XSL-FO

Massimiliano Tambini

12 maggio 2000

Introduzione

Oggi esaminiamo in breve:

- XSLFO, ovvero un vocabolario di elementi che specificano una semantica di formattazione per documenti XML.

Evoluzione di XSL

- 27 agosto 1997: prima nota del W3C che stabilisce la filosofia generale del linguaggio.
- 18 agosto 1998: primo Working Draft che separa nettamente la fase di trasformazione dalla fase di visualizzazione. Cambia la sintassi. Sono introdotti i namespace.
- 21 aprile 1999: separazione di XSL in due Draft distinti XSLT e XSL-FO.
- Situazione attuale: XSLT è recommendation. XSL-FO è ancora in fase di discussione.

XSL-FO

Scopi del linguaggio:

- definire la fase di formattazione.
- definire un vocabolario di elementi di formattazione indipendenti dal tipo di supporto utilizzato per l'output.

Il legame tra le due definizioni è ovviamente molto stretto in quanto la fase di formattazione interpreta l'albero che risulta dall'eventuale trasformazione in base alla semantica degli oggetti di formattazione che lo costituiscono.

Introduzione alla formattazione

Nell'output su supporti visuali per astrarre dal tipo di scrittura usato (ad esempio occidentale o orientale) si introduce il writing-mode che definisce:

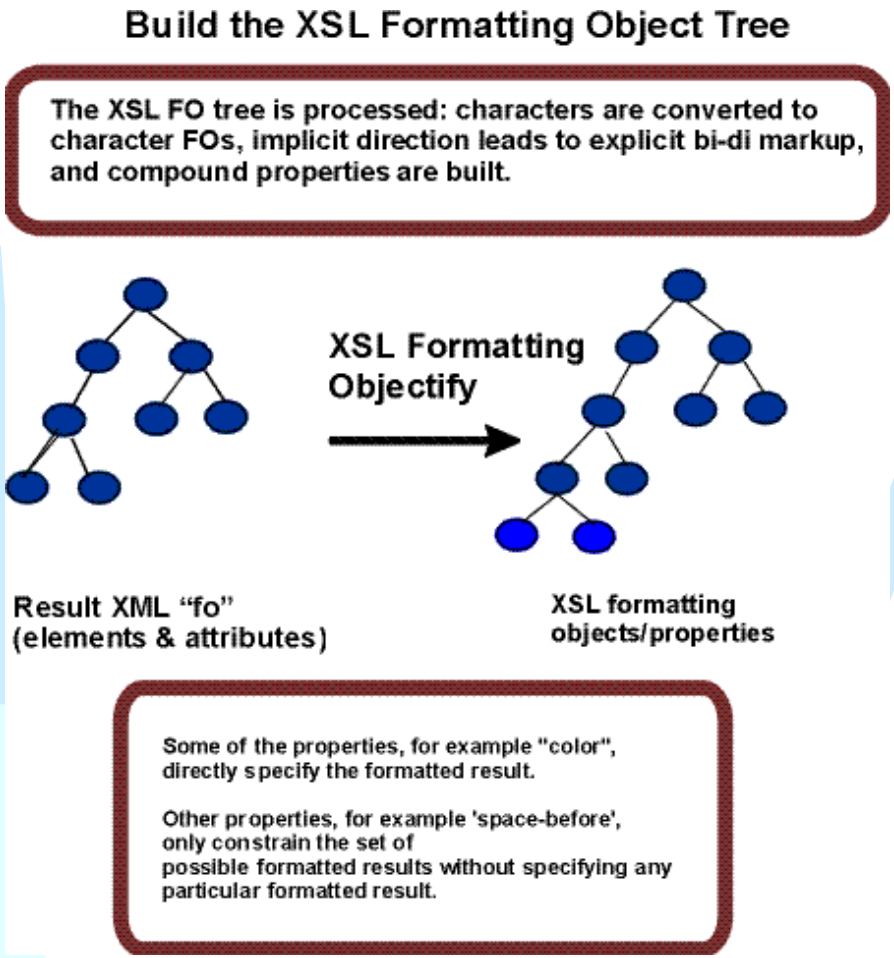
- Direzioni relative (block- e inline- progression-direction)
- Riferimenti relativi (before, after, start o end)

Fasi della formattazione

La fase di formattazione è schematizzabile in 4 passi:

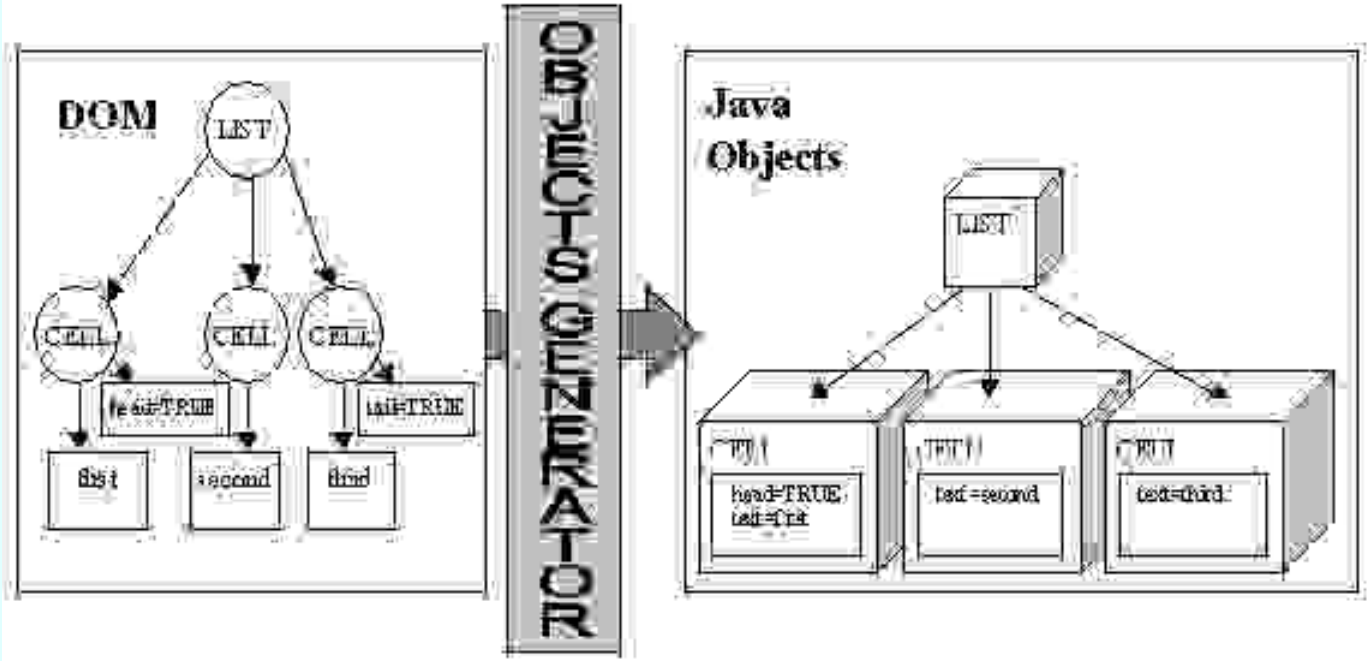
- Trasformazione dell'albero ottenuto attraverso un' elaborazione XSLT del documento XML iniziale in uno costituito, non da elementi e attributi, ma da oggetti di formattazione e loro proprietà.
- Raffinamento dell'albero degli oggetti di formattazione ovvero mapping dalle proprietà nelle caratteristiche.
- Costruzione dell'albero delle aree.
- Rendering

Costruzione dell'albero dei FO (1)

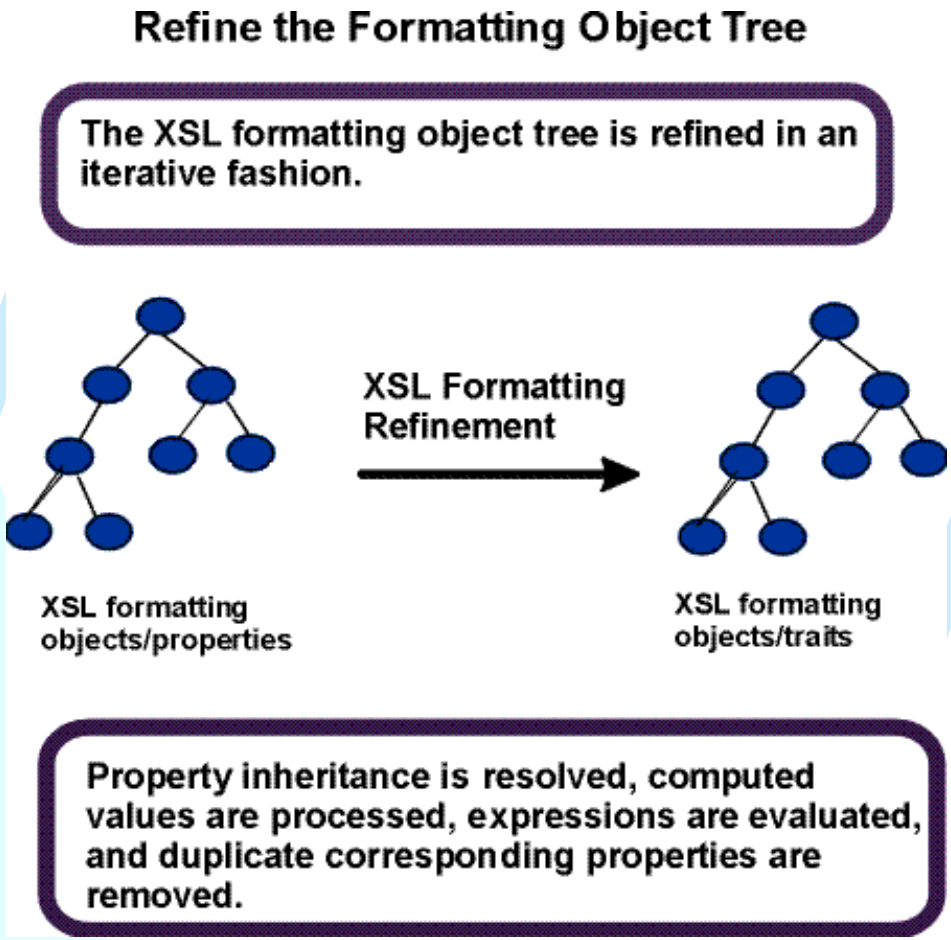


Costruzione dell'albero dei FO (2)

Scelta del modello: XMLC.



Raffinamento dell'albero (1)



Raffinamento (2)

Valori specificati, calcolati e applicabili.

- Valore in %
- Valore assoluto
- Valore espresso in unità di misura utilizzate dal supporto di output (es. pixel)

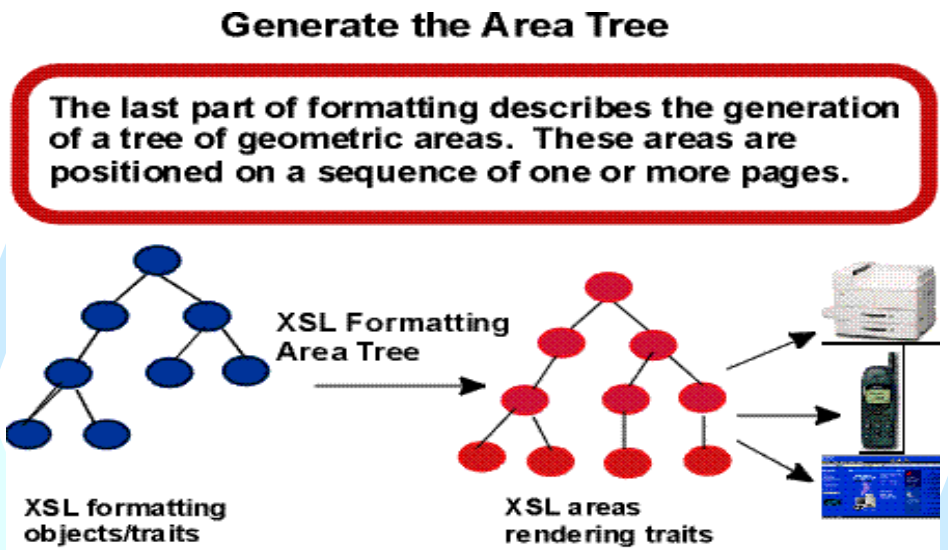
Proprietà espresse in forma breve

- Border: "**`10px solid blue`**"

Mapping di proprietà relative

- Border-before = border-top se writing-mode è lr-tb

Costruzione albero delle aree e rendering



Modello delle aree

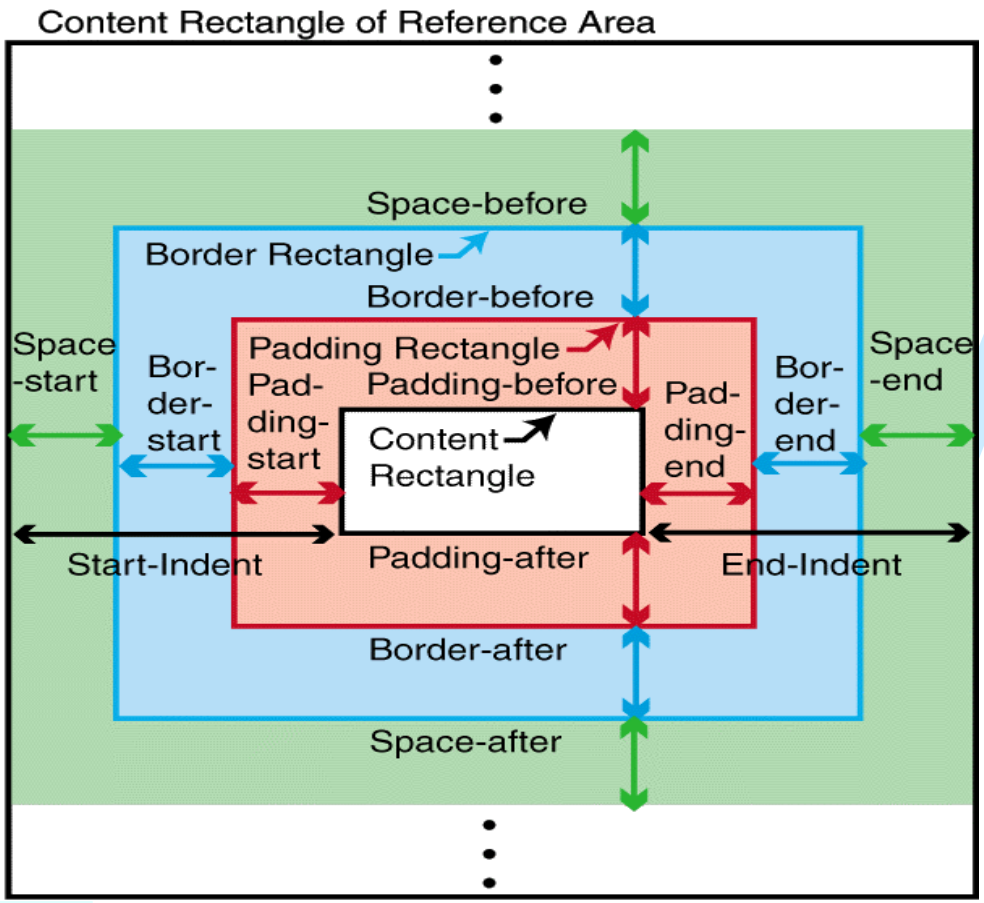
Il modello delle aree definisce: aree rettangolari e spazi. Le aree rettangolari, generate dagli oggetti di formattazione, riservano spazio e racchiudono contenuto. Gli spazi riservano spazio prima e dopo le aree rettangolari e non hanno contenuto.

Aree Rettangolari

- Block-area
- Inline-area.

La loro differenza principale sta nel modo in cui normalmente il formatter le posiziona. Esse vengono tipicamente posizionate seguendo rispettivamente la block-progression-direction e la inline-progression-direction, in alcuni casi però è possibile che siano posizionate esplicitamente (ad esempio in base alle absolute-position-properties).

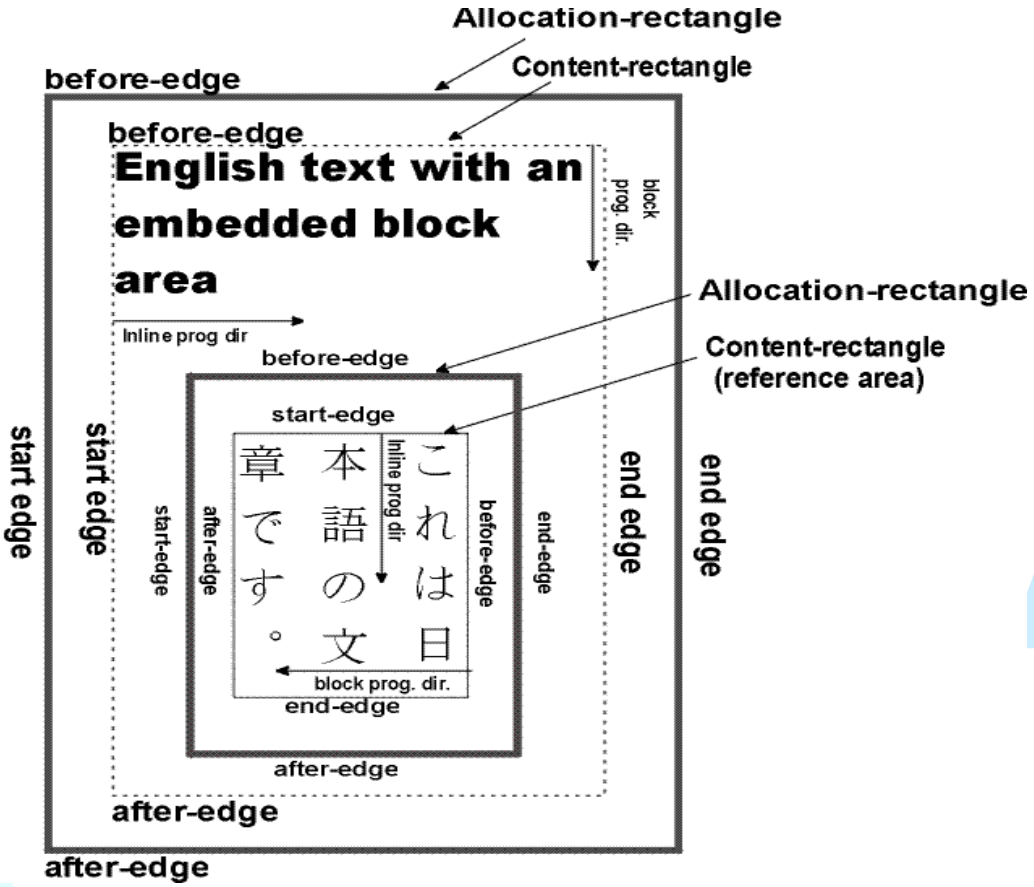
Aree Rettangolari, caratteristiche comuni



Caratteristiche comuni (2)

- Content rectangle.
- Padding Rectangle.
- Border Rectangle.
- Allocation Rectangle: definisce la dimensione usata per allocare spazio quando l'area viene posizionata nell'area padre.

Aree Rettangolari e writing-mode



Ordinamento delle aree Rettangolari

Ci serve per definire termini come area in testa, in coda, seguente o precedente.

L'ordine non è univocamente definito ma dipende dal tipo di visita che si esegue sull'albero:

- Previsita
- Postvisita

Ordinamento delle aree Rettangolari (2)

L’ordine delle aree può essere definito solo per quelle aree dell’albero che in fase di posizionamento rispettano:

- l’annidamento definito dall’albero stesso
- la direzione di posizionamento dipendente dal tipo di area (block-progression-direction o inline-progression-direction).

Non ne sono coinvolte ad esempio le aree posizionate in maniera assoluta, le celle di una tabella o gli elementi di una lista. In questi casi l’area si definisce out-of-sequence.

Aree rettangolari consecutive

Se A e B sono block-area non out-of-sequence si dice che A precede consecutivamente B (o B segue consecutivamente A) se:

- A precede B e non esistono aree non out-of-sequence tra A e B.
- Tutti gli antenati di A che precedono B hanno border-after e padding-after nullo.
- Tutti gli antenati di B che seguono A hanno border-before e padding-before nullo.

La stessa definizione vale per le inline-area con la differenza che devono essere nulli border-, paddind-end e border-, padding-start.

Aree consecutive (2)

Se A e B sono block-area non out-of-sequence si dice che A è in testa a B se:

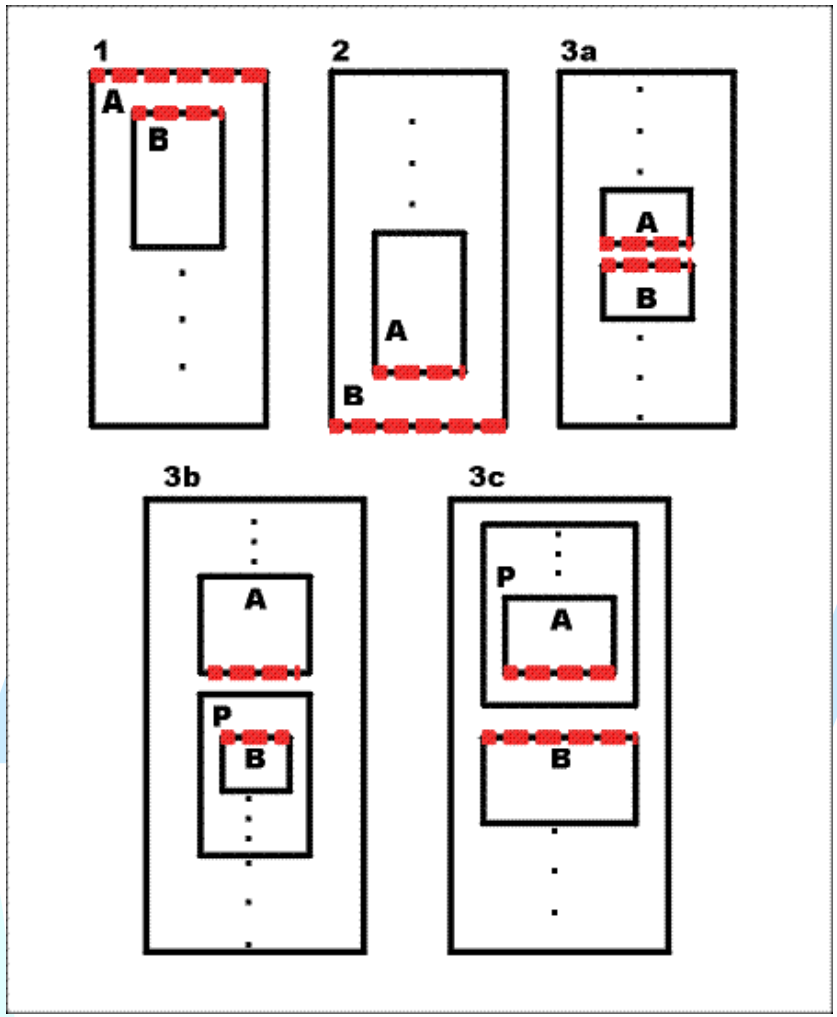
- A è il primo figlio di B.
- A è il primo figlio di C, C è in testa a B ed ha border- e padding-before nulli.

Se A e B sono block-area non out-of-sequence si dice che A è in coda a B se:

- A è l'ultimo figlio di B.
- A è l'ultimo figlio di C, C è in coda a B ed ha border- e padding-after nulli.

In entrambi i casi A e B sono considerate consecutive.

Aree Rettangolari adiacenti



Spazi

Gli spazi riservano spazio prima e dopo le aree rettangolari e non hanno contenuto. Essi vengono definiti attraverso un tipo di dato composto dalle seguenti informazioni:

- Space.minimum
- Space.maximum
- Space.optimum
- Space.conditionality
- Space.precedence

In generale gli stessi formatting-object che generano aree rettangolari generano gli spazi ad esse associati.

Regole di risoluzione degli spazi (1)

Dato uno spazio S, per calcolare il suo corrispondente spazio risolto, è necessario considerare la massima sequenza di spazi consecutivi contenenti S.

Gli spazi sono consecutivi quando sono associati ad aree consecutive.

Regole di risoluzione degli spazi (2)

Regola 1

- Se qualcuno degli spazi nella sequenza è conditional ed è in testa ad una reference-area o line-area, allora tale spazio sarà soppresso. Tutti gli spazi conditional che lo seguono consecutivamente saranno altresì soppressi. Lo stesso dicasi per gli spazi conditional in coda ad una reference-area o line-area.

Regole di risoluzione degli spazi (3)

Regola 2

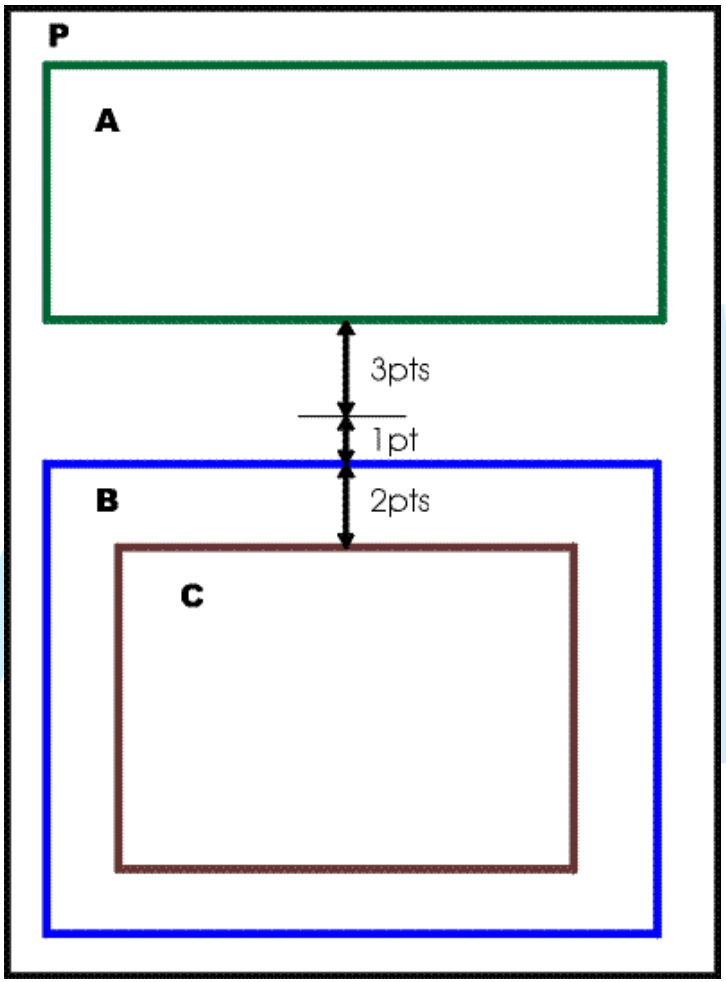
- Se qualcuno degli spazi rimanenti e forcing allora ogni spazio non forcing è soppresso dalla sequenza. In questo caso lo spazio risultante avrà le dimensioni optimum, minimum e maximum pari alla somma dei valori corrispondenti di ogni spazio rimasto nella sequenza. Tale spazio sarà non conditional.

Regole di risoluzione degli spazi (4)

Regola 3

- Se tutti gli spazi sono non forcing saranno selezionati gli spazi con valore precedence più alto e a parità di questo con più grande valore optimum tutti gli altri spazi saranno soppressi dalla sequenza. A questo punto se rimane un solo spazio nella sequenza esso sarà preso come spazio risultante. Altrimenti il risultato sarà un nuovo spazio con valori: optimum pari al valore comune, minimum pari al più grande dei valori minimi e maximum pari al più piccolo dei valori massimi. Tale spazio sarà non conditional.

Esempio



Tipi particolari di block-area

- Reference-area
- Page-area
- Line-area

Posizionamento delle block-area

Data un'area P i cui figli sono block-area, gli elementi in P si dicono propriamente posizionati se:

Per ogni block-area B figlia di P sia ha che:

I lati before e after dell'allocation-rectangle di B devono essere paralleli ai corrispondenti lati del content-rectangle di P.

I lati start e end di B sono paralleli e coincidenti con i corrispondenti lati del content-rectangle dell'area P.

Per ogni coppia di aree B e B' figlie di P, la distanza tra esse (ovviamente nella block-progression-direction) è definita applicando le regole di risoluzione alla massima sequenza di spazi tra B e B'.

Tipi particolari di inline-area

- Reference-area
- Glyph-area

Posizionamento delle inline-area

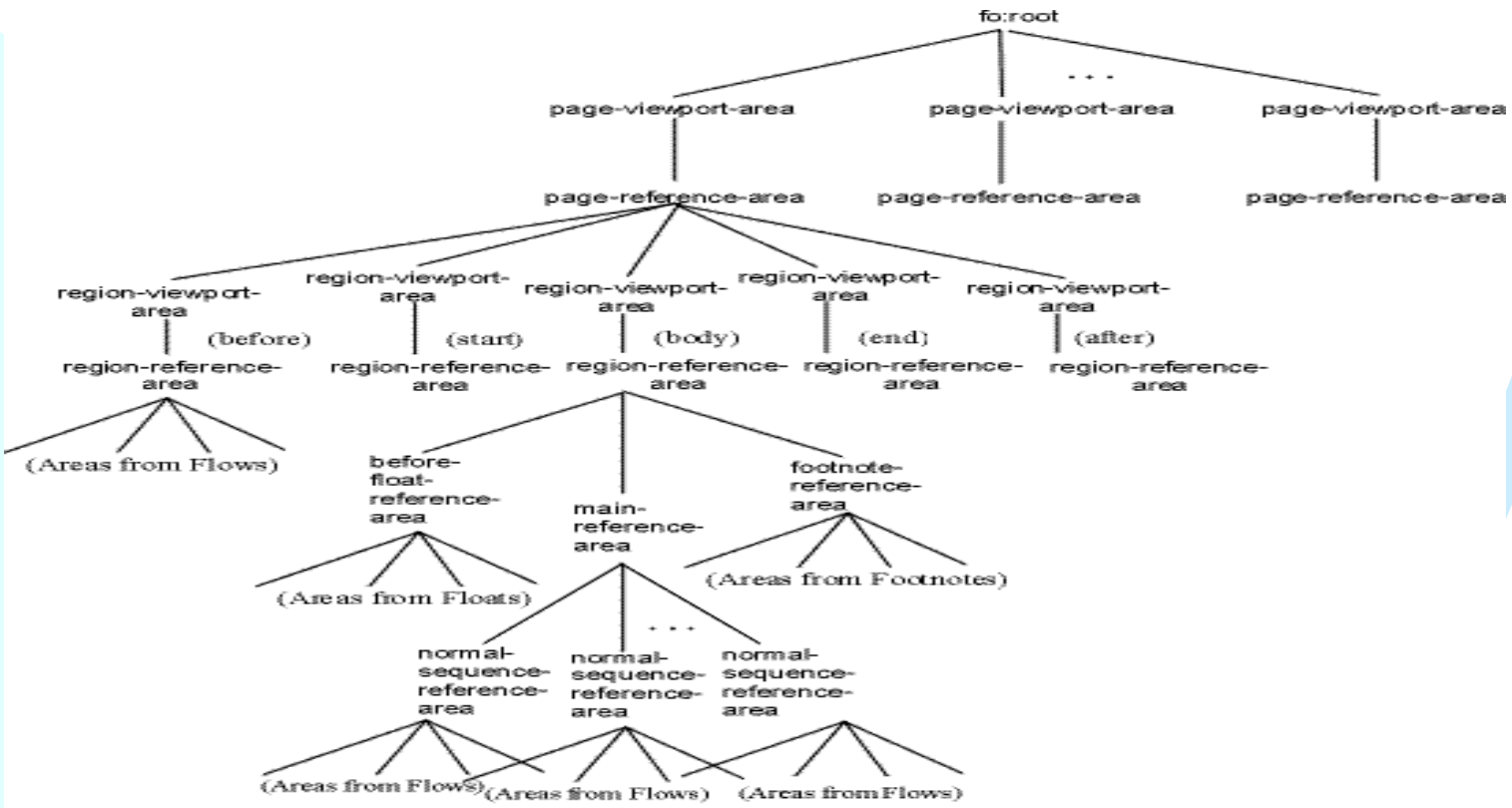
Data un'area P i cui figli sono inline-area, gli elementi in P si dicono propriamente posizionati se le seguenti condizioni sono verificate:

- Per ogni inline-area I in P, i lati start, end, after e before dell'allocation-rectangle di I sono paralleli ai corrispondenti lati del content-rectangle dell'area P.
- Per ogni coppia di aree I e I' in P, la distanza tra esse (ovviamente nella inline-progression-direction) è definita applicando le regole di risoluzione alla massima sequenza di spazi tra I e I'.

Viewport

Ci sono situazioni in cui il contenuto di un'area non viene completamente visualizzato. Questo può accadere se tale contenuto genera un overflow e l'area che lo contiene ha la proprietà overflow uguale a hidden. Un altro caso può verificarsi se il designer riduce la zona visibile di un'area attraverso la proprietà clip. Date queste situazioni si definisce viewport la zona visibile di un area, ad esempio page-viewport-area o block-viewport-area.

Esempio di albero delle aree



A Typical Area Tree

I formatting-object

Ci sono tre tipi di formatting-object: quelli che generano aree, quelli che restituiscono aree modificandone eventualmente le caratteristiche e quelli usati da altri formatting-object per generare aree. I primi due tipi sono comunemente chiamati flow-object , il terzo tipo può essere detto layout-object o auxiliary-object.

FO di impaginazione e layout

Permettono di definire sia la struttura di layout di una pagina o frame (dimensioni e posizione del body e sua eventuale suddivisione in colonne, header, footer, side-bar) sia le regole attraverso cui il contenuto XML di partenza è sistemato in questi contenitori attraverso:

- fo:simple-page-master
- fo:page-sequence

Block-level FO

- Sono generalmente usati per la formattazione di titoli, paragrafi, didascalie di immagini, tabelle o liste.
- Generano block-area

Esempio di Block-level FO

```
<xsl:template match="chapter">
   <fo:block break-before="page">
      <xsl:apply-templates/>
   </fo:block>
</xsl:template>
<xsl:template match="chapter/title">
   <fo:block text-align="center"
        space-after="8pt"
        space-before="16pt"
        space-after.precedence="3">
      <xsl:apply-templates/>
    </fo:block>
</xsl:template>
```


Inline-level FO

- Il loro utilizzo più comune è per la gestione di immagini, la formattazione di elementi all'interno di linee come ad esempio assegnare bordi colorati a caratteri o parole e la formattazione di elementi di link.

Esempio di Inline-level FO

```
<xsl:template match="p">
   <fo:block>
      <fo:initial-property-set
            font-variant="small-caps"/>
      <xsl:apply-templates/>
   </fo:block>
</xsl:template>
```


Altri FO

- fo:wrapper

Permette di attribuire un set di proprietà a tutti i suoi elementi figli.

- Utilizzo classico:

```
<fo:wrapper font-style="italic">
   Testo in stile corsivo
</fo:wrapper>
```


Proprietà dei FO

- Absolute properties
- Aural properties
- Border, padding and background properties
- Font properties
- Hyphenation properties
- Keeps and Break properties
- Margin properties block
- Margin properties inline

Proprietà dei FO (2)

- Pagination and layout properties
- Table properties
- Character properties
- Link properties
- Miscellaneous properties

XSLFO e CSS

- I FO sono nati per essere applicati ad elementi privi di semantica, i CSS per essere applicati ad elementi HTML.
- I FO hanno introdotto un complesso meccanismo di impaginazione. I CSS nascono per la visualizzazione sul web.
- I FO hanno esteso i meccanismi di internazionalizzazione.

Implementazioni esistenti

- FOP James Tauber
  - **http://xml.apache.org/fop/**
- INDelv
  - **http://www.indelv.com/**
- REXP
  - **http://www.esng.dibe.unige.it/REXP**

Reperibili anche al sito
http://www.w3.org/style/XSL

XMLCFO

- Implementazione Java dei fo block-level e inline-level basata su displet.
- Amplia le caratteristiche delle tabelle e introduce la gestione per la sovrapposizione disgiunta di aree.
- Tratta le immagini sia a livello inline che di blocco.
- Non splitta ogni carattere ma delega gli oggetti fo:block e fo:inline alla gestione del testo.