# TP - XSLT+SVG

### Objectifs
Écrire une feuille de transformation **générique**.
Définir et utiliser des fonctions et paramètres en XSLT.
Utiliser l'application SVG (*Scalable Vector Graphics*).

### Références
W3C - syntaxe XSLT *www.w3.org/TR/xslt.html*
W3C - syntaxe SVG *www.w3.org/TR/SVG*
W3C - syntaxe XHTML *www.w3.org/TR/xhtml1*
W3C - syntaxe CSS *www.w3.org/TR/CSS*
Lire attentivement les annexes placées à la fin de ce sujet de TP (XSLT, SVG)

### Ressources
Dans le répertoire DOC/TP/SVG2 de la formation (M2).

### Environnement de développement XML
*Eclipse*, *XML Spy*, navigateurs et *plugins* SVG

### Présentation du sujet de ce TP
L'objectif final est de transformer un document XML de modèle quelconque en un graphe SVG représentant l'arbre du document. La méthode proposée consiste à procéder en trois étapes :

1. transformer la structure du document XML en un document XHTML dans lequel l'arborescence est visualisée par des indentations (question 1),
2. transformer la structure du document XML en SVG (question 2),
3. transformer la structure et le contenu du document XML en SVG (question 3).

## Exercice 1 - Production de la représentation XHTML arborescente

### Objectif
Transformer n'importe quel document XML en un document XHTML dans lequel l'arborescence est visualisée par des indentations.

### Exemple
Voici un document XML :

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<systeme_solaire>
<etoile>
        <nom>Soleil</nom>
        <type_spectral>G2</type_spectral>
        <age unit="milliard d'annees">5</age>
</etoile>
<planete type="tellurique">
        <nom>Mercure</nom>
        <distance unit="UA">0.4</distance>
        <masse unit="masse terrestre">0.06</masse>
        <diametre unit="diamètre terrestre">0.4</diametre>
</planete>
</systeme_solaire>
```

Voici la représentation du document XHTML obtenue sur un navigateur à l'issue de cette première transformation :

```
systeme_solaire
...etoile
......nom
......type_spectral
......age
...planete
......nom
......distance
......masse
......diametre
```

### Question 1

Écrire une feuille de style XSLT qui transforme n'importe quel document XML en un document XHTML où l'arborescence du document XML est visualisée à raison d'une ligne par nœud de l'arbre avec une indentation proportionnelle à la profondeur du noeud (chaque nom de nœud est dans un élément <p> et est précédé par un nombre de points proportionnel à la profondeur du nœud dans l'abre XML d'origine).

CONSEIL : pour calculer la profondeur d'un nœud utiliser une récursivité ayant en paramètre la profondeur courante.

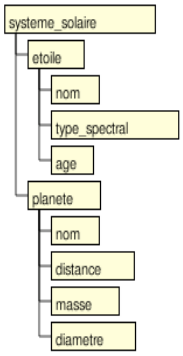
## Exercice 2 - Production de la représentation graphique SVG

### Objectif

Transformer la structure d'un document XML en son graphe SVG.

### Exemple

Pour le document XML ci-dessus, le graphe SVG résultat obtenu sera le suivant :




Les deux fichiers suivants sont fournis :

1. un squelette de feuille de style dans le fichier **squelettexml2svg.xslt**
2. une fonction `string-width(s)` dans le fichier **string-width.xslt**

   Cette fonction, qui est importée dans le squelette, prend une chaîne *s* en paramètre d'entrée et rend en sortie une valeur décimale correspondant à la longueur, en nombre de pixels, de la chaîne *s* représentée avec la police ArialMT de taille 12. Cette fonction vous permet de calculer la longueur des encadrements rectangulaires.

### Question 2

En vous inspirant de la première feuille de style de la question 1 et en utilisant les ressources fournies, écrire une feuille de style qui produit un graphe SVG représentant l'arborescence d'un document XML (les contenus textuels du document sont ignorés).

## Exercice 3 - Représentation des contenus

### Objectif

Transformer la structure et le contenu d'un arbre XML en un graphe SVG.

### Question 3

Même question qu'en 2. mais en visualisant les contenus textuels des élements XML comme des feuilles de l'arbre.

**Tous les résultats doivent évidemment être validés.**

---

## *ANNEXE : INFORMATIONS COMPLÉMENTAIRES*

### *A - Rappel sur le passage de paramètre (résursif)*

- Dans l'exemple ci-dessous, le paramètre entier N est incrémenté récursivement de 1 à chaque appel de la fonction :

```
<xsl:apply-templates>
   <xsl:with-param name="N" select="$N +1">
</xsl:apply-templates>
```

### B - Tracé d'un arbre de boîtes en SVG (*Scalable Vector Graphics*)

Le point origine des tracés de graphiques en SVG est en haut à gauche. Tout point est repéré par rapport à cette origine avec des coordonnées <x,y> telles que, x est interprété en positif vers le droite et y est interprété en positif vers le bas.

Chaque nœud de l'arbre sera représenté par quatre composants graphiques :

1. un cadre,
2. un texte,
3. un trait vertical pour la poignée,
4. un trait horizontal pour relier les poignées.

Chaque composant graphique est représenté par un élément SVG comme suit :

```
<!-- Cadre rectangulaire -->
<rect x="…" y="…" width="…" height="…"/>

<!-- Texte -->
<text x="…" y="…" style="…">SimpleType</text>

<!-- Trait vertical d'origine <$x,$y> de $n pixels de long vers le haut
-->
<path style="…" d="M$x,$y v-$n"/>

<!-- Trait horizontal d'origine <$x,$y> de $n pixels de long vers la
gauche -->
<path style="…" d="M$x,$y h-$n"/>
```

L'origine <x,y> d'un élément <rect> est son coin en haut à gauche. L'origine <x,y> d'un texte <text> est sur la ligne de base d'écriture, à gauche du premier caractère (les lettres comme g et p dépassent sous cette ligne). Les noms des autres attributs de <rect> et <text> parlent d'eux mêmes. L'attribut *d* de l'élément <path> est un chemin décomposé en deux parties :

- "M$x,$y" précise l'origine du chemin,
- "h-$n" décrit un trait horizontal vers la gauche
- "v-$n" décrit un trait vertical vers le haut.

Le squelette qui est fourni comporte un certain nombre de variables que vous devrez utiliser pour effectuer une transformation correcte. En particulier, les variables *graph.style*, *handle.path.style* et *text.style*, fournissent directement les valeurs à introduire dans les attributs style des éléments respectifs <g>, <path> et <text>.

Le contenu du fichier squelette fourni est le suivant :

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL
/Transform">
 <xsl:output method="xml" version="1.0" encoding="ISO-8859-15"
indent="yes"/>
 <xsl:param name="text.style" select="fill:#000000;stroke:none;"/>
 <xsl:param name="handle.path.style" select="stroke-width:0.5;stroke-
miterlimit:3;fill:none;"/>
 <xsl:param name="graph.style">
fill-rule:nonzero;clip-rule:nonzero;fill:#FFFFD0;
```

```
stroke:#000000;stroke-miterlimit:4; font-family:'ArialMT'; font-size:12;
 </xsl:param>
 <xsl:param name="handles" select="true"/>
 <xsl:param name="root.handle" select="false"/>
 <xsl:param name="boxes" select="true"/>
 <xsl:param name="letterHeight" select="8"/>
 <xsl:param name="letterWidth" select="8"/>
 <xsl:param name="x0" select="20"/>
 <xsl:param name="y0" select="20"/>
 <xsl:param name="rowHeight" select="20"/>
 <xsl:param name="columnWidth" select="20"/>
 <xsl:param name="boxHeight" select="16"/>
 <xsl:include href="string-width.xslt"/>
 <!-- Pour rejeter tous les textes -->
 <xsl:template match="text()"/>

 <!-- contenu à compléter en TP ... -->

</xsl:stylesheet>
```

Les paramètres booléens *handles*, *root.handle* et *boxes* de ce squelette sont destinés à être utilisés pour paramétrer la présence ou l'absence respective, des poignées, de la poignée de la racine et du cadre rectangulaire autour des noms d'élément dans le graphe résultat.

Les paramètres *x0* et *y0* sont les coordonnées de la position initiale, en nombre de pixels, de la représentation du premier élément (c'est à dire l'élément racine du document).

Le paramètre *columnWidth* est la valeur du décalage horizontal, en nombre de pixel, entre le début d'un nœud et celui de ses fils.

Les autres paramètres parlent d'eux mêmes.

---