TP Servlet

Olivier AUBERT

29 avril 2002

1 Configuration d'apache

Installez apache à partir des paquetages *Debian* disponibles sur le serveur `134.214.90.7`. Installez de plus le paquetage `jserv`.

Pour activer *jserv* sous apache, il faut charger le module correspondant. La phase de la configuration du paquetage doit le faire automatiquement. Consultez le fichier de configuration d'apache pour vérifier que le module correspondant est bien chargé.

2 Configuration de jserv

Le répertoire de configuration de jserv est `/etc/jserv`. On peut y trouver les fichiers `jserv.conf`, `jserv.properties` et `jserv.secret.key`. De plus, le fichier `/etc/jserv/zones/root.properties` définit les propriétés essentielles du serveur de servlet par défaut. Une zone est l'équivalent pour les servlets des *VirtualHost* d'apache. Chaque zone va donc définir un répertoire contenant les servlets qu'elle pourra exécuter. La correspondance entre une zone et l'URL à laquelle on pourra l'atteindre est définie dans apache par la directive `ApJServMount /servlets /root`.

Vous pouvez dans un premier temps vérifier que jserv est bien installé en accédant à l'URL `http://localhost/servlets/IsItWorking`. Le code source de la servlet donnant ce résultat est le fichier `/usr/share/doc/jserv/examples/Hello.java`.

L'URL `http://localhost/jserv/` active un handler permettant d'obtenir des informations sur la configuration du serveur de servlet (de manière analogue au *handler* `server-info` que nous avions installé sur apache). Trouvez dans quel fichier de configuration cet handler est précisé. Modifiez le fichier de propriétés de jserv pour pouvoir visualiser le statut du moteur de servlet (désactivé par défaut).

Vérifiez que le processus `jserv` existe bien et est bien distinct du processus apache.

3 Création d'une servlet simple

Nous allons créer une servlet effectuant une opération mathématique quelconque (factorielle, produit, etc). Pour cela, on l'invoquera via un formulaire HTML qui demandera à l'utilisateur les données nécessaires, et affichera la réponse.

Recopiez le fichier `Hello.java` dans `/tmp/MathTest.java`. Effectuez les modifications nécessaires. Pour le compiler, il faut spécifier que l'on utilise le contenu de la bibliothèque `servlet-2.0.jar`, d'une manière similaire à la ligne :

```
javac -classpath /usr/lib/jdk1.1/lib/classes.zip:/usr/share/java/servlet-2.0.jar MathTest.java
```

Vous trouverez la documentation de l'API java à l'adresse `http://134.214.90.7/java/` ainsi que celle de l'API servlet à l'adresse `http://134.214.90.7/javax/`

Placez le fichier `.class` généré dans le répertoire approprié. Créez le formulaire HTML permettant d'activer la servlet. Testez que cette dernière fonctionne correctement.

4 Gestion de cookies

Le constructeur `javax.servlet.http.Cookie` permet de créer un cookie avec un nom et une valeur spécifiés. Attention : si votre servlet renvoie une réponse en utilisant le *Writer*, prenez soin de créer le cookie et de l'ajouter à la réponse avec la méthode `addCookie` avant d'accéder au *PrintWriter* de la réponse. En effet, le cookie est transmis au navigateur dans les entêtes HTTP, et non dans les données.

Créez une applet qui va manipuler un cookie appelé `DATE`, qui contiendra la date de dernier accès à la servlet, et un autre appelé `TOTAL` qui contiendra la nombre d'accès effectués sur la servlet. Pour savoir si le cookie est déjà positionné, on utilise la méthode `getCookies` sur l'objet de requête. La servlet retournera en outre un document HTML indiquant si les cookies sont positionnés, et dans la positive, quelles sont leurs valeurs.

Vérifiez que le cookie est bien déclaré dans le navigateur. Pour Netscape, les cookies sont stockés dans le répertoire `~/.netscape/cookies`. Il est intéressant de modifier les préférences de Netscape (menu `Edit/Preferences/Advanced` et de valider l'option *Warn me before accepting a cookie.* Une autre solution est d'utiliser `lynx` qui affiche par défaut un message lorsque le serveur tente de positionner un cookie.

Manipulez les délais d'expiration pour vérifier que les cookies sont bien supprimés du navigateur.

5 Cycle de vie

Nous allons utiliser la méthode `log` des servlet, qui permet de générer des messages d'historique dans le fichier d'historique des servlets, pour étudier le cycle de vie des servlets.

Créez une servlet `LifeCycle.java` qui utilisera la méthode `log` dans les méthodes `init`, `doGet` et `destroy`. Effectuez un accès à la servlet. Vérifiez dans le fichier de log à quels moments sont effectués les appels aux différentes méthodes.

Pour la méthode `init`, il est important de bien en respecter l'interface originale et d'invoquer la méthode `super.init (sc);` de la classe parent afin que les initialisations nécessaires soient effectuées.

6 Accès aux informations de contexte

Les informations de contexte et de configuration de la servlet sont accessibles via les méthodes `getServletContext` et `getServletConfig`.

Écrivez une applet renvoyant l'ensemble des attributs (nom et valeur) disponibles via le contexte de la servlet, ainsi que l'ensemble des paramètres (nom et valeur) fournis à la servlet par l'objet `ServletConfig`. Ajoutez dans le fichier `/etc/jserv/zones/root.properties` un paramètre qui sera passé à votre servlet. Vérifiez que la servlet a bien accès au paramètre fixé.