TP DOCBOOK – XSLT et transformation SVG

Wassim CHEGHAM
Gurval LE BOUTER
Master2 MITIC - ISTIC

Table of Contents

1. Compte Rendus du TP SVG

1.1. Objectifs

Le but du TP est de nous familiariser au modèle SVG (Scalable Vector Graphics), l’écriture de feuilles de transformation générique grâce notamment à l'utilisation des fonctions avec paramètres en XSLT.

1.2. Environnement de développement XML

Pour réaliser ce TP nous avions utilisé l’IDE Eclipse et son mode WTP (Web Tools Platform), ainsi que les navigateur Firefox/Chrome pour la visualisation des documents produits. Nous avions également eu recours aux consoles de débogages de Google chrome et au plugin Firebug de Firefox, pour mieux inspecter les modèles SVG générés.

1.3. Utilisation d’un document externe pour la transformation

Dans cette partie, nous avions écris une feuille de style utilisant deux fichiers XML en entrée: un premier fichier XML comme source de données, et un second comme source de style. Dans un premier temps, nous devions lire le fichier de style dans une variable `$style`, comme ceci:

```
...
<xsl:variable name="style" select="document('../style.xml')" />
...
```

Ensuite, nous avions écris une règle XSL pour parcourir tous les mois du fichier de données, et ainsi généré un tag `<rect />` permettant d'avoir une forme SVG rectangulaire représentant les barres de l'histogramme :

```
...
<xsl:for-each-group select="annee/mois" group-by="@numero">
 <xsl:variable name="y" select="100 - number(./text())"/>
 <rect x="{@numero * 10}" y="{ $y }" width="10" height="{./text()}"
  fill="{$style/style/color[ number(current-group()/@numero) ][position()]}"
  stroke="black" />
</xsl:for-each-group>
...
```

Et afin de donner une couleur à chaque barre, il nous a suffit d'écrire une requête XPATH afin de récupérer le code de la couleur renseigné dans les tags `<color />` du fchier style.xml.

Et voici donc le résultat produit:

Histogramme 4

1.4. Histogramme SVG commenté

Pour produire un histogramme SVG commenté, il a suffit d'ajouter un tag `<text />` au dessus de chaque barre indiquant le pourcentage de chacune d'elle. Et voici le code correspondant :

```
...
<xsl:for-each-group select="annee/mois" group-by="@numero">
 <xsl:variable name="y" select="100 - number(./text())"/>
 <rect x="{@numero * 10}" y="{ $y }" width="10" height="{./text()}"
  fill="{$style/style/color[ number(current-group()/@numero) ][position()]}"
  stroke="black" />

<text x="{ (@numero * 10)}" y="{$y - 5}" font-size="5">
  <xsl:value-of select="./text()"/>%

</text>
</xsl:for-each-group>
...
```

Ce qui nous donne cet histogramme :

Histogramme 5

1.5. Référence à un document SVG

Afin de produire un document XHTML intégrant une référence au fichier annee.svg, une simple balise `<img />` suffit :

```
...
<img src="../Question5/annee.svg" alt="histogramme par mois"
 height="80%" width="80%" />
...
```

1.6. Document XHTML composite

Nous avions également écrit une feuille de transformation afin de produire un fichier XHTML contenant à la fois les résultats en chiffre sous forme de liste et l'histogramme en SVG. L'écriture de cette feuille XSL n'était pas si compliquée en soit puisque cela consistait simplement à refaire ce que nous avions réalisé lors des questions précédentes. De plus, la majorité des navigateurs Internet interprètent nativement les graphiques SVG ce qui permet donc d'intégrer ces derniers au sein du document XHTML. Cela permet, par exemple, au navigateur d'éviter de refaire une seconde requête HTTP pour charger ces graphiques, dans le cas où ces derniers étaient référencés dans la balise `<img />`.

1.7. Production de résultats statistiques

Lors de cette partie, nous devions écrire une feuille de style permettant de produire un document XHTML présentant le nombre de films par décade. Nous avions définis deux groupes SVG: l'un contenant les barres de l'histogramme et l'autre contenant l'axe des Y. Ce qui nous a posé des difficultés pour réaliser cette tâche, n'était pas l'écriture des règles de transformations mais plutôt de trouver les bonnes formules mathématiques qui nous permettaient d'arriver au résultat attendue.

Histogramme 8

2. TP Docbook

2.1. Logiciels utilisés

Pour l'édition des fichiers XML (fichiers ANT et Docbook) nous avons utilisé Eclipse. Pour les transformation XSLT, nous avons utilisé Saxon. Enfin, pour générer les fichiers XML-FO, nous avons utilisé la librairie Fop (Formatting Objects Processor) dans sa version 1.0.

2.2. Fichiers Docbook

2.2.1. Version

La version de Docbook utilisée est la 4.5, ce que l'on peut constater grâce à sa DTD:

```
<!DOCTYPE article PUBLIC "-//OASIS//DTD DocBook XML V4.5//EN"
 "../docbook-4.5-dtd/docbookx.dtd">
```

2.2.2. Répertoires

Les fichiers Docbook se trouvent dans le répertoire input. Les fichiers générés se trouvent dans le répertoire output, dans lequel on retrouvent deux sous-répertoires: un pour les sorties HTML et un deuxième pour les sorties PDF.

2.2.3. Type de document

Docbook permet d'écrire deux types de document: “book” et “article”, selon la balise utilisée. Les compte rendus seront des articles, ceux ci étant plus adaptés et plus concis pour la rédaction de documents technique. La principale différence avec le type “book” est que celui-ci peut contenir une balise “chapter” et également une balise “article”. Un article, lui ne peut pas contenir de “book”.

2.2.4. Balises utilisées

`<sect1>` : permet de structurer un document, on peut y retrouver:

- `<title>`: le titre de la section
- `<para>`: pour créer un paragraphe
- `<link>`: pour créer un lien

Pour créer un lien vers une ancre dans un document, on procède comme ceci:

```
<link linkend="image-id">figure 1</link>
```

Il ne nous reste plus qu'à insérer l'image vers laquelle pointe le lien:

```
<figure id="image-id">
   <title><link linkend="image-id">Texte ici</link></title>
   <mediaobject>
    <imageobject>
     <imagedata fileref="image.jpg" format="JPG" role="pdf" />
    </imageobject>
    <imageobject>
     <imagedata fileref="image.jpg" format="JPG" role="html"/>
    </imageobject>
   </mediaobject>
</figure>
```

Notons que l'on peut différencier l'image en fonction du document généré via l'attribut "role" de "imagedata".

2.3. Script ANT

2.3.1. Impression

C'est la création du script ANT qui a été la partie la plus compliquée et qui a pris le plus de temps dans ce TP. Le script ANT "build.xml" se situe à la racine du répertoire Docbook.

2.3.2. Utilisation

Le script ANT possède plusieurs cibles, dont deux qui nous intéressent:

- build-html: génère les fichiers *.html à partir des fichiers Docbook
- build-pdf-xalan: génère les fichiers *.fo pour ensuite les transformer en *.pdf

2.3.3. Syntaxe

Définir une variable:

```
<property name="ma_variable" value="ma_valeur" />
```

Avec name, le nom de la variable, et value, sa valeur. Pour utiliser une variable dans un script, on l'appelle comme ceci: `${ma_variable}`

Définir une cible:

```
<target name="ma_cible" depends="dependance"
```

```
 description="La description de la cible">
```

Avec name, le nom de la cible. C'est cette valeur là qu'il faudra utiliser pour exécuter le script ant avec cette cible:

```
> ant ma_cible
```

L'attribut depends permet d'exécuter d'autres cibles avant celle-ci.

Dans une cible, on peut exécuter plusieurs actions, dont les plus simples sont:

créer le répertoire définit par la variable "mon_repertoire":

```
<mkdir dir="${mon_repertoire}" />
```

supprime "mon_repertoire":

```
<delete dir="${mon_repertoire}" />
```

copie le répertoire "a_copier" vers "destination":

```
<copy toDir="${destination}/assets" >
 <fileset dir="${a_copier}/assets" />
</copy>
```

2.3.4. Pourquoi utiliser XML-FO pour générer des PDF?

Il n'est pas possible de générer des fichiers PDF directement à partir des fichiers Docbook. Il faut soit générer des fichiers LaTeX, soit des fichiers XML-FO, pour ensuite les convertir en PDF. L'avantage d'utiliser XML-FO au lieu de LaTex, est qu'il s'agit d'un fichier XML. La génération se fait grâce à une transformation XSL. Pour générer le fichier PDF à partir du XML-FO, on utilise la librairie Fop, qui est développée en Java. Pour résumer, utiliser XML-FO permet d'utiliser des technologies que l'on connait bien et surtout en rapport avec la matière DOC.

A. Images

Figure A.1. Histogramme Question 4

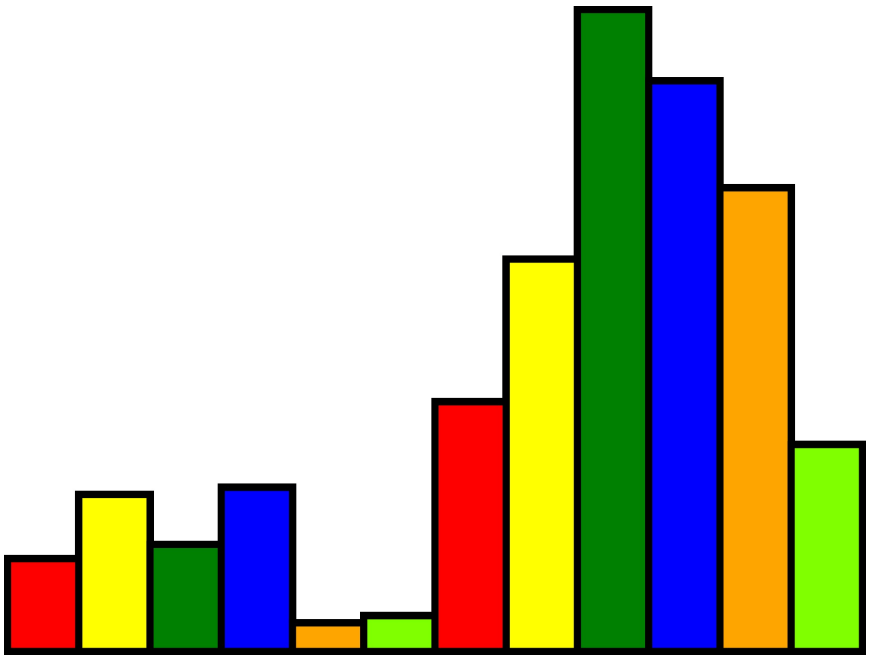


Figure A.2. Histogramme Question 5

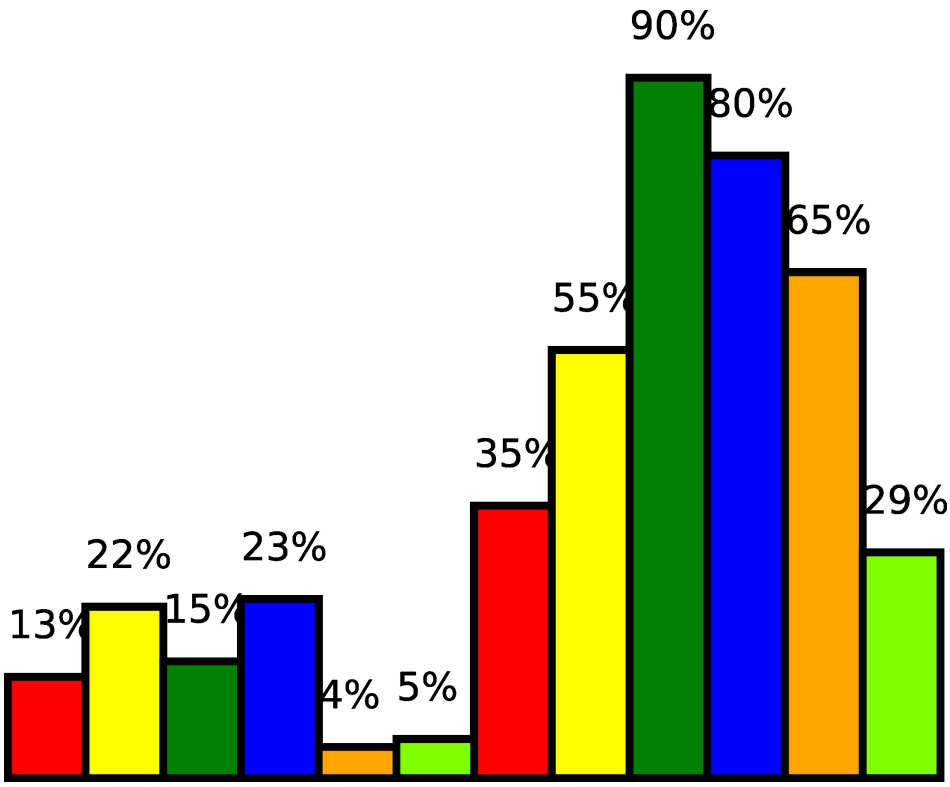
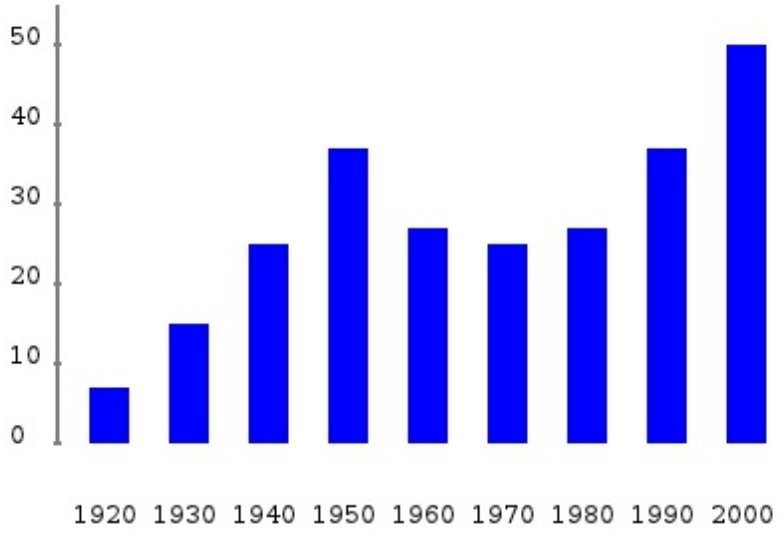


Figure A.3. Histogramme Question 8



Bibliographie

www.w3school.com [http://www.w3schools.com/svg/default.asp] . *SVG tutorial*.

www.w3.org [http://www.w3.org/TR/SVG/struct.html] . *Document structure*.

www.docbook.org [http://www.docbook.org/] . *Site officiel de Docbook*.

www.docbook.org [http://www.docbook.org/tdg/en/html/docbook.html] . *Documentation de Docbook*.