Java - Servlets

Code: java-servl

Originaux

url: http://tecfa.unige.ch/guides/tie/html/java-servl/java-servl.html

url: http://tecfa.unige.ch/guides/tie/pdf/files/java-servl.pdf

- Version: 0.2 (modifié le 17/1/01)

Auteurs et version

- Daniel K. Schneider - Vivian Synteta

Prérequis

- Java de Base

  Module technique précédent: java-intro
- Avoir une idée du standard "CGI"

  Module technique précédent: cgi-intro
- Pages JSP, pas obligatoire mais très utile

  Module technique précédent: java-jsp

Exercices / Activités

Module d'exercices: act-servlets

Objectifs

- Servlets de base
- Simple traitement de requêtes

1. Table des matières détaillée

2. Introduction aux Servlets

2.1 Servlets: Informations générales

A. Tutoriels:

- Sun Tutorial: http://tecfa.unige.ch/guides/java/tutorial/servlets/index.html (local) or http://java.sun.com/docs/books/tutorial/servlets/index.html
- Stefan Zeigers: http://www.novocode.com/doc/servlet-essentials/
- Voir la page Java pour d'autres: http://tecfa.unige.ch/guides/java/pointers.html

B. Exemples:

- A Tecfa: http://tecfa.unige.ch:/guides/java/staf2x/ex/servlets/

C. Les packages javax.* nécessaires pour compiler un servlet

Il existe 2 archives (les deux marchent avec notre serveur):

- veille version générique: jsdk.jar
- version qui vient avec tomcat: appelée souvent servlet.jar ou servlet2_x.jar

Documentation et package pour le server Tomcat sur tecfa.unige.ch

url: http://tecfa.unige.ch/guides/tomcat/

- Copie de l'archive servlet.jar: http://tecfa.unige.ch/guides/java/classes/servlet.jar
- Script qui inialise l'environnement sous Unix (+ MySQL + xerces)

```
source /local/env/java12-sql-xml-ser.csh
```

Le vieux package JSDK (qui marche avec presque tous les serveurs)

url: http://java.sun.com/products/servlet

- Copie locale du package: /local/java/classes/JSDK2.0/lib/jsdk.jar

  url: API local: http://tecfa2.unige.ch/guides/java/jsdk2/doc/apidoc/packages.html
- Script qui inialise l'environnement sous Unix (+ MySQL + XML/IBM)

```
source /local/env/java-sql-xml-ser.csh
```

Installation d'un servlet

- Avec le serveur Tomcat il est possible d'allouer des "contextes web application" aux utilisateurs. Un servlet var dans le répertoire <context>/WEB-INF/classes
- Sur d'autres serveurs, un servlet doit être installé par l'administrateur du serveur (comme les cgi)

D. Le servlet "Life Cycle"

- Une fois lancé, un servlet vit jusqu'à ce qu'il soit détruit par le serveur
- Cela rend relativement facile la programmation d'applis groupware

2.2 Architecture du Package Servlet

- Voir la Class hierarchy: /guides/tomcat/docs/api/overview-summary.html
- Abstraction centrale: l'interface javax.servlet.Servlet
  - déclare, mais n'implémente pas des méthodes pour gérer des servlets et pour communiquer avec les clients

Implémentations disponibles de Servlet à utiliser

- javax.servlet.GenericServlet (Internet générique)
- javax.servlet.http.HttpServlet (Server WWW)

Exemple:

```
public class HelloServlet extends HttpServlet {
    // a method ....
}
```

A. L'interaction avec un client

Quand un servlet accepte une requête d'une client, il reçoit deux objets:

1. un objet qui implémente ServletRequest et qui encapsule la communication du client vers le serveur (ServletInputStream ou HttpServletRequest)
2. un objet qui qui implémente ServletResponse et qui encapsule la communication du servlet vers le client (ServletOutStream ou HttpServletResponse)

Votre code:

- Doit implémenter des méthodes de la classe GenericServlet ou HttpServlet qui utilisent ces objets

  url: http://tecfa.unige.ch/guides/tomcat/docs/api/javax/servlet/http/HttpServlet.html
- Il faut respecter les interfaces pour ces méthodes (voir plus tard)

  Exemples à titre d'information:

```
protected void doGet(HttpServletRequest request, HttpServletResponse response)
 throws ServletException, IOException {
.....
}
```

```
     out = response.getWriter();
```

3. Hello avec un Servlet

Exemple 3-1: Hello World avec un Java Servlet

url: http://tecfa.unige.ch/guides/java/staf2x/ex/servlet/HelloServlet

url: http://tecfa.unige.ch/guides/java/staf2x/ex/WEB-INF/classes/HelloServlet.java

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

public class HelloServlet extends HttpServlet {
    // Handle the Get Method

protected void doGet(HttpServletRequest request, HttpServletResponse response)
 throws ServletException, IOException {
     PrintWriter out ;
     // Content type and other HTTP response header fields
     response.setContentType("text/html");

     // get a print writer, to write out the data of the response
     out = response.getWriter();
     // Write ...
     String header = "<HTML><HEAD><TITLE>Hello Client!</TITLE></HEAD>";
     out.println(header + "<body>");
     out.println("<h1>Hello Client!</h1> How is life at Tecfa ?");
     out.println("</BODY></HTML>");
     out.close();
```

```
    }

    public String getServletInfo() {
 return "HelloClientServlet by DKS";
    }
}
```

Packages à importer:

```
Java:          import java.io.*;
Duservlet kit: import javax.servlet.*;
               import javax.servlet.http.*;
```

Création d'une classe de type HttpServlet (un peu comme un applet)

```
public class HelloServlet extends HttpServlet { ... }
```

Implémentation de la méthode doGet

```
protected void doGet(HttpServletRequest request,HttpServletResponse response)
     throws ServletException, IOException { .... }
```

Le servlet passe deux variables à cette méthode:

1. un objet qui encapsule la communication du *client vers le serveur*
   - classe: HttpServletRequest)
2. un objet qui encapsule la communication du *servlet vers le client*:
   - classe: HttpServletResponse

Méthodes utilisées des objets request et response:

- Lecture des données de la requête
  - pas nécessaire dans cet exemple .....
- Gestion de la réponse (avec l'objet response)
  - définir les headers de la réponse (au moins content type et encoding)

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

  - accéder au output ou writer stream

```
out = response.getWriter();
```

  - écrire les données

```
out.println( ....)
```

  - fermer l'output stream à la fin (IMPORTANT, sinon votre servlet coincera)

```
out.close()
```

4. Analyse de requêtes POST avec un servlet

Exemple 4-1: Simple calcul (Formulaire + Servlet)

url: http://tecfa.unige.ch/guides/java/staf2x/ex/servlets/simple-calcul.html

url: source: /guides/java/staf2x/ex/WEB-INF/classes/SimpleCalculServlet.java

- Même logique que pour l'exemple "Simple calcul (Formulaire + JSP)"

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

public class SimpleCalculServlet extends HttpServlet {

    public void  doPost ( HttpServletRequest  req, HttpServletResponse  res )
     throws ServletException, IOException {

     res.setContentType ( "text/html" );
     PrintWriter  out = res.getWriter ( );
     try {
         String title = "Simple Calcul Java Servlet";
         ........
         String  choice = req.getParameter ( "choice" );
         String  choice2 = req.getParameter ( "choice2" );
         ..........
         int score = Integer.parseInt(choice) + Integer.parseInt(choice2);

         out.print("<h3>Votre score est de " + score + "</h3>");
         ....
```

- Ce servlet étend classe HttpSevlet et implémente la méthode doPost
  - doPost fonctionne comme doGet (dans l'exemple précédent)

Exemple 4-2: Simple Calcul avec persistance

url: http://tecfa.unige.ch/guides/java/staf2x/ex/servlets/simple-calcul2.html
url: Source JAVA: /guides/java/staf2x/ex/WEB-INF/classes/SimpleCalculServlet2.java

- Même principe, mais le servlet mémorise de l'information

```
public class SimpleCalculServlet2 extends HttpServlet {

int totScore;
int nClicks;
int moyScore;
  public void  doPost (HttpServletRequest req, HttpServletResponse res)
     throws ServletException, IOException {

    ........

     nClicks++;
     totScore = totScore + score;
     moyScore = totScore / nClicks;

    out.println("<h3>Le score moyen est " + moyScore + " pour " + nClicks + "
participants</h3>");

    ..........
```

- Note: dans une application "industrial strength" il faudrait sauver ces variables sur le serveur.

Exemple 4-3: Simple calcul contrôlant GET (Formulaire + Servlet)

url: http://tecfa.unige.ch/guides/java/staf2x/ex/servlet/SimpleCalculServletGet
url: source: /guides/java/staf2x/ex/WEB-INF/classes/SimpleCalculServletGET.java

- Dans l'exemple 4-1 "Simple calcul (Formulaire + Servlet)" [11] on a travaillé avec POST, on aurait pu utiliser "GET" (laissé au lecteur)
- Si vous avez un servlet qui traite une requête POST venant d'une page HTML, il est utile d'ajouter une méthode GET qui renvoye les utilisateurs vers le formulaire.

```
   protected void doGet(HttpServletRequest request,
                        HttpServletResponse response)
        throws ServletException, IOException
    {
        PrintWriter out ;
        response.setContentType("text/html");
        out = response.getWriter();
        String header = "<HTML><HEAD><TITLE>Hello Client!</TITLE></HEAD>";
        out.println(header + "<body>");
        out.println("<h1>Sorry, get is not supported, try use the <a href='/
guides/java/staf2x/ex/servlets/simple-calcul.html'>form</a> please !</h1>");
        out.println("</BODY></HTML>");
        out.close();
    }
```

Exemple 4-4: Simple calcul avec all-in-one servlet

url: http://tecfa.unige.ch/guides/java/staf2x/ex/servlet/SimpleCalculServletAllinOne
url: source: /guides/java/staf2x/ex/WEB-INF/classes/SimpleCalculServletAllinOne.java

- Il est bien sûr possible de faire un all-in-1 servlet (sans page HTML)
- Typiquement cela se ferait plutôt avec une page JSP
- La logique est la même que pour l'exemple 4-3 "Simple calcul contrôlant GET (Formulaire + Servlet)" [13].
- On ajoute simplement une méthode GET qui affiche le formulaire.

```
String html = "<HTML>" + "<HEAD>" +
        "<TITLE>Un simple test avec un All-in-One Java Servlet </TITLE>" +
.....
 "<form action='SimpleCalculServletAllinOne' method=post>" +
.....
 "<input type='submit' value='Voir le résultat!'>" + "</form><hr>" + .. ;

 protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException
    {
        PrintWriter out ;
        response.setContentType("text/html");
        out = response.getWriter();
        out.print(html);
        out.close();
    }
```

5. Server deployment

La Java Servlet Specification Version 2.3 définit une méthodes uniforme pour installer des servlets dans un serveur Java.

Voici brièvement la logique pour le serveur Tomcat

5.1 Deployment au niveau server administration

1. L'administrateur du serveur définit un contexte pour une "Web application" (webapp) pour chaque projet.
   - Dans Tomcat cela se fait dans le fichier tomcat/conf/server.xml

```
<Context path="/staf/staf-e/paraskev"
    docBase="/web/staf/staf-e/paraskev"
    debug="0"
    reloadable="true">
</Context>
```

2. Dans un install Apache/Tomcat/mod_jk comme à TECFA:
   - on ajoute en plus dans le fichier conf/mod_jk.conf:

```
JkMount /staf/staf-e/paraskev/servlet/* ajp13
<Location "/staf/staf-e/paraskev/WEB-INF/">
    AllowOverride None
    deny from all
</Location>
```

5.2 Installation par le développeur dans son contexte

1. Pour qu'un URL de type suivant marche
   `http://tecfa.unige.ch/staf/staf-e/paraskev/servlet/vivianSqltables`
   - il faut demander à l'administrateur un context (voir page précédante) pour
     `http://tecfa.unige.ch/staf/staf-e/paraskev/`
     `A TECFA cela correspond à: /web/staf/staf-e/paraskev/`
2. Créer un sous-répertoire "WEB-INF" et dedans un sous-rep "classes"
   `/web/staf/staf-e/paraskev/WEB-INF`
   `/web/staf/staf-e/paraskev/WEB-INF/classes`
   - mettre vos classes dans WEB-INF / classes
3. Ceci est une configuration minimaliste !
   - on peut également y installer des packages dans
     `../WEB-INF/lib`
   - définir des comportements, paramètres, etc. etc. dans un fichier:
     `../WEB-INF/web.xml`

Pour aller plus loin:

- Voir la Servlet 2.3 specification (chapitres 9 et 11 surtout)
  url: http://java.sun.com/aboutJava/communityprocess/first/jsr053/index.html
- La configuration des exemples JSP qui viennent avec Tomcat (ou Cocoon)
  url: http://tecfa.unige.ch/guides/tomcat/examples/jsp/
  `(web/guides/tomcat/examples/jsp/WEB-INF depuis UNIX !)`