Gestion de l'Information sur l'Internet
Exercices

Philippe Rigaux

22 septembre 2003

La plupart des exemples sur lesquels s'appuient ces exercices sont sur le site :

http://www.lri.fr/~rigaux/GII

Vous pouvez également trouver d'autres documents et exemples sur le site :

http://cortes.cnam.fr:8080/XSLT

1 Environnement

L'environnement est constitué

1. du parseur XML *Xerces* qui contient les API DOM et SAX ;
2. du processeur XSLT *Xalan* pour appliquer des transformations aux documents XML ;
3. de la servlet XSQL d'ORACLE qui permet de publier dynamiquement des documents à partir de fichiers XML, d'une base de données ORACLE et de programmes XSLT ;

Pour consulter les documents : vous avez *Mozilla* pour HTML/XML, *acrobat reader* pour le PDF et le navigateur Deckit pour WML.

Xerces et Xalan sont des librairies Java qui permettent d'effectuer de la programmation et des transformations par des commandes en ligne. Vous pouvez les trouver à l'adresse suivante :

1. *http://xml.apache.org/dist/xerces-j/*
2. *http://xml.apache.org/dist/xalan-j/*

XSQL vous servira à créer votre site et est déjà installé. La première chose à faire est d'initialiser votre environnement et de vérifier que ça marche en faisant tourner les exemples fournis sur le site. Cela vous donne également un point de départ pour vos propres développements.

1.1 Les librairies Java

Xerces et Xalan doivent être référencés dans votre variable d'environnement `CLASSPATH`. Initialement, cette variable doit au moins contenir la référence au répertoire courant, ce qui s'obtient avec la commande :

```
setenv CLASSPATH .
```

Pour avoir accès aux parseurs, ajoutez les commandes suivantes dans votre fichier *.cshrc* (vérifiez les versions qui peuvent avoir changé, et remplacez INSTALL par le répertoire d'installation) :

```
setenv CLASSPATH INSTALL/xalan-j_2_4_0/bin/xalan.jar:$CLASSPATH
setenv CLASSPATH INSTALL/xerces-1_4_4/bin/xerces.jar:$CLASSPATH
```

Pour vérifier que c'est correct, récupérez les fichiers *Vertigo.xml* et *Film.xsl* sur le site et effectuez la commande suivante :

```
java org.apache.xalan.xslt.Process -in Vertigo.xml -xsl Film.xsl
                -out Vertigo.html
```

La commande devrait produire un fichier *Vertigo.html*. En principe vous devriez obtenir le même résultat en passant par Mozilla qui effectue des transformations XSLT. Il faut alors inclure l'instruction suivante pour indiquer quel est le programme XSLT à appliquer :

```
<?xml-stylesheet type="text/xsl" href="Film.xsl"?>
```

Vous pouvez récupérer sur le site l'archive *sitecours.tar* qui contient la petite application XSLT qui produit les documents HTML de ce même site. Décompressez l'archive dans *public_html* avec les commandes suivantes :

```
cd
cd public_html
tar xvf sitecours.tar
```

Cela crée un sous-répertoire *SITECOURS*. En vous plaçant dans ce répertoire et en lançant les commandes :

```
make clean
make
```

vous produisez les documents HTML par transformations XSLT. Pour voir ce que ça donne, accédez à l'URL suivante (il va sans dire que « rigaux » est là pour l'exemple) :

```
http://www.ie2.u-psud.fr:8080/~rigaux/SITECOURS
```

Vous pouvez regarder le *Makefile* et les différents fichiers (notamment *sitemap.xml* et *style.xsl*) pour comprendre comment ça marche.

1.2 XSQL

XSQL vous a été présenté en cours. Il suffit de donner le suffixe `.xsql` à un fichier et d'y accéder en passant par le serveur Tomcat (en écoute sur *http://www.ie2.u-psud.fr:8080*) pour que le fichier soit traité par XSQL.

Pour que Tomcat puisse accéder à vos fichiers, il faut impérativement qu'il soient sous le répertoire *webapps* qui se trouve sur */net/djembe/webapps*. Chacun d'entre vous dispose d'un sous-répertoire à son nom, contenant déjà l'ensemble des exemples vus en cours.

Pour la version WML, il faut utiliser le navigateur Deckit que vous pouvez récupérer sur le site.

2 Documents XML

Ces premiers exercices doivent permettre de se familiariser avec la création de documents « à balise », d'abord HTML, puis XML. Dans un deuxième temps ces documents sont manipulés sous leur forme arborescente avec l'API DOM. Une pratique minimale du langage java est nécessaire.

Les premiers exercices servent essentiellement à prendre en main l'environnement. Si vous ne connaissez **rien** au Web et à ses langages, un chapitre explicatif est fourni sur le site.

Exercice 2.1 *Si vous ne connaissez pas HTML, voici quelques manipulations qui vous permettront de vous familiariser avec l'essentiel des balises.*

1. *Allez sur le site www.ie2.u-psud.fr/~rigaux et récupérez les quelques exemples qui sont donnés.*

2. *Installez-les dans votre répertoire et testez-les.*

3. *Éditez ces fichiers et faites diverses modifications.*

Exercice 2.2 *Créez votre page personnelle en HTML (pas trop longue), présentant des informations comme :*

1. *votre état civil (nom, prénom, date de naissance, etc) ;*

2. *une description de votre formation (diplômes, études) ;*

3. *vos hobbies ;*

4. *en vrac : vos sites, vos livres, vos disques, vos films préférés, avec les liens vers les URL.*

Présentez cette page comme vous le souhaitez, en vous inspirant au besoin des exemples que vous pouvez trouver absolument partout sur le Web. Faites en sorte que dans cette page on trouve les principaux éléments de présentation de HTML, à savoir

- *un ou plusieurs tableaux ;*
- *des ancres vers d'autres sites ;*
- *des listes ;*
- *éventuellement une image ;*
- *etc.*

Exercice 2.3 *Reprenez le* contenu *du document précédent, et créez cette fois un document InfosPerso.xsql avec XML. Veillez à ce que chaque constituant du contenu soit balisé de manière à ce qu'il soit possible de le traiter spécifiquement par la suite (par exemple marquez les URL avec un balisage spécifique). Inspirez-vous des exemples fournis si nécessaire.*

Faites en sorte que dans le document XML on trouve (au moins) les types de nœuds suivants :

1. *des éléments et du texte ;*

2. *des sections littérales ;*

3. *des commentaires.*

Exercice 2.4 *Appliquez à votre document XML le programme XSLT generic.xsl que vous pourrez trouver sur notre site. Ce programme affiche en HTML l'arborescence de n'importe quel document XML. Il suffit de placer une instruction de traitement dans le prologue du document :*

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<?xml-stylesheet href="generic.xsl" type="text/xsl"?>

<!-- Suite du document  -->
```

L'application de ce programme XSLT est également l'occasion de vérifier que votre document est bien formé.

3 XPath

Vous pouvez exécuter des intructions XPath avec le programme *ApplyXPath* fourni avec Xalan. Récupérez le fichier *ApplyXPath.java* (sur le site) et compilez-le avec la commande :

```
javac ApplyXPath.java
```

On obtient un fichier *ApplyXPath.class* qui peut être exécuté par une machine java avec la commande [1] :
`java ApplyXpath` *`fichierXML expressionXPath`*
Pour faciliter la tâche du programme il est préférable d'encadrer *`expressionXPath`* par des guillemets doubles (") et les chaînes de caractères par des guillemets simples, ou l'inverse.

Exercice 3.1 *Expliquez la différence entre les deux expressions suivantes, et donnez un document pour laquelle on n'obtient pas le même résultat :*

- *`//B[position() = 1]`*
- *`/descendant::B[position() = 1]`*

Dans le premier cas, la notation étendue est `/descendant-or-self::node()/B[1]`. On a donc le premier fils de type `B` d'un des éléments du document. Dans le second cas on obtient le premier élément de type `B` rencontré dans le document.
Exemple d'une instance : `/A[C[B][B]D[B]]`.

Exercice 3.2 *Expliquez la signification des expressions suivantes :*

- *`//COURS[INTITULE='XML']`*
- *`//COURS[INTITULE=XML]`*

Donnez une instance pour laquelle le résultat est identique.

La première donne les éléments de type `COURS` ayant un élément fils `INTITULE` dont la valeur textuelle est `XML`. La seconde donne les éléments de type `COURS` ayant un élément fils `INTITULE` **et** un élément de type `XML` dont les valeurs textuelles sont identiques. On en déduit facilement une instance pour laquelle l'évaluation donne le même résultat.

Exercice 3.3 *Récupérez sur le site le document Films.xml qui contient un ensemble d'informations sur des films. Appliquez à ce document les expressions XPath pour rechercher les informations suivantes.*

1. *La liste des titres de films.*
2. *Les titres des films parus en 1990*
3. *Le résumé d'Alien*
4. *Titre des films avec Bruce Willis*
5. *Quels films ont un résumé ?*
6. *Quels films n'ont pas de résumé ?*
7. *Donner les titres des films vieux de plus de trente ans.*
8. *Quel rôle joue Harvey Keitel dans Reservoir dogs ?*
9. *Quel est le dernier film du document ?*
10. *Quel est le titre du film qui précède immédiatement Shining (dans l'ordre du document).*

1. Le fichier *ApplyXPath.class* doit être dans un répertoire référencé par la variable `CLASSPATH`.

11. *Qui a mis en scène Vertigo ?*
12. *Donnez les titres des films qui contiennent un « V » (utiliser la fonction contains())*
13. *Donner les nœuds qui ont exactement trois descendants (utiliser la fonction count()).*
14. *Donner les nœuds dont le nom contient la chaîne « TU » (fonction name())*

1. `//TITRE`
2. `/FILMS/FILM[@Annee=1990]/TITRE`
3. `/FILMS/FILM[TITRE='Alien']/RESUME`
4. `/FILMS/FILM[.//NOM='Willis' and .//PRENOM='Bruce']/TITRE`
5. `/FILMS/FILM[RESUME]/TITRE`
6. `/FILMS/FILM[not(RESUME)]/TITRE`
7. `/FILMS/FILM[2001 - @Annee > 30]/TITRE`
8. `/FILMS/FILM[TITRE='Reservoir dogs']/ROLES/ROLE[NOM='Keitel']/INTITULE`
9. `/FILMS/FILM[position() = last()]/TITRE`
10. `/FILMS/FILM[TITRE='Shining']/preceding::FILM[position()=1]/TITRE`
11. `/FILMS/ARTISTE[@id=/FILMS/FILM[TITRE='Vertigo']/MES/@idref]/ACTNOM`
12. `/FILMS/FILM/TITRE[contains (text(), 'V')]`
13. `//*[count(descendant::*) = 3]`
14. `//*[contains(name(), 'TU')]`

Exercice 3.4 *À partir du programme Preodre.java des exercices DOM, créez une version de ApplyXPath qui évalue le sous-ensemble des expressions XPath suivant :*

- *seuls les axes* `child` *et* `attribute` *sont reconnus ;*
- *les prédicats consistent uniquement en expressions XPath.*

Le programme doit prendre en entrée le nom du fichier XML, le numéro du nœud contexte (dans l'ordre du document) et l'expression XPath.

4 Programmation XSLT

Pour commencer, voici quelques exercices d'introduction consistant à créer et évaluer des programmes très simples.

Exercice 4.1 *Écrire un programme sans aucune règle et l'appliquer à Alien.xml.*

Exercice 4.2 *Écrire un programme avec une seule règle s'appliquant à la racine du document (attribut* `match='/'`*). Créer dans cette règle un document HTML contenant toutes les informations sur Alien, en utilisant seulement des* `xsl:value-of` *et des expressions XPath.*

Exercice 4.3 *Maintenant écrivons un programme avec des* `xsl:apply-templates`*. Le programme s'applique à Films.xml, et construit une représentation HTML avec les règles suivantes :*

- *une règle s'appliquant à la racine du document pour produire le « cadre » HTML (balises* `<html>`*,* `<head>`*,* `<body>` *; dans* `<body>`*, demander l'application de règles à tous les éléments* `<FILM>` *;*

– *une règle s'appliquant à un élément* `<FILM>` *; afficher le titre, le genre, le pays ; demander l'application de règles pour tous les rôles ;*

– *une règle s'appliquant aux rôles : les afficher sous forme de liste HTML, avec le prénom et le nom de l'acteur, et l'intitulé du rôle.*

Voici maintenant une série d'exercices consistant à créer des programmes XSLT « génériques » qui s'appliquent à tout type de document XML.

Exercice 4.4 *Écrire un programme qui affiche le nom et la valeur de tous les attributs du document.*

Exercice 4.5 *Écrire un programme qui affiche le nom des éléments d'un document, et pour chaque élément :*

– *le nombre de ses attributs ;*

– *le nombre de descendants de type* ***Element*** *;*

– *son numéro dans l'ordre du document.*

Rappel : le nombre de nœuds dans un ensemble est obtenu avec la fonction XPath count().

Exercice 4.6 *Cet exercice permet de produire, à partir de n'importe quel document XML en entrée, un document XML en sortie qui en décrit – en partie – la structure et le contenu. Le principe est de transformer tous les nœuds,* quel que soit leur type, *en élément dans le document résultat. Le document résultat contient les informations suivantes :*

– *Pour chaque nœud du document source, il existe un* élément *dans le document résultat dont le nom est, selon le type du nœud,* `DOCUMENT`, `ELEMENT`, `COMMENT`, `ProcessingInstruction`, `ATTRIBUT` *ou* `TEXT`[2]

– *Chaque élément* `ELEMENT` *du document résultat a un attribut* `nom` *avec le nom de l'élément dans le document source ; les autres éléments (sauf* `DOCUMENT`*) ont un attribut* `valeur` *avec la valeur du nœud dans le document source. (Rappel : un attribut est produit avec* `xsl:attribute`*).*

– *Chaque élément du document résultat contient un attribut* `position` *qui représente la position hiérarchique du nœud correspondant dans le document source. Si par exemple l'élément racine est le second fils de la racine du document, il porte le numéro 1.2.*

Exercice 4.7 *Écrire un programme qui recopie le document source en remplaçant les commentaires par un élément* `COMMENT` *(suggestion : utilisez les règles avec priorités).*

Les exercices suivants s'appliquent au document *Biblio.xml* qui contient une liste de publications scientifiques.

Exercice 4.8 *À partir de Biblio.xml, produire un document HTML contenant :*

1. *Une table des matières avec la liste des années de publication d'articles triées en ordre descendant.*

2. *La liste des articles, triés par année en ordre descendant. Pour chaque article figurera : les auteurs, le titre en italiques, le résumé*

Bien entendu la table des matières doit être constituée de liens menant vers une des années référencées.

Exercice 4.9 *Donner la liste des auteurs avec, pour chaque auteur, le nombre de publications.*

2. NB : on travaille sur une représentation des documents XML qui ne fait plus la différence entre **Text** et **CDataSection**.

Les exercices qui suivent s'appliquent aux deux documents *Station.xml* et *Client.xml*. Les personnes décrites dans *Client.xml* sont référencées dans *Station.xml*. Il est donc nécessaire d'utiliser la fonction *document*() pour créer des documents résultats à partir d'informations réparties dans ces deux documents sources.

Exercice 4.10 *Produire un document HTML donnant, pour chaque station, la liste des séjours avec le nom, le prénom, et la région d'origine du client.*

Exercice 4.11 *Produire un document HTML donnant, pour chaque client, la liste des séjours avec le nom de la station. Calculer le prix payé par chaque client en multipliant le tarif par le nombre de places réservées.*

Exercice 4.12 *Pour finir, reprenez votre document InfosPerso.xml et transformez-le en page HTML.*

5 DOM et SAX

Les exercices qui suivent permettent de manipuler des documents XML via les interfaces de programmation DOM et SAX. Même si n'importe quel parseur devrait faire l'affaire, nos exemples sont basés sur les parseurs Xerces fournis par la fondation Apache, et sur les modules de sérialisation inclus dans Xalan.

Exercice 5.1 *Établir (sur papier) la représentation de votre document InfoPerso sous forme d'arbre DOM.* ***Attention*** *à bien représenter les nœuds de textes constitués uniquement d'espaces.*

Pour chaque nœud, vous indiquerez :

1. *son type ;*
2. *son numéro dans l'ordre du document ;*
3. *son nom (s'il a un nom) ;*
4. *sa valeur (s'il en a une).*

Voici maintenant quelques manipulations et explications pour préparer les exercices suivants. Récupérez sur notre site le programme *Preordre.java* et compilez-le avec :

```
javac Preordre.java
```

Ce programme analyse avec DOM un document XML contenu dans un fichier, parcourt tous les nœuds dans l'ordre du document, et ajoute aux nœuds de type **Text** leur numéro dans l'ordre du document. Enfin le document modifié est sérialisé. Voici par exemple comment l'appliquer à *Alien.xml*

```
java Preordre Alien.xml
```

Les exercices qui suivent consistent à effectuer des modifications du programme *Preordre*. Vous pouvez appliquer ces programmes à votre document *InfosPerso*.

Exercice 5.2 *Écrivez une version du programme qui affiche la liste de tous les nœuds avec leur numéro, leur nom, leur type et leur valeur.*

Exercice 5.3 *Écrivez une version du programme qui affiche, pour tous les éléments ayant des attributs, le nom de l'élément, les noms des attributs et leurs valeurs.*

Exercice 5.4 *Écrivez une version du programme qui prend en argument un nom de fichier XML, un nom d'élément, et affiche tous les éléments du document ayant ce nom.*

Exercice 5.5 *Écrivez une version du programme qui prend en argument un nom de fichier XML F, un nom d'élément E, une chaîne de caractères C, et ajoute un élément de type E avec le contenu textuel C comme dernier fils de l'élément racine de F.*

Exercice 5.6 *Écrivez une version du programme qui supprime tous les nœuds de type* ***Text*** *ne contenant que des espaces.*

Exercice 5.7 *Créez un annuaire XML Personnes.xml qui contient des éléments* <PERSONNE> *avec un attribut* `id` *et – au moins – les trois sous-éléments suivants :* <NOM>, <PRENOM> *et* <EMAIL>. *Par exemple :*

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<ANNUAIRE>
  <PERSONNE id='1'>
    <NOM>Amann</NOM>
    <PRENOM>Bernd</PRENOM>
    <EMAIL>amann@cnam.fr</EMAIL>
  </PERSONNE>
  <PERSONNE id='2'>
    <NOM>Rigaux</NOM>
    <PRENOM>Philippe</PRENOM>
    <EMAIL>rigaux@lri.fr</EMAIL>
  </PERSONNE>
</ANNUAIRE>
```

Écrivez un programme qui ajoute des éléments <PERSONNE> *à ce document. Voici trois versions possibles.*

1. *Créez un programme qui prend en argument, sur la ligne de commande, les trois sous-éléments de* <PERSONNE>.
2. *Placez un ou plusieurs nouveaux éléments* <PERSONNE> *dans un fichier temporaire et faites l'union de Personnes.xml et de ce fichier.*
3. *Utiisez un formulaire HTML pour saisir les données, et une servlet pour les récupérer et insérer le nouvel élément.*

Pour compléter, associez au nouvel élément un commentaire XML indiquant la date d'entrée, et associez automatiquement un nouvel identifiant à la personne insérée sous forme d'un attribut `id`. *Vous devez utiliser pour cela la méthode setAttributeNode(Attr* `newAttr`*) de la classe* ***Element*** *qui ajoute un nœud attribut.*

6 Projet

Voici le projet proposé. Le but est de mettre en œuvre de la manière la plus complète possible les techniques et outils présentés dans le cours, et ce dans le cadre d'une application de publication et d'échange de données raisonnablement réaliste.

L'environnement préconisé est composé d'une base de données relationnelle (MySQL, et peut-être ORACLE), du conteneur de servlets TOMCAT, d'un outil de transformation dynamique de données relationnelles en documents XML (XSQL d'ORACLE est conseillé), de parseurs DOM et SAX (Xerces conseillé) et d'un parseur XSLT (Xalan ou le parseur d'ORACLE).

Le projet consiste à gérer des informations, soit dans une base de données, soit dans des fichiers texte, à unifier ces informations sous forme de documents XML, et à les publier, dans un environnement Web, en différents formats : HTML, WML, FO, SVG, SMIL, VRML, ... Les deux premiers sont nécessaires et suffisants, mais vous êtes libres de partir à l'aventure. D'une manière générale les initiatives s'écartant du cadre ci-dessus sont bienvenues, mais mieux vaut en parler à l'enseignant au préalable.

Voici les points importants qui doivent être traités dans le cadre du projet :

1. Une partie au moins des informations doit être présente dans une base de données (ORACLE ou MySQL) et extraite dynamiquement sous forme de document XML. Vous pouvez utiliser la technique de votre choix, XSQL d'ORACLE étant l'outil préconisé et proposé.

2. La publication doit s'effectuer dans au moins deux formats de sortie, par exemple HTML, WML. Un troisième format apprécié est XSL-FO/PDF. Si vous souhaitez en utiliser d'autres (par exemple SVG ou SMIL), voir avec l'enseignant.

3. La publication avec des langages hypertextes (HTML, WML et même PDF) doit utiliser intensivement les liens pour naviguer dans les données.

4. Incluez un petit moteur de recherche pour les versions HTML et WML afin de permettre à l'utilisateur d'entrer des critères de recherche.

5. La mise à jour des informations s'effectue avec des formulaires web et des servlets ou des JSP côté serveur (une introduction à ces techniques sera proposée en cours). Utilisez JDBC pour des mises à jour de la base relationnelle, et DOM ou SAX pour la mise à jour de documents XML.

Au début du projet, priviliégiez les aspects bases de donnnées et publication HTML/WML. La mise à jour et la production de PDF ou autre viennent après. Ne négligez pas non plus la qualité de la présentation, même si ce n'est pas l'essentiel.

6.1 Sujet 2003 : petites annonces logements

La motivation du projet est simple et concrète : imaginez que vous arrivez dans un belle ville de province dotée d'une université réputée, malheureusement située loin de votre domicile habituel. Vous partez donc à la recherche d'un logement et vous vous apercevez que c'est difficile et cher. Ce n'est qu'après de longues et difficiles recherches que vous prenez connaissance de l'existence d'une association d'étudiants qui gère, plus ou moins efficacement, une liste de logements. D'où l'idée de faciliter la recherche des étudiants qui viendront après vous en créant un site donnant accès à cette liste et permettant de l'enrichir.

Dans un deuxième temps vous vous rendrez compte que beaucoup d'autres personnes ont eu la même idée et ont développé un site analogue dans d'autres université (voire dans la même...). Vous décidez alors de mettre en commun vos informations. Une DTD commune est définie, et tout le monde rend public sa liste de logements dans cette DTD. Il devient possible d'intégrer les informations disponibles pour produire un guide coopératif des logements étudiants sur le Web, dans beaucoup de formats différents.

Le projet consiste à réaliser ces deux phases en utilisant les techniques XML vues en cours :

– **Le site de votre université**. Dans une première phase, vous travaillez par groupes de 2 ou 3, indépendamment des autres, et vous réalisez un site destiné aux étudiants de votre université, dans votre ville.

– **Le site coopératif**. Dans un deuxième temps on vous donnera une DTD et vous devrez, comme tous les autres groupes, publier vos données dans cette DTD. Vous pourrez alors étendre votre site en récupérant les informations issues des autres groupes. NB : selon le temps disponible et l'avancement des différents groupes cette phase pourra éventuellement être considérée comme facultative.

Rien ne vous empêche de consulter les sites analogues éventuellement existant pour vous donner des idées sur la présentation, le contenu, les fonctionnalités, etc.

Voici quelques indications pour la réalisation, que vous êtes invités à respecter pour faciliter l'intégration des données de chaque groupe. En fonction du temps dont vous disposez, vous êtes libre d'inventer des compléments qui vous paraissent plus intéressants et/ou amusants.

6.2 Le site de votre université

Le site doit être relatif à une université précise, située dans une ville et une région. Ensuite on donnera une liste des logements, avec pour chaque logement :

– son adresse complète ;

– le nombre de pièces ;

– le loyer mensuel ;

– la distance pour aller à l'université ;

– les dates de disponibilité ;

– une description (environnement, état, commentaires sur les propriétaires, etc).

Les informations ci-dessus sont nécessaires. Ensuite, le site étant destiné aux nouveaux arrivants, il doit comprendre des informations diverses et variées sur le contexte universitaire : nombre d'étudiants, moyens de transport, départements d'enseignement, coût moyen du logement, etc. Épicez selon vos goûts, en ajoutant par exemple des zones de commentaires sur les logements déposés par des locataires antérieurs, des photos, etc.

Toute idée créative est bienvenue. Vous pouvez par exemple envisager de gérer des logements en co-location, ce qui est un peu plus compliqué.

Représentation des données

Vous devez définir la représentation de ces informations en XML, en en plaçant une partie dans la base de données. Par exemple la liste des logements peut être en base, et le reste du site en XML « pur ». Réflechissez aux avantages/inconvénients des diverses solutions techniques, et préparez-vous à défendre votre point de vue.

Mises à jour

Vous pouvez vous contenter, surtout dans un premier temps, d'effectuer les mises à jour manuellement en éditant directement les documents XML. Il serait bien sûr préférable de passer par l'intermédiaire de formulaires associés à des servlets ou JSP, aussi bien pour la base de données que pour la manipulation des documents XML. Essayez de garder un peu de temps à la fin du projet pour cela.

Publication

Enfin produisez au moins deux versions différentes de vos informations, dont une en HTML, et l'autre en WML. Une édition PDF (via XSL-FO) sera appréciée, mais il faut être sûr de pouvoir en venir à bout dans les délais impartis.

Pour le site web, il doit être possible de saisir avec un formulaire HTML des critères de recherche (la même chose est possible, de manière simplifiéee, en WML). On devrait pouvoir effectuer des recherches pour un nombre de pièces, un loyer, une période, etc. N'oubliez de fournir des liens hypertextes pour pouvoir naviguer dans le site, ou pour générer un PDF permettant d'imprimer le contenu affiché à l'écran.

6.3 Site coopératif

On vous donnera, 3 ou 4 semaines avant la remise du projet, une DTD commune pour les logements universitaires. Vous devrez alors :

1. transformez vos informations dans cette DTD, et la rendre accessible à une URL publique ;
2. tous les autres groupes ayant fait de même, vous avez alors la possibilité d'intégrer les données de tous et de les publier dans votre site.

Vous pouvez adapter votre code à la DTD commune, ou bien au contraire transformer la DTD commune vers votre propre DTD. Pour l'intégration vous pouvez utiliser le système d'entités externes, ou un document XML qui référence toutes les URL. À vous d'effectuer les bons choix techniques. Les enseignants sont là pour vous conseiller bien entendu.

Bon courage !