INE3 : Communication P2P en temps réel sur le web

TP1 : Les servlets http

Introduction

L'objectif de ce TP est de manipuler le protocole HTTP en utilisant la plateforme JEE surtout sa composante servlet. Nous allons commencer par présenter le fondement théorique de la plateforme JEE. Ensuite, nous abordons la partie pratique en développant une petite application à base de HTML et HttpServlet avec le conteneur web Tomcat et l'IDE Eclipse EE. Enfin, des extensions sont demandées pour couvrir le reste des aspects des servlets et de la couche présentation.

I- Partie théorique

Cette partie donne un aperçu sur le développement web en JEE.

La plateforme JEE

Les développements JEE reposent sur un découpage de leurs applications en couches ou tiers dont les principales sont la couche présentation (tiers Web), la couche métier (tiers Métier ou tiers Business), la couche service (tiers service), et la couche stockage des informations (tiers Enterprise Information System). Autrement dit, les applications sont découpées en plusieurs composants réalisant des fonctionnalités spécifiques et installés sur une machine serveur ou sur plusieurs tiers distribués. Les composants JEE sont des unités autonomes assemblées dans une application JEE composée de classes Java et de fichiers XML, et communiquant avec d'autres composants. De même, le code métier écrit est indépendant de la couche de présentation, ce qui est utile pour changer cette dernière ou pour l'afficher sur divers supports. Les composants Clients ou tiers Client sont des applications clientes (logiciel installé en local ou navigateur Web ou Applets) s'exécutant sur la machine des utilisateurs. Les composants Web ou tiers Web sont les technologies Servlets, JavaServer Pages et Java- Server Faces, etc.

Les composants métier ou tiers Métier sont des composants Enterprises JavaBeans (EJB) représentant la logique métier, s'exécutant sur le serveur Java EE et dialoguant avec la couche de stockage (EIS : Enterprise Information System). Les composants service sont utilisés pour exposer les fonctionnalités de la couche métier aux différents clients tout en découplant ces derniers de la complexité de la logique métier, et ce en utilisant les technologies REST ou SOAP.

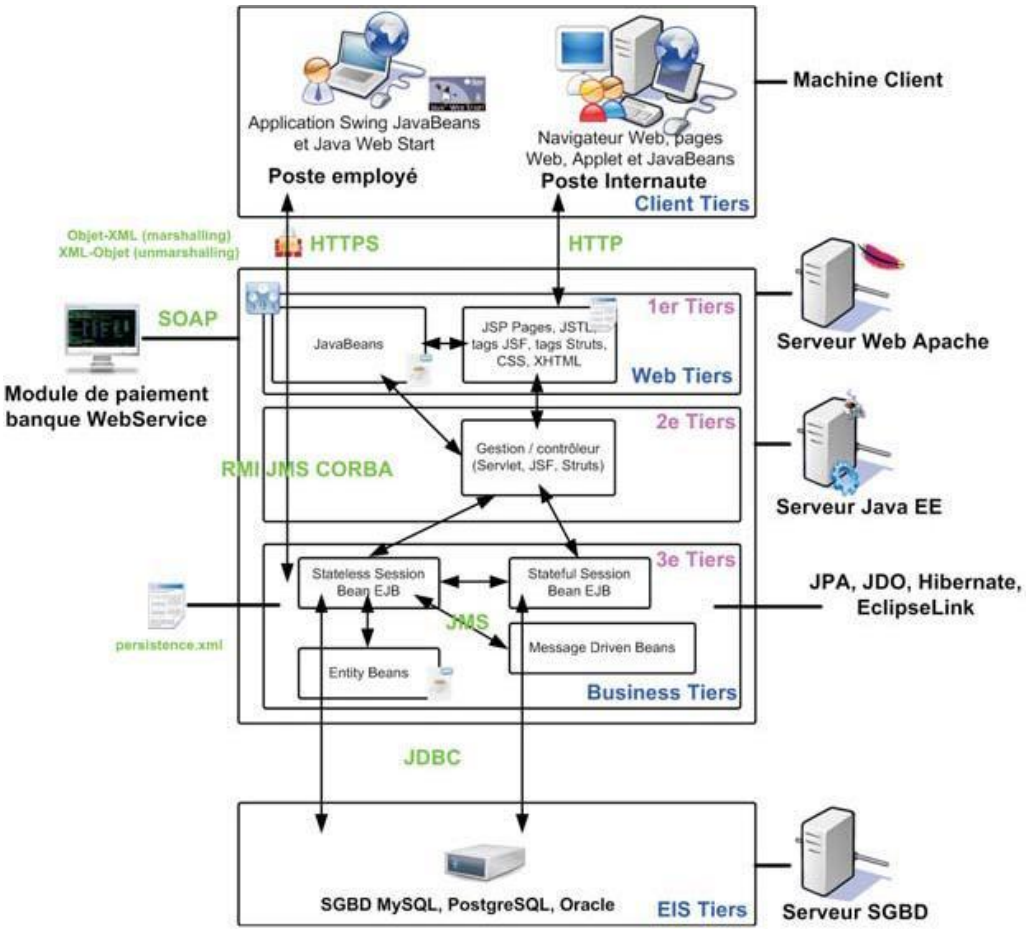

Figure 1 : Architecture d'une application JEE

La plate-forme Java EE propose trois types de clients : les clients Web, les Applets et les applications clientes riches type Java SE. Un client Web consiste en des pages Web de différents types (HTML, XHTML, XML, JavaScript ou autres) générées à partir de composants exécutés côté serveur dans un conteneur Web et capables de répondre aux requêtes HTTP en provenance du navigateur Internet. Ces programmes côté serveur sont représentés en Java EE par les Servlets, pages JSP et JSF. Ces programmes serveur ne réalisent en général pas directement les accès aux bases de données mais s'occupent de la logique applicative. Les Applets sont des interfaces graphiques Java SE exécutées dans un navigateur Web. Ces applications utilisent une interface graphique évoluée de type SWING et sont exécutées dans une machine virtuelle Java installée dans le navigateur. Cependant cette technique est plus contraignante à maintenir, requiert des accès et des droits pour la sécurité et nécessite un plug-in pour les navigateurs. Les clients Web sont donc préférables pour la création du tiers Client. Une application de type client est un logiciel riche, qui s'exécute sur la machine du client et fournit un ensemble de services aux utilisateurs par l'intermédiaire d'une interface graphique évoluée encore appelée Graphical User Interface (GUI).

Les clients et le serveur dialoguent ensemble par l'intermédiaire d'un composant standardisé nommé JavaBean ou classe Plain Old Java Object (POJO). Un composant JavaBean peut être vu comme la plus petite unité de communication entre les couches ou tiers Java EE. Les composants JavaBeans sont des objets, instances de classes POJO, composés de propriétés et de leurs accesseurs associés pour accéder aux données. Les JavaBeans s'exécutent sur le serveur Java EE et dialoguent avec les clients ou avec la base de données (tiers EIS).

Les composants Web Java EE sont des Servlets et/ou des pages JavaServer Pages et/ou des JavaServer Faces. Les Servlets sont des classes Java, capables d'intercepter et de gérer les requêtes du protocole HTTP. Les pages JSP sont des documents textuels exécutés comme des Servlets apportant une solution simplifiée pour la programmation de pages Web. La technologie JavaServer Faces est construite à partir de Servlets et fournit un framework de développement pour accélérer la création d'applications Web.

Les composants métier ou tiers Métier représentent la couche business, avec les données du système, et sont de deux types :

- Les entités beans (entity bean ou bean entity) peuvent être exécutées par un conteneur léger (pas forcément un serveur Java EE) et permettent de réaliser la persistance des données à partir des JavaBeans et de Java Persistence API (JPA).
- Les Enterprise JavaBeans offrent des possibilités plus riches comme la gestion des transactions, les accès directs par clients riches ou encore la gestion automatique des sessions utilisateur, mais sont exécutés sur un conteneur lourd, c'est-à-dire compatible Java EE.
- La partie stockage des données, nommée tiers Enterprise Information System (EIS) est directement liée au tiers Métier et correspond dans la majorité des cas, aux systèmes de gestion de bases de données (Derby, MySQL, PostgreSQL, Oracle ou autres) ou à un ERP, ou n'importe quel système de stockage évolué.

Les conteneurs JEE

Les serveurs Java EE proposent plusieurs types de conteneurs (containers) pour chaque type de composant. Chaque conteneur a un rôle bien défini et offre un ensemble de services pour les développeurs tels que l'annuaire de nommage d'accès aux ressources : Java Naming and Directory Interface (JNDI), l'injection dynamique de ressources, la gestion des accès aux bases de données, la gestion de la sécurité, le paramétrage des transactions, etc.

Une application Java EE de type Web nécessite un conteneur Web pour son exécution alors qu'une application utilisant les EJB nécessite un conteneur EJB pour son exécution. Chaque conteneur propose un ensemble de services avec ses avantages et ses contraintes.

Anatomie d'un projet JEE

Toute application web JEE doit respecter une structure de dossiers standards, qui est définie dans les spécifications de la plate-forme, comme illustré sur la figure 2.

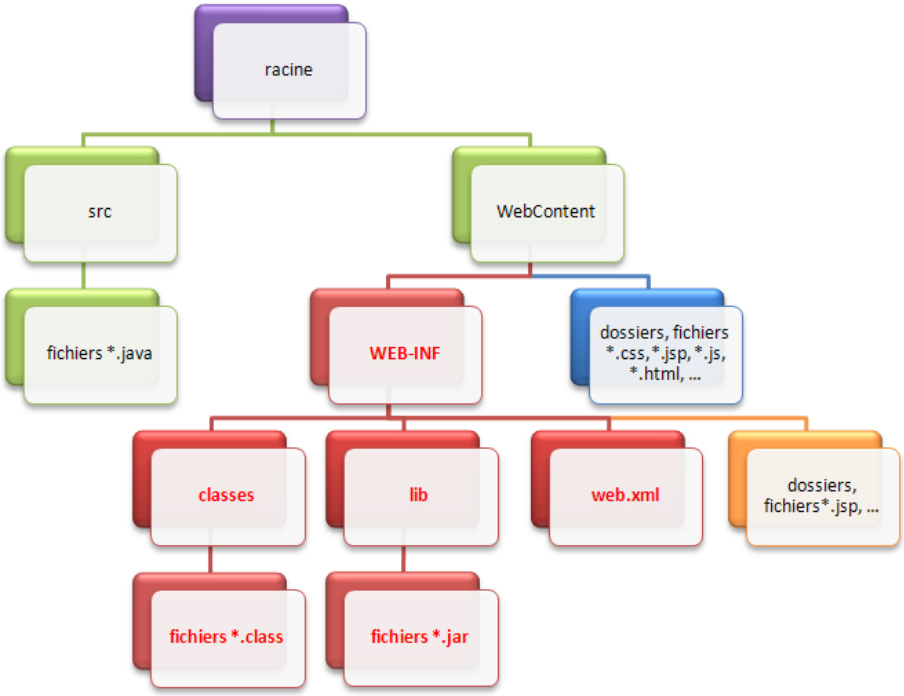

Figure 2 : Structure d'une application web JEE.

- La racine de l'application est le dossier qui porte le nom de votre projet et qui contient l'intégralité des dossiers et fichiers de l'application.
- Le dossier **src** qui contient le code source de vos classes (les fichiers .java).
- Le dossier nommé **WEB-INF** est un dossier spécial. Il doit obligatoirement exister et être placé juste sous la racine de l'application.
- Le fichier de configuration de l'application (web.xml).
- Un dossier nommé **classes**, qui contient à son tour les classes compilées (fichiers .class).
- Un dossier nommé **lib**, qui contient à son tour les bibliothèques nécessaires au projet (archives .jar).
- Les fichiers et dossiers personnels placés directement sous la racine sont publics et donc accessibles directement par le client via leurs URL.

- Les fichiers et dossiers personnels placés sous le répertoire **WEB-INF** sont privés et ne sont donc pas accessibles directement par le client.

II- Partie pratique

Cette section permet de mettre en pratique les concepts décrits précédemment. Nous allons développer une application web permettant à un utilisateur de saisir son nom et d'avoir en réponse un message sur son navigateur l'informant combien il a gagné virtuellement en loterie. Les technologies utilisées sont le HTML, le HTTP et les servlets.

Pour ce faire, vous devez d'abord installer un IDE pour JEE tel que Eclipe EE ou NetBeans ainsi qu'un conteneur web tel que Tomcat. Ensuite, vous devez créer un projet web dynamique contenant :

- Un formulaire HTML pour saisir le nom de l'utilisateur et soumettre la requête en utilisant les commandes http telles que GET et POST.
- Une classe servlet héritant de la classe prédéfinie HttpServlet pour implémenter les méthodes de traitement des requêtes de l'utilisateur telles que doGet() et doPost().
- Un fichier de description de l'application web.xml contenant les composants de l'application ainsi que leurs configurations.

Voici le contenu de base des différents fichiers demandés :

Le formulaire HTML

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Welcome to this amazing web application</title>
</head>
<body>
<CENTER><H1> Tenter votre chance a cette lotterie virtuelle!</H1></CENTER>
<FORM ACTION = "./hello" METHOD="post">
   Votre nom svp:  <INPUT TYPE = "TEXT" NAME = "nom"><BR>
   <BR>
   <CENTER><INPUT TYPE = "SUBMIT" NAME = "Submit Query"></CENTER>
</FORM>
</body>
</html>
```

La servlet

```
package exple1;

import java.io.*;
```

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


@SuppressWarnings("serial")
public class GreetingServlet extends HttpServlet {
  @Override
  public void doGet(HttpServletRequest request,HttpServletResponse
response)
      throws ServletException, IOException {
    response.setContentType("text/html");
    PrintWriter out = response.getWriter();
    String docType =
      "<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.0 " +
      "Transitional//EN\">\n";
    String title = null;
    String votreNom = null;
    String nomPrenom = "Anonymous";
    votreNom = request.getParameter("nom");
    if (votreNom != null)
      nomPrenom = votreNom.toUpperCase();
    title =  "<H1>Greetings " + nomPrenom + "!</H1>\n";
    out.println(docType +
                "<HTML>\n" +
                "<HEAD><TITLE>Greetings Servlet</TITLE></HEAD>\n" +
                "<BODY BGCOLOR=\"#FDF5E6\">\n" +
                title +
                "</BODY></HTML>");
    out.println("Vous avez gagne: "+ Math.random()* 10);
    out.println(" millions de dollars!");
  }

  public void doPost(HttpServletRequest request,
HttpServletResponse response)
              throws ServletException, IOException {
     doGet(request,response);
  }

}// end class Servlet
```

Le descriptif web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://java.sun.com/xml/ns/javaee"
xmlns:web="http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd" id="WebApp_ID" version="2.5">
  <display-name>websecurity1</display-name>
```

```
  <welcome-file-list>
    <welcome-file>greetings.html</welcome-file>
    <welcome-file>greetings.jsp</welcome-file>
    </welcome-file-list>
  <servlet>
    <description></description>
    <display-name>GreetingServlet</display-name>
    <servlet-name>GreetingServlet</servlet-name>
    <servlet-class>exple1.GreetingServlet</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>GreetingServlet</servlet-name>
    <url-pattern>/hello</url-pattern>
  </servlet-mapping>
</web-app>
```

III- Extensions à faire

Pour apprendre davantage sur les applications JEE surtout concernant la couche présentation, je vous demande de réaliser les extensions suivantes:

- Refaire l'application développée précédemment en utilisant une base de données MySQL pour stocker et consulter les gains de la loterie des utilisateurs en se basant sur la technologie JDBC.
- Refaire l'application pour empêcher un utilisateur d'accéder aux services de l'application (c'est-à-dire, la servlet) si l'utilisateur ne possède pas le rôle "tomcat".
- Refaire l'application en utilisant les filtres pour empêcher un utilisateur d'accéder aux services de l'application (c'est-à-dire, la servlet) si le nom introduit figure parmi ceux d'une liste noire (blacklist) déjà établie et configurée dans l'application.
- Refaire l'application en utilisant la technologie JSP au lieu de HTML.
- Refaire l'application en implémentant et en utilisant les autres méthodes de HttpServlet.
- Refaire l'application en intégrant la gestion des erreurs (4xx, 5xx, …) par configuration dans web.xml.
- Dresser une synthèse de tous les concepts pratiqués dans ce TP.

Bon apprentissage !