

Berthet Laurent

Pehme Taavi

Compte Rendu du TP de XML

Master TIC ISC

Avril 2007

Sommaire :

Sommaire :	2
Introduction	3
Accès aux services web d'Amazon	4
Récupération des résultats	5
Création des requêtes	5
Analyse du résultat	5
Stockage dans un fichier XML.....	7
Création d'un fichier XSL.....	10
Démonstration de l'application	13
Conclusion.....	15

Introduction

Le but de ce TP était d'accéder aux services web d'Amazon et d'afficher les résultats à l'aide de descripteur de cartes comme SVG ou X3D. Pour réaliser ce sujet, nous avons décidé cette application dans le langage Java et de cartographier nos résultats à l'aide SVG (Scalable Vector Graphics).

SVG est le standard de **dessin vectoriel en 2D** et d'applications graphiques. Il s'agit d'un langage défini par le **W3C** et respectant la norme **XML**.

Dans un premier temps, nous pensions utiliser le langage PHP pour réaliser cette application, mais on a rencontré des problèmes pour accéder aux services web d'Amazon donc nous avons décidé d'utiliser le langage Java pour de nombreuses raisons : nous pensons maîtriser suffisamment ce langage pour créer une application. De plus, il est très facile de parser un document XML à l'aide de librairie (comme JDom).

Accès aux services web d'Amazon

Pour accéder aux services web d'Amazon, nous avons utilisé la librairie Apache Axis v 1.1 qui est une implémentation de SOAP ("Simple Object Access Protocol"). Cette librairie nous a permis de générer toutes les classes et donc les méthodes pour accéder à Amazon. Pour ce faire il suffit de connaître l'adresse du WSDL (Web Services Description Language) d'Amazon

Voici le code pour générer les classes :

```
import org.apache.axis.wsdl.WSDL2Java;

public class Init {
    private static String AMAZON=
    "http://soap.amazon.com/schemas3/AmazonWebServices.wsdl";

    public static void main(String[] args){
        String[] arg={"-v","-a","-o","classes",AMAZON};
        WSDL2Java.main(arg);
    }
}
```

Après l'exécution de cette méthode. On trouve dans le dossier « classes » toutes les classes qui viennent d'être créés.

Pour pouvoir faire des requêtes auprès d'Amazon, il faut s'enregistrer sur leur site car ils donnent un numéro que l'on appellera devTag.

Récupération des résultats

Création des requêtes

Pour avoir un résultat de la part du service web, il faut faire une requête. Pour réaliser une requête il faut 4 variables :

```
private final String devTag = "DIBLHQV6QXF19";  
private AmazonSearchService amazonSearchService = new AmazonSearchServiceLocator();  
private AmazonSearchPort remote = amazonSearchService.getAmazonSearchPort();  
private ProductInfo pInfo;
```

pInfo contiendra le résultat de la requête.

remote permet d'effectuer la recherche sur un artiste, un auteur, un acteur. On peut même faire une recherche avec la référence Asin d'un produit. Il y a de nombreuses possibilités.

Prenons par exemple la recherche d'un groupe de musique.

La méthode artistSearchRequest prend comme paramètre un ArtistRequest donc on en crée un avec comme paramètres, le nom de l'artiste que l'on cherche, ensuite la page que l'on veut, le mode, le tag, le type, le devTag, (sort, variations, locale). Une requête Amazon renvoie dix résultats à la fois. En changeant la page, on peut avoir tous les résultats de l'artiste recherché.

```
pInfo = remote.artistSearchRequest(new ArtistRequest("Metallica", "1", "music",  
"webservices-20", "lite", devTag, "2.0", null, null));
```

Analyse du résultat

Notre variable pInfo contient le résultat de notre requête. Grâce à la méthode getTotalPages() de pInfo on peut connaître le nombre de pages de notre requête. Si nous

voulons toutes les pages, il faut recommencer la requête avec une page différente et stocker le résultat dans une autre variable. Pour connaître le contenu de notre requête, il faut aller voir dans pInfo qui contient un tableau de Details. Un Detail est un resultat, donc pour connaître tous les renseignements de ce résultat, il faut accéder aux méthodes de Details. Il existe une multitude de méthodes, nous en utiliserons qu'une partie.

Après de nombreux essais, nous nous sommes rendu compte que certaines valeurs de Details étaient toujours vide, donc pour la suite, nous le les avons pas utilisées.

Stockage dans un fichier XML

Pour pouvoir afficher nos résultats, il faut les stocker dans un fichier adéquat pour qu'ensuite on puisse lui appliquer une transformation. C'est pour cela qu'on va le stocker dans un fichier XML. Ce fichier devra contenir toutes les informations nécessaires pour l'affichage. Pour réaliser cette opération, nous avons utilisé la librairie JDOM pour parser les informations.

Avant de faire notre fichier XML, nous faisons une DTD (Document Type Definition) qui va définir notre XML.

Voici la DTD :

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT Discographie (nom,suivant, precedent,CD+)>
<!ELEMENT nom (#PCDATA)>
<!ELEMENT suivant (#PCDATA) >
<!ELEMENT precedent (#PCDATA) >
<!ELEMENT CD (url, position, status,listePrice
,urlComplete,Asin,ProductName,catalog,avab, releaseDate, manufacturer,distributor,
usedPrice)>
<!ELEMENT Asin (#PCDATA) >
<!ELEMENT url (#PCDATA) >
<!ELEMENT position (#PCDATA) >
<!ELEMENT avab (#PCDATA) >
<!ELEMENT ProductName (#PCDATA) >
<!ELEMENT releaseDate (#PCDATA) >
<!ELEMENT urlComplete (#PCDATA) >
<!ELEMENT manufacturer (#PCDATA) >
<!ELEMENT usedPrice (#PCDATA) >
<!ELEMENT listePrice (#PCDATA) >
<!ELEMENT status (#PCDATA) >
<!ELEMENT catalog (#PCDATA) >
<!ELEMENT distributor (#PCDATA) >
```

La racine du fichier XML sera « Discographie » qui contiendra trois éléments : l'élément « nom » est le nom de l'artiste que l'on cherche. L'élément « suivant » correspond à la page suivante des résultats. L'élément CD lui contient toutes les informations que l'on aura récupéré grâce au service web. L'élément « position » contient un nombre qui permettra l'affichage des informations.

Voici la classe qui me permet d'ajouter des informations dans mon XML

```
public class ConvertToXml {

public ConvertToXml(){
    type = new DocType("Discographie", "test.dtd");
    racine = new Element("Discographie");
    document = new Document(racine,type);

    position = 0;
}

public static void ajouteCd(){
    CD =new Element("CD");

    racine.addContent(CD);

}

public static void ajouteNom(String n){
    nom = new Element("nom");
    nom.setText(n);
    racine.addContent(nom);
}

public static void ajouteSuivant(String n){
    suivant = new Element("suivant");
    suivant.setText(n);
    racine.addContent(suivant);
}

public static void ajouteElts(int i,Details[] p){

    Element url = new Element("url");
    url.setText(p[i].getImageUrlLarge());
    CD.addContent(url);

    Element avallibility= new Element("avab");
    avallibility.setText(p[i].getAvailability());
    CD.addContent(avallibility);

    ...

}
```

La methode ajouteElts() ajoute toutes les inforamtions d'un détails dans le XML. La méthode ajouteSuivant() met dans la balise « suivant » l'Url du fichier de la page suivante. Les deux autres méthodes ajoutent la balise CD et nom.

Voici un extrait du fichier XML qui est crée :

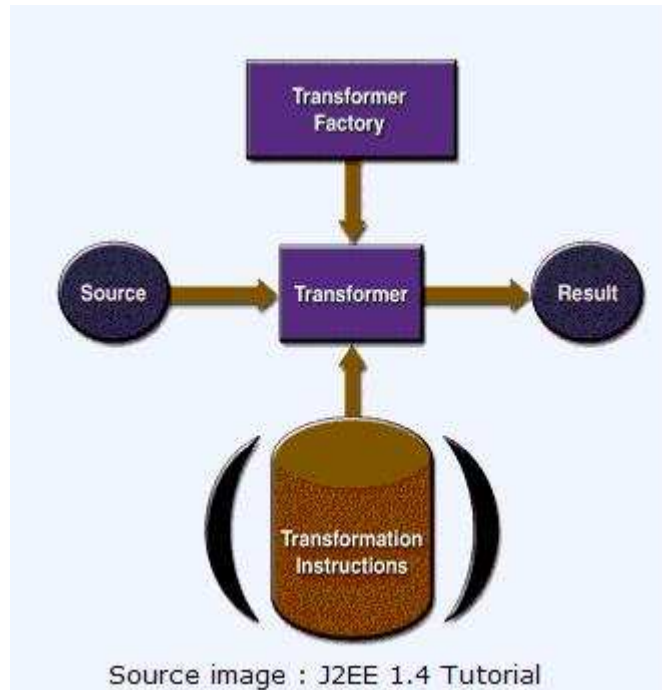
```
<Discographie>
  <nom>madonna</nom>
  <suivant>file:///D:/Mes Cours/workspace/t/madonna3.svg</suivant>
  <CD>
    <url>http://images.amazon.com/images/P/B0000071SI.01.LZZZZZZZ.jpg</url>
    <position>0</position>
    <status />
    <listePrice>$11.99</listePrice>
    <urlComplete>http://www.amazon.com/gp/product/B0000071SI%3ftag=webservices-20%26link_code=sp1%26camp=2025%26dev-t=DIBLHQV6QXF19</urlComplete>
    <Asin>B0000071SI</Asin>
    <ProductName>Into Groove/Who's/Causing Comm</ProductName>
    <catalog>Music</catalog>
    <avab>This item is currently not available.</avab>
    <releaseDate>01 January, 1992</releaseDate>
    <manufacturer>Wb</manufacturer>
    <distributor />
    <usedPrice>$24.44</usedPrice>
  </CD>
```

On retrouve la racine Discographie qui contient un nom, un suivant un precedent un ou plusieurs CD. Dans le cd on retrouve bien tous les éléments que l'on a défini dans la dtd

Création d'un fichier XSL

Pour faire l'affichage de nos résultats à partir du XML, nous devons faire un XSL (eXtensible Stylesheet Language)

Voici une illustration du mécanisme de transformation du XML :



On va voir comment créer un XSL pour qu'il puisse générer un fichier SVG :

Il faut penser que le point d'origine est le coin supérieur gauche.

L'ordre de l'ajout d'élément est important car si on pose deux éléments au même endroits le deuxième élément va écraser le premier.

Pour chaque élément que l'on veut insérer on crée un élément avec son nom.

Cet élément va afficher un texte, les attributs de cet élément indiquent la position du texte, sa taille et son contenu.

```
<xsl:element name="text">  
  <xsl:attribute name="x"> 100</xsl:attribute>  
  <xsl:attribute name="y">100</xsl:attribute>  
  <xsl:attribute name="transform">  
    translate(0,0)  
  </xsl:attribute>
```

```

    <xsl:attribute name="font-size"><xsl:value-of select="48"/></xsl:attribute>
    <xsl:value-of select="nom/text()"/>
</xsl:element>

```

Resultat après transformation :

```
<text x=" 100" y="100" transform=" translate(0,0)" font-size="48">metallica</text>
```

Voici le code pour insérer un rectangle :

```

<xsl:element name="rect">
  <xsl:attribute name="transform">
    translate(0,155)
  </xsl:attribute>
  <xsl:attribute name="x">100 </xsl:attribute>
  <xsl:attribute name="y"> <xsl:value-of
select="position/text()"/>+400</xsl:attribute>
  <xsl:attribute name="width">200 </xsl:attribute>
  <xsl:attribute name="height">90</xsl:attribute>
  <xsl:attribute name="style">fill:red </xsl:attribute>

  <animate attributeType="CSS" attributeName="opacity"
    from="1" to="0" dur="5s" repeatCount="4" />

</xsl:element>

```

L'attribut « animate » permet d'animer l'élément c'est-à-dire de le faire disparaître petit à petit un certain temps, ou de le faire se déplacer d'un point à un autre. Cet attribut peut être mis dans n'importe quel élément. Il existe de nombreuses animations.

Cet élément va afficher une image dont on récupère l'url dans le fichier XML. Pour pouvoir mettre un lien il faut mettre un namespace une adresse pour indiquer à l'élément que l'on va utiliser un attribut avec une adresse.

```

<xsl:element name="image" namespace="http://www.w3.org/2000/svg">
  <xsl:attribute name="transform">
    translate(0, 155)
  </xsl:attribute>
  <xsl:attribute name="x">100</xsl:attribute>
  <xsl:attribute name="y">
    <xsl:value-of select="position/text()"/>
  </xsl:attribute>
  <xsl:attribute name="width">200</xsl:attribute>

```

```
<xsl:attribute name="height">200</xsl:attribute>
<xsl:attribute name="xlink:href" ><xsl:value-of
select="url/text()"/></xsl:attribute>
</xsl:element>
```

Voici l'élément qui va insérer une ligne

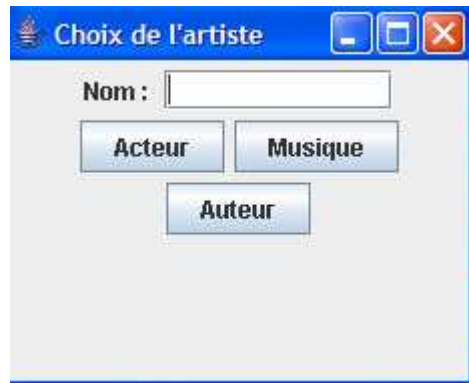
```
<xsl:element name="line">
  <xsl:attribute name="transform">
    translate(0,155)
  </xsl:attribute>
  <xsl:attribute name="x1">100</xsl:attribute>
  <xsl:attribute name="y1"><xsl:value-of select="position/text()"/></xsl:attribute>
  <xsl:attribute name="x2">800</xsl:attribute>
  <xsl:attribute name="y2"><xsl:value-of select="position/text()"/></xsl:attribute>
  <xsl:attribute name="style">fill:white;stroke:black;stroke-width:2</xsl:attribute>
</xsl:element>
```

L'attribut stroke-width indique l'épaisseur du trait.

C'est élément crée un lien

```
<xsl:element name="a" namespace="http://www.w3.org/2000/svg">
  <xsl:attribute name="xlink:href" ></xsl:attribute>
</xsl:element>
```

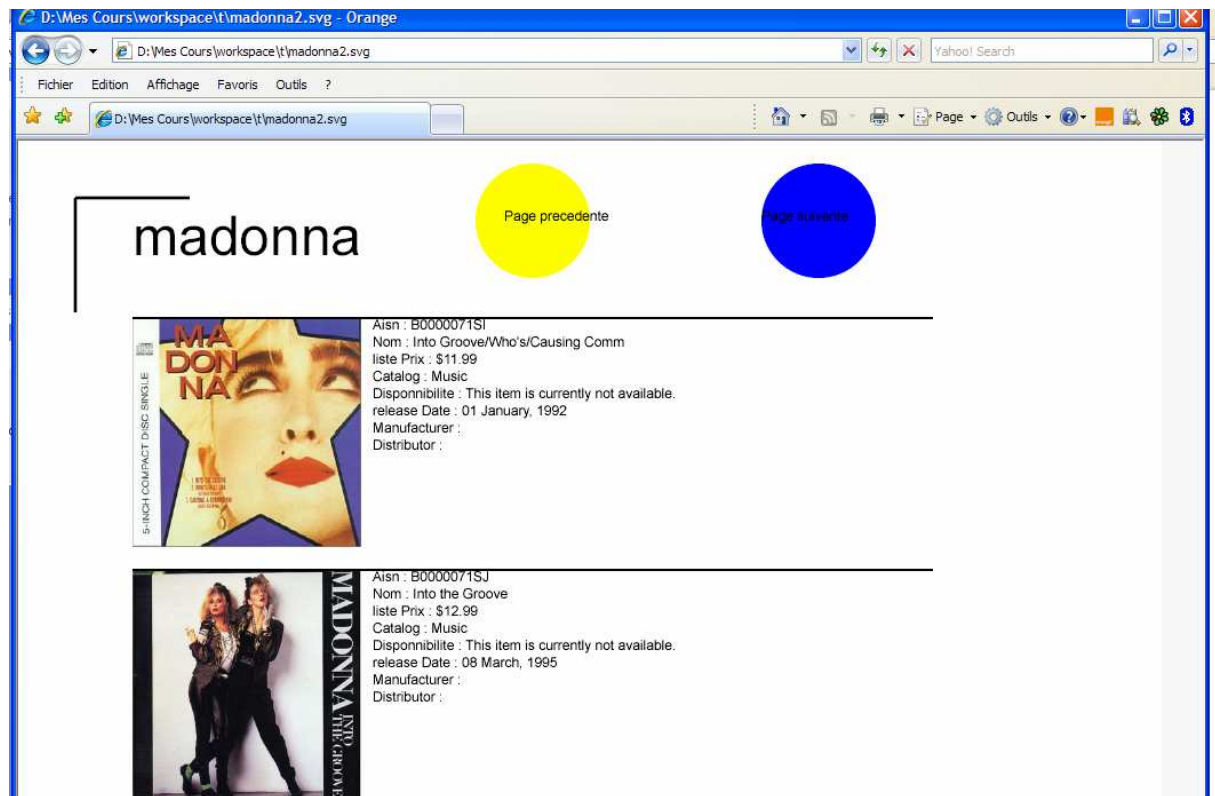
Démonstration de l'application



Au lancement de l'application se lance pour que l'on entre notre recherche.



Après avoir saisi sa recherche, on clique sur le type de recherche et on attend qu'une nouvelle fenêtre s'ouvre avec les résultats. Il se peut que ça prenne un peu de temps car il y a des recherches qui donnent de nombreux résultats.



Le résultat de la recherche s'ouvre dans le navigateur par défaut du système d'exploitation

Conclusion

Nous avons rencontré quelques problèmes liées à l'accès du service web d'Amazon. Après avoir réussi à récupérer des résultats, il a fallu comprendre comment on écrit un fichier XSL pour générer le SVG.

La grosse difficulté que l'on a rencontré pour écrire le XSL a été la gestion des coordonnées pour insérer les éléments dans notre page. Ensuite, il a fallu trouver la méthode pour insérer des liens (namespace avec l'adresse du xlink). Nous avons trouvé dommage qu'Amazon ne fournisse pas les titres des chansons lors d'une recherche pour un groupe de musique.

Ce TP nous a permis de nous rendre compte de toute la puissance de SVG et de l'importance des services web.