Département de génie logiciel et des TI

LOG-720 ARCHITECTURE DISTRIBUÉE ORIENTÉE OBJET

Tutoriel JEE: Servlets, JSP, Eclipse WTP, m2eclipse, Tomcat, PostgreSQL, Jetty

---

Version initiale : Nicholas Côté, étudiant

Révision : Roger Champagne, professeur

Révision : 2012-Oct-12

Table des matières

Préalables

1. Assurez-vous que la variable d'environnement **`JAVA_HOME`** pointe vers votre JDK (ex: **`C:\Oracle\Java\jdk`**)
2. Assurez-vous d'avoir une installation d'Apache Maven.
3. Assurez-vous d'avoir une installation d'eclipse pour Java édition entreprise (EE), qui inclut plusieurs plugiciels utiles pour le développement JEE. Notez que l'utilisation d'Eclipse n'est pas obligatoire, mais c'est le seul IDE supporté pour les besoins de ce cours.
4. Assurez-vous d'avoir le plugiciel eclipse m2e (m2eclipse) si vous souhaitez utilisez des projets maven dans eclipse.
5. Assurez-vous d'avoir le plugiciel eclipse d'intégration m2e-wtp si vous souhaitez utiliser des projets web dynamiques (JEE) maven dans eclipse.
6. Assurez-vous d'avoir une installation d'Apache Tomcat. Si vous installez Tomcat sur votre propre ordinateur, il est suggéré de prendre une version ZIP et de simplement la désarchiver. L'installateur EXE installe forcément Tomcat comme un service, ce qui complique un peu les choses si vous n'avez pas les privilèges d'administrateur sur le poste concerné.

Voir le site web du cours (page "Logiciels") pour les liens vers les versions courantes de tous les logiciels requis.

Configuration de Tomcat sur les postes du laboratoire

Afin d'assurer une installation stable de Tomcat sur les postes du laboratoire, tous les répertoires de Tomcat sont accessibles en lecture seulement. Or, Tomcat doit pouvoir écrire dans certains de ces répertoires dans le cadre de son fonctionnement normal.

Pour pouvoir utiliser Tomcat sur les postes au laboratoire, il suffit de définir la variable d'environnement **`CATALINA_BASE`** dans Windows. Cette variable doit pointer vers un répertoire "de travail" accessible en lecture/écriture (par exemple **`j:\tomcat`**) où vous copiez une partie de l'arborescence de Tomcat. À partir de ce moment, à chaque fois que vous démarrez Tomcat, ce sont vos répertoires "de travail" qui sont utilisés. La seule chose de la distribution originale qui continue à être utilisée est le répertoire **`bin`**, qui contient les scripts de démarrage et d'arrêt de Tomcat, entre autres.

1. Déterminez où sera votre répertoire de travail (ex: **`j:\tomcat`**).
2. Ajoutez une variable d'environnement **`CATALINA_BASE`** qui pointe vers votre répertoire de travail (sous Windows, via Paramètres →Panneau de configuration →Système →Avancé →Variables d'environnement, dans la section "usager")
3. Copiez les répertoires suivants de la distribution de Tomcat installée sur les postes (**`C:\apache-tomcat-X.Y.Z`**, ci-après identifié par **`CATALINA_HOME`**) à votre répertoire de travail:
    a. conf
    b. lib
    c. temp: ce répertoire est vide mais doit être présent pour que Tomcat fonctionne);
    d. webapps: ce répertoire contient un certain nombre d'exemples d'applications web utiles. Si vous ne souhaitez pas le copier au complet, copier au moins l'application ROOT, qui est utile pour déboguer l'installation de Tomcat
4. Démarrez Tomcat à l'aide du script "startup" approprié pour votre plate-forme dans **`CATALINA_HOME\bin`**. Vérifiez qu'il n'y a pas d'erreur grave dans la chaine de messages et que celle-ci se termine par quelque chose qui ressemble à:

```
14-Oct-2011 8:12:24 AM org.apache.catalina.startup.Catalina start
INFO: Server startup in 840 ms
```

5. Pointez votre fureteur vers l'adresse suivante: http://localhost:8080/. Si vous avez au moins inclus **`webapps/ROOT`** dans votre répertoire de travail, vous devriez avoir la page d'accueil de Tomcat, indiquant que votre configuration est à point.

Accès à la base de données

Comme ce laboratoire utilise une base de données, il est important de la configurer avant d'aller plus loin.

Ce cours utilise le SGBD PostgreSQL, qui est installé sur le serveur dédié au cours. Il existe deux façons d'accéder à ce SGBD, soit:

1. à la ligne de commande (**`psql`**): sur les postes du laboratoire, les outils sont installés dans le répertoire **`C:\Program Files\pgAdmin III\1.12`**, qui doit être sur votre PATH;
2. via un outil graphique (pgadmin): accessible via Démarrer → Programmes →pgadmin III →pgadmin

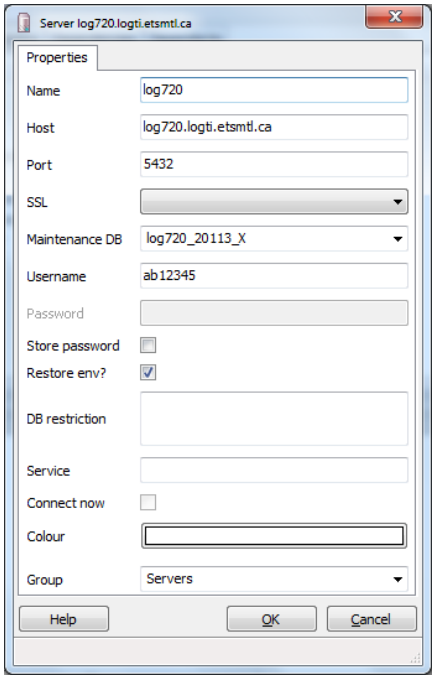
Voici les informations utiles pour accéder au SGBD, peu importe le mode choisi:

- hôte: **`log720.logti.etsmtl.ca`**
- nom de votre BD: **`log720_20123_X`**, où X est votre numéro d'équipe
- port : **`5432`**

À la ligne de commande, la commande à invoquer pour se connecter à votre BD serait la suivante, en remplaçant l'information surlignée en jaune par votre numéro d'équipe et votre code universel **en minuscules** (après quoi on vous demandera votre mot de passe) :

```
psql -h log720.logti.etsmtl.ca log720_20123_X ab12345
```

Si vous choisissez d'utiliser **`pgadmin`**, voici comment saisir les informations pour ajouter le serveur à votre configuration :



NOTE IMPORTANTE: un mot de passe générique, commun à tout le groupe, est initialement utilisé. Ce mot de passe vous sera communiqué verbalement au cours ou au lab. Vous êtes FORTEMENT encouragés à changer ce mot de passe dès que possible (chaque usager a son mot de passe, même si la BD est commune à l'équipe). Pour changer votre mot de passe, connectez-vous à la BD de votre équipe via **`psql`** tel qu'indiqué plus haut, en utilisant le mot de passe générique, puis une fois connecté, lancez la commande **`\password`** à l'invite **`psql`**. On vous demandera de saisir (deux fois) votre nouveau mot de passe. Vous pouvez également changer votre mot de passe via pgadmin (menu File → Change Password…).

Ajout d'une table dans la base de données via la ligne de commandes

Les indications qui suivent supposent que vous êtes déjà connectés à la base de données en mode ligne de commande, tel que décrit à la section précédente.

1. Lancer la commande suivante pour créer la table « testdata » (l'exemple suppose l'équipe 34):

```
log720_20123_34=> create table testdata (
          id serial primary key,
          foo varchar(25),
          bar int);
```

2. Lancer la commande suivante pour ajouter un enregistrement dans la table « testdata » :

```
log720_20123_34=> insert into testdata (foo, bar)
       values('hello', 12345);
```

   Le résultat devrait ressembler à :

```
INSERT 17312 1

log720_20123_34=>
```

3. Lancer la commande suivante pour vérifier l'ajout de l'enregistrement:

```
log720_20123_34=> select * from testdata;
```

   Le résultat devrait ressembler à :

```
id  |   foo     |  bar
----+-----------+-------
  1 | hello     | 12345
 (1 row)

log720_20123_34=>
```

   Votre base de données est maintenant prête pour la suite. La commande pour mettre fin à une session psql est **`\q`**.

Création d'un projet Web dynamique dans eclipse via le plugiciel m2e

1. Lancez Eclipse édition EE (identifiée eclipse720 au laboratoire).
2. Assurez-vous d'être dans la perspective "Java EE"
3. Créez un nouveau projet maven via File→New→Project...→Maven →Maven Project, puis en cliquant sur **`Next>`**.
4. Sur l'écran suivant, pointez eclipse vers le bon "workspace" et appuyez sur **`Next>`**.
5. Spécifiez la localisation de votre projet en choisissant une destination pour vos fichiers et appuyez sur **`Next>`**.
6. **NOTE:** par défaut, le plugin m2e tente de mettre à jour sa liste d'archétypes à chaque démarrage d'eclipse, et ceci est assez long (plusieurs minutes). Vous devez laisser ceci se faire complètement au moins une fois. Vous verrez le message "Updating indexes" dans le coin inférieur droit de la fenêtre d'eclipse pendant que la liste des archétypes est mise à jour. Une fois que cette liste est remplie, vous pouvez désactiver cette option (pour accélérer le démarrage d'eclipse les prochaines fois) en allant dans les options de configuration via Window→Preferences →Maven, décochez l'option "Download repository index updates on startup".
7. Dans la fenêtre de création de nouveau projet Maven, entrez **`webapp`** dans le champ **`Filter`**. Ceci réduira considérablement la longueur de la liste des archétypes à consulter.
8. Choisissez l'archétype qui a comme groupId **`org.codehaus.mojo.archetypes`** et comme artifactId **`webapp-jee5`**. Cliquez sur **`Next>`**.
9. Saisissez les informations suivantes:
    a. Group Id: **`ca.etsmtl.log720.tutoriel_lab2`**
    b. Artifact Id: **`TestJeeDB`**
10. Cliquez sur **`Finish`**.

Remplacez le contenu du fichier pom.xml généré à la racine du projet par ce qui suit:

```
<project
  xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
  http://maven.apache.org/xsd/maven-4.0.0.xsd">

  <modelVersion>4.0.0</modelVersion>
  <groupId>ca.etsmtl.log720.Tutoriel_lab2</groupId>
  <artifactId>TestJeeDB</artifactId>
  <version>1.0</version>
  <packaging>war</packaging>

  <build>
    <!-- Evite d'avoir -SNAPSHOT* en suffixe au nom de l'archive WAR -->
    <finalName>${project.artifactId}</finalName>

    <plugins>
      <plugin>
        <groupId>org.mortbay.jetty</groupId>
        <artifactId>jetty-maven-plugin</artifactId>
        <version>8.0.3.v20111011</version>

        <!-- Contexte (racine) de l'application -->
        <configuration>
          <webAppConfig>
            <contextPath>/${project.artifactId}</contextPath>
          </webAppConfig>
        </configuration>
      </plugin>
    </plugins>
  </build>

  <dependencies>
    <!-- Driver JDBC -->
    <dependency>
      <groupId>postgresql</groupId>
      <artifactId>postgresql</artifactId>
      <version>9.0-801.jdbc4</version>
      <scope>runtime</scope>
    </dependency>

    <!-- Pour le "expression language" (EL) dans les JSP -->
    <dependency>
      <groupId>taglibs</groupId>
      <artifactId>standard</artifactId>
      <version>1.1.2</version>
      <scope>runtime</scope>
    </dependency>
    <dependency>
      <groupId>javax.servlet</groupId>
      <artifactId>jstl</artifactId>
      <version>1.1.2</version>
      <scope>runtime</scope>
    </dependency>
  </dependencies>
</project>
```

Il est possible qu'après cette modification, eclipse vous signale un problème avec votre configuration Maven. Appliquez simplement la "réparation" suggérée par eclipse, consistant à mettre à jour votre configuration dans eclipse.

Configuration de la source de données JNDI[1] en ciblant Tomcat

1. Créez le repertoire **`META-INF`** sous **`src/main/webapp`**.
2. Afin que l'application web puisse accéder à la base de données, configurez votre source de données JNDI en ajoutant le fichier **`context.xml`** tel que montré ci-bas sous le répertoire **`META-INF`** de votre projet Web.

```
<?xml version="1.0" encoding="UTF-8"?>
<Context path="/TestJeeDB" docBase="TestJeeDB"
        debug="5" reloadable="true" crossContext="true">

    <!-- maxActive: Maximum number of dB connections in pool.    -->

    <!-- maxIdle: Maximum number of idle dB connections to retain in pool.
         Set to -1 for no limit.  See also the DBCP documentation on this
         and the minEvictableIdleTimeMillis configuration parameter.
         -->

    <!-- maxWait: Maximum time to wait for a dB connection to become available
         in ms, in this example 10 seconds. An Exception is thrown if
         this timeout is exceeded.  Set to -1 to wait indefinitely.
         -->

    <!-- username and password: dB username and password for dB connections  -->

    <!-- driverClassName: Class name for the JDBC driver   -->

    <!-- url: The JDBC connection url for connecting to your dB.  -->

    <Resource name="jdbc/TestJeeDB" auth="Container" type="javax.sql.DataSource"
        maxActive="100" maxIdle="30" maxWait="10000" username="ab12345"
          password="*****" driverClassName="org.postgresql.Driver"
          url="jdbc:postgresql://log720.logti.etsmtl.ca:5432/log720_20123_X"/>
</Context>
```

NOTE: dans la balise **`<Resource>`**, remplacer:

- "**`ab12345`**" par votre nom d'usager (code universel en minuscules);
- les **`"******"`** du champ **`password`** par votre mot de passe (pour le SGBD);
- le "**`X`**" de **`log720_20123_X`** par votre numéro d'équipe.

[1] Cette section est tirée de la documentation de Tomcat 6 (CATALINA_HOME\webapps\docs\jndi-datasource-examples-howto.html).

Configuration de la source de données JNDI en ciblant Jetty

Afin que l'application web puisse accéder à la base de données, configurez votre source de données JNDI en ajoutant le fichier **`jetty-env.xml`** tel que montré ci-bas sous le répertoire **`WEB-INF`** de votre projet Web.

```
<?xml version="1.0"?>
<!DOCTYPE Configure PUBLIC "-//Mort Bay Consulting//DTD Configure//EN"
  "http://jetty.mortbay.org/configure.dtd">
<Configure class="org.eclipse.jetty.webapp.WebAppContext">
  <New id="TestJeeDB" class="org.eclipse.jetty.plus.jndi.Resource">
    <Arg></Arg>
    <Arg>jdbc/TestJeeDB</Arg>
    <Arg>
      <New class="org.postgresql.ds.PGSimpleDataSource">
        <Set name="User">ab12345</Set>
        <Set name="Password">******</Set>
        <Set name="DatabaseName">log720_20123_X</Set>
        <Set name="ServerName">log720.logti.etsmtl.ca</Set>
        <Set name="PortNumber">5432</Set>
      </New>
    </Arg>
  </New>
</Configure>
```

NOTE: remplacer:

- "**`ab12345`**" par votre nom d'usager (code universel en minuscules);
- les **`"******"`** du champ **`password`** par votre mot de passe (pour le SGBD);
- le "**`X`**" de **`log720_20123_X`** par votre numéro d'équipe.

Descripteur de déploiement `web.xml` (indépendant du serveur utilisé)

Remplacer le contenu de **`WEB-INF/web.xml`** par ce qui suit

```
<?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>TestJeeDB</display-name>
  <welcome-file-list>
    <welcome-file>TestJeeDB.jsp</welcome-file>
  </welcome-file-list>
  <resource-ref>
    <description>DB Connection</description>
    <res-ref-name>jdbc/TestJeeDB</res-ref-name>
    <res-type>javax.sql.DataSource</res-type>
    <res-auth>Container</res-auth>
  </resource-ref>
</web-app>
```

Création d'une page JSP qui affiche le contenu de la BD

NOTE IMPORTANTE: accéder à une base de données à partir d'une page JSP N'EST PAS UNE BONNE IDÉE. Vous ne devez PAS reproduire cette façon de faire dans votre laboratoire. Nous utilisons cette mécanique ici pour valider rapidement l'infrastructure.

Créez-vous une page JSP ayant comme nom **`TestJeeDB.jsp`** sous le répertoire **`webapp`**. Ajoutez-y le contenu suivant.

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
        pageEncoding="ISO-8859-1"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/sql" prefix="sql"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>

<sql:query var="rs" dataSource="jdbc/TestJeeDB">
    select id, foo, bar from testdata
</sql:query>

<html>
<head>
<title>Test JEE with DB</title>
</head>
<body>

<h2>Results</h2>

<c:forEach var="row" items="${rs.rows}">
    Foo ${row.foo}<br />
    Bar ${row.bar}<br />
</c:forEach>

</body>
</html>
```

Déploiement de votre projet Web dans Tomcat avec Eclipse

But : Cette étape permet le déploiement de votre projet Web sur un serveur Apache Tomcat.

1. Si ce n'est pas déjà fait, ajouter un serveur Apache Tomcat avec un clique droit sur **`Server`** (onglet dans le bas du "Workbench" eclipse) → New → Server.

2. Choisissez la bonne version de votre serveur soit « Tomcat v6.0 Server », pointez eclipse vers le répertoire d'installation de Tomcat 6 sur le poste (**`C:\apache\tomcat\6.Y.Z`**) et cliquez sur « Next ».

3. Ajouter votre projet Web et cliquez sur « Finish ».

4. Démarrez votre serveur en appuyant sur « Start » et attendez d'obtenir la ligne suivante dans la console eclipse:

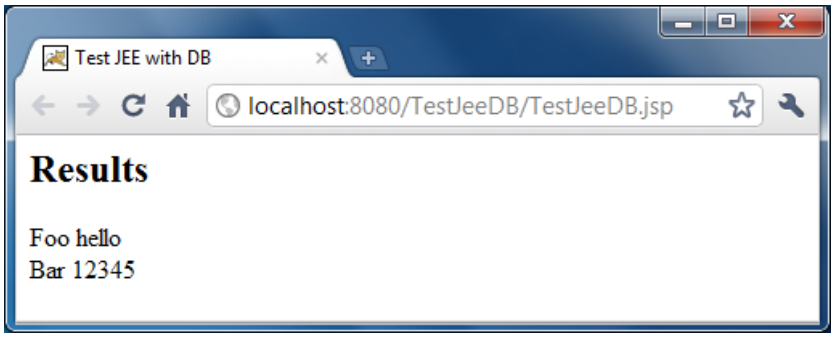
```
2008-02-05 11:18:47 org.apache.catalina.startup.Catalina start
INFO: Server startup in 453 ms
```

Déploiement de votre projet Web dans Jetty avec Maven

Exécutez la commande **`mvn compile jetty:run`** à la racine de votre projet maven. Notez que vous ne pouvez pas faire ceci pendant qu'une instance d'un autre serveur (comme Tomcat) est active, car le même port par défaut est utilisé par tous les serveurs d'applications web dynamiques (typiquement 8080).

Visualiser votre projet Web dans un fureteur

Ouvrir votre navigateur, pointer à l'adresse http://localhost:8080/TestJeeDB/TestJeeDB.jsp.



À ce stade, nous avons fait le tour de l'ensemble des outils requis pour réaliser le reste du laboratoire (en termes d'infrastructure).

Les sections suivantes servent d'introduction aux servlets, aux Java Beans, et à l'approche suggérée pour utiliser le patron MVC en JEE.

Création d'un servlet

Afin de conserver le travail réalisé à date, et comme la suite est différente de ce qui a été fait plus haut, il est suggéré de créer un nouveau projet Web dynamique nommé "TestJeeMvc".

Group Id: **`ca.etsmtl.log720.tutoriel_lab2`**
Artifact Id: **`TestJeeMvc`**

1. Une fois le projet maven créé, remplacer le contenu du POM par ce qui suit:

```
<project
  xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
  http://maven.apache.org/xsd/maven-4.0.0.xsd">

  <modelVersion>4.0.0</modelVersion>
  <groupId>ca.etsmtl.log720.tutoriel_lab2</groupId>
  <artifactId>TestJeeMvc</artifactId>
  <version>1.0</version>
  <packaging>war</packaging>

  <build>
    <finalName>${project.artifactId}</finalName>
    <plugins>
      <plugin>
        <groupId>org.mortbay.jetty</groupId>
        <artifactId>jetty-maven-plugin</artifactId>
        <version>8.0.3.v20111011</version>
        <configuration>
          <webAppConfig>
            <contextPath>/${project.artifactId}</contextPath>
          </webAppConfig>
        </configuration>
      </plugin>
    </plugins>
  </build>
  <dependencies>
    <dependency>
      <groupId>javax.servlet</groupId>
      <artifactId>servlet-api</artifactId>
      <version>2.5</version>
      <scope>provided</scope>
    </dependency>
    <dependency>
      <groupId>javax.servlet.jsp</groupId>
      <artifactId>jsp-api</artifactId>
      <version>2.1</version>
      <scope>provided</scope>
    </dependency>
  </dependencies>
</project>
```

2. Créez un paquetage **servlets** sous le dernier paquetage de votre projet (**ca.etsmtl.log720.tutoriel_lab2.TestJeeMvc**).
3. Clique droit sur votre projet : « New → Servlet »
4. Choisissez le paquetage **ca.etsmtl.log720.tutoriel_lab2.TestJeeMvc.servlets** et donnez un nom (par exemple **MyServlet**) pour votre Servlet.
5. Voici à quoi votre servlet devrait ressembler.

```
package ca.etsmtl.log720.tutoriel_lab2.TestJeeMvc.servlets;

import java.io.IOException;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class MyServlet extends javax.servlet.http.HttpServlet
    implements javax.servlet.Servlet  {

  public MyServlet() {
    super();
  }

  protected void doGet(HttpServletRequest request,
    HttpServletResponse response) throws ServletException, IOException  {
      // Insérer du code plus tard
  }

  protected void doPost(HttpServletRequest request,
    HttpServletResponse response) throws ServletException, IOException {
      // Insérer du code plus tard
  }
}
```

6. Éditez votre fichier **WebContent/WEB-INF/web.xml** pour qu'il ressemble à ceci :

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app id="WebApp_ID" version="2.5"
     xmlns="http://java.sun.com/xml/ns/j2ee"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee
     http://java.sun.com/xml/ns/j2ee/web-app_2_5.xsd">
  <display-name>Client_Web</display-name>
  <servlet>
    <servlet-name>MyServlet</servlet-name>
    <servlet-class>
      ca.etsmtl.log720.tutoriel_lab2.TestJeeMvc.servlets.MyServlet
    </servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>MyServlet</servlet-name>
    <url-pattern>/MyServlet</url-pattern>
  </servlet-mapping>
</web-app>
```

Création d'un JavaBean

Note : Un JavaBean est une classe java qui contient un constructeur public sans argument, est sérialisable, et utilise des méthodes get/set pour accéder à ses attributs privés.

1. Créez un paquetage **`beans`** au même niveau que le paquetage **`servlets`**.

2. Créez une nouvelle classe Java dans le paquetage **`beans`** et donnez-lui le nom **`MyBean`**.

3. Définissez des méthodes get/set pour les utiliser dans vos pages JSP.

```
package ca.etsmtl.log720.tutoriel_lab2.TestJeeMvc.beans;

public class MyBean
{
  // Cette méthode permettra de tester votre Pojo
  public String getSalut_FromJSP() {
    return "Votre JavaBean vous salue avec votre JSP directement.";
  }

  // Cette méthode permettra de tester votre Servlet
  public String getSalut_FromServlet() {
    return "Votre JavaBean vous salue avec votre Servlet.";
  }
}
```

4. Initialisez les JavaBeans à partir de votre Servlet (« MyServlet.java »)

```
import ca.etsmtl.log720.tutoriel_lab2.TestJeeMvc.beans.MyBean;

…

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

     MyBean bean = new MyBean();
     request.getSession().setAttribute("myBean", bean); // Scope: Session
     response.sendRedirect
       ("/TestJeeMvc/TestClientWebFromServlet.jsp"); // redirection
}
```

Création de pages JSP liées à des JavaBeans et servlets

But : Créer un fichier JSP qui sera votre page Web affichant le contenu de votre JavaBean à l'aide de votre servlet. Deux pages JSP sont nécessaires pour démontrer deux méthodes d'accès aux JavaBeans.

1. Clique droit sur votre projet : « New → JSP File ».
2. Créez la page sous **webapp** et donnez-lui comme nom **TestClientWeb.jsp**.
3. Remplacez le contenu de **TestClientWeb.jsp** par[2] :

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
  pageEncoding="ISO-8859-1"%>
<!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>Test_Client_Web </title>
</head>
<body>
  <jsp:useBean id="myBean"
    class="ca.etsmtl.log720.tutoriel_lab2.TestJeeMvc.beans.MyBean"
    scope="request">
  </jsp:useBean>
  ${myBean.salut_FromJSP}
</body>
</html>
```

4. Clique droit sur votre projet : « New → JSP File ».
5. Créez la page sous **webapp** et donnez-lui comme nom **TestClientWebFromServlet.jsp**.
6. Remplacez le contenu de **TestClientWebFromServlet.jsp** par :

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
  pageEncoding="ISO-8859-1"
  import="ca.etsmtl.log720.tutoriel_lab2.TestJeeMvc.beans.MyBean" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
  "http://www.w3.org/TR/html4/loose.dtd">
<% MyBean myBean = (MyBean)request.getSession().getAttribute("myBean"); %>
<html>
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
  <title>Test_Client_Web_From_Servlet </title>
</head>
<body>
  <%=myBean.getSalut_FromServlet()%>
</body>
</html>
```

5. Déployez votre projet selon une des deux méthodes expliquées plus haut.

---

[2] L'utilisation de la balise « jsp:useBean » permet à la page JSP d'utiliser directement vos JavaBean. **${myBean.salut_FromJSP}** équivaut à **myBean.getSalut_FromJSP();**

Visualiser votre projet Web dans un fureteur

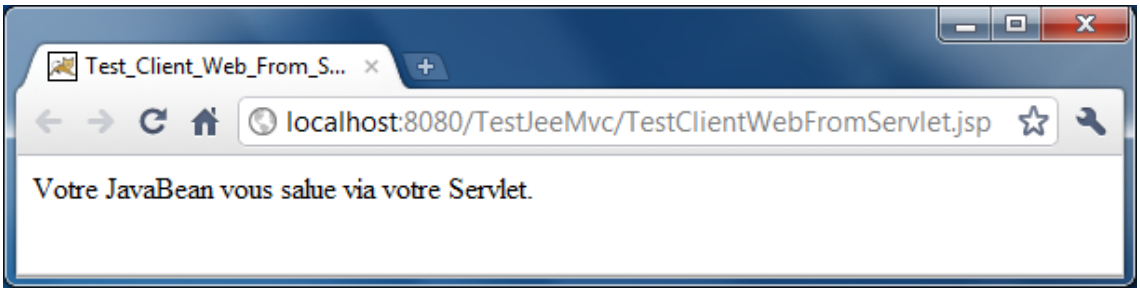
But : Cette étape permet de visualiser votre site Web à l'aide des pages JSP.

Méthode 1 : En appelant une page JSP qui utilise la balise **`<jsp:useBean>`**

1. Ouvrir votre navigateur au http://localhost:8080/TestJeeMvc/TestClientWeb.jsp

Méthode 2 : En appelant directement le servlet par l'URL et qui utilise une redirection.

2. Ouvrir votre navigateur au http://localhost:8080/TestJeeMvc/MyServlet.



Vous pouvez constater la redirection vers la nouvelle page JSP **`TestClientWebFromServlet.jsp`**. Quoique subtile ici, la redirection d'une requête http à partir d'un servlet vers une page JSP via **`RequestDispatcher`** est une bonne pratique d'implémentation du patron MVC en JEE.