FOPPATH:/usr/share/java/xalan2.jar"
FOPPATH="$FOPPATH:/usr/share/java/xerces.jar"
FOPPATH="$FOPPATH:/usr/share/java/logkit.jar"
FOPPATH="$FOPPATH:/usr/share/java/avalon-framework.jar"
FOPPATH="$FOPPATH:/usr/share/java/batik.jar"
FOPPATH="$FOPPATH:/usr/share/java/jimi-1.0.jar"
CLASSPATH="$FOPPATH:$CLASSPATH"
export CLASSPATH

java $JAVA_OPTS org.apache.fop.apps.Fop "$@"
```

Come si può intendere, in questo caso si considera che i file Java di FOP e di Xalan siano contenuti nella directory '`/usr/share/java/`'; inoltre, viene considerato il contenuto della variabile di ambiente ***JAVA_OPTS*** per le opzioni da passare all'interprete Java. Infine, per quanto riguarda le prove effettuate, '**`java`**' è in realtà un collegamento all'eseguibile che svolge il compito di interpretazione dei programmi Java (per esempio '`/usr/lib/kaffe/bin/java`', oppure '`/usr/bin/kaffe`').

Per verificare subito il funzionamento di FOP si può provare con il sorgente XSL-FO seguente:

```
<?xml version="1.0" encoding="UTF-8"?>
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
    <fo:layout-master-set>
        <fo:simple-page-master master-name="prova">
            <fo:region-body />
        </fo:simple-page-master>
    </fo:layout-master-set>
    <fo:page-sequence master-reference="prova">
        <fo:flow flow-name="xsl-region-body">
            <fo:block>
                Ciao a tutti.
            </fo:block>
        </fo:flow>
    </fo:page-sequence>
</fo:root>
```

Supponendo che si tratti del file '`prova.fo`', si può generare la sua composizione con il comando seguente:

```
$ fop -fo prova.fo -pdf prova.pdf [ invio ]
```

Utilizzando Kaffe come interprete Java è probabile che appaiano molti errori non gravi; a parte questi, ciò che si dovrebbe vedere durante la composizione sono i messaggi seguenti:

```
[INFO] FOP 0.20.4
[INFO] building formatting object tree
[INFO] [1]
[INFO] Parsing of document complete, stopping renderer
```

Se non si arriva a vedere la frase finale in cui si conferma la conclusione dell'analisi e della composizione, qualcosa è andato storto. Quello che si deve ottenere è quindi il file '`prova.pdf`', contenente la scritta «Ciao a tutti.», collocata a partire dall'angolo superiore sinistro del foglio, senza alcun margine (il foglio dovrebbe avere il formato Lettera, ovvero 8,5 in × 11 in).

A parte la facilità con cui è stato utilizzato FOP per questo esempio, si possono presentare problemi difficili da comprendere, perché FOP è un progetto ancora all'inizio del suo sviluppo e non genera informazioni sufficienti a trovare gli errori sintattici nel sorgente XSL-FO. Per esempio, nel momento in cui si scrive questo capitolo, se si commette un piccolo errore come quello seguente (alla quarta riga, il marcatore iniziale dell'elemento '**`fo:simple-page-master`**' non è concluso dal simbolo '**`>`**'), i messaggi di XSL-FO non aiutano a capirlo:

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
3      <fo:layout-master-set>
4          <fo:simple-page-master master-name="prova"
5              <fo:region-body />
6          </fo:simple-page-master>
7      </fo:layout-master-set>
8      <fo:page-sequence master-reference="prova">
9          <fo:flow flow-name="xsl-region-body">
10             <fo:block>
11                 Ciao a tutti.
12             </fo:block>
13         </fo:flow>
14     </fo:page-sequence>
15 </fo:root>
```

```
$ fop -fo prova.fo -pdf prova.pdf [invio]

[INFO] FOP 0.20.4
[INFO] building formatting object tree
[ERROR] Can't find bundle for base name ↩
↪org.apache.xerces.impl.msg.XMLMessages, locale en_US
```

Date le difficoltà, può essere conveniente l'abbinamento con un DTD, da verificare prima della composizione; eventualmente si può usare quello presentato nella sezione 52.3.4. Purtroppo il DTD non può verificare tutti i vincoli reali del formato XSL-FO, ma almeno consente di evitare errori grossolani.

Se si salva il file del DTD con il nome '`fo.dtd`', nella directory corrente, si può modificare l'intestazione dei file XSL-FO nel modo seguente:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE fo:root SYSTEM "fo.dtd">
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
...
</fo:root>
```

Per l'analisi sintattica si può usare il pacchetto SP di James Clark (come descritto nella sezione 51.2), ma prima occorre procurarsi anche una dichiarazione XML standard assieme al suo catalogo. Si suppone di avere così anche i file '`xml.dcl`' e '`xml.cat`' (si riveda eventualmente quanto spiegato all'inizio del capitolo), entrambi collocati nella directory corrente, assieme al DTD. Naturalmente il file '`xml.cat`' deve contenere un riferimento al file '`xml.dcl`' nella directory corrente:

```
SGMLDECL "xml.dcl"
```

Per avviare la verifica del file '`prova.fo`' che contiene la dichiarazione corretta del DTD, si può procedere con il comando seguente:

```
$ cat prova.fo | nsgmls -s -c ./xml.cat [invio]
```

Data questa premessa, negli esempi proposti di file XSL-FO viene inserita sempre l'intestazione che dichiara il DTD (salvo eccezioni), anche se si tratta di un'informazione superflua per la composizione.

52.3.2 Struttura generale del documento secondo XSL-FO

Un foglio di stile XSL-FO è un documento XML abbinato al dominio applicativo (*namespace*) *http://www.w3.org/1999/XSL/Format* , a cui si associa normalmente il prefisso '**`fo:`**'. L'elemento principale di questo documento, ovvero quello che contiene tutti gli altri, è '**`fo:root`**':

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE fo:root SYSTEM "fo.dtd">
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
    ...
</fo:root>
```

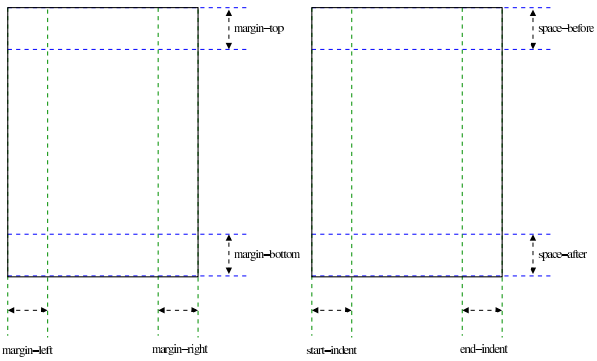
Questo elemento contiene obbligatoriamente '**`fo:layout-master-set`**' e uno o più elementi '**`fo:page-sequence`**'. Il primo serve a descrivere la pagina e l'impaginazione generale, mentre il secondo serve a incorporare il contenuto di ciò che si vuole stampare effettivamente (il testo, le immagini, ecc.):

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE fo:root SYSTEM "fo.dtd">
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
    <fo:layout-master-set>
        ...
    </fo:layout-master-set>
    <fo:page-sequence>
        ...
    </fo:page-sequence>
    ...
</fo:root>
```

52.3.3 Margini e direzione

Prima di poter descrivere la struttura della pagina secondo XSL-FO, è necessario comprendere alcuni concetti generali legati alla gestione dei margini: i «blocchi», ovvero gli oggetti che per loro natura possono essere considerati come dei rettangoli, ovvero delle scatole bidimensionali, hanno una gestione comune dei margini. Si osservi la figura 52.65.

Figura 52.65. Margini nelle scatole XSL-FO.



La figura mostra due scatole con margini definiti attraverso gruppi differenti di attributi: '**`margin-top`**', '**`margin-bottom`**', '**`margin-left`**' e '**`margin-right`**'; '**`space-before`**', '**`space-after`**', '**`start-indent`**' e '**`end-indent`**'. Il primo gruppo di attributi riguarda letteralmente il margine superiore, il margine inferiore, il margine sinistro e il margine destro, mentre il secondo gruppo tiene conto della direzione della scrittura. In pratica, la figura mostra l'uso del secondo gruppo di attributi quando la scrittura avviene da sinistra a destra e dall'alto in basso; si può comprendere intuitivamente come cambino significato questi margini se l'andamento della scrittura cambia.

Questi due gruppi di definizione dei margini sono alternativi e se si usano assieme, viene scelto un tipo di margine in base a delle precedenze. Tuttavia, in generale è meglio usare il secondo gruppo di margini che è più generale.

Per quanto riguarda la direzione, questa può essere controllata con l'attributo '**`writing-mode`**', a cui si possono attribuire stringhe come '**`lr-tb`**' e '**`rl-tb`**' (in questo caso riferite rispettivamente alla

scrittura da sinistra a destra e dall'alto in basso, oppure da destra a sinistra e dall'alto in basso). Tuttavia, è probabile che gli strumenti di elaborazione di file XSL-FO, nel loro sviluppo iniziale, non siano in grado di adeguarsi al cambiamento di direzione.

52.3.4 Un DTD per XSL-FO

All'indirizzo *allegati/sgml/fo.dtd* è disponibile un DTD realizzato appositamente durante lo studio che ha preceduto la realizzazione degli esempi che sono proposti in questi capitoli, attingendo dalle informazioni contenute nel documento *xsl:fo short reference*, il quale fa parte della documentazione di FOP, oltre che dal documento *Extensible stylesheet language (XSL)*, presso *http://www.w3.org/TR/xsl/* .

Si osservi che nel DTD manca la possibilità di utilizzare attributi generici riferiti ai bordi (per esempio '**`border`**', '**`border-color`**', ecc.), la presenza dei quali risulterebbe in un errore. Si ritiene che non sia il caso di utilizzare queste proprietà complessive come si può fare con i fogli di stile CSS; tuttavia, non sarebbe difficile aggiungere tali proprietà al DTD se lo si ritiene necessario.

52.4 XSL-FO: impaginazione

L'impaginazione secondo XSL-FO viene definita attraverso dei modelli a cui si fa riferimento per mezzo di un nome. Quando si dichiara il modello, l'elemento relativo utilizza l'attributo '**`master-name`**'; quando vi si fa riferimento, l'elemento utilizza l'attributo '**`master-reference`**'.

52.4.1 Impaginazione semplificata

La definizione minima della pagina e del contenuto si ottiene con una struttura XSL-FO simile a quella seguente, dove però non sono ancora stati annotati gli attributi obbligatori:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE fo:root SYSTEM "fo.dtd">
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
    <fo:layout-master-set>
        <fo:simple-page-master>
        ...
        </fo:simple-page-master>
        ...
        <fo:region-body />
    </fo:layout-master-set>
    <fo:page-sequence>
        <fo:flow>
        ...
        </fo:flow>
        ...
    </fo:page-sequence>
    ...
</fo:root>
```

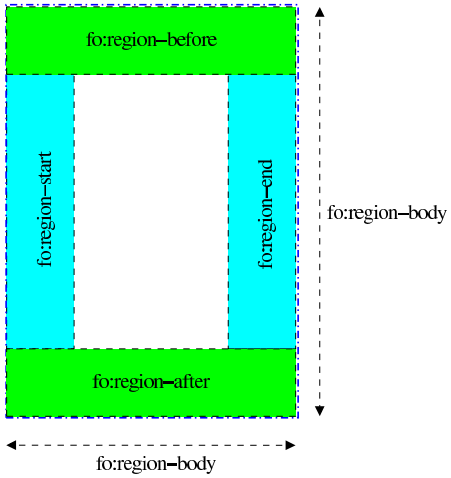
All'interno di '**`fo:layout-master-set`**' si collocano gli elementi che descrivono le caratteristiche delle pagine. Nella situazione più semplice si usano elementi '**`fo:simple-page-master`**', all'interno dei quali si definiscono delle «regioni»:

```
<fo:simple-page-master
    master-name="nome_tipo_impaginazione"
    [page-height="altezza_pagina"]
    [page-width="ampiezza_pagina"]
    [writing-mode="direzione"]
    [space-before="spazio_prima"]
    [space-after="spazio_dopo"]
    [start-indent="rientro_iniziale"]
    [end-indent="rientro_finale"]
    [altri_attributi]>

    regioni

</fo:simple-page-master>
```


Il significato degli attributi mostrati nello schema è abbastanza intuitivo (in modo particolare il problema dei margini delle scatole è già descritto nella sezione 52.3), a parte l'attributo obbligatorio '**`master-name`**', che ha lo scopo di dare un nome al tipo di impaginazione che viene dichiarato in quel contesto. Successivamente, quando si descrivono le componenti del corpo della pagina, viene fatto riferimento a questo nome per richiamare il tipo di impaginazione desiderato.

All'interno dei margini fissati con gli attributi appropriati dell'elemento '**`fo:simple-page-master`**' si collocano delle regioni, dichiarate attraverso gli elementi '**`fo:region-before`**', '**`fo:region-after`**', '**`fo:region-start`**', '**`fo:region-end`**' e '**`fo:region-body`**'.

Figura 52.67. Regioni all'interno dei margini di una pagina.



Guardando la figura 52.67, che rappresenta lo spazio interno ai margini di una pagina, si deve osservare che la scatola definita dall'elemento '**`fo:region-body`**' contiene, sovrapposte, le altre quattro regioni. Inoltre, è importante osservare i nomi delle regioni, che rappresentano una collocazione relativa alla direzione della scrittura; per esempio, con una scrittura che procede dal basso verso l'altro, si scambiano di posto le regioni '**`fo:region-before`**' e '**`fo:region-after`**' rispetto alla figura; inoltre, una scrittura che procede da destra verso sinistra fa sì che si scambi la collocazione delle regioni dichiarate dagli elementi '**`fo:region-start`**' e '**`fo:region-end`**'.

```
<fo:region-body
    [space-before="spazio_prima"]
    [space-after="spazio_dopo"]
    [start-indent="rientro_iniziale"]
    [end-indent="rientro_finale"]
    [writing-mode="direzione"]
    [reference-orientation="rotazione"]
    [altri_attributi] />
```

```
<fo:region-{before|after|start|end}
    extent="dimensione"
    [writing-mode="direzione"]
    [reference-orientation="rotazione"]
    [altri_attributi] />
```

Gli elementi che dichiarano le regioni sono vuoti. A esclusione dell'elemento '**`fo:region-body`**', gli altri hanno tutti un attributo obbligatorio che ne definisce l'estensione (l'ampiezza per '**`fo:region-start`**' e '**`fo:region-end`**'; l'altezza per '**`fo:region-before`**' e '**`fo:region-after`**').

Nei modelli sintattici mostrati appare un attributo che non è anco-

ra stato preso in considerazione: '**reference-orientation**'. Con questo è possibile fa ruotare il contenuto, in multipli di 90 gradi, cosa che può risultare particolarmente utile per il contenuto delle regioni laterali.

Si comprende intuitivamente il significato di queste regioni che si trovano ai bordi del corpo della pagina: consentono di inserire delle intestazioni, dei piè di pagina, delle note a margine e simili. Ciò che è importante è rendersi conto che l'area che descrivono si sovrappone al corpo della pagina; pertanto, solitamente il contenuto che si va a inserire nel corpo viene controllato da margini ulteriori.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE fo:root SYSTEM "fo.dtd">
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
    <fo:layout-master-set>
        <fo:simple-page-master
                master-name="normale"
                page-width="210mm"
                page-height="297mm"
                margin-top="2cm"
                margin-bottom="2cm"
                margin-left="2cm"
                margin-right="2cm">
            <fo:region-body   margin-top="1cm"
                              margin-bottom="1cm"/>
            <fo:region-before extent="1cm" />
            <fo:region-after  extent="1cm" />
        </fo:simple-page-master>
        ...
    </fo:layout-master-set>
    <fo:page-sequence>
        ...
    </fo:page-sequence>
    ...
</fo:root>
```

L'esempio che appare sopra mostra l'uso di un elemento '**fo:simple-page-master**', con il quale si dichiara un tipo di impaginazione denominato '**normale**', con le dimensioni di un foglio A4 (21 cm × 29,7 cm), con margini uguali di 2 cm, all'interno dei quali si trova un corpo, che a sua volta ha margini superiori e inferiori, pari all'estensione delle regioni rispettive.

Per riempire effettivamente le pagine di contenuti, occorre intervenire all'interno di elementi '**fo:page-sequence**', con cui si fa riferimento al tipo di impaginazione che si vuole usare:

```
<fo:page-sequence
    master-reference="nome_tipo_impaginazione"
    [initial-page-number="numero_iniziale"]
    [force-page-count="auto"|"odd"|"even"|altro]
    [country="nazionalità_iso_3166"]
    [language="lingua_iso_639"]
    [altri_attributi]>

    contenuti

</fo:page-sequence>
```

Dal modello sintattico si vede che il riferimento al tipo di impaginazione si ottiene con l'attributo '**master-reference**', che pertanto è obbligatorio. Sono inoltre da considerare gli attributi '**initial-number**' e '**force-page-count**': il primo consente di stabilire il numero di pagina iniziale e il secondo permette di stabilire se il gruppo di pagine in questione deve essere complessivamente dispari, pari o se debbano essere rispettate altre regole.

Gli attributi '**country**' e '**language**', nell'insieme servono a definire le caratteristiche locali del testo, stabilendo la nazionalità e il linguaggio. Formalmente, le sigle che si utilizzano per questo sono quelle definite nel documento RFC 1766, che in pratica corrisponde agli standard ISO 3166 (tabella 13.5) e ISO 639 (tabella 13.4).

Un elemento '**fo:page-sequence**' contiene necessariamente l'elemento '**fo:flow**' (uno solo), che a sua volta contiene ciò che viene

distribuito nelle pagine (di solito il testo), come se fosse un flusso di informazioni. L'elemento '**fo:flow**' deve dichiarare, attraverso l'attributo '**flow-name**', in quale regione della pagina si inserisce il flusso in questione:

```
<fo:flow
    flow-name="xsl-region-{body|before|after|start|end}"

    flusso

</fo:flow>
```

L'esempio seguente riprende quanto già presentato a proposito della dichiarazione del tipo di impaginazione, associato a un flusso che riguarda il corpo della pagina stessa:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE fo:root SYSTEM "fo.dtd">
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
    <fo:layout-master-set>
        <fo:simple-page-master
                master-name="normale"
                page-width="210mm"
                page-height="297mm"
                margin-top="2cm"
                margin-bottom="2cm"
                margin-left="2cm"
                margin-right="2cm">
            <fo:region-body   margin-top="1cm"
                              margin-bottom="1cm"/>
            <fo:region-before extent="1cm" />
            <fo:region-after  extent="1cm" />
        </fo:simple-page-master>
    </fo:layout-master-set>
    <fo:page-sequence
            master-reference="normale"
            initial-page-number="1"
            force-page-count="even"
            language="it"
            country="it">
        <fo:flow flow-name="xsl-region-body">
            <fo:block>
                Bla bla bla Bla bla bla bla bla bla bla bla
                bla bla bla bla bla bla bla bla bla bla bla
                bla bla bla bla bla bla bla bla bla bla bla
                bla bla bla bla bla bla bla bla bla bla bla
                bla bla bla.
            </fo:block>
            <fo:block>
                Bla bla bla Bla bla bla bla bla bla bla bla
                bla bla bla bla bla bla bla bla bla bla bla
                bla bla bla bla bla bla bla bla bla bla bla
                bla bla bla bla bla bla bla bla bla bla bla
                bla bla bla.
            </fo:block>
        </fo:flow>
    </fo:page-sequence>
</fo:root>
```

Quello che si ottiene dalla trasformazione di questo file XSL-FO sono due pagine (l'ultima è bianca) che contengono il testo che si può vedere, suddiviso in due capoversi. Si può osservare che il margine superiore è di 3 cm, perché 2 cm definiti dall'impaginazione si sommano al margine del corpo della pagina.

Le pagine ottenute sono due perché è stato richiesto espressamente di impiegare una quantità pari (*even*) di pagine con l'attributo '**force-page-content**' dell'elemento '**fo:page-sequence**'.

Se si tralasciano gli attributi, la sintassi completa di '**fo:page-sequence**' corrisponde al modello seguente:

```
<fo:page-sequence attributi>
    [<fo:title attributi>
        titolo
    </fo:title>]
    [<fo:static-content flow-name="regione">
        contenuto_statico
    </fo:static-content>]
    …
    <fo:flow flow-name="regione">
        flusso
    </fo:flow>
</fo:page-sequence>
```

L'elemento '**`fo:title`**' consente di attribuire un titolo, che non fa parte, necessariamente, del risultato della composizione. Gli elementi '**`fo:static-content`**' si comportano in modo simile a '**`fo:flow`**', per quanto riguarda la selezione della regione di competenza, ma servono per definire del testo che si vuole appaia su tutte le pagine (come può essere per una riga di intestazione o alla base della pagina).

L'esempio seguente espande quanto già visto in quello precedente, aggiungendo un'intestazione e una riga a piè pagina, dove, in particolare, viene collocato anche il numero della pagina. Ovviamente, per poter apprezzare il fatto che l'intestazione e il piè di pagina rimangono costanti, occorre mettere qualcosa di più nel corpo:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE fo:root SYSTEM "fo.dtd">
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
    <fo:layout-master-set>
        <fo:simple-page-master
                master-name="normale"
                page-width="210mm"
                page-height="297mm"
                margin-top="2cm"
                margin-bottom="2cm"
                margin-left="2cm"
                margin-right="2cm">
            <fo:region-body   margin-top="1cm"
                              margin-bottom="1cm"/>
            <fo:region-before extent="1cm" />
            <fo:region-after  extent="1cm" />
        </fo:simple-page-master>
    </fo:layout-master-set>
    <fo:page-sequence
            master-reference="normale"
            initial-page-number="1"
            force-page-count="even"
            language="it"
            country="it">
        <fo:static-content flow-name="xsl-region-before">
            <fo:block>
                Un esempio per cominciare.
            </fo:block>
        </fo:static-content>
        <fo:static-content flow-name="xsl-region-after">
            <fo:block>
                pagina <fo:page-number />
            </fo:block>
        </fo:static-content>
        <fo:flow flow-name="xsl-region-body">
            <fo:block>
                Bla bla bla Bla bla bla bla bla bla bla bla
                bla bla bla bla bla bla bla bla bla bla bla
                bla bla bla bla bla bla bla bla bla bla bla
                bla bla bla bla bla bla bla bla bla bla bla
                bla bla bla.
            </fo:block>
            <!-- Inserire qui altri fo:block per ottenere
                 più pagine -->
            <fo:block>
                Bla bla bla Bla bla bla bla bla bla bla bla
                bla bla bla bla bla bla bla bla bla bla bla
                bla bla bla bla bla bla bla bla bla bla bla
                bla bla bla bla bla bla bla bla bla bla bla
                bla bla bla.
            </fo:block>
        </fo:flow>
    </fo:page-sequence>
</fo:root>
```

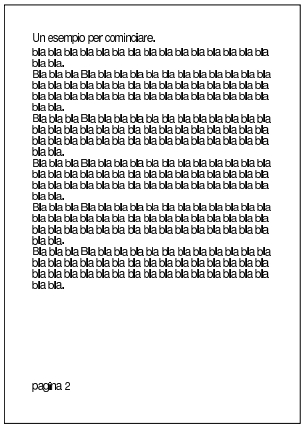

Si osservi che gli esempi devono essere salvati in file con codifica UTF-8, altrimenti diventa indispensabile usare delle entità che facciano riferimento al punto di codifica, quando si usano simboli al di sopra di U+007F. Pertanto, se necessario, anche il commento va scritto nel modo corretto, per non creare problemi al programma che lo elabora:

```
<!-- Inserire qui altri fo:block per ottenere
     pi&#x00F9; pagine -->
```

Oppure:

```
<!-- Inserire qui altri fo:block per ottenere
     piu' pagine -->
```

Figura 52.73. L'aspetto di una pagina ottenuta dalla composizione dell'esempio. Si osservi che i caratteri sono stati ingranditi rispetto a quanto si otterrebbe normalmente, considerando le proporzioni.



52.4.2 Impaginazione più articolata

Nella sezione precedente è stata mostrata la possibilità di richiedere espressamente una quantità pari o dispari di pagine per una certa sequenza di pagine. Tuttavia questo non basta per controllare l'estetica di un bel libro, perché di solito si vuole che le pagine cambino impostazione in modo automatico all'inizio di un capitolo. Per raggiungere risultati come questo, al posto di usare l'elemento '**`fo:simple-page-master`**', si usa piuttosto '**`fo:page-sequence-master`**', che però si avvale in pratica delle dichiarazioni di altri elementi '**`fo:simple-page-master`**':

```
<fo:page-sequence-master
    master-name="nome_tipo_impaginazione">
    [<fo:single-page-master-reference
        master-reference=nome_tipo_impaginazione />]
    [<fo:repeatable-page-master-reference
        master-reference=nome_tipo_impaginazione
        [maximum-repeat=max_ripetizioni] />]
    [<fo:repeatable-page-master-alternatives
        [maximum-repeat=max_ripetizioni]>
        <fo:conditional-page-master-reference
            master-reference="nome_tipo_impaginazione"
            page-position="posizione"
            [altri_attributi] />
        …
    </fo:repeatable-page-master-alternatives>]
    …
</fo:page-sequence-master>
```

Inizialmente, l'insieme è piuttosto complesso. L'elemento più esterno, '**fo:page-sequence-master**' definisce un tipo di impaginazione a cui in seguito è possibile fare riferimento. All'interno di questo elemento possono apparire tre tipi di elementi, anche ripetutamente, con lo scopo di descrivere in sequenza i tipi di pagina da usare.

L'elemento '**fo:single-page-master-reference**', che si riferisce a un certo tipo di pagina in base all'attributo '**master-reference**', stabilisce che deve apparire una sola pagina con quelle caratteristiche. L'elemento '**fo:repeatable-page-master-reference**' stabilisce di utilizzare un certo tipo di pagina, attraverso l'attributo '**master-reference**', per una quantità imprecisata di pagine (salvo l'uso dell'attributo '**maximum-repeat**'. Si osservi l'esempio seguente:

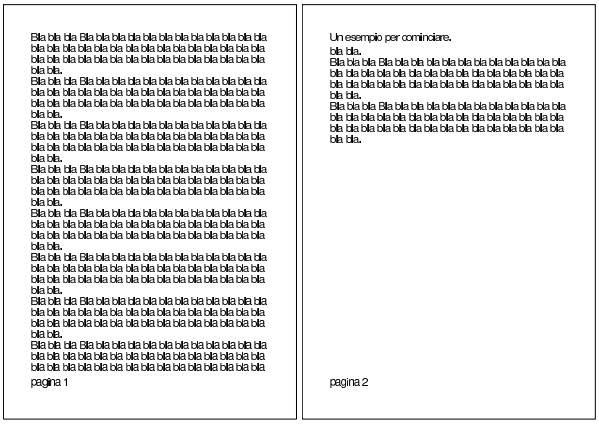
```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <!DOCTYPE fo:root SYSTEM "fo.dtd">
3  <fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
4      <fo:layout-master-set>
5          <fo:simple-page-master
6              master-name="normale"
7              page-width="210mm"
8              page-height="297mm"
9              margin-top="2cm"
10             margin-bottom="2cm"
11             margin-left="2cm"
12             margin-right="2cm">
13             <fo:region-body
14                 margin-top="1cm"
15                 margin-bottom="1cm"/>
16             <fo:region-before extent="1cm" />
17             <fo:region-after  extent="1cm" />
18         </fo:simple-page-master>
19         <fo:simple-page-master
20             master-name="speciale"
21             page-width="210mm"
22             page-height="297mm"
23             margin-top="2cm"
24             margin-bottom="2cm"
25             margin-left="2cm"
26             margin-right="2cm">
27             <fo:region-body
28                 margin-bottom="1cm"/>
29             <fo:region-after
30                 extent="1cm" />
31         </fo:simple-page-master>
32         <fo:page-sequence-master
33             master-name="capitolo">
34             <fo:single-page-master-reference
35                 master-reference="speciale" />
36             <fo:repeatable-page-master-reference
37                 master-reference="normale" />
38         </fo:page-sequence-master>
39     </fo:layout-master-set>
40     <fo:page-sequence
41             master-reference="capitolo"
42             initial-page-number="1"
43             force-page-count="even"
44             language="it"
45             country="it">
46         <fo:static-content
47             flow-name="xsl-region-before">
48             <fo:block>
49                 Un esempio per cominciare.
50             </fo:block>
51         </fo:static-content>
52         <fo:static-content
53             flow-name="xsl-region-after">
54             <fo:block>
55                 pagina <fo:page-number />
56             </fo:block>
57         </fo:static-content>
58         <fo:flow flow-name="xsl-region-body">
59             <fo:block>
60                 Bla bla bla Bla bla bla bla bla bla bla
61                 bla bla bla bla bla bla bla bla bla bla
62                 bla bla bla bla bla bla bla bla bla bla
63                 bla bla bla bla bla bla bla bla bla bla
```



```
64                 bla bla bla bla bla bla bla.
65             </fo:block>
66             <!-- Inserire qui altri fo:block per
67                  ottenere più pagine -->
68             <fo:block>
69                 Bla bla bla Bla bla bla bla bla bla bla
70                 bla bla bla bla bla bla bla bla bla bla
71                 bla bla bla bla bla bla bla bla bla bla
72                 bla bla bla bla bla bla bla bla bla bla
73                 bla bla bla bla bla bla bla.
74             </fo:block>
75         </fo:flow>
76     </fo:page-sequence>
77 </fo:root>
```

Questa volta vengono dichiarati due tipi di impaginazione «semplice»; una denominata '**normale**', l'altra denominata '**speciale**' (righe da 5 a 31). La differenza tra i due tipi di impaginazione sta nell'assenza, nel secondo caso, della regione superiore, con la conseguente assenza del margine superiore nel corpo. In questo modo, si vuole fare sì che un capitolo utilizzi come prima pagina questo secondo tipo di impaginazione, senza usare la riga di intestazione. Pertanto, si definisce un tipo di impaginazione più complesso, denominato proprio '**capitolo**', che prevede come prima pagina l'impaginazione '**speciale**' e come pagine successive l'impaginazione '**normale**'. Per quanto riguarda l'elemento '**fo:page-sequence**' nulla è cambiato rispetto all'esempio precedente.

Figura 52.75. L'aspetto delle prime due pagine ottenute dalla composizione dell'esempio. Si osservi che i caratteri sono stati ingranditi rispetto a quanto si otterrebbe normalmente, considerando le proporzioni.



Per ottenere effetti più complessi si può usare l'elemento '**fo:repeatable-page-master-alternatives**', che contiene necessariamente elementi '**fo:conditional-page-master-referenc**

```
<fo:repeatable-page-master-alternatives
        [maximum-repeat=max_ripetizioni]>
    <fo:conditional-page-master-reference
        master-reference="nome_tipo_impaginazione"
        page-position="first"|"last"|"rest"|"any"
        [altri_attributi] />
    ...
</fo:repeatable-page-master-alternatives>
```

In pratica, attraverso gli elementi '**conditional-page-master-refere** si vanno a individuare dei sottogruppi di pagine in base a una condizione. La sintassi mostra la selezione più semplice, attraverso l'attributo '**page-position**', con il quale si può individuare la prima pagina, l'ultima, le pagine successive alla prima oppure qualunque pagina. Tuttavia sono disponibili altri attributi per condizioni più sofisticate, che qui non vengono descritti. L'esempio seguente svolge lo stesso lavoro di quello precedente, con l'uso di queste selezioni condizionali:

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <!DOCTYPE fo:root SYSTEM "fo.dtd">
3  <fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
4      <fo:layout-master-set>
5          <fo:simple-page-master
6              master-name="normale"
7              page-width="210mm"
8              page-height="297mm"
9              margin-top="2cm"
10             margin-bottom="2cm"
11             margin-left="2cm"
12             margin-right="2cm">
13             <fo:region-body margin-top="1cm"
14                 margin-bottom="1cm"/>
15             <fo:region-before extent="1cm" />
16             <fo:region-after  extent="1cm" />
17         </fo:simple-page-master>
18         <fo:simple-page-master
19             master-name="speciale"
20             page-width="210mm"
21             page-height="297mm"
22             margin-top="2cm"
23             margin-bottom="2cm"
24             margin-left="2cm"
25             margin-right="2cm">
26             <fo:region-body margin-bottom="1cm"/>
27             <fo:region-after extent="1cm" />
28         </fo:simple-page-master>
29         <fo:page-sequence-master master-name="capitolo">
30             <fo:repeatable-page-master-alternatives>
31                 <fo:conditional-page-master-reference
32                     master-reference="speciale"
33                     page-position="first" />
34                 <fo:conditional-page-master-reference
35                     master-reference="normale"
36                     page-position="rest" />
37             </fo:repeatable-page-master-alternatives>
38         </fo:page-sequence-master>
39     </fo:layout-master-set>
40     <fo:page-sequence
41         master-reference="capitolo"
42         initial-page-number="1"
43         force-page-count="even"
44         language="it"
45         country="it">
46         <fo:static-content
47             flow-name="xsl-region-before">
48             <fo:block>
49                 Un esempio per cominciare.
50             </fo:block>
51         </fo:static-content>
52         <fo:static-content
53             flow-name="xsl-region-after">
54             <fo:block>
55                 pagina <fo:page-number />
56             </fo:block>
57         </fo:static-content>
58         <fo:flow flow-name="xsl-region-body">
59             <fo:block>
60                 Bla bla bla Bla bla bla bla bla bla bla
61                 bla bla bla bla bla bla bla bla bla bla
62                 bla bla bla bla bla bla bla bla bla bla
63                 bla bla bla bla bla bla bla bla bla bla
64                 bla bla bla bla bla bla bla.
65             </fo:block>
66             <!-- Inserire qui altri fo:block per
67                  ottenere più pagine -->
68             <fo:block>
69                 Bla bla bla Bla bla bla bla bla bla bla
70                 bla bla bla bla bla bla bla bla bla bla
71                 bla bla bla bla bla bla bla bla bla bla
72                 bla bla bla bla bla bla bla bla bla bla
73                 bla bla bla bla bla bla bla.
74             </fo:block>
75         </fo:flow>
76     </fo:page-sequence>
77 </fo:root>
```

La differenza rispetto all'esempio precedente si trova precisamente nelle righe da 29 a 38.

52.4.3 Contenuto

A proposito del contenuto del documento, è già stato mostrato l'uso e la struttura successiva all'elemento '**`fo:page-sequence`**':

```
<fo:page-sequence
        master-reference="nome_tipo_impaginazione"
        [initial-page-number="numero_iniziale"]
        [force-page-count="auto"|"odd"|"even"|altro]
        [country="nazionalità_iso_3166"]
        [language="lingua_iso_639"]
        [altri_attributi]>
    [<fo:title attributi>
        titolo
    </fo:title>]
    [<fo:static-content flow-name="regione">
        contenuto_statico
    </fo:static-content>]
    …
    <fo:flow flow-name="regione">
        flusso
    </fo:flow>
</fo:page-sequence>
```

L'elemento '**`fo:page-sequence`**' rappresenta letteralmente una sequenza di pagine, come potrebbe essere un capitolo o una parte di un libro, dove, tra le altre cose, può essere utile ripartire con una nuova numerazione delle pagine. Pertanto, un documento può avere più elementi '**`fo:page-sequence`**' distinti.

Gli elementi '**`fo:flow`**', o '**`fo:static-content`**', rappresentano in pratica la destinazione del contenuto, che va inserito nella regione appropriata. Questi elementi possono contenere dei «blocchi», o più precisamente elementi '**`fo:block`**', '**`fo:block-container`**', '**`fo:list-block`**', '**`fo:table`**', '**`fo:table-and-caption`**'.

L'elemento '**`fo:block`**' ha la particolarità di contenere sia testo lineare, sia altri blocchi (quindi anche se stesso). Pertanto viene usato indifferentemente come un equivalente dell'elemento '**`DIV`**' o dell'elemento '**`P`**' di HTML.

52.5 XSL-FO: contenuto

Il contenuto di una regione del corpo definita dall'impaginazione, è fatto sommariamente di blocchi, come l'elemento '**`fo:block`**', e di componenti lineari, in base al contesto. Generalmente, un blocco non può essere contenuto in un contesto lineare, a meno che questo sia incorporato in un elemento apposito, come '**`fo:inline-container`**'.

Un blocco è idealmente un oggetto a due dimensioni, rettangolare, per il quale potrebbero essere previsti dei margini su tutti i lati; un componente lineare è invece inteso come un oggetto monodimensionale che può avere margini solo prima e dopo.

In generale, gli elementi che descrivono un blocco o una componente lineare condividono delle opzioni comuni per definire i margini, il bordo, il colore di fondo e altre caratteristiche. La tabella 52.77 elenca alcuni attributi abbastanza comuni.

Tabella 52.77. Alcuni attributi abbastanza comuni negli elementi che contengono del testo, sia in forma di blocco, sia in forma lineare.

Attributo	Descrizione
`space-start="`*spazio_iniziale*`"` `space-end="`*spazio_finale*`"`	Riguarda i componenti lineari e richiede uno spazio iniziale o finale.

Attributo	Descrizione

`*before*="`***prima***`"`
`*after*="`***dopo***`"`
`*start*="`***inizio***`"`
`*end*="`***fine***`"`
	Un attributo che contiene le parole chiave evidenziate rappresenta una proprietà relativa alla posizione rispettiva, tenendo conto della direzione di scrittura.

`space-before="`***spazio_prima***`"`
`space-after="`***spazio_dopo***`"`
`start-indent="`***rientro_iniziale***`"`
`end-indent="`***rientro_finale***`"`
	Riguarda i componenti a blocchi e definisce i margini.

`border-before-color="`***colore***`"`
`border-after-color="`***colore***`"`
`border-start-color="`***colore***`"`
`border-end-color="`***colore***`"`
	Definisce il colore del bordo.

`border-before-style="`***stile***`"`
`border-after-style="`***stile***`"`
`border-start-style="`***stile***`"`
`border-end-style="`***stile***`"`
	Definisce lo stile del bordo. Possono essere usate parole chiave come: '**`none`**', '**`dotted`**', '**`dashed`**', '**`solid`**', '**`double`**'.

`border-before-width="`***spessore***`"`
`border-after-width="`***spessore***`"`
`border-start-width="`***spessore***`"`
`border-end-width="`***spessore***`"`
	Definisce lo spessore del bordo.

`font-family="`***nome***`"` `font-family="serif"`↩ ↪`|"sans-serif"`↩ ↪`|"cursive"|"fantasy"`↩ ↪`|"monospace"`	Dichiara il nome di un carattere tipografico. Il nome può essere indicato in modo preciso, oppure attraverso parole chiave che si riferiscono alla caratteristica di massima.

`font-size="`***altezza***`"`
`font-size="xx-small"`↩
↪`|"x-small"|"small"`↩
↪`|"medium"|"large"|"x-large"`↩
↪`|"xx-large"`
`font-size="larger"|"smaller"`
`font-size="`***n***`%"`
`font-size="`***n***`em"`
	Dichiara la dimensione del carattere tipografico. La dimensione può essere data con un valore assoluto, espresso attraverso un numero e la sua unità di misura; un valore assoluto specificato attraverso una parola chiave; un valore relativo specificato attraverso una parola chiave; un valore relativo specificato in modo percentuale o in quadratoni («em»).

`font-style="normal"`↩ ↪`|"italic"|"oblique"`↩ ↪`|"backslant"`	Dichiara stile del carattere, inteso come la sua inclinazione o la sua forma corsiva.
`font-stretch="ultra-condensed"`↩ ↪`|"extra-condensed"|"condensed"`↩ ↪`|"semi-condensed"|"normal"`↩ ↪`|"semi-expanded"|"expanded"`↩ ↪`|"extra-condensed"`↩ ↪`|"ultra-expanded"`	Dichiara la larghezza del carattere.
`font-variant="normal"|"small-caps"`	Dichiara l'utilizzo o meno del maiuscoletto.

Attributo	Descrizione
`font-weight="normal"|"bold"`↩ ↪`|"bolder"`↩ ↪`|"lighter"`	Dichiara lo spessore del carattere (normale, nero, nerissimo, chiaro).
`line-height="normal"|"`***altezza***`"`↩ ↪`|"`***n***`"|"`***n***`%"`	Dichiara l'altezza della riga. Se si indica un numero privo di unità di misura, si fa riferimento alla dimensione del carattere (***n*** volte l'altezza del carattere); se si indica una percentuale, questa si riferisce all'altezza del carattere.
`text-align="start"|"center"|"end"`↩ ↪`|"justify"|…`	Riguarda i componenti a blocchi e serve per dichiarare l'allineamento da usare. Al posto di indicare un allineamento a destra o a sinistra, si preferiscono definizioni riferite alla direzione di scrittura.
`text-indent="`***dimensione***`"|"`***n***`%"`	Riguarda i componenti a blocchi e serve per dichiarare il rientro della prima riga. Un valore percentuale si riferisce all'ampiezza del blocco che lo contiene.

Le sezioni successive non esauriscono l'argomento trattato, perché diversi elementi e molti attributi non vengono descritti affatto. Per ottenere le informazioni mancanti e in modo più dettagliato, si consulti la documentazione annotata nella bibliografia che appare alla fine del capitolo.

52.5.1 Blocchi di testo comuni

Un blocco di testo comparabile all'elemento '**`P`**' di HTML, si ottiene con l'elemento '**`fo:block`**', che però può contenere anche altri blocchi, oltre al testo lineare puro e semplice:

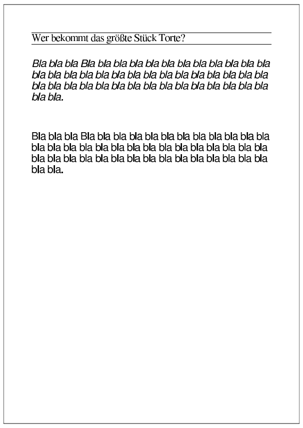
```
<fo:block
        [attributi]>
    …
</fo:block>
```

L'esempio seguente mostra l'uso di alcuni attributi per modificare il linguaggio, per inserire dei bordi, per modificare il carattere e per inserire dei margini:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE fo:root SYSTEM "fo.dtd">
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
    <fo:layout-master-set>
        <fo:simple-page-master master-name="normale"
            page-width="210mm" page-height="297mm"
            margin-top="2cm"   margin-bottom="2cm"
            margin-left="2cm"  margin-right="2cm">
            <fo:region-body />
        </fo:simple-page-master>
    </fo:layout-master-set>
    <fo:page-sequence master-reference="normale"
        language="it" country="it">
        <fo:flow flow-name="xsl-region-body">
            <fo:block language="de"       country="de"
                      font-family="serif" font-size="7mm"
                      border-before-style="solid"
                      border-after-style="solid">
                Wer bekommt das gr&#x00F6;&#x00DF;te
                St&#x00FC;ck Torte?
            </fo:block>
            <fo:block font-style="italic" font-size="7mm"
                      space-before="1cm"  space-after="1cm">
                Bla bla bla Bla bla bla bla bla bla bla bla
                bla bla bla bla bla bla bla bla bla bla bla
                bla bla bla bla bla bla bla bla bla bla bla
                bla bla bla bla bla bla bla bla bla bla bla
```

```
                bla bla bla.
            </fo:block>
            <fo:block font-size="7mm"
                      space-before="1cm" space-after="1cm">
                Bla bla bla Bla bla bla bla bla bla bla bla
                bla bla bla bla bla bla bla bla bla bla bla
                bla bla bla bla bla bla bla bla bla bla bla
                bla bla bla bla bla bla bla bla bla bla bla
                bla bla bla.
            </fo:block>
        </fo:flow>
    </fo:page-sequence>
</fo:root>
```

Il risultato della composizione si può osservare nelle figura seguente:



Dal momento che tutti gli elementi '**`fo:block`**' dell'esempio utilizzano un carattere di 7 mm, si può sfruttare l'ereditarietà della proprietà, inserendo tutti questi blocchi in un elemento '**`fo:block`**' complessivo (riga 15) in cui si dichiara una volta sola tale dimensione:

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <!DOCTYPE fo:root SYSTEM "fo.dtd">
3  <fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
4      <fo:layout-master-set>
5          <fo:simple-page-master master-name="normale"
6              page-width="210mm" page-height="297mm"
7              margin-top="2cm" margin-bottom="2cm"
8              margin-left="2cm" margin-right="2cm">
9              <fo:region-body />
10         </fo:simple-page-master>
11     </fo:layout-master-set>
12     <fo:page-sequence  master-reference="normale"
13         language="it" country="it">
14         <fo:flow flow-name="xsl-region-body">
15             <fo:block font-size="7mm">
16                 <fo:block language="de" country="de"
17                     font-family="serif"
18                     font-size="7mm"
19                     border-before-style="solid"
20                     border-after-style="solid">
21                     Wer bekommt das
22                     gr&#x00F6;&#x00DF;te
23                     St&#x00FC;ck Torte?
24                 </fo:block>
25                 <fo:block font-style="italic"
26                     space-before="1cm"
27                     space-after="1cm">
28                     Bla bla bla Bla bla bla bla bla bla
29                     bla bla bla bla bla bla bla bla bla
30                     bla bla bla bla bla bla bla bla bla
31                     bla bla bla bla bla bla bla bla bla
32                     bla bla bla bla bla bla bla bla bla
33                     bla bla.
34                 </fo:block>
35                 <fo:block space-before="1cm"
36                     space-after="1cm">
37                     Bla bla bla Bla bla bla bla bla bla
38                     bla bla bla bla bla bla bla bla bla
39                     bla bla bla bla bla bla bla bla bla
40                     bla bla bla bla bla bla bla bla bla
41                     bla bla bla bla bla bla bla bla bla
42                     bla bla.
43                 </fo:block>
44             </fo:block>
45         </fo:flow>
46     </fo:page-sequence>
47 </fo:root>
```


Il risultato che si ottiene è lo stesso già visto nella figura precedente.

52.5.2 Testo lineare

In un contesto lineare, si può inserire testo o elementi che non creano un blocco. In modo particolare, l'elemento '**`fo:inline`**', con l'uso dei suoi attributi, consente di modificare le caratteristiche del testo, come il carattere, lo spostamento sulla linea di riferimento (apici e pedici) e il colore:

```
<fo:inline
    [font-family="nome"|"serif"|"sans-serif"
        |"cursive"|"fantasy"|"monospace"]
    [font-size="altezza"|"xx-small"|"x-small"
        |"small"|"medium"|"large"|"x-large"
        |"xx-large"|"larger"|"smaller"
        |"n%"|"nem"]
    [font-style="normal"|"italic"|"oblique"
        |"backslant"]
    [font-stretch="ultra-condensed"|"extra-condensed"
        |"condensed"|"semi-condensed"
        |"normal"|"semi-expanded"
        |"expanded"|"extra-condensed"
        |"ultra-expanded"]
    [font-variant="normal"|"small-caps"]
    [font-weight="normal"|"bold"|"bolder"|"lighter"]
    [baseline-shift="sub"|"super"|...]
    [color="colore"]
    [altri_attributi]>
    [contenuto_lineare]
</fo:inline>
```

L'esempio seguente mostra solo un estratto di un file XSL-FO, con un blocco che contiene testo lineare, all'interno del quale appaiono alcuni elementi '**`fo:inline`**' con lo scopo di modificare il carattere, il colore e lo spostamento verticale:

```
<fo:block>Bla bla bla <fo:inline font-style="italic">testo
in corsivo</fo:inline> bla bla bla <fo:inline
font-weight="bold">H<fo:inline
baseline-shift="sub">2</fo:inline>O</fo:inline> bla bla bla
<fo:inline font-family="monospace">ls -l | sort &gt;
prova</fo:inline> bla bla bla <fo:inline color="blue">testo
in blu</fo:inline> bla bla bla <fo:inline
font-variant="small-caps">testo in maiuscoletto</fo:inline>
bla bla bla.</fo:block>
```

Nell'ambito di un contesto lineare, eccezionalmente, è possibile inserire un blocco, attraverso l'elemento '**`fo:inline-container`**':

```
<fo:inline-container
        [width="ampiezza"]
        [altri_attributi]>
    [contenuto_blocco]
</fo:inline-container>
```

52.5.3 Elenchi

Gli elenchi sono definiti attraverso l'elemento '**`fo:list-block`**', che è, evidentemente, un blocco:

```
<fo:list-block [attributi]>
    <fo:list-item [attributi]>
        <fo:list-item-label [attributi]>
            blocchi
        </fo:list-item-label>
        <fo:list-item-body [attributi]>
            blocchi
        </fo:list-item-body>
    <fo:/list-item>
    …
</fo:list-block>
```

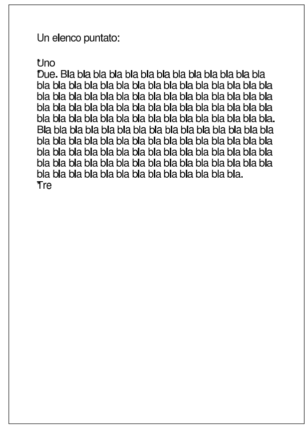
In pratica, l'elemento '**`fo:list-block`**' contiene obbligatoriamente elementi '**`fo:list-item`**', i quali, a loro volta, contengono la coppia '**`fo:list-item-label`**' e '**`fo:list-item-body`**'. L'esempio seguente costruisce un elenco puntato molto semplice, dove i lembi che segnalano l'inizio di ogni punto sono costituiti da un asterisco:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE fo:root SYSTEM "fo.dtd">
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
    <fo:layout-master-set>
        <fo:simple-page-master     master-name="normale"
                page-width="210mm" page-height="297mm"
                margin-top="2cm"   margin-bottom="2cm"
                margin-left="2cm"  margin-right="2cm">
            <fo:region-body />
        </fo:simple-page-master>
    </fo:layout-master-set>
    <fo:page-sequence master-reference="normale"
        language="it" country="it">
        <fo:flow flow-name="xsl-region-body">
            <fo:block font-size="7mm">
                <fo:block space-after="1cm">
                    Un elenco puntato:
                </fo:block>
                <fo:list-block  space-after="1cm">
                    <fo:list-item>
                        <fo:list-item-label>
                            <fo:block>*</fo:block>
                        </fo:list-item-label>
                        <fo:list-item-body>
                            <fo:block>Uno</fo:block>
                        </fo:list-item-body>
                    </fo:list-item>
                    <fo:list-item>
                        <fo:list-item-label>
                            <fo:block>*</fo:block>
                        </fo:list-item-label>
                        <fo:list-item-body>
                            <fo:block>
                                Due. Bla bla bla bla bla bla
                                bla bla bla bla bla bla bla
                                bla bla bla bla bla bla bla
                                bla bla bla bla bla bla bla
                                bla bla bla bla bla bla bla
                                bla bla bla bla bla bla bla
                                bla bla bla bla bla bla bla
                                bla bla bla bla bla bla bla
                                bla bla bla bla bla bla bla
                                bla bla bla bla bla bla bla
                                bla bla bla bla.
                            </fo:block>
                            <fo:block>
                                Bla bla bla bla bla bla bla
                                bla bla bla bla bla bla bla
                                bla bla bla bla bla bla bla
                                bla bla bla bla bla bla bla
                                bla bla bla bla bla bla bla
                                bla bla bla bla bla bla bla
                                bla bla bla bla bla bla bla
                                bla bla bla bla bla bla bla
                                bla bla bla bla bla bla bla
                                bla bla bla bla bla bla bla
                                bla bla bla.
                            </fo:block>
                        </fo:list-item-body>
```



```
                    </fo:list-item>
                    <fo:list-item>
                        <fo:list-item-label>
                            <fo:block>*</fo:block>
                        </fo:list-item-label>
                        <fo:list-item-body>
                            <fo:block>Tre</fo:block>
                        </fo:list-item-body>
                    </fo:list-item>
                </fo:list-block>
            </fo:block>
        </fo:flow>
    </fo:page-sequence>
</fo:root>
```

Tuttavia, se si esegue la composizione si può osservare che gli asterischi si trovano a essere sovrapposti al corpo delle voci:

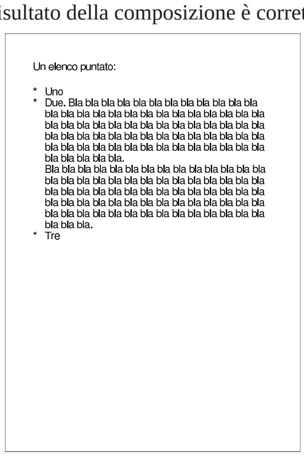


Per evitare che ciò accada, è necessario intervenire con dei rientri appropriati, utilizzando gli attributi dei blocchi che sono contenuti, sia come etichette, sia come corpo delle voci elencate. Per la precisione, si usa l'attributo '**`end-indent`**' nei blocchi delle etichette e l'attributo '**`start-indent`**' nei blocchi del corpo. Per fare in modo che queste distanze vengano calcolate automaticamente, si usano delle funzioni interne a XSL-FO, in grado di restituire il valore appropriato al contesto. Si tratta rispettivamente di ***label-end()*** e di ***body-start()***. Pertanto, l'esempio va modificato come segue:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE fo:root SYSTEM "fo.dtd">
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
    <fo:layout-master-set>
        <fo:simple-page-master master-name="normale"
            page-width="210mm" page-height="297mm"
            margin-top="2cm" margin-bottom="2cm"
            margin-left="2cm" margin-right="2cm">
            <fo:region-body />
        </fo:simple-page-master>
    </fo:layout-master-set>
    <fo:page-sequence master-reference="normale"
        language="it" country="it">
        <fo:flow flow-name="xsl-region-body">
            <fo:block font-size="7mm">
                <fo:block space-after="1cm">
                    Un elenco puntato:
                </fo:block>
                <fo:list-block space-after="1cm">
                    <fo:list-item>
                        <fo:list-item-label>
                            <fo:block
                                end-indent="label-end()">
                                *
                            </fo:block>
                        </fo:list-item-label>
                        <fo:list-item-body>
                            <fo:block
                                start-indent="body-start()">
                                Uno
                            </fo:block>
                        </fo:list-item-body>
                    </fo:list-item>
```

```
                    <fo:list-item>
                        <fo:list-item-label>
                            <fo:block
                                end-indent="label-end()">
                                *
                            </fo:block>
                        </fo:list-item-label>
                        <fo:list-item-body>
                            <fo:block
                                start-indent="body-start()">
                                Due. Bla bla bla bla bla bla
                                bla bla bla bla bla bla bla
                                bla bla bla bla bla bla bla
                                bla bla bla bla bla bla bla
                                bla bla bla bla bla bla bla
                                bla bla bla bla bla bla bla
                                bla bla bla bla bla bla bla
                                bla bla bla bla bla bla bla
                                bla bla bla bla bla bla bla
                                bla bla bla bla bla bla bla
                                bla bla bla bla.
                            </fo:block>
                            <fo:block
                                start-indent="body-start()">
                                Bla bla bla bla bla bla bla
                                bla bla bla bla bla bla bla
                                bla bla bla bla bla bla bla
                                bla bla bla bla bla bla bla
                                bla bla bla bla bla bla bla
                                bla bla bla bla bla bla bla
                                bla bla bla bla bla bla bla
                                bla bla bla bla bla bla bla
                                bla bla bla bla bla bla bla
                                bla bla bla bla bla bla bla
                                bla bla bla.
                            </fo:block>
                        </fo:list-item-body>
                    </fo:list-item>
                    <fo:list-item>
                        <fo:list-item-label>
                            <fo:block
                                end-indent="label-end()">
                                *
                            </fo:block>
                        </fo:list-item-label>
                        <fo:list-item-body>
                            <fo:block
                                start-indent="body-start()">
                                Tre
                            </fo:block>
                        </fo:list-item-body>
                    </fo:list-item>
                </fo:list-block>
            </fo:block>
        </fo:flow>
    </fo:page-sequence>
</fo:root>
```

Questa volta il risultato della composizione è corretto:


Un elenco puntato:

* Uno
* Due. Bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla.
  Bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla.
* Tre

52.5.4 Tabelle

Come si può immaginare, la realizzazione di una tabella con XSL-FO è un'operazione abbastanza complessa, anche nelle situazioni più semplici:

```
<fo:table-and-caption [attributi]>
    [<fo:table-caption [attributi]>
        blocchi
    </fo:table-caption>]
    <fo:table
            [table-layout="auto"…"fixed"]
            [altri_attributi]>
        [<fo:table-column
                [table-column="ampiezza"]
                [attributi] />]
        …
        [<fo:table-header [attributi]>
            righe
        </fo:table-header>]
        [<fo:table-footer [attributi]>
            righe
        </fo:table-footer>]
        <fo:table-body [attributi]>
            righe
        </fo:table-body>
    </fo:table-caption>
</fo:table-and-caption>
```

Il significato dei vari elementi che appaiono nel modello sintattico dovrebbe essere intuitivo; si osservi in particolare che una tabella si può realizzare anche partendo soltanto dall'elemento '**`fo:table`**', quando non è importante l'inserimento di una didascalia con l'elemento '**`fo:table-caption`**'. Le righe interne all'intestazione, al piè di pagina e al corpo, vengono inserite nel modo seguente:

```
<fo:table-row [attributi]>
    <fo:table-cell [attributi]>
        blocchi
    </fo:table-cell>
</fo:table-row>
```

L'ampiezza delle colonne, i bordi e altre caratteristiche della tabella vengono gestiti attraverso gli attributi. Teoricamente, l'elemento '**`fo:table`**' dispone dell'attributo '**`table-layout`**', con il quale si potrebbe richiedere un adattamento automatico della larghezza delle colonne; in pratica questa funzionalità è abbastanza difficile da trovare, pertanto si è costretti a usare l'attributo '**`column-width`**' degli elementi '**`fo:table-column`**'.

L'esempio seguente mostra una tabella molto semplice e anche molto povera, perché manca qualunque bordo e perché viene definita senza l'elemento '**`fo:table-and-caption`**', in modo da garantire il funzionamento con le prime edizioni di FOP:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE fo:root SYSTEM "fo.dtd">
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
    <fo:layout-master-set>
        <fo:simple-page-master master-name="normale"
            page-width="210mm" page-height="297mm"
            margin-top="2cm"   margin-bottom="2cm"
            margin-left="2cm"  margin-right="2cm">
            <fo:region-body />
        </fo:simple-page-master>
    </fo:layout-master-set>
    <fo:page-sequence master-reference="normale"
        language="it" country="it">
        <fo:flow flow-name="xsl-region-body">
            <fo:block font-size="7mm">
                <fo:block>Tabella di esempio. Bla bla bla
                bla bla bla bla bla bla bla bla bla bla bla
                bla bla bla bla bla bla bla bla bla bla bla
                bla bla bla bla bla bla bla bla bla bla bla
```

```
                bla bla bla bla bla bla bla bla bla bla bla
                bla bla bla bla bla bla bla bla bla
                bla</fo:block>
                <fo:table table-layout="fixed">
                    <fo:table-column column-width="5cm" />
                    <fo:table-column column-width="7cm" />
                    <fo:table-column column-width="5cm" />
                    <fo:table-header>
                        <fo:table-row>
                            <fo:table-cell>
                                <fo:block>
                                    Parola di controllo
                                </fo:block>
                            </fo:table-cell>
                            <fo:table-cell>
                                <fo:block>
                                    Competenza
                                </fo:block>
                            </fo:table-cell>
                            <fo:table-cell>
                                <fo:block>
                                    Condizione o valore
                                    predefinito
                                </fo:block>
                            </fo:table-cell>
                        </fo:table-row>
                    </fo:table-header>
                    <fo:table-body>
                        <fo:table-row>
                            <fo:table-cell>
                                <fo:block>
                                    \hoffset
                                </fo:block>
                            </fo:table-cell>
                            <fo:table-cell>
                                <fo:block>
                                    Posizione iniziale dei
                                    paragrafi nella pagina.
                                </fo:block>
                            </fo:table-cell>
                            <fo:table-cell>
                                <fo:block>
                                    0
                                </fo:block>
                            </fo:table-cell>
                        </fo:table-row>
                        <fo:table-row>
                            <fo:table-cell>
                                <fo:block>
                                    \hsize
                                </fo:block>
                            </fo:table-cell>
                            <fo:table-cell>
                                <fo:block>
                                    Larghezza del paragrafo
                                    a partire da \hoffset.
                                </fo:block>
                            </fo:table-cell>
                            <fo:table-cell>
                                <fo:block>
                                    6,5 pollici
                                </fo:block>
                            </fo:table-cell>
                        </fo:table-row>
                        <fo:table-row>
                            <fo:table-cell>
                                <fo:block>
                                    \parindent
                                </fo:block>
                            </fo:table-cell>
                            <fo:table-cell>
                                <fo:block>
                                    Rientro della prima
                                    riga.
                                </fo:block>
                            </fo:table-cell>
                            <fo:table-cell>
                                <fo:block>
                                    20 punti
                                </fo:block>
                            </fo:table-cell>
                        </fo:table-row>
                        <fo:table-row>
                            <fo:table-cell>
                                <fo:block>
                                    \baselineskip
                                </fo:block>
                            </fo:table-cell>
                            <fo:table-cell>
                                <fo:block>
                                    Distanza tra la base di
                                    una riga e la base della
                                    riga successiva.
                                </fo:block>
                            </fo:table-cell>
                            <fo:table-cell>
                                <fo:block>
                                    12 punti
                                </fo:block>
                            </fo:table-cell>
                        </fo:table-row>
                        <fo:table-row>
                            <fo:table-cell>
                                <fo:block>
                                    \parskip
                                </fo:block>
                            </fo:table-cell>
                            <fo:table-cell>
                                <fo:block>
                                    Distanza aggiuntiva tra
                                    i paragrafi.
                                </fo:block>
                            </fo:table-cell>
                            <fo:table-cell>
                                <fo:block>
                                    0
                                </fo:block>
                            </fo:table-cell>
                        </fo:table-row>
                        <fo:table-row>
                            <fo:table-cell>
                                <fo:block>
                                    \raggedright
                                </fo:block>
                            </fo:table-cell>
                            <fo:table-cell>
                                <fo:block>
                                    Allinea il testo a
                                    sinistra.
                                </fo:block>
                            </fo:table-cell>
                            <fo:table-cell>
                                <fo:block>
                                    allineato
                                    simultaneamente a
                                    sinistra e a destra
                                </fo:block>
                            </fo:table-cell>
                        </fo:table-row>
                    </fo:table-body>
                </fo:table>
            </fo:block>
        </fo:flow>
    </fo:page-sequence>
</fo:root>
```

Ecco il risultato:

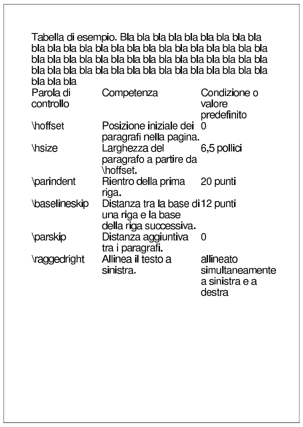


Tabella di esempio. Bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla

Parola di controllo	Competenza	Condizione o valore predefinito
\hoffset	Posizione iniziale dei paragrafi nella pagina.	0
\hsize	Larghezza del paragrafo a partire da \hoffset.	6,5 pollici
\parindent	Rientro della prima riga.	20 punti
\baselineskip	Distanza tra la base di una riga e la base della riga successiva.	12 punti
\parskip	Distanza aggiuntiva tra i paragrafi.	0
\raggedright	Allinea il testo a sinistra.	allineato simultaneamente a sinistra e a destra

Per introdurre dei bordi, almeno teoricamente, si può intervenire dal blocco contenuto nella singola cella, fino all'elemento '**`fo:table`**' più esterno; tuttavia, a seconda del programma di elaborazione che si utilizza, può darsi che in alcuni punti i bordi non vengano presi in considerazione.

52.5.5 Inserzione di immagini

«

L'inserzione di un'immagine si ottiene con l'elemento '**`fo:external-graphic`**', che è vuoto e appartiene a un contesto lineare:

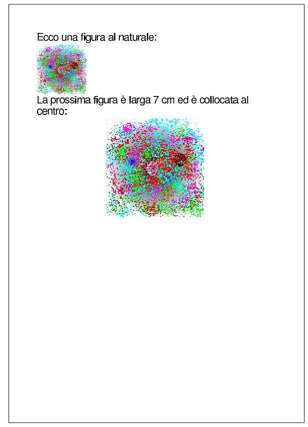
```
<fo:external-graphic>
        src="file" | "indirizzo_uri"
        [scaling="uniform" | "non-uniform"]
        [height="auto" | "altezza"]
        [width="auto" | "larghezza"]
        [altri_attributi]  />
```

Se non si indicano delle dimensioni attraverso gli attributi '**`height`**' o '**`width`**', l'immagine viene inserita alla sua dimensione naturale. Se si modifica una sola dimensione, normalmente si ottiene l'adattamento dell'altro valore in modo relativo; tuttavia, attraverso l'opzione '**`scaling="non-uniform"`**', è possibile deformare l'immagine.

Il file o l'indirizzo URI dell'immagine deve fare riferimento a un formato compatibile con le capacità elaborative del sistema usato per la composizione. Di solito vanno bene almeno i formati GIF e JPG.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE fo:root SYSTEM "fo.dtd">
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
    <fo:layout-master-set>
        <fo:simple-page-master master-name="normale"
            page-width="210mm" page-height="297mm"
            margin-top="2cm"   margin-bottom="2cm"
            margin-left="2cm"  margin-right="2cm">
            <fo:region-body />
        </fo:simple-page-master>
    </fo:layout-master-set>
    <fo:page-sequence master-reference="normale"
        language="it" country="it">
        <fo:flow flow-name="xsl-region-body">
            <fo:block font-size="7mm">
                <fo:block>
                    Ecco una figura al naturale:
                </fo:block>
                <fo:block>
                    <fo:external-graphic
                            src="esempio.jpg" />
                </fo:block>
                <fo:block>
                    La prossima figura &#x00E8; larga 7 cm
                    ed &#x00E8; collocata al centro:
                </fo:block>
                <fo:block   text-align="center">
                    <fo:external-graphic
                            src="esempio.jpg"
                            scaling="uniform"
                            width="7cm" />
                </fo:block>
            </fo:block>
        </fo:flow>
    </fo:page-sequence>
</fo:root>
```


L'esempio mostra l'inserzione della stessa immagine due volte, in modi differenti. In particolare, la seconda volta l'immagine viene centrata orizzontalmente per mezzo dell'attributo '**`text-align`**' dell'elemento '**`fo:block`**' che la contiene.



52.5.6 Linee orizzontali

«

Si può realizzare facilmente una linea orizzontale con un elemento '**`fo:block`**' bordato, magari anche vuoto:

```
<fo:block
        border-after-style="stile"
        [border-after-width="spessore"]
        [border-after-color="colore"]
        [altri_attributi]  />
```

Per inserire delle linee orizzontali in un contesto lineare, ovvero all'interno del testo, si usa l'elemento '**`fo:leader`**':

```
<fo:leader
        leader-pattern="use-content"
        [leader-length="lunghezza"]
        [altri_attributi]>
    modello
</fo:leader>
```

```
<fo:leader
        leader-pattern="space" | "rule" | "dots"
        [leader-length="lunghezza"]
        [rule-style="stile"]
        [rule-thickness="spessore"]
        [altri_attributi]  />
```

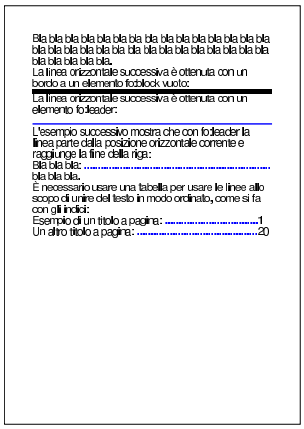
Se l'elemento contiene qualcosa e all'attributo '**`leader-pattern`**' viene associato il valore '**`use-content`**', questo viene usato per costruire la linea (per esempio si potrebbe realizzare una linea fatta di asterischi o di sequenze più complesse). Tuttavia, di solito si vogliono realizzare soltanto linee continue, spezzate o puntate, pertanto l'elemento si usa prevalentemente vuoto.

È importante osservare che la linea che si ottiene raggiunge la fine dello spazio orizzontale disponibile, pertanto, per realizzare linee di collegamento come quelle degli indici, dove appare un titolo alla sinistra e un numero di pagina alla destra, occorre inserire tutto in una tabella. Nell'esempio seguente vengono mostrati diversi ca-

si, compresa la simulazione di un indice elementare, inserito in una tabella:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE fo:root SYSTEM "fo.dtd">
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
    <fo:layout-master-set>
        <fo:simple-page-master master-name="normale"
            page-width="210mm" page-height="297mm"
            margin-top="2cm"   margin-bottom="2cm"
            margin-left="2cm"  margin-right="2cm">
            <fo:region-body />
        </fo:simple-page-master>
    </fo:layout-master-set>
    <fo:page-sequence master-reference="normale"
        language="it" country="it">
        <fo:flow flow-name="xsl-region-body">
            <fo:block font-size="7mm">
                <fo:block>
                    Bla bla bla bla bla bla bla bla bla bla
                    bla bla bla bla bla bla bla bla bla bla
                    bla bla bla bla bla bla bla bla bla bla
                    bla bla bla bla bla.
                </fo:block>
                <fo:block>
                    La linea orizzontale successiva &#x00E8;
                    ottenuta con un bordo a un elemento
                    fo:block vuoto:
                </fo:block>
                <fo:block
                        border-after-style="solid"
                        border-after-width="3mm"
                        border-after-color="black" />
                <fo:block>
                    La linea orizzontale successiva &#x00E8;
                    ottenuta con un elemento fo:leader:
                </fo:block>
                <fo:block>
                    <fo:leader
                            leader-pattern="rule"
                            rule-style="solid"
                            rule-thickness="1mm"
                            color="blue"/>
                </fo:block>
                <fo:block>
                    L'esempio successivo mostra che con
                    fo:leader la linea parte dalla posizione
                    orizzontale corrente e raggiunge la fine
                    della riga:
                </fo:block>
                <fo:block>
                    Bla bla bla:
                    <fo:leader
                        leader-alignment="reference-area"
                        leader-pattern="dots"
                        rule-style="solid"
                        rule-thickness="1mm"
                        color="blue"/>
                    bla bla bla.
                </fo:block>
                <fo:block>
                    &#x00C8; necessario usare una tabella
                    per usare le linee allo scopo di unire
                    del testo in modo ordinato, come si fa
                    con gli indici:
                </fo:block>
                <fo:table   table-layout="fixed">
                    <fo:table-column column-width="16cm"/>
                    <fo:table-column column-width="1cm"/>
                    <fo:table-body>
                        <fo:table-row>
                            <fo:table-cell>
                                <fo:block>
                                    Esempio di un titolo a
                                    pagina:
                                    <fo:leader
                                      leader-pattern="dots"
                                      rule-style="solid"
                                      rule-thickness="1mm"
                                      color="blue"/>
                                </fo:block>
                            </fo:table-cell>
                            <fo:table-cell>
                                <fo:block>
                                    1
                                </fo:block>
                            </fo:table-cell>
                        </fo:table-row>
                        <fo:table-row>
                            <fo:table-cell>
                                <fo:block>
                                    Un altro titolo a
                                    pagina:
                                    <fo:leader
                                       leader-pattern="dots"
                                       rule-style="solid"
                                       rule-thickness="1mm"
                                       color="blue"/>
                                </fo:block>
                            </fo:table-cell>
                            <fo:table-cell>
                                <fo:block>
                                    20
                                </fo:block>
                            </fo:table-cell>
                        </fo:table-row>
                    </fo:table-body>
                </fo:table>
            </fo:block>
        </fo:flow>
    </fo:page-sequence>
</fo:root>
```

Ecco il risultato della composizione dell'esempio:



52.5.7 Riferimenti incrociati

XSL-FO ha una gestione molto sofisticata dei riferimenti ipertestuali; tuttavia, qui si vuole mostrare solo l'uso più semplice. In generale, la maggior parte degli elementi può contenere l'attributo '**id**', con lo scopo di associare una stringa di identificazione univoca, in modo da consentire successivamente di raggiungere il contenuto di questi elementi attraverso un riferimento interno.

Per fare riferimento a qualcosa, si usa l'elemento '**fo:basic-link**' (nell'ambito di un contesto lineare), specificando se si tratta di un riferimento interno o esterno, con l'uso alternativo degli attributi '**internal-destination**' e '**external-destination**':

```
<fo:basic-link
        {internal-destination="stringa_id"}↩
↪       |{external-destination="indirizzo_uri"}
        [altri_attributi]>
    contenuto
</fo:basic-link>
```

Il testo contenuto nell'elemento serve per individuare la zona su cui funziona la selezione del riferimento, come nell'esempio seguente, dove la parola «qui» serve come superficie su cui fare un clic con il mouse:

```
Per raggiungere il sito, basta fare un clic <fo:basic-link
external-destination="http://www.brot.dg">qui</fo:basic-link>.
```

Per fare riferimento a una pagina, in modo da ottenerne il numero nella composizione stampata, si usano due elementi per due situazioni differenti:

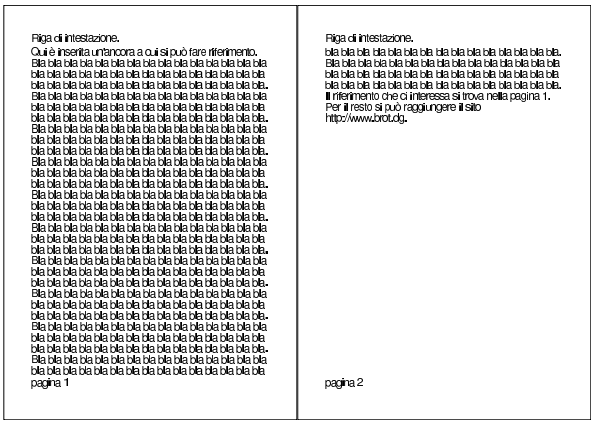
```
<fo:page-number [attributi] />
```

```
<fo:page-number-citation
        ref-id="stringa_a_cui_si_fa_riferimento"
        [attributi] />
```

Nel primo caso, si ottiene il numero della pagina corrente; nel secondo, si ottiene il numero di pagina a cui si fa riferimento con la stringa indicata con l'attributo '**ref-id**'. L'esempio seguente riassume l'uso di questi elementi, dove in particolare si inserisce il numero di pagina nella regione inferiore:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE fo:root SYSTEM "fo.dtd">
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
    <fo:layout-master-set>
        <fo:simple-page-master
                master-name="normale"
                page-width="210mm"
                page-height="297mm"
                margin-top="2cm"
                margin-bottom="2cm"
                margin-left="2cm"
                margin-right="2cm">
            <fo:region-body margin-top="1cm"
                margin-bottom="1cm"/>
            <fo:region-before extent="1cm" />
            <fo:region-after  extent="1cm" />
        </fo:simple-page-master>
    </fo:layout-master-set>
    <fo:page-sequence
            master-reference="normale"
            initial-page-number="1"
            force-page-count="even"
            language="it"
            country="it">
        <fo:static-content flow-name="xsl-region-before">
            <fo:block font-size="7mm">
                Riga di intestazione.
            </fo:block>
        </fo:static-content>
        <fo:static-content flow-name="xsl-region-after">
            <fo:block font-size="7mm">
                pagina <fo:page-number />
            </fo:block>
        </fo:static-content>
        <fo:flow flow-name="xsl-region-body">
            <fo:block font-size="7mm">
                <fo:block id="riferimento">
                    Qui &#x00E8; inserita un'ancora a cui si
                    pu&#x00F2; fare riferimento.
                </fo:block>
                <fo:block>
                    Bla bla bla bla bla bla bla bla bla bla bla
                    bla bla bla bla bla bla bla bla bla bla bla
                    bla bla bla bla bla bla bla bla bla bla bla
                    bla bla bla bla bla bla bla bla bla bla bla
                    bla bla bla bla bla.
                </fo:block>
                <!-- Inserire qui altri fo:block per
                     ottenere piu' pagine -->
                <fo:block>
                    Il riferimento che ci interessa si trova
                    nella <fo:basic-link
                    internal-destination="riferimento">pagina
                    <fo:page-number-citation
                    ref-id="riferimento"/></fo:basic-link>.
                    Per il resto <fo:basic-link
                external-destination="http://www.brot.dg">si
                    pu&#x00F2; raggiungere il sito
                    http://www.brot.dg</fo:basic-link>.
                </fo:block>
            </fo:block>
        </fo:flow>
    </fo:page-sequence>
</fo:root>
```


Ecco come potrebbe apparire il risultato della composizione:



52.5.8 Componenti autonomi rispetto al testo

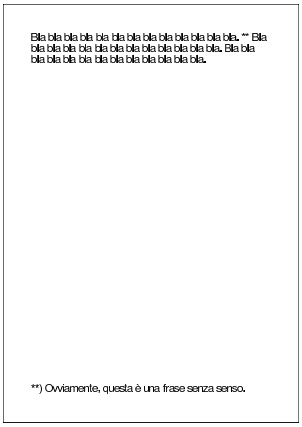
Esistono due tipi di componenti che, nella composizione, non si collocano nella posizione in cui appaiono nel corso del documento. Si tratta di note a piè pagina e di oggetti fluttuanti. Le note a piè pagina si inseriscono nell'ambito di un contesto lineare e si ottengono con questa sintassi:

```
<fo:footnote>
    <fo:inline [attributi]>
        simbolo
    </fo:inline>
    <fo:footnote-body>
        blocchi
    </fo:footnote-body>
</fo:footnote>
```

In pratica, l'elemento '**fo:footnote**' contiene necessariamente un elemento '**fo:inline**', il cui scopo è quello di inserire un simbolo o qualunque cosa serva per rinviare il lettore alla nota a piè pagina; subito dopo, l'elemento '**fo:footnote-body**' contiene uno o più blocchi che contengono la nota vera e propria. Si tenga in considerazione il fatto che, probabilmente, si vuole inserire all'inizio delle note lo stesso simbolo usato per il richiamo.

```
<fo:block>
    Bla bla bla bla bla bla bla bla bla bla bla bla bla.
    <fo:footnote>
        <fo:inline>**</fo:inline>
        <fo:footnote-body>
            <fo:block>
                **) Ovviamente, questa &#x00E8; una
                frase senza senso.
            </fo:block>
        </fo:footnote-body>
    </fo:footnote>
    Bla bla bla bla bla bla bla bla bla bla bla bla bla.
    Bla bla bla bla bla bla bla bla bla bla bla bla bla.
</fo:block>
```

L'esempio mostra solo una porzione di un file XSL-FO, che, nella composizione finale potrebbe apparire in modo simile a questo:



Gli oggetti fluttuanti si inseriscono all'interno dell'elemento '**`fo:float`**', che si comporta come un blocco:

```
<fo:float
        float="before"|"start"|"end"|…
        [clear="posizione"]>
    blocchi
</fo:float>
```

In pratica, con l'aiuto dell'attributo '**`float`**' si può stabilire in che modo debba essere collocato l'oggetto fluttuante, mentre l'attributo '**`clear`**' serve a definire un comportamento in prossimità di altri oggetti fluttuanti.

Non vengono mostrati esempi sull'uso di questo elemento, anche se si può intuire la sua importanza in presenza di immagini e piccole tabelle che è bene collocare in modo dinamico nel corpo di un documento.

52.6 XSLT e XSL-FO

Fino a questo punto, XSLT e XSL-FO sono stati descritti separatamente, per evitare confusione, mentre fanno parte di un sistema integrato. Una volta compreso il meccanismo di XSLT e di XSL-FO, non dovrebbe essere difficile unire le due cose per trasformare un documento XML in un file XSL-FO, allo scopo di arrivare poi alla composizione finale.

In precedenza è stato mostrato un file XML molto semplice, associando a questo la trasformazione in formati comuni:

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE relazione [
    <!ELEMENT relazione     (titolo?, data, contenuto)>
    <!ATTLIST relazione
        xml:lang   CDATA    "">
    <!ELEMENT titolo        (#PCDATA)>
    <!ELEMENT data          (#PCDATA)>
    <!ELEMENT contenuto     (paragrafo+, firma+)>
    <!ELEMENT paragrafo     (#PCDATA)>
    <!ELEMENT firma         (#PCDATA)>
]>

<relazione xml:lang="it">
<titolo>Relazione introduttiva su XML</titolo>

<data>11/11/2011</data>

<contenuto>

<paragrafo>XML sta per Extensible Markup Language.    bla
bla bla... Perché,... così,... perciò,...
sarà...</paragrafo>

<paragrafo>Bla, bla, bla....</paragrafo>

<firma>Pinco Pallino</firma>

</contenuto>
</relazione>
```


Per trasformare questo tipo di file in un foglio di stile XSL-FO, si può usare un foglio di stile XSLT come quello seguente; si osservi anche la presenza della dichiarazione del dominio applicativo associato al prefisso '**`fo:`**'.

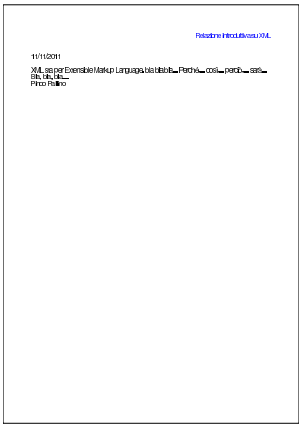
```
<xsl:stylesheet
    version="1.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:fo="http://www.w3.org/1999/XSL/Format">
    <xsl:output
        method="xml"
        indent="yes"
        encoding="UTF-8" />
    <xsl:template match="/">
        <fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
            <fo:layout-master-set>
                <fo:simple-page-master master-name="normale"
                    page-width="210mm" page-height="297mm"
                    margin-top="2cm"   margin-bottom="2cm"
                    margin-left="2cm"  margin-right="2cm">
                    <fo:region-body />
                </fo:simple-page-master>
            </fo:layout-master-set>
            <fo:page-sequence master-reference="normale"
                language="{relazione/@xml:lang}">
                <fo:flow flow-name="xsl-region-body">
        <xsl:apply-templates />
                </fo:flow>
            </fo:page-sequence>
        </fo:root>
    </xsl:template>
    <xsl:template match="titolo">
        <fo:block font-size="4mm" space-after="1cm"
            text-align="right" color="blue">
            <xsl:apply-templates />
        </fo:block>
    </xsl:template>
    <xsl:template match="data">
        <fo:block space-after="5mm">
            <xsl:apply-templates />
        </fo:block>
    </xsl:template>
    <xsl:template match="paragrafo">
        <fo:block>
            <xsl:apply-templates />
        </fo:block>
    </xsl:template>
    <xsl:template match="firma">
        <fo:block>
            <xsl:apply-templates />
        </fo:block>
    </xsl:template>
</xsl:stylesheet>
```

In questo caso la trasformazione genera il file XSL-FO seguente:

```
<?xml version="1.0" encoding="UTF-8"?>
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
<fo:layout-master-set>
<fo:simple-page-master master-name="normale"
    page-width="210mm" page-height="297mm" margin-top="2cm"
    margin-bottom="2cm" margin-left="2cm"
    margin-right="2cm">
<fo:region-body/>
</fo:simple-page-master>
</fo:layout-master-set>
<fo:page-sequence master-reference="normale" language="it">
<fo:flow flow-name="xsl-region-body">
<fo:block font-size="4mm" space-after="1cm"
    text-align="right"
    color="blue">Relazione introduttiva su XML</fo:block>

<fo:block space-after="5mm">11/11/2011</fo:block>



<fo:block>XML sta per Extensible Markup Language.    bla
bla bla... Perché,... così,... perciò,...
sarà...</fo:block>

<fo:block>Bla, bla, bla....</fo:block>
```

```
<fo:block>Pinco Pallino</fo:block>


</fo:flow>
</fo:page-sequence>
</fo:root>
```

Il risultato della composizione:



Per il resto, si veda quanto già descritto in precedenza a proposito di XSLT nella sezione 52.2.

52.7 XMLTeX e PassiveTeX

XMLTeX[4] è un sistema per comporre file XML attraverso TeX; PassiveTeX[5] è un insieme di fogli di stile aggiuntivi che, in particolare, consentono a XMLTeX di elaborare file XSL-FO.

52.7.1 XMLTeX

XMLTeX è un sistema che si affianca normalmente a LaTeX per trasformare un file XML in un documento pronto per la stampa, attraverso un metodo di trasformazione degli elementi e degli attributi del file di origine in comandi di LaTeX. Per fare questo occorre predisporre un «file di configurazione» nel quale si dichiarano queste sostituzioni. Viene proposto il solito esempio di file XML molto semplice, dove in particolare l'attributo '**`linguaggio`**' contiene un nome valido per LaTeX:

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE relazione [
    <!ELEMENT relazione     (titolo?, data, contenuto)>
    <!ATTLIST relazione
        linguaggio  CDATA   "">
    <!ELEMENT titolo        (#PCDATA)>
    <!ELEMENT data          (#PCDATA)>
    <!ELEMENT contenuto     (paragrafo+, firma+)>
    <!ELEMENT paragrafo     (#PCDATA)>
    <!ELEMENT firma         (#PCDATA)>
]>

<relazione linguaggio="italian">
<titolo>Relazione introduttiva su XML</titolo>

<data>11/11/2011</data>

<contenuto>

<paragrafo>XML sta per Extensible Markup Language.    bla
bla bla... Perché,... così,... perciò,...
sarà...</paragrafo>

<paragrafo>Bla, bla, bla...</paragrafo>

<paragrafo>... ma soprattutto vorrei sorvolare sopra
questioni così complesse e arzigogolate come qualcuno
invece vorrebbe...</paragrafo>

<firma>Pinco Pallino</firma>
```



```
</contenuto>
</relazione>
```

Supponendo di dare a questo file il nome '`relazione.xml`', si può realizzare il file '`relazione.cfg`', con l'indicazione delle trasformazioni da apportare:

```
\XMLelement{relazione}
    {\XMLattribute{linguaggio}
        {\mylanguageattribute}{english}}{%
    \documentclass{article}
    \usepackage[OT2,OT1]{fontenc}
    \usepackage[\mylanguageattribute]{babel}
    \frenchspacing
    \begin{document}
}
{%
    \end{document}
}

\XMLelement{titolo}{}{%
    \xmlgrab
}
{
    \section{#1}
}

\XMLelement{data}{}{%
}
{
    \par
}

\XMLelement{contenuto}{}{%
}
{
}

\XMLelement{paragrafo}{}{%
}
{
    \par
}

\XMLelement{firma}{}{%
}
{
    \par
}
```

Come si vede si tratta di comandi TeX con cui si dichiarano gli abbinamenti tra elementi e attributi con comandi corrispondenti di LaTeX.

```
\XMLelement{nome_elemento}{[dichiarazione_attributi]}↩
↪{codice_iniziale}{codice_finale}
```

```
\XMLelement{nome_elemento}{[dichiarazione_attributi]}↩
↪{\xmlgrab}{codice_complessivo}
```

Il comando '**`\XMLelement`**' consente di dichiarare la trasformazione dei marcatori di un elemento in comandi LaTeX: gli ultimi due argomenti contengono rispettivamente il codice LaTeX di apertura e il codice di chiusura dell'elemento. Tuttavia, se nell'argomento destinato a ospitare il codice LaTeX di apertura si inserisce soltanto il comando '**`\xmlgrab`**', si può mettere tutto nell'argomento finale, usando la sigla '**`#1`**' per fare riferimento al contenuto dell'elemento.

Nel secondo argomento del comando '**`\XMLelement`**' si possono inserire comandi '**`\XMLattribute`**' per prelevare le informazioni sugli attributi:

```
\XMLattribute{nome_attributo}{nome_comando_tex}{[valore_predefinito]}
```

Osservando l'esempio del file '`relazione.cfg`' si può vedere che nella dichiarazione riferita all'elemento '**`relazione`**'

appare l'associazione dell'attributo '**linguaggio**' al comando '**\mylanguageattribute**', che poi viene usato successivamente. In pratica, il comando di LaTeX che nel file '`relazione.cfg`' appare come la riga seguente, è come se venisse sostituito con quella successiva:

```
    \usepackage[\mylanguageattribute]{babel}
```

```
    \usepackage[italian]{babel}
```

Nel file di configurazione, ovvero il file '`.cfg`', possono essere collocati altri comandi importanti; fortunatamente molte indicazioni essenziali fanno già parte della configurazione generale, che potrebbe trovarsi nel file '`/etc/texmf/xmltex/xmltex.cfg`'. Eventualmente si può approfondire la cosa leggendo la documentazione originale di XMLTeX.

Disponendo del file di configurazione per la trasformazione del documento XML, prima di passare alla composizione con LaTeX, si deve realizzare un piccolo file di collegamento, denominato, in questo caso, '`relazione.tex`':

```
\def\xmlfile{relazione.xml}
\input xmltex.tex
```

Si può passare così alla composizione di questo file '`relazione.tex`':

```
$ latex relazione.tex [invio]

This is TeX, Version 3.14159 (Web2C 7.4.5)
(./relazione.tex
LaTeX2e <2001/06/01>
Babel <v3.7h> and hyphenation patterns for american, french,
german, ngerman, danish, dutch, finnish, greek, italian,
latin, spanish, nohyphenation, loaded.
(/usr/share/texmf/tex/xmltex/base/xmltex.tex
xmltex version: 2002/06/25 v1.9 (Exp)

Encoding = utf-8
(/usr/share/texmf/tex/xmltex/config/xmltex.cfg)
(./relazione.cfg) (./relazione.xml
 Default: relazione linguaggio=""
 (/usr/share/texmf/tex/latex/base/article.cls
Document Class: article 2001/04/21 v1.4e Standard LaTeX
document class
(/usr/share/texmf/tex/latex/base/size10.clo))
(/usr/share/texmf/tex/latex/base/fontenc.sty
(/usr/share/texmf/tex/latex/cyrillic/ot2enc.def)
(/usr/share/texmf/tex/latex/base/ot1enc.def))
(/usr/share/texmf/tex/generic/babel/babel.sty
(/usr/share/texmf/tex/generic/babel/italian.ldf
(/usr/share/texmf/tex/generic/babel/babel.def)))
(./relazione.aux)
(/usr/share/texmf/tex/latex/cyrillic/ot2cmr.fd)
[1] (./relazione.aux) ) ) )
Output written on relazione.dvi (1 page, 692 bytes).
Transcript written on relazione.log.
```

Si ottiene così il file '`relazione.dvi`', da cui si può generare facilmente un file PostScript:

```
$ dvips -o relazione.ps relazione.dvi [invio]

This is dvips(k) 5.92b Copyright 2002 Radical Eye Software
(www.radicaleye.com) ' TeX output 2003.08.09:1533' ->
relazione.ps
<texc.pro><f7b6d320.enc><texps.pro>.
<cmr10.pfb><cmbx12.pfb>[1]
```

In modo analogo si può ottenere una composizione in formato PDF, utilizzando '**pdflatex**':

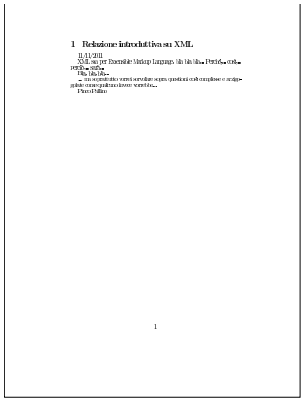
```
$ pdflatex relazione.tex [invio]
```

Eventualmente, entrambe le cose si possono fare in modo più semplice, senza bisogno di realizzare il file '`relazione.tex`', con i comandi seguenti, che servono a ottenere, rispettivamente, la composizione in formato DVI e PDF:

```
$ latex "&xmltex" relazione.tex [Invio]

$ pdflatex "&pdfxmltex" relazione.tex [Invio]
```


Nel caso dovessero essere presenti riferimenti incrociati, è necessario eseguire la composizione con LaTeX per tre volte di seguito (sempre con lo stesso comando).

Per concludere viene mostrato il risultato della composizione dell'esempio:



52.7.2 PassiveTeX

Come accennato all'inizio del capitolo, PassiveTeX è un insieme di fogli di stile da usare con XMLTeX per comporre direttamente file XSL-FO, attraverso LaTeX. Viene ripreso un esempio già apparso in un altro capitolo di file XSL-FO:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE fo:root SYSTEM "fo.dtd">
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
    <fo:layout-master-set>
        <fo:simple-page-master master-name="normale"
            page-width="210mm" page-height="297mm"
            margin-top="2cm"   margin-bottom="2cm"
            margin-left="2cm"  margin-right="2cm">
            <fo:region-body />
        </fo:simple-page-master>
    </fo:layout-master-set>
    <fo:page-sequence master-reference="normale"
        language="it" country="it">
        <fo:flow flow-name="xsl-region-body">
            <fo:block font-size="7mm">
                <fo:block>
                    Bla bla bla <fo:inline
                    font-style="italic">testo in
                    corsivo</fo:inline> bla bla bla
                    <fo:inline
                    font-weight="bold">H<fo:inline
            baseline-shift="sub">2</fo:inline>O</fo:inline>
                    bla bla bla <fo:inline
                    font-family="monospace">ls -l | sort
                    &gt; prova</fo:inline> bla bla bla
                    <fo:inline color="blue">testo in
                    blu</fo:inline> bla bla bla <fo:inline
                    font-variant="small-caps">testo in
                    maiuscoletto</fo:inline> bla bla bla.
                </fo:block>
            </fo:block>
        </fo:flow>
    </fo:page-sequence>
</fo:root>
```

In modo molto semplice, si arriva alla composizione di questo file con il comando seguente, supponendo che si tratti del file '`esempio-014.fo`':

```
$ latex "&xmltex" esempio-014.fo [Invio]
```

In questo modo si genera il file '`esempio-014.dvi`', dal quale si può ottenere un file PostScript nel modo consueto:

```
$ dvips -o esempio-014.ps esempio-014.dvi [Invio]
```

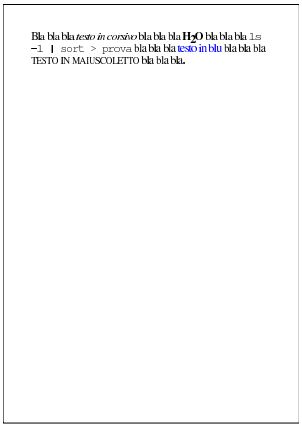
Ovviamente si può anche eseguire una composizione in formato

PDF direttamente con il comando seguente:

```
$ pdflatex "&pdfxmltex" esempio-014.fo [ Invio ]
```

Nel caso dovessero essere presenti riferimenti incrociati, sarebbe necessario eseguire la composizione con LaTeX per tre volte di seguito.

Ecco il risultato della composizione:



52.8 Riferimenti

- W3C, *Extensible Markup Language (XML) 1.0*, http://www.w3.org/TR/REC-xml/
- James Clark, *Comparison of SGML and XML*, http://www.w3.org/TR/NOTE-sgml-xml-971215.html
- Peter Flynn, *The XML FAQ*, http://xml.silmaril.ie/
- Norman Walsh, *A Technical Introduction to XML*, http://nwalsh.com/docs/articles/xml/
- W3C, *Namespaces in XML*, http://www.w3.org/TR/REC-xml-names/
- James Clark, *XML namespaces*, http://www.jclark.com/xml/xmlns.htm
- O'Reilly, *xml.com*, http://www.xml.com
- *Cover Pages*, http://xml.coverpages.org/xml.html
- W3C, *XSL Transformations (XSLT)*, http://www.w3.org/TR/xslt
- W3C, *XML Path Language (XPath)*, http://www.w3.org/TR/xpath
- Elliotte Rusty Harold, *XML Bible*, http://rf4.us/lib/old/books/programming/XMLBible.pdf
- Michele Sciabarrà, *Tutorial XML/XSLT*, http://www.corsojava.it/testi/xml/
- Michele Sciabarrà, *Linux e programmazione web*, McGraw-Hill, 2001, ISBN 88-386-4177-3; in particolare il capitolo *XML e XSLT*
- Apache software foundation, *FOP*, http://xmlgraphics.apache.org/fop/
- W3C, *Extensible stylesheet language (XSL)*, http://www.w3.org/TR/xsl/
- W3C, *Extensible stylesheet language*, http://www.w3.org/TR/xsl/
- Herong Yang, *Herong's notes on XSL-FO and XHTML*, http://www.herongyang.com/xhtml/
- David Carlisle, *xmltex: A non validating (and not 100% conforming) namespace aware XML parser implemented in TeX**, http://www.dcarlisle.demon.co.uk/xmltex/manual.html

- Sebastian Rahtz, *PassiveTeX*, http://wayback.archive.org/web/*/http://www.tei-c.org/Software/passivetex

[1] **Xalan** software libero con licenza speciale

[2] **FOP** software libero con licenza speciale

[3] Kaffe è un vecchio interprete di applicazioni Java, sviluppato prima che fosse rilasciato il pacchetto OpenJDK con una licenza libera. Attualmente, le distribuzioni GNU/Linux comuni dovrebbero disporre di una «macchina virtuale» Java, ed eventualmente di un ambiente di sviluppo Java, attraverso OpenJDK, GCJ e altri componenti liberi, senza complicazioni per l'utilizzatore.

[4] **XMLTeX** LPPL

[5] **PassiveTeX** Software libero con licenza speciale