Servlet

A.-E. Ben Salem

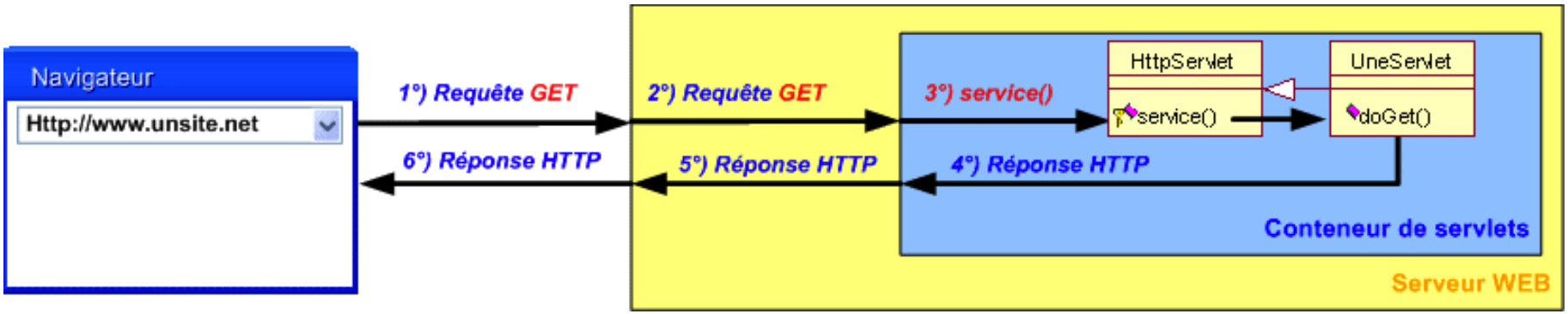
LRDE and LIP6

17 Octobre 2011

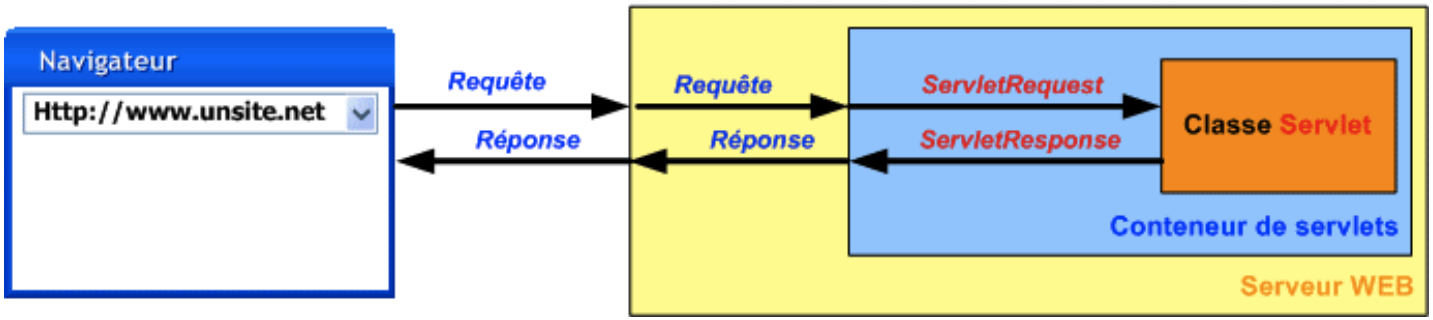
Plan

Servlet

- une Servlet est :
  - Code Java permettant la génération (dynamique) de pages HTML
  - c'est une Classe Java héritant de la classe HttpServlet (bibliothèque J2EE)
  - et redéfinissant les méthodes (doGet, doPost ...) afin de construire la réponse à envoyer au client
- Donc, en général pour développer une Servlet, **il suffit d'écrire la méthode doGet(...) et/ou doPost(...)**



Servlet




```
import java.io.*; import java.text.*; import java.util.*;

import javax.servlet.*;
import javax.servlet.http.*;

public class HelloWorldExample extends HttpServlet {
   public void doGet(HttpServletRequest request, HttpServletResponse response)
      throws IOException, ServletException
   {
   response.setContentType("text/html");
   PrintWriter out = response.getWriter();

   out.println("<html>"); out.println("<head>");
   ...
   ...
   out.println("</body>");
   out.println("</html>");
   }
}
```

contient les données envoyées par le client

écriture du début de page html

contient la spécif pour l'envoi (dont le flux de sortie)

Ecriture d'une servlet

1. Ecrire une nouvelle classe Java, héritant de `HTTPServlet`
2. Redéfinir éventuellement `init()` et `destroy()`
3. Genérer la Réponse selon le type de la Requête en redéfinissant l'une des méthodes :
   - doGet(HttpServletRequest req, HttpServletResponse resp)
   - doPost(HttpServletRequest req, HttpServletResponse resp)
   - HttpServletRequest : permet de manipuler la Requête reçue
   - HttpServletResponse : permet de générer la réponse à envoyer au client

Ecriture d'une servlet

- Récupérer les informations de la Requête
- Utiliser les méthodes de HttpServletRequest :
  - `String getParameter(String key)`
  - `String[] getParameterValues(String key)`
  - `Enumeration getParameterNames()`

Ecriture d'une servlet

- Construire la réponse
- méthodes de HttpServletResponse :
  - `setContentType(String)` : fixe le type MIME de la réponse HTTP, par exemple "text/html".
  - `PrintWriter getWriter()` : Créer un objet de sortie, qui contiendra le corps de la réponse.
    Pour du contenu de type texte. A remplir avec println().
  - `ServletOutputStream getOutputStrem()` : A réserver pour produire un contenu binaire.

Ecriture d'une servlet

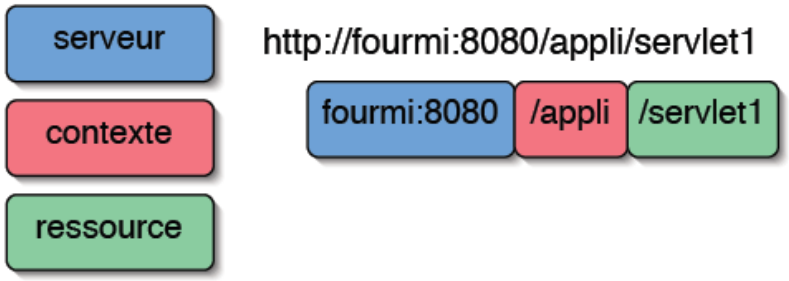
Exemple :

```
public class myClass extends HttpServlet {
        public void doGet ( HttpServletRequest request , HttpServletResponse
        response ) throws ServletException , IOException
{
                response.setContentType("text/html") ;
                PrintWriter out = response.getWriter() ;
                out.println("<html>" +"<head><title >Hello </ title ></head>" ) ;
                out.println("<body>") ;
                String nom = request.getParameter("monNom'');
                out.println(" < h2 > Hello" + nom + " < /h2 > ");

                out.println("</body></html>") ;
                out.close() ;
}
```

Déploiement et exécution d'une Servlet

Chaque Servlet est associée à un nom de ressource (ou plusieurs) défini dans web.xml. Voici un Exemple de Déploiement de la Servlet `MyClass.class` :




```
|-- WEB-INF
|    |-- classes
|    |    '-- MyClass.class
|    '-- web.xml
```

La ressource `/servlet1` est associée à la servlet `MyClass.class` dans

"web.xml" :

```
<web-app>
        <servlet>
                <servlet-name>myServlet1</servlet-name>
                <servlet-class>myClass</servlet-class>
        </servlet>
        <servlet-mapping>
                <servlet-name>myServlet1</servlet-name>
                <url-pattern>/servlet1</url-pattern>
        </servlet-mapping>
```