Université ChouaibDoukkali
Faculté des Sciences
Département d'Informatique
EL Jadida

AU : 2019/2020
Filière SMI
Semestre 6

Technologie J2EE

Atelier 1 : Introduction aux Servlets

Définition

Une servlet (Server-side applet) est un programme Java utilisé pour étendre les fonctionnalités d'un serveur Web et pour accéder à des systèmes déjà existants. C'est une application :

- Exécutée côté serveur
- Utilisée pour générer du contenu dynamique
- Chargée dynamiquement quand elle est demandée.

Fonctionnement

- Lorsqu'une servlet est appelée par un client, la méthode **service()** est exécutée. Celle-ci est le principal pointd'entrée de toute servlet et accepte deux objets enparamètres:
  - l'objet **ServletRequest** encapsulant la requête du client, c'est-à-dire qu'il contient l'ensemble desparamètres passés à la servlet (informations sur l'environnement du client, cookies du client, URL demandée, ...)
  - l'objet **ServletResponse** permettant de renvoyer une réponse au client (envoyer des informations aunavigateur). Il est ainsi possible de créer des en-têtes HTTP (headers), d'envoyer des cookies aunavigateur du client, ...
- Afin de développer une servlet fonctionnant avec leprotocole HTTP, il suffit de créer une classe étendant**HttpServlet** (qui implémente elle-même l'interface Servlet).
- La classe **HttpServlet** (dérivant de **GenericServlet**) permet de fournir une implémentation de l'interface Servletspécifique à HTTP. La classe **HttpServlet** surcharge laméthode service en lisant la méthode HTTP utilisée par leclient, puis en redirigeant la requête vers une méthodeappropriée.
- Les deux principales méthodes du protocole HTTP étant **GET** et **POST**, il suffit de surcharger la méthode adéquate afin de traiter la requête; Ainsidonc:
  - Si la méthode utilisée est **GET**, il suffit de redéfinir la méthode public **voiddoGet(HttpServletRequest req, HttpServletResponseres);**
  - Si la méthode utilisée est POST, il suffit de redéfinir la méthode public **voiddoPost(HttpServletRequest req, HttpServletResponse res);**

Exemple

Voici un exemple simple de servlet dont le seul but est d'afficher du texte sur le navigateur du client :

```
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*
public class PremiereServlet extends HttpServlet {
public void init() {
}
```

```
public void doGet(HttpServletRequest req, HttpServletResponse res)throws
ServletException, IOException {
        res.setContentType("text/html");
        PrintWriter out = res.getWriter();
        out.println("<HTML>");
        out.println("<HEAD><TITLE> Titre </TITLE></HEAD>");
        out.println("<BODY>");
        out.println("Ma première servlet");
        out.println("</BODY>");
        out.println("</HTML>");
        out.close();
        }
}
```

La première étape consiste àimporter les packages nécessaires àla création de la servlet, il faut donc

```
importer javax.servlet,
javax.servlet.http et javax.io
```

Afin de mettre en place l'interface Servlet nécessaire au conteneur de servlet, la classe **HttpServlet** a été étendue

```
public class PremiereServlet extendsHttpServlet {
}
```

Lorsque la servlet est instanciée, il peut être intéressant d'effectuer des opérations qui seront utiles tout au long du cycle de vie de la servlet (par exemple se connecter à une base de données, ouvrir un fichier, ...).
Pour ce faire, il s'agit de surcharger la méthode **init()** de la servlet et de définir les opérations d'initialisation.

```
public void init() {}
```

A chaque requête, la méthode **service()** est invoquée. Celle-ci détermine le type de requête dont ils'agit, puis transmet la requête et la réponse à la méthode adéquate (**doGet() ou doPost()**). Dans notre cas, on ne s'intéressequ'à la méthode **GET**, c'est la raison pour laquelle la méthode **doGet()** a été surchargée

```
public void doGet(HttpServletRequest req, HttpServletResponse res)throws
ServletException, IOException {
}
```

L'objet **HttpServletRequest** permet de connaître les éventuels paramètres passés à la servlet (dans le cas d'un formulaire HTML par exemple), mais l'exemple ci-dessus n'en a pas l'utilité.
Par contre l'objet **HttpServletResponse** permet de renvoyer une page à l'utilisateur.
La première étape consiste à définir le type de données qui vont être envoyées au client. Généralement il s'agit d'une page HTML, la méthode **setContentType()** de l'objet**HttpServletResponse** doit donc prendre comme paramètre le type MIME associé au format HTML (text/html) :

```
res.setContentType("text/html");
```

Ensuite la création d'un objet **PrintWriter** grâce à la méthode **getWriter()** de l'objet**HttpServletResponse**permet d'envoyer du texte formatté au navigateur(pour envoyer un flot de données, il faudrait utiliserla méthode **getOutputStream()**)

```
PrintWriter out = res.getWriter();
```

Enfin il faut utiliser la méthode **println()** de l'objet **PrintWriter** afin d'envoyer les données textuelles au navigateur, puis fermer l'objet **PrintWriter** lorsqu'iln'est plus utile avec sa méthode close()

```
out.println("<HTML>");
```

```
out.println("<HEAD><TITLE> Titre
</TITLE></HEAD>");
out.println("<BODY>");
out.println("Ma première servlet");
out.println("</BODY>");
out.println("</HTML>");
out.close();
```