XSL qqs notions

Quelle est la syntaxe d'une règle modèle ?

Elle est la suivante :

```
<xsl:template match="motif"> <!-- Spécifie à quels noeuds la règle est
applicable -->


    <!-- Insertions dans le fichier cible de données prélevées/calculées à partir des données
    du fichier source. Appel éventuel d'autres règles modèles -->

</xsl:template>
```

Que fait une règle modèle une fois qu'elle est activée ?

Par défaut elle ne fait rien -- ce qui revient à dire que ce sur quoi elle est positionnée (et qui peut être un sous-arbre) sera absent (ne sera pas reproduit) dans le fichier cible. Ainsi, si vous avez écrit une règle modèle qui capture toutes les balises `<para>` et qui est vide, les balises `<para>` et tous leurs contenus (qui sont peut-être des sous-arbres) ne seront pas exploités/reproduits dans le fichier cible. (Si l'on accepte l'idée que celui-ci est une image, plus ou moins déformée, du fichier source, on peut dire que les balises `<para>` et tout leur contenu auront été effacés.)

Si vous souhaitez que le modèle fasse quelque chose, ne serait-ce que rendre la main à d'autres modèles, eh bien il faut le lui demander explicitement !

Vous pouvez, par exemple lui demander de simplement remplacer les balises <para> et tout ce qu'elles contiennent par le texte "trouvé !". En ce cas il vous suffira d'écrire :

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


    trouvé !

</xsl:template>
```

Si vous désirez que ce texte "trouvé !" apparaisse à l'intérieur d'une nouvelle balise, par exemple `<p>`, alors vous écrirez :

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


    <p>trouvé !</p>

</xsl:template>
```

Si vous voulez que le texte qui se trouvait précédemment à l'intérieur de la balise `<para>` apparaisse maintenant à l'intérieur de la balise `<p>`, vous écrirez :

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


    <p><xsl:value-of select="."></p>

</xsl:template>
```

Mais ceci ne va reproduire que les noeuds *textuels* descendants de la balise `<para>`. Que se

passera-t-il si la balise <para> contient des balises "descendantes" -- par exemple une balise <important> ? Eh bien, ces balises descendantes seront ignorées et seul leur contenu textuel appararaîtra dans le résultat final.

Alors, comment faire ? C'est ici qu'apparaît tout l'intérêt de l'instruction "reine" de XSLT : <xsl:apply-templates /> .

Que fait cette instruction ? Eh bien, elle permet à la règle modèle active de demander à la cantonade si, par hasard, d'autres règles modèles ne voudraient pas reprendre le travail là où elle l'a laissé... Et si il ne s'en trouve pas ? Eh bien tant pis, elle s'en désintéresse !

XSLT est basé sur la division du travail et la coopération. En XSLT la division du travail et la coopération sont un peu difficiles à mettre en place et à faire fonctionner. Mais, elles se révèlent une fois en place extrêmement efficaces et d'un fonctionnement très souple.

Prenez par exemple l'instruction que nous avons écrite ci-dessus dans notre modèle :

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

On peut la remplacer par <xsl:apply-templates /> et créer par ailleurs une règle modèle générique unique chargée de la reproduction des noeuds textuels de tout un fichier XML. Cette règle modèle générique "reproductrice de feuilles" (qui fait fort opportunément partie des règles modèle internes de XSLT) aura l'allure suivante :

```
<xsl:template match="text()">

    <xsl:value-of select=".">

</xsl:template>
```

L'avantage de cette façon de travailler est que le <xsl:apply-templates /> en question va pouvoir faire face à tous les cas possibles :

- le cas où la balise <para> ne contient que du texte
- le cas ou elle contient des balises filles -- auquel cas il faudra bien entendu que des règles modèles spécifiques aient été définies pour traiter ces balises, du genre :

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

Qqs fonctions :

Fonction XSLT	Description
xsl:for-each	Boucle (enfin pas tout à fait !)
xsl:if	Si conditionnel
xsl:choose, xsl:when et xsl:otherwise	Suite de codes conditionnels : instruction switch en C

xsl:template, xsl:call-template, xsl:param et xsl:with-param	Déclarer et appeler une fonction, avec ou sans paramètre(s).
xsl:number	Numérotation/compteur