Introduction aux Servlets

Laurent Réveillère
http://reveille.vvv.enseirb-matmeca.fr/

Dans ce TP vous allez déployer vos premiers servlets et découvrir la structure d'un conteneur de servlet (Tomcat). Pour le moment, pas d'IDE (ex: Eclipse), on code à l'ancienne. Les seules choses dont vous avez besoin pour réaliser les tâches demandées sont un terminal et un navigateur web (on peut aussi lancer un navigateur web dans un terminal (Lynx par exemple)).

Q1.1 Installation de Tomcat

- Accédez à votre machine virtuelle se trouvant sur morpheus via ssh.
- Mettre à jour le gestionnaire de paquets `sudo apt-get update`.
- Assurez vous que le SDK Java est bien installé sur votre VM, en lançant la commande `javac -version`. Si ce n'est pas le cas, installez java avec la commande `sudo apt-get install openjdk-7-jdk`
- Téléchargez Tomcat à l'adresse suivante:

  `http://www.us.apache.org/dist/tomcat/tomcat-8/v8.0.30/bin/apache-tomcat-8.0.30.zip`
- Extraire l'archive
- Éditez le fichier `conf/server.xml` pour que Tomcat utilise le port 80 au lieu de 8080.
- Ajouter les droits en exécution aux fichiers d'extension *sh* dans le répertoire `bin`.
- Démarrez le serveur en exécutant le script `bin/startup.sh` en tant que super-utilisateur (seuls `root` ou le super-utilisateur peuvent démarrer un serveur utilisant le port 80)
- Les messages d'erreurs de Tomcat sont enregistrés dans le fichier logs/catalina.out , vous pouvez vérifier ici que le serveur s'est lancé sans erreur.
- Pour vérifier que le serveur est bien lancé, utilisez la commande `lynx http://localhost/`, la page d'accueil de Tomcat devrait s'afficher.
- Vérifiez que le serveur est bien accessible depuis votre machine, dans la barre d'adresse de votre navigateur, l'adresse: `http://{votre-login}.rmorpheus.enseirb.fr/`
- Vous pouvez stopper le serveur avec le script `bin/shutdown.sh`.

Q1.2 Votre premier Servlet : Hello World!

Les différentes applications présentes dans le serveur Tomcat se trouvent dans le dossier `webapps`. C'est ici que nous allons créer notre premier servlet. Dans ce dossier, créez un répertoire *hello*. Les fichiers de ce répertoire seront accessibles via `http://{votre-login}.rmorpheus.enseirb.fr/hello`.

```java
package fr.enseirb.t2.servlet;

import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
import javax.servlet.annotation.*;

@WebServlet(urlPatterns="/world")
public class HelloServlet extends HttpServlet
{
  @Override
  protected void doGet(HttpServletRequest req, HttpServletResponse resp)
    throws ServletException, IOException {

    resp.setContentType("text/html");
    PrintWriter out = resp.getWriter();
    out.println("<html>");
    out.println("<body>");
    out.println("Hello World!");
    out.println("</body>");
    out.println("</html>");
    }
}
```

Figure 1: HelloServlet.java

Dans le dossier *hello*, créez un dossier *WEB-INF*. C'est ici que seront situées les informations liées à la configuration de l'application. Dans le dossier *WEB-INF*, ajoutez un dossier *classes*. Comme son nom l'indique, c'est ici que devront être déployées les différentes classes utilisées par l'application.

Créez enfin un dossier *src* dans le dossier *webapps/hello*, puis ajoutez dans ce dossier un nouveau fichier *HelloServlet.java*, contenant le code de la figure 1.
Vous pouvez ensuite compiler la classe HelloServlet avec la commande suivante.

```
javac -classpath ../../../lib/servlet-api.jar HelloServlet.java
```

Déplacez le fichier *.class* obtenu dans *WEB-INF/classes/fr/enseirb/t2/servlet/*. Notez que la fin de l'arborescence correspond au nom du package de HelloServlet.

NB: mkdir -p permet de créer une arborescence de fichiers en une seule commande.

Redémarrez votre serveur (shutdown.sh puis startup.sh).

En tapant l'adresse `adresse-du-serveur/hello/world`, le serveur fera un appel à la méthode doGet, et affichera une page html avec le contenu ajouté par le PrintWriter.

Q1.3 Traitement des paramètres d'une requête GET
Il est possible de passer des paramètres dans une requête `HTTP GET`. L'url prend alors la forme suivante: `http://localhost/hello/world?name=toto`
Modifiez la classe HelloServlet afin que l'url ci-dessus affiche une page html contenant le texte:

"Hello toto."

Pour récupérer la valeur d'un paramètre, il faut utiliser :

```
String parameterValue = req.getParameter("parameterName");
```

Pour éviter de redémarrer le serveur à chaque fois qu'on fait une modification des fichiers .class, une solution est d'éditer le fichier `conf/context.xml` et d'ajouter un attribut à la balise `Context` :

```
<Context reloadable="true">
....
```

Redémarrez le serveur.

Lorsque vous ferez des modifications des fichiers `.class`, le message suivant devrait s'afficher dans les logs de Tomcat : `INFO: Reloading Context with name [/hello] is completed.`

Q1.4 Requêtes POST
Un servlet peut également traiter des requêtes POST. Ce type de requête peut être envoyé via un formulaire html. Ajoutez dans le dossier webapps/hello le fichier index.html suivant:

```
<html><body>
  <form action="world" method="POST">
    <h1>Please Fill the Registration Form</h1>
    Enter Your Name <input type="text" name="name"/>
    <input type="submit" value="send"/>
  </form>
</body></html>
```

Listing 1: index.html

Si vous ouvrez la page `http://adresse-du-serveur/hello`, vous verrez apparaître le formulaire. Si vous cliquez sur le bouton *send*, une exception HTTP sera renvoyée par le serveur, signifiant que la méthode POST n'est pas supportée par le serveur.

Il faut donc ajouter le support de la méthode POST à notre servlet. Ajoutez dans le fichier HelloWorld.java la méthode doPost:

```
@Override
protected void doPost(HttpServletRequest req, HttpServletResponse resp)
  throws ServletException, IOException {

  resp.setContentType("text/plain");
  // TODO: Traiter les parametres de la requete

}
```

Notez ici que le type de réponse est *text/plain*, ce qui signifie qu'il est possible de mettre du simple texte dans la réponse, au lieu de texte formaté en Html.
Modifiez le code pour afficher "Hello toto!" lorsque "toto" est entré dans le formulaire.

Q1.5 Headers

Un certain nombre d'informations sont disponibles dans les headers des requêtes, comme le nom de l'hôte, le navigateur du client, etc. La fonction suivante permet d'afficher les headers d'une requête dans une table html. Copiez-la dans votre servlet, puis testez-la avec un appel à cette fonction dans la méthode `doGet()`.

```
private void printHeaders(HttpServletRequest req, PrintWriter out) {
  out.println("<br><b>Request Method: </b>" +
    req.getMethod() + "<br>\n" +
    "<b>Request URI: </b>" + req.getRequestURI() + "<br>\n" +
    "<b>Request Protocol: </b>" + req.getProtocol() + "<br><br>\n" +
    "<table border=1 align=center>\n" + "<tr bgcolor=\"#FFAD00\">\n" +
    "<th>Header Name<th>Header Value");
  Enumeration headerNames = req.getHeaderNames();
  while(headerNames.hasMoreElements()) {
    String headerName = (String)headerNames.nextElement();
    out.println("<tr><td>" + headerName);
    out.println("  <td>" + req.getHeader(headerName));
  }
  out.println("</table>");
}
```

Listing 2: Affichage des headers

Pensez à ajouter l'import suivant:

```
import java.util.Enumeration;
```

Faites la même chose pour la méthode `doPost` (pensez à modifier le content-type de la réponse en text/html).

Vous remarquerez qu'un nouveau header *content-length* est apparu. Vous pouvez (et devez) donc lire le contenu de la requête et l'afficher grâce à l'instruction suivante.

```
String content = req.getReader().readLine();
```

Q1.6 Sessions

Il est possible de conserver des données le temps d'une session. Une session se termine lorsque le timeout expire ou lorsque le serveur s'arrête. Modifiez votre servlet pour que le nom de la dernière personne soit affiché si aucun nom n'est mis en paramètre.
La session se comporte comme un table de hachage, associant des clés de type `String` à des éléments de type `Object`.
Pour enregistrer un élément dans la session, utilisez la méthode :

```
req.getSession().setAttribute("name", userName);
```

Et pour récupérer la valeur d'un élément, utilisez :

```
String userName = (String) req.getSession().getAttribute("name");
```

Q1.7 Forward

Différentes parties de l'application peuvent traiter une même requête. Cette opération s'appelle le forwarding, et elle est transparente du point de vue du client. Le forwarding se fait grâce à la méthode suivante: `req.getRequestDispatcher("/servletToto").forward(req, resp);`

Créez un servlet qui *forward* les requêtes GET et/ou POST vers un servlet différent.

Q1.8 Redirection
Il est possible de rediriger le navigateur du client vers une autre URL, qui peut être externe au serveur (e.g. www.google.fr), graçe à la méthode sendRedirect de la classe HttpServletResponse.
Créez un formulaire qui redirige le navigateur du client sur Google, en ayant lancé la recherche avec les mots clés contenus dans le champ de saisie.

Q1.9 Identification
Mettre en place l'exemple d'identification par mot de passe vu en cours en utilisant des sessions et des redirections. Ajouter un formulaire de création de compte (*login, password*) et stocker les informations d'identification dans le contexte de l'application sous la forme d'une simple table.