Master Informatique
Galaxie XML
Introduction au projet de Géolocalisation "Générique"

Pierre Pompidor

But du projet

Le but (de départ) de ce projet est de permettre à un internaute de localiser sur un plan des informations géolocalisées. Je vous propose de nous concentrer cette année sur la localisation d'œuvres d'art dans un musée, et plus particulièrement du Musée Fabre de Montpellier.
Liens intéressants :

- http ://museefabre.montpellier-agglo.com/index.php/connaitre
- → cliquez sur "Les grands parcours"
- → cliquez sur "Détails section par section"
- → cliquez sur "Détail du parcours salle par salle", et certaines salles :
  - sont localisées sur le plan du musée ;
  - listent les oeuvres exposées ;
  - voire même permettent leur visite via une vue panoramique !

Cette géolocalisation sera effectuée en plusieurs étapes :

- Modélisation de vos données dans un fichier XML (et dans la grammaire associée) ;
- Création d'un formulaire HTML de recherche des informations géolocalisées par une feuille de style (maquette.xsl) codée "en dur", c'est à dire spécifiant directement les balises/attributs que vous avez choisis ;
- Création d'un formulaire HTML de recherche par une feuille de style générique (geolocalisation.xsl) à partir du document XML (et/ou de la grammaire associée au document XML). **Attention, cette feuille de style devra fonctionner quelques soient les données qui lui seront fournies** → première étape décrite dans cet énoncé
- Application d'une seconde feuille de style (recherche.xsl) paramétrée par les critères de recherche de l'utilisateur qui visualisera sur le plan les objets correspondants aux critères retenus → deuxième étape décrite ultérieurement

Attention : Le but de cette première partie du projet est de créer une interface de visualisation d'objets géographiques la plus générique possible. C'est à dire que dans les feuilles de styles finales ne devront jamais appraître le nom d'une balise ou d'un attribut du document XML (hormis les attributs *x* et *y* (et peut-être *z*), voire ci-après).
Cela signifie aussi que votre visualisateur devra fonctionner sur diverses données géographiques emboîtées...

Les données et l'analyse de leur structure

Les données :

Les données XML géographiques devront spécifier des structures géographiques enboîtées sur différents niveaux (par exemple secteurs/salles/...), et au niveau le plus bas les objets qui devront être localisés. Ces objets devront être accompagnés par **des attributs les caractérisant et qui permettront de les rechercher**, et par **leur positionnement** en x et y (voire z) :

- x représentant la position en abscisse sur le plan de l'objet
- y représentant la position en ordonnée sur le plan de l'objet

Ces informations de positionnement ne seront renseignées que plus tard par rapport à un plan réel.

→ Par rapport à un positionnement absolu, une autre idée possible est que le positionnement des objets finaux soit relatif au positionnement des structures englobantes (par exemple les produits par rapport au rayon et le rayon par rapport au secteur...).

En exemple, une feuille de style analysant un document XML : analyse_xml.xsl

Voici à titre d'exemple les modèles permettant d'afficher la structure d'un document XML (noms des balises et des attributs les accompagnant). Vous remarquerez l'utilisation intensive de la fonction **name()** qui affiche le nom du nœud contexte (c'est à dire la nom du nœud sélectionné par l'expression XPath la plus proche).

```
<xsl:template match="/">
    <html> <body>
        <xsl:apply-templates select="//*" />  <!-- sélection de tous les noeuds de l'arbre DOM -->
    </body> </html>
</xsl:template>

<xsl:template match="*">  <!-- modèle invoqué pour n'importe quel noeud de type élément -->
    balise <xsl:value-of select="name()" />
    <ul> <xsl:apply-templates select="@*" />  <!-- sélection de tous les attributs  --> </ul>
</xsl:template>

<xsl:template match="@*">  <!-- modèle invoqué pour n'importe quel noeud de type attribut -->
    <li> <xsl:value-of select="name()" /> :  <xsl:value-of select="." />  </li>
</xsl:template>
```

Création du formulaire

Dans un premier temps votre feuille de style sera une maquette spécifiant en dur les noms des balises et des attributs, dans un second temps elle sera générique. Dans tous les cas elle devra générer une liste déroulante par critère de recherche possible. Dans ces listes déroulantes, chaque valeur ne devra apparaître **qu'une seule fois**.

Modèles permettant le sélection de listes uniques par nom de balises

Une première tâche va être de pouvoir lister tous les élements de mêmes noms. Voici quelques pistes :

```
    ...
    <xsl:apply-templates select="//*" />
    ...

<xsl:template match="*[not(preceding::node()[name()=name(current())])]">
     Pour le noeud <xsl:value-of select="name()" /> :
     <ul>
          <xsl:for-each select="//*[name()=name(current())]">  <!-- for-each technique :-) -->
                <li> <xsl:value-of select="name()" /> avec <xsl:apply-templates select="@*" /> </li>
          </xsl:for-each>
     </ul>
</xsl:template>

<xsl:template match="@*">
    <xsl:value-of select="name()" />=<xsl:value-of select="." /> 
</xsl:template>
```

Rappels sur les points difficiles relatifs aux expressions XPath à utiliser :

- **not()** : négation de l'expression XPath
- (rappel **.** : nœud contexte (celui sélectionné par l'expression XPath la plus proche))
- **name()** : nom du nœud contexte
- **current()** : nœud courant (nœud ayant provoqué l'invocation du template)
- **name(current())** : nom du nœud courant
- et donc *[not(preceding : :node()[name()=name(current())])] :
  n'importe quel nœud de type élément dont le nom n'est pas celui d'un élément déjà traité !