Indexation et Recherche d'Information
Interrogation d'une base de données SQL via une Servlet

LO 17

1 Introduction

Vous allez dans ce TD utiliser des servlets Java pour interroger une base de données Postgresql à partir d'une page html.

Le principe de fonctionnement est le suivant : une application Java, une *servlet*, dont la fonction est de prendre en compte des requêtes SQL reçues d'un client web, de les transmettre à un serveur de base de données et de renvoyer au client web les résultats obtenus, doit être réalisée et déployée en tant que *webapp* dans un conteneur Tomcat sur votre poste de travail. Tomcat traitera des requêtes HTTP que vous lui ferez parvenir en utilisant un URL commençant par `http://localhost:8080/`.

2 Préparation de l'environnement

- Si vous utilisez votre propre matériel informatique, sachez que l'Eclipse accessible par le menu du *Dispositif enseignement* est assorti du *'Plugin Tomcat'*.

- Assurez-vous que votre *workspace* Eclipse se situe sur votre *Disque Z* et non au niveau de votre profil itinérant (par exemple, sur votre *Bureau*). Sinon, il y a un fort risque de saturation de votre quota disque.

- Le plus simple est de créer un répertoire spécifique dans votre *workspace* dans lequel vous mettrez tous les fichiers que vous téléchargerez.

- Téléchargez depuis la zone de téléchargement du site web de l'UV le fichier **apache-tomcat-6.0.37.zip**, et **EXTRAYEZ**-en le contenu vers votre *Disque Z*.

- Dans Eclipse, au niveau du menu Fenêtre -> Préférences -> Tomcat, spécifiez la bonne version de Tomcat (**6.x**) et le **RÉPERTOIRE** dans lequel votre Tomcat est installé (`Z/../apache-tomcat-6.0.37`).

- Au niveau du menu Fenêtre -> Préférences -> Tomcat -> Application 'Tomcat Manager', saisissez et *Ajoutez au fichier tomcat-users.xml* un nom d'utilisateur (votre **login lo17**) et un mot de passe (votre **mot de passe lo17**) qui serviront à vous authentifier lorsque vous accéderez à `http://localhost:8080/manager`.

- Note : (vous n'avez en principe rien à changer ici car la configuration est en principe correctement initialisée) Puisque Tomcat a obligatoirement besoin d'un JDK plutôt que d'un simple JRE, vous devez 'faire pointer' votre Eclipse vers un JDK que vous trouverez quelque part sur le disque de votre poste de travail (par exemple, `C/Program Files/Java/jdk1.6.0_27`).

- **Créez un nouveau projet Tomcat dans Eclipse (cliquez sur le dossier Java qui est tout en bas du menu des projets) :**
  New -> Projet -> Java -> Projet Tomcat

3 Les fichiers et répertoires qui constituent votre *webapp*

Votre projet Tomcat va être déployé en *webapp* par votre serveur Tomcat au démarrage de ce dernier (réalisé par un clic de l'icône correspondant dans Eclipse).

- Les documents que vous placerez directement en dessous du répertoire du projet (c'est à dire dans `Z:/<votre workspace>/<nom de votre projet>`) seront accessibles tels quels au client web, et c'est donc à ce niveau que vous devez mettre vos fichiers .html, etc.

- Le sous-répertoire `WEB-INF` contient les éléments que Tomcat utilisera pour construire des pages dynamiquement, notamment des servlet.

  - Les fichiers source de vos servlet (fichiers .java) sont à placer dans `WEB-INF/src`.

  - Les classes Java correspondantes (fichiers .class) sont créées de manière transparente par Eclipse dans `WEB-INF/classes` (que par défaut le *'Plugin Tomcat'* n'affiche pas).

  - Toute bibliothèque dont vos servlet pourraient avoir besoin (en dehors de celles qui sont automatiquement rendues disponibles par le conteneur Tomcat) sont à placer dans `WEB-INF/lib`. Pour vous il s'agira notamment du fichier **postgresql-9.0-801.jdbc4.jar**, que vous téléchargerez depuis le site web de l'UV.

- Vous devez également créer, dans le répertoire `WEB-INF`, un fichier `web.xml` sur le modèle de celui que vous pouvez télécharger depuis le site web de l'UV. Ce fichier fait correspondre les servlet que vous pourriez mettre à disposition à des chemins d'URL permettant à des clients web distants de les invoquer.

4 Préparation de votre page html

Vous trouverez dans la rubrique TDSERVLET, du répertoire TELECHARGEMENT du site de lo17 un exemple de page html qui accepte une requête en entrée et affiche le résulat dans un cadre (frame). Il faudra changer les extensions « `.txt` » en « `.html` » pour les deux fichiers concernés. Inspirez-vous de cet exemple pour créer votre propre application de façon à ce qu'elle permette, par exemple, de mettre en colonnes (en tableau) les différentes réponses si la requête porte sur des items multiples (auteur et année et fichier) et que lorsque la réponse contient des noms de fichiers, ceux-ci soient des liens *clickables* vers les fichiers en question.

Pour ce qui est des URL à utiliser, le nom que vous choisissez pour votre projet fera partie du chemin. Si vous déployez un fichier `LanceRequete.html` dans un projet que vous avez appelé `monProjetTomcatLCI`, l'URL correspondant sera `http://localhost:8080/monProjetTomactLCI/LanceRequete.html`. Faîtes attention aux *chemins relatifs*. **Dans web.xml**, un chemin renseigné par la balise `<url-pattern>`, même si le premier caractère est `/`, reste *relatif à la racine de votre projet*, et par conséquent

```
<url-pattern>/servlet/LanceRequete</url-pattern>
```

désigne l'URL `http://localhost:8080/`*`monProjetTomcatLCI`*`/servlet/LanceRequete` (notez que `/servlet` ne correspond pas à un sous-répertoire, mais dans le contexte d'un `<url-pattern>` est interprété par Tomcat comme une chaîne de caractères quelconque). En revanche, **dans une page html** faisant partie de votre projet, le caractère `/` initial dans un URL indique que l'URL est *relatif à la racine de votre serveur*. Si vous désirez qu'il soit relatif à la racine de votre *projet* le caractère `/` doit être omis. Ainsi :

`<form action = "/servlet/LanceRequete">` -> `http://localhost:8080/servlet/LanceRequete`
(mauvais URL)

`<form action = "servlet/LanceRequete">` -> `http://localhost:8080/`*`monProjetTomcatLCI`*`/servlet/LanceRequete`
(bon URL)