Corrigé type partie 4 : activité « Transformez votre document XML en un document HTML »

Afin que vous puissiez noter correctement les exercices que vous devez corriger, je vous propose de revenir ensemble sur l'ensemble des critères de correction. Cela nous permettra de parcourir le corrigé type par la même occasion.

Lisibilité du document XSLT

La première chose à noter est la lisibilité du document XSLT. Par lisibilité, on parle de la "forme" du document. Typiquement, si l'on jette un coup d'œil au corrigé-type, même si on l'ouvre dans un bloc-note "primaire" sans coloration syntaxique, on arrive quand même à le déchiffrer. En effet, le contenu est espacé, clair et l'indentation permet de se rendre facilement compte de l'architecture du document. Les commentaires, facilement repérables grâce à leur syntaxe spécifique, aident d'autant plus à se repérer dans le document.

Quand vous ouvrez le document XSLT que vous devez corriger, êtes-vous en face d'un document parfaitement lisible, lisible ou peu lisible ? Est-ce que l'indentation permet de se rendre compte de l'architecture du document ? Le document est-il commenté ?

Si ce critère peut sembler subjectif, il est pourtant très important car le jour où vous travaillerez à plusieurs sur un document, il convient que chacun puisse le prendre en main le plus rapidement possible sans perdre du temps à le déchiffrer. N'oubliez jamais qu'un document bien présenté permet de facilité sa prise en main.

Le document XSLT est-il bien formé ?

Pour pouvoir fonctionner, un document XSLT doit être bien formé, c'est-à-dire qu'il ne comporte aucune erreur de syntaxe et que tous les mots clefs utilisés existent bel et bien dans la technologie.

Pour vérifier que le document XSLT que vous devez corriger est bien formé, le plus simple est encore d'utiliser un outil spécifique comme par exemple EditiX. Si vous ne savez pas comment vérifier que le document XSLT est bien formé à l'aide d'EditiX, vous pouvez consulter cette partie du cours.

Validité du document HTML

Cette partie est purement visuelle. Pour la corriger, il convient à partir du document XSLT à corriger et du document XML fourni dans l'énoncé de produire le document HTML. Pour ce faire, vous pouvez utiliser un outil comme EditiX et vous baser sur cette partie du cours.

Une fois le document HTML produit, placez au même niveau le document CSS et ouvrez le dans un navigateur web. Visuellement, le document HTML est-il conforme aux captures d'écran fournies dans l'énoncé ?

Utilisation d'un élément "for-each" dans le template "principal"

Il convient ici de vérifier que le document XSLT créé est un minimum générique et que si demain, on souhaite ajouter 1, 2 ou 3 livres au document XML à transformer, celui-ci continue de fonctionner correctement. Pour ce faire, il convient que les livres soient traités de manière automatique, peu importe le nombre contenu dans le document XML.

Cela est possible via l'utilisation d'un élément "for-each" qui va parcourir l'ensemble des livres de la bibliothèque. Dans le corrigé type, nous retrouvons bien la présence de cet élément dans le template "principal" :

```
<!-- utilisation d'un élément for-each pour boucler sur l'ensemble des livres -->

<xsl:for-each select="bibliotheque/livre">

   <!-- ... -->

</xsl:for-each>
```

L'affichage des autres titres d'un livre est conditionné

Tous les livres du document XML n'ont pas forcément plusieurs titres. Si l'on regarde les captures d'écran du document HTML de l'énoncé, on s'aperçoit alors que pour les livres qui ne possèdent pas plusieurs titres, il convient de ne pas afficher le titre "Autres titres". Cet affichage est conditionné. En effet, il convient de vérifier que le livre a bien plusieurs titres.

Il existe plusieurs possibilités pour conditionner l'affichage. Vous pouvez par exemple systématiquement appeler le template "titres" qui se chargera alors de vérifier que le livre possède bien plusieurs titres, ou vous pouvez conditionner l'appel du template. Dans ce cas là, si le template est appelé, c'est qu'il a forcément plusieurs titres. C'est cette seconde solution que j'ai choisi d'implémenter dans la correction type.

Ainsi, si l'on regarde du côté du template "book" qui se charge de structurer l'affichage d'un livre, on y découvre les lignes suivantes :

```
<!-- on conditionne l'appel au template "titres" en vérifiant que le livre à au moins un autre titre
-->

<xsl:if test="count($livre/titres) != 0">

   <xsl:call-template name="titres">

       <xsl:with-param name="titres" select="$livre/titres" />

   </xsl:call-template>

</xsl:if>
```

Comme vous pouvez le constater, nous vérifions via un test, que la balise <titres> existe pour le livre. Si elle existe, nous considérons alors qu'elle contient des balises <titre>.

Les titres et leurs langues sont bien affichés

Petit point facile à gagner ;)

Il convient ici de vérifier que pour pour chacun des titres d'un livre, toutes les informations sont bien affichées. En effet, vous deviez afficher le titre et sa langue à chaque fois. Pour ce faire, vous deviez afficher la valeur de balise ainsi que la valeur de son unique attribut. Si l'on regarde du côté du document XSLT proposé dans la correction type, nous avons bien cet affichage.

Tout d'abord au niveau du template "titreOriginal" :

```
<xsl:value-of select="$titre" /> (<xsl:value-of select="$titre/@lang" />)
```

Puis au niveau du template "titres" :

```
<xsl:value-of select="." /> (<xsl:value-of select="./@lang" />)
```

Pour rappel, " " permet de forcer l'affichage d'un espace dans le document HTML qui va être produit par la transformation XSLT.

Le template "informations" n'affiche que les informations connues

Vous deviez ici conditionner l'affichage des informations relatives à un livre dans la section "Informations". En effet, pour certains livre, le type n'est pas précisé et l'auteur n'a pas forcément un nom et un prénom. Pour conditionner l'affichage de ces informations, il convient alors de vérifier la valeur des paramètres fournis au template. Si les paramètres sont différents d'une chaîne de caractères, c'est que l'information existe.

Ainsi, si l'on se penche sur le document XSLT fourni dans la correction type, et plus précisément au niveau du template "informations", on remarque que l'affichage du nom d'un auteur est bien conditionné :

```
<!-- on conditionne l'affichage du nom de l'auteur qui n'est pas systématiquement présent -->

<xsl:if test="$auteur/nom != ''">

    <xsl:value-of select="$auteur/nom" />

</xsl:if>
```

Il en va de même pour le type :

```
<!-- on conditionne l'affichage du type du livre qui n'est pas systématiquement présent -->

<xsl:if test="$type != ''">

   <li><span class="bold">Type : </span><xsl:value-of select="$type" /></li>

</xsl:if>
```