

MASTER INFORMATIQUE - SEMESTRE 5

Cours : Développement Web

Séquence 5: JavaServer Faces





Apache Tomcat 8
Version 8.5.24, Nov 27 2017



Partie 5: JSF et Bases de données



JavaServer™ Faces

JSF



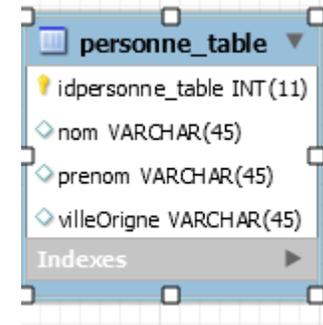
Contenu

- Architectures Multi-tiers (Rappel)
- La plate-forme Java EE (Rappel)
- JavaServer Faces
 - Environnement de développement
 - Une séparation de la couches présentation des autres couches
 - Un mapping entre l'HTML et L'objet
 - JSF et Bases de données
 - Un ensemble de composants riches et utilisables
 - Une liaison simple entre les actions côte Client et côte serveur
 - Composants additionnels JSF: Primefaces
 - Combinaison de plusieurs composants pour aboutir à un composant plus complexe

JSF et Base de données MySQL

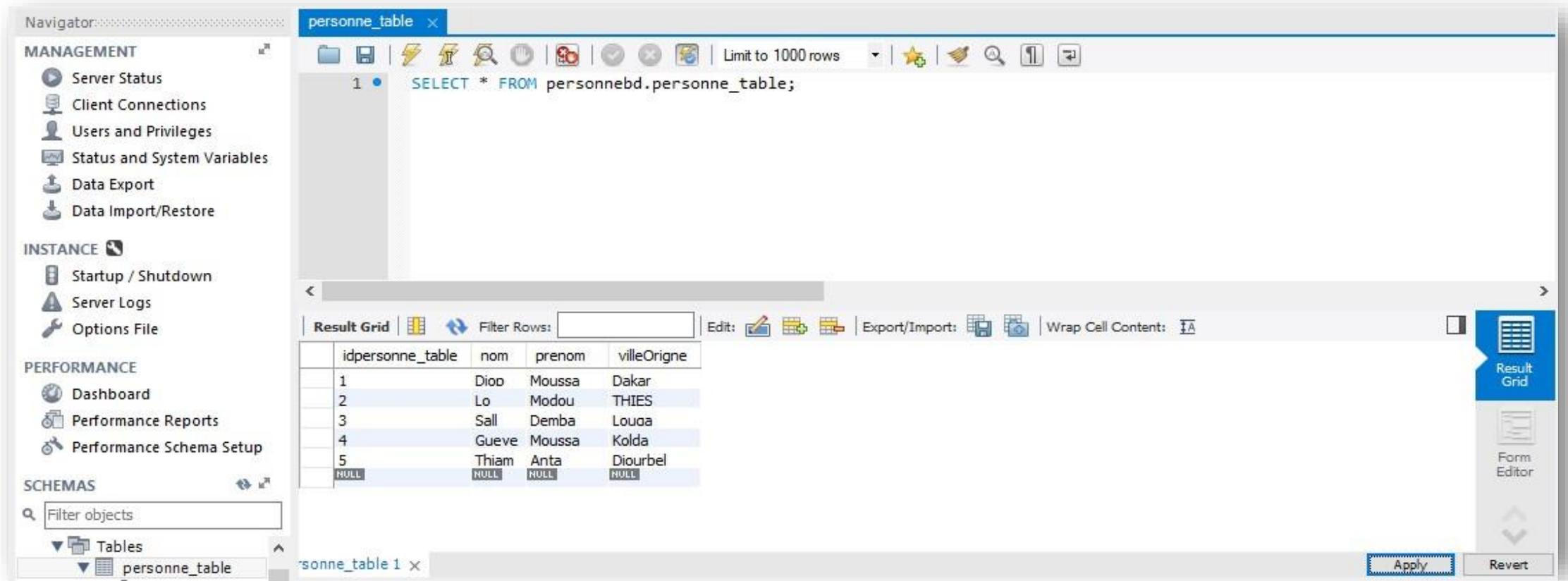
- Créer une base données nommée `personnebd`
- Créer une table nommée ***personne_table*** avec la structure suivante:

```
CREATE TABLE `personnebd`.`personne_table` (  
  `idpersonne_table` INT NOT NULL AUTO_INCREMENT, `nom`  
  VARCHAR(45) NULL, `prenom` VARCHAR(45) NULL,  
  `villeOrigine` VARCHAR(45) NULL, PRIMARY KEY  
  (`idpersonne_table`));
```



JSF et Base de données MySQL

- Ajouter des données à la BD



The screenshot displays a MySQL management tool interface. On the left, a sidebar contains navigation menus for MANAGEMENT, INSTANCE, PERFORMANCE, and SCHEMAS. The main window shows a query editor with the SQL statement: `SELECT * FROM personnebd.personne_table;`. Below the query editor, a "Result Grid" displays the following data:

| | idpersonne_table | nom | prenom | villeOrigine |
|---|------------------|-------|--------|--------------|
| 1 | | Dioo | Moussa | Dakar |
| 2 | | Lo | Modou | THIES |
| 3 | | Sall | Demba | Louqa |
| 4 | | Gueve | Moussa | Kolda |
| 5 | | Thiam | Anta | Diourbel |
| | NULL | NULL | NULL | NULL |

The interface also includes a toolbar with various icons, a "Filter Rows" input field, and buttons for "Apply" and "Revert" at the bottom right.

JSF et Base de données MySQL: création d'une Pool de Connection sous Tomcat

L'expression employée pour nommer cette pratique est le « **connection pooling** », souvent très sauvagement francisée « **pool** de connexions ». Pour faire simple, le **pool** de connexions va pré-initialiser un nombre donné de connexions au SGBD lorsque l'application démarre. 10 mars 2017

Gérer un pool de connexions avec BoneCP - Créez votre application ...
<https://openclassrooms.com/courses/...ee/gerer-un-pool-de-connexions-avec-bonecp>

Un pool de connexions est un mécanisme permettant de réutiliser les connexions créées. En effet, la création systématique de nouvelles instances de *Connection* peut parfois devenir très lourd en consommation de ressources. Pour éviter cela, un pool de connexions ne ferme pas les connexions lors de l'appel à la méthode *close()*. Au lieu de fermer directement la connexion, celle-ci est « retournée » au pool et peut être utilisée ultérieurement.

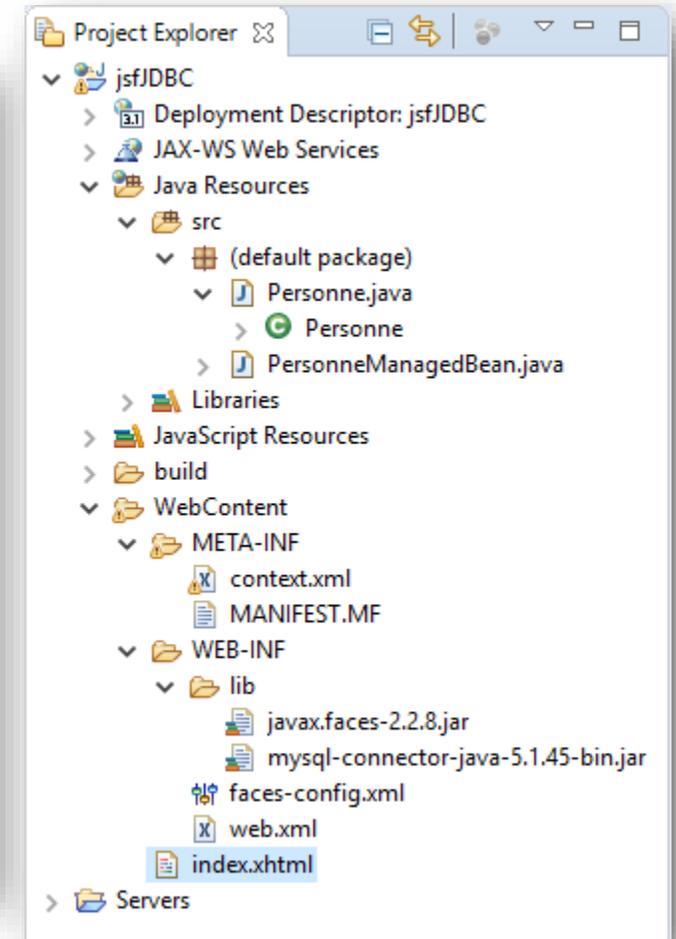
JSF et Base de données MySQL: création d'une Pool de Connection sous Tomcat



http://localhost:8081/jsfJDBC/faces/index.xhtml

Exemple JDBC et Pool de Connexion

| ID | Nom | Prénom | Ville d'origine |
|----|-------|--------|-----------------|
| 1 | Diop | Diop | Dakar |
| 2 | Lo | Lo | THIES |
| 3 | Sall | Sall | Louga |
| 4 | Gueye | Gueye | Kolda |
| 5 | Thiam | Thiam | Diourbel |

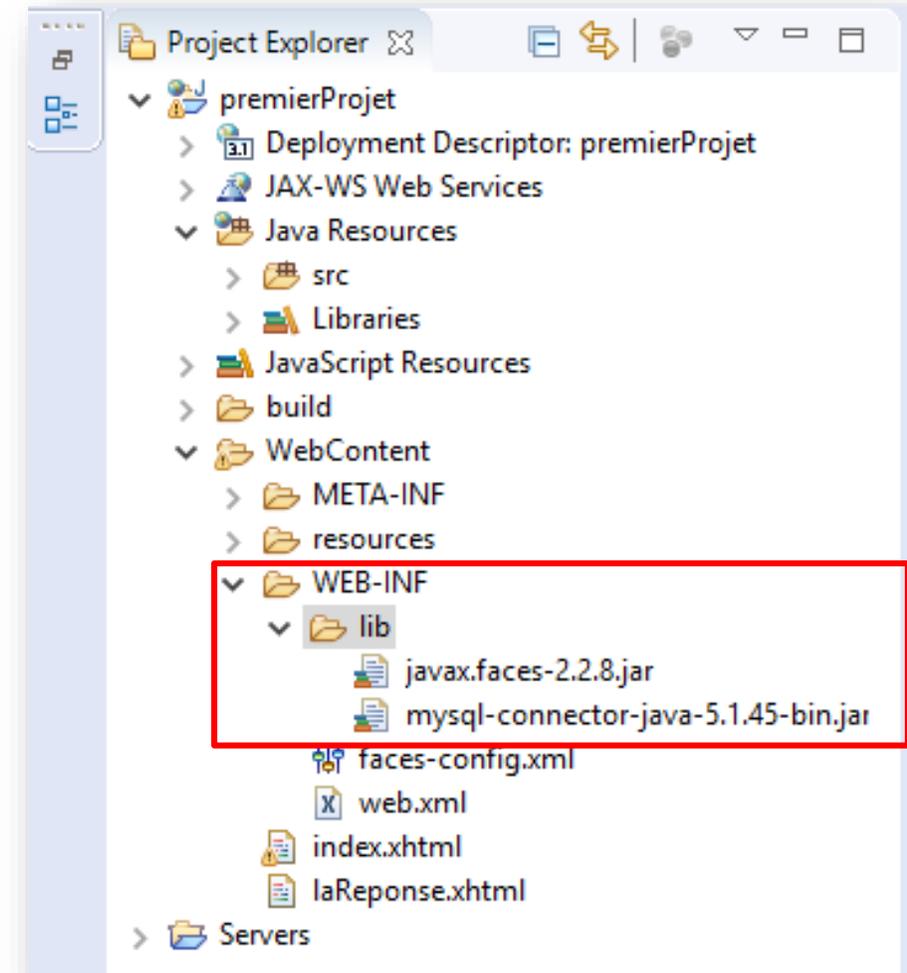


Project Explorer

- jsfJDBC
 - Deployment Descriptor: jsfJDBC
 - JAX-WS Web Services
 - Java Resources
 - src
 - (default package)
 - Personne.java
 - Personne
 - PersonneManagedBean.java
 - Libraries
 - JavaScript Resources
 - build
 - WebContent
 - META-INF
 - context.xml
 - MANIFEST.MF
 - WEB-INF
 - lib
 - javax.faces-2.2.8.jar
 - mysql-connector-java-5.1.45-bin.jar
 - faces-config.xml
 - web.xml
 - index.xhtml
 - Servers

JSF et Base de données MySQL

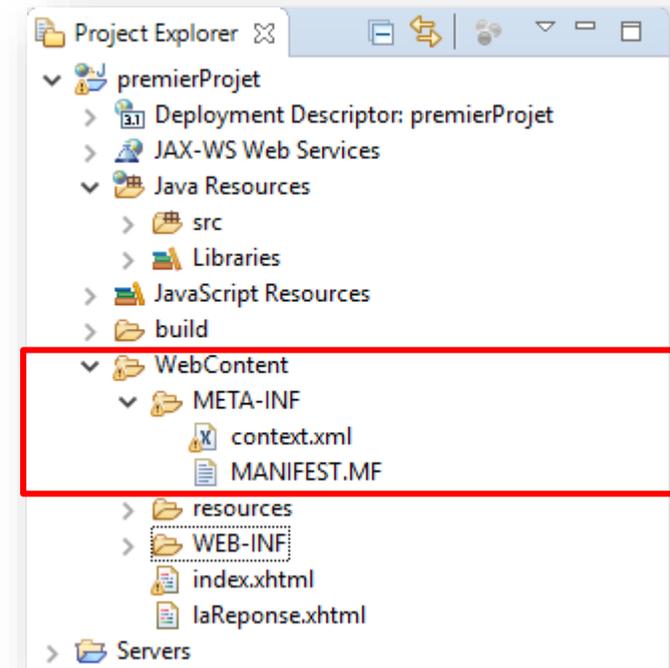
1. Télécharger le driver JDBC de MySQL <https://dev.mysql.com/downloads/> et le placer dans le dossier **WEB-INF/lib**
2. Définir la pool de connexion dans META-INF/context.xml
3. Définir une ressource de référence dans le fichier WEB-INF/web.xml
4. Référencer la pool de connexion dans le code Java



JSF et Base de données MySQL: création d'une Pool de Connection sous Tomcat

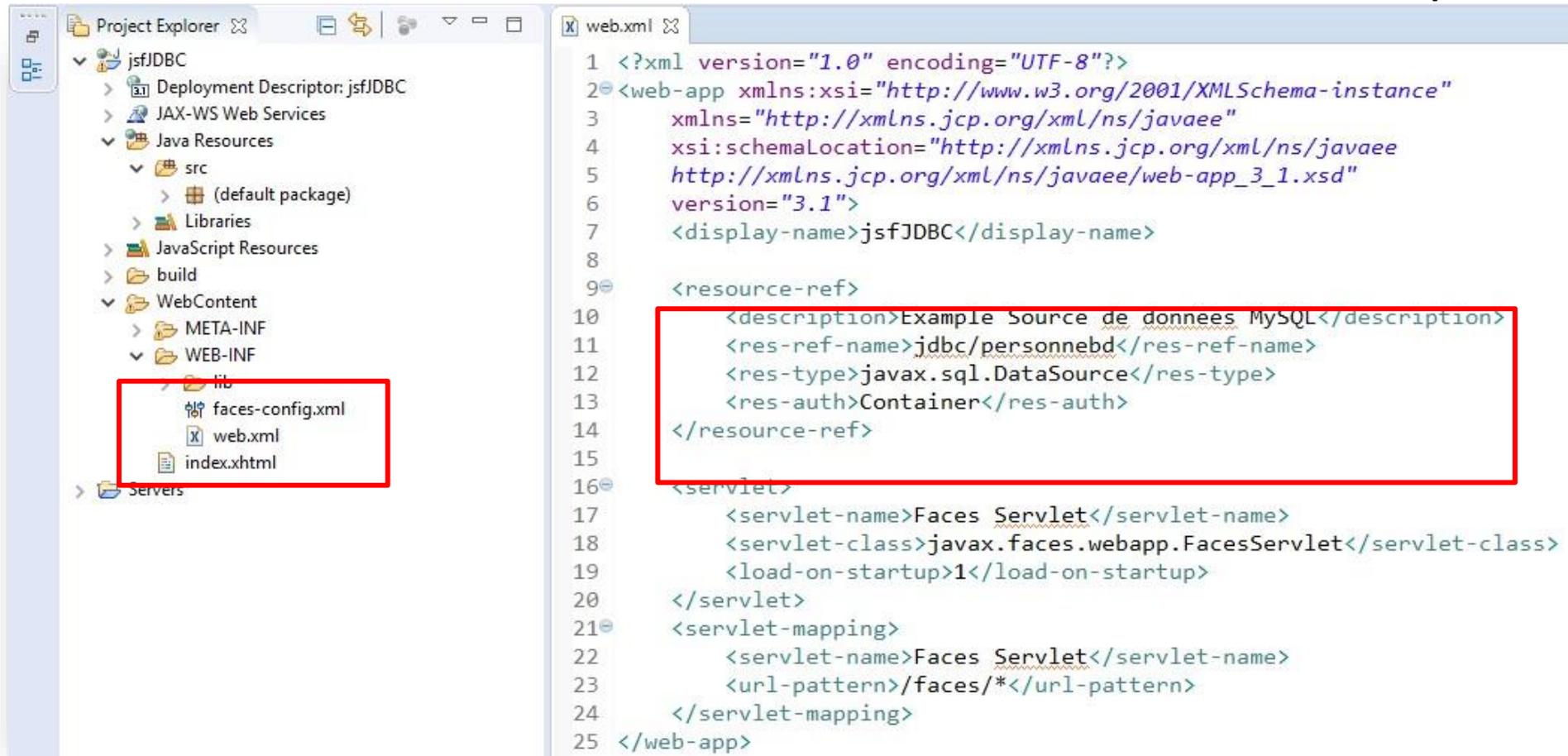
2. Définir la pool de connexion dans **META-INF/context.xml**

```
context.xml
1 <Context>
2   <Resource name="jdbc/personnebd"
3     auth="Container" type="javax.sql.DataSource"
4     maxActive="100" maxIdle="30" maxWait="10000"
5     username="osall" password="osall"
6     driverClassName="com.mysql.jdbc.Driver"
7     url="jdbc:mysql://localhost:3306/personnebd" />
8 </Context>
```



JSF et Base de données MySQL: création d'une Pool de Connection sous Tomcat

3. Définir une ressource de référence dans le fichier WEB-INF/web.xml



```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3   xmlns="http://xmlns.jcp.org/xml/ns/javaee"
4   xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
5   http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd"
6   version="3.1">
7   <display-name>jsfJDBC</display-name>
8
9   <resource-ref>
10    <description>Example Source de donnees MySQL</description>
11    <res-ref-name>jdbc/personnebd</res-ref-name>
12    <res-type>javax.sql.DataSource</res-type>
13    <res-auth>Container</res-auth>
14  </resource-ref>
15
16  <servlet>
17    <servlet-name>Faces Servlet</servlet-name>
18    <servlet-class>javax.faces.webapp.FacesServlet</servlet-class>
19    <load-on-startup>1</load-on-startup>
20  </servlet>
21  <servlet-mapping>
22    <servlet-name>Faces Servlet</servlet-name>
23    <url-pattern>/faces/*</url-pattern>
24  </servlet-mapping>
25 </web-app>
```

JSF et Base de données MySQL: création d'une Pool de Connection sous Tomcat

4. Référencer la pool de connexion dans le code Java. Plusieurs options possibles parmi lesquels:
 - Option 1: Injection de ressources avec les servlets
 - **Option 2: Utiliser JNDI(Java Naming Directory Interface)**

Voir également ce lien sur votre serveur d'application Tomcat

<http://localhost:8080/docs/jndi-datasource-examples-howto.html>



Apache Tomcat 8
Version 8.5.24, Nov 27 2017



JSF et Base de données MySQL: création d'une Pool de Connection sous Tomcat

```
Personne.java ✕
1
2 public class Personne {
3     private int idPersonne;
4     private String nom;
5     private String prenom;
6     private String villeOrigine;
7
8     public Personne() {
9         // TODO Auto-generated constructor stub
10    }
11
12    public int getIdPersonne() {
13
14    }
15
16    public void setIdPersonne(int idPersonne) {
17
18    }
19
20    public String getNom() {
21
22    }
23
24    public void setNom(String nom) {
25
26    }
27
28    public String getPrenom() {
29
30    }
31
32    public void setPrenom(String prenom) {
33
34    }
35
36    public String getVilleOrigine() {
37
38    }
39
40    public void setVilleOrigine(String villeOrigine) {
41
42    }
43 }
```

Créer une entité nommée Personne

JSF et Base de données MySQL: création d'une Pool de Connection sous Tomcat

```
PersonneManagedBean.java
1 import java.sql.Connection;
15 @ManagedBean
16 @SessionScoped
17 public class PersonneManagedBean {
18     // Injection de ressources
19     @Resource(name = "jdbc/personnebd")
20     private DataSource ds;
21
22     private List<Personne> listePersonnes;
23     // Si l'injection de ressources n'est pas prise en charge,
24     // vous pouvez toujours l'obtenir manuellement.
25     public PersonneManagedBean() {
26         try {
27             Context ctx = new InitialContext();
28             ds = (DataSource) ctx.lookup("java:comp/env/jdbc/personnebd");
29         } catch (NamingException e) {
30             e.printStackTrace();
31         }
32     }
33
34     public List<Personne> getListePersonnes() {
35         return listePersonnes;
36     }
37
38     public void setListePersonnes(List<Personne> listePersonnes) {
39         this.listePersonnes = listePersonnes;
40     }
41
42     // Se connecter à DB et obtenir la liste des personnes
43     public void chargerListePersonnes() throws SQLException {
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71 }
```

Soit le Managed bean

JSF et Base de données MySQL: création d'une Pool de Connection sous Tomcat

```
// Se connecter à DB et obtenir la liste des personnes
public void chargerListePersonnes() throws SQLException {
    listePersonnes = new ArrayList<Personne>();
    if (ds == null)
        throw new SQLException("Can't get data source");

    // Obtenir la connexion à la base
    Connection con = ds.getConnection();

    if (con == null)
        throw new SQLException("Can't get database connection");

    PreparedStatement ps = con.prepareStatement("select * from personne_table");

    // Obtenir des données de personne de la base de données
    ResultSet result = ps.executeQuery();

    while (result.next()) {
        Personne pers = new Personne();

        pers.setIdPersonne(result.getInt("idpersonne_table"));
        pers.setNom(result.getString("nom"));
        pers.setPrenom(result.getString("nom"));
        pers.setVilleOrigine(result.getString("villeOrigine"));

        // Stocker toutes les données dans une liste
        listePersonnes.add(pers);
    }
}
```

Soit le Code de la méthode permettant de charger les données de la base de données

JSF et Base de données MySQL: création d'une Pool de Connection sous Tomcat

preRenderViewEvent permet d'appeler une méthode avant le chargement de la page.

```
index.xhtml ✕
1 <?xml version="1.0" encoding="UTF-8"?>
2 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
3 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
4 <html xmlns="http://www.w3.org/1999/xhtml"
5       xmlns:h="http://java.sun.com/jsf/html"
6       xmlns:f="http://java.sun.com/jsf/core">
7 <h:head>
8   <h:outputStylesheet library="css" name="table-style.css" />
9 </h:head>
10 <f:metadata>
11   <f:event listener="#{personneManagedBean.chargerListePersonnes()}" type="preRenderView"/>
12 </f:metadata>
13 <h:body>
14   <h1>Exemple JDBC et Pool de Connexion</h1>
15   <h:dataTable value="#{personneManagedBean.ListePersonnes}" var="c">
16     <h:column>
17       <f:facet name="header"> ID </f:facet> #{c.idPersonne}
18     </h:column>
19     <h:column>
20       <f:facet name="header"> Nom </f:facet> #{c.nom}
21     </h:column>
22     <h:column>
23       <f:facet name="header"> Prénom </f:facet> #{c.prenom}
24     </h:column>
25     <h:column>
26       <f:facet name="header"> Ville d'origine </f:facet> #{c.villeOrigine}
27     </h:column>
28   </h:dataTable>
29
30 </h:body>
31 </html>
```

Soit le Code de la méthode permettant de charger les données de la base de données

JSF et Base de données MySQL: Ajouter une personne

- Transformer l'entité Personne en Manager bean

```
Personne.java
1 import javax.faces.bean.ManagedBean;
2
3 @ManagedBean
4 public class Personne {
5     private int idPersonne;
6     private String nom;
7     private String prenom;
8     private String villeOrigine;
9
10    public Personne() {}
13
14    public int getIdPersonne() {}
17
18    public void setIdPersonne(int idPersonne) {}
21
22    public String getNom() {}
25
26    public void setNom(String nom) {}
29
30    public String getPrenom() {}
33
34    public void setPrenom(String prenom) {}
37
38    public String getVilleOrigine() {}
41
42    public void setVilleOrigine(String villeOrigine) {}
45 }
```

http://localhost:8081/jsfJDBC/faces/index.xhtml

Exemple JDBC et Pool de Connexion

Nom:

Prénom:

Ville d'origine:

| ID | Nom | Prénom | Ville d'origine |
|----|-------|--------|-----------------|
| 1 | Diop | Diop | Dakar |
| 2 | Lo | Lo | THIES |
| 3 | Sall | Sall | Louga |
| 4 | Gueye | Gueye | Kolda |
| 5 | Thiam | Thiam | Diourbel |
| 6 | lll | lll | pppp |
| 7 | jj | jj | hhh |

JSF et Base de données MySQL: Ajouter une personne

Ajouter la méthode ajouterPersonne au Managed bean

```
PersonneManagedBean.java
1+ import java.sql.Connection;
15
16 @ManagedBean
17 @SessionScoped
18 public class PersonneManagedBean {
19     // Injection de ressources
20     @Resource(name = "jdbc/personnebd")
21     private DataSource ds;
22
23     private List<Personne> listePersonnes;
24
25     public String ajouterPersonne(Personne personne) throws SQLException {
26         if (ds == null)
27             throw new SQLException("Can't get data source");
28         // Obtenir la connexion à la base
29         Connection con = ds.getConnection();
30         if (con == null)
31             throw new SQLException("Can't get database connection");
32         PreparedStatement myStmt = null;
33         String sql = "insert into personne_table (nom, prenom, villeOrigine) values (?, ?, ?)";
34         myStmt = con.prepareStatement(sql);
35         // set params
36         myStmt.setString(1, personne.getNom());
37         myStmt.setString(2, personne.getPrenom());
38         myStmt.setString(3, personne.getVilleOrigine());
39
40         myStmt.execute();
41
42         return "index?faces-redirect=true";
43     }
44 }
```

JSF et Base de données MySQL: Ajouter une personne

```
index.xhtml
1 <?xml version="1.0" encoding="UTF-8"?>
2 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
3 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
4 <html xmlns="http://www.w3.org/1999/xhtml" xmlns:h="http://java.sun.com/jsf/html"
5     xmlns:f="http://java.sun.com/jsf/core">
6 <h:head>
9 <f:metadata>
13 <h:body>
14     <h1>Exemple JDBC et Pool de Connexion</h1>
15     <h:form id="form">
16         <h:panelGrid columns="3">
17             <h:outputLabel>Nom:</h:outputLabel>
18             <h:inputText value="#{personne.nom}" id="nom" />
19             <br />
20
21             <h:outputLabel>Prénom:</h:outputLabel>
22             <h:inputText value="#{personne.prenom}" id="prenom" />
23             <br />
24
25             <h:outputLabel>Ville d'origine:</h:outputLabel>
26             <h:inputText value="#{personne.villeOrigine}" id="ville" />
27             <h:message for="email" />
28             <br />
29
30             <h:outputLabel />
31             <h:commandButton value="Sauvegarder" styleClass="sauvegarder"
32                 action="#{personneManagedBean.ajouterPersonne(personne)}" />
33         </h:panelGrid>
34     </h:form>
35     <h:dataTable value="#{personneManagedBean.listePersonnes}" var="c">
49 </h:body>
50 </html>
```

Le formulaire dans le fichier index.xhtml

JSF et Base de données MySQL: Ajouter une personne

http://localhost:8081/jsfJDBC/faces/index.xhtml

Exemple JDBC et Pool de Connexion

Nom:

Prénom:

Ville d'origine:

| ID | Nom | Prénom | Ville d'origine |
|----|-------|--------|-----------------|
| 1 | Diop | Diop | Dakar |
| 2 | Lo | Lo | THIES |
| 3 | Sall | Sall | Louga |
| 4 | Gueye | Gueye | Kolda |
| 5 | Thiam | Thiam | Diourbel |
| 6 | lll | lll | pppp |
| 7 | jj | jj | hhh |

http://localhost:8081/jsfJDBC/faces/index.xhtml

Exemple JDBC et Pool de Connexion

Nom:

Prénom:

Ville d'origine:

| ID | Nom | Prénom | Ville d'origine |
|----|-------|--------|-----------------|
| 1 | Diop | Diop | Dakar |
| 2 | Lo | Lo | THIES |
| 3 | Sall | Sall | Louga |
| 4 | Gueye | Gueye | Kolda |
| 5 | Thiam | Thiam | Diourbel |
| 6 | lll | lll | pppp |
| 7 | jj | jj | hhh |
| 8 | Wade | Wade | Kebemer |

JSF et Base de données MySQL: Supprimer une personne

- Ajouter la méthode de suppression au managed bean `PersonneManagedBean`

```
public void supprimerPersonne(int idPersonne) throws SQLException {
    if (ds == null)
        throw new SQLException("Can't get data source");
    // Obtenir la connexion à la base
    Connection con = ds.getConnection();
    if (con == null)
        throw new SQLException("Can't get database connection");
    PreparedStatement myStmt = null;
    String sql = "delete from personne_table where idpersonne_table=?";

    myStmt = con.prepareStatement(sql);

    // set params
    myStmt.setInt(1, idPersonne);

    myStmt.execute();
}
```

JSF et Base de données MySQL: Supprimer une personne

```
index.xhtml http://localhost:8081/jsfJDBC/faces/index.xhtml
14 <h:body>
15   <h1>Exemple JDBC et Pool de Connexion</h1>
16 <h:form id="form">
36   <h:form>
37     <h:dataTable value="#{personneManagedBean.ListePersonnes}" var="c">
38       <h:column>
39         <f:facet name="header"> ID </f:facet> #{c.idPersonne}
40       </h:column>
41       <h:column>
42         <f:facet name="header"> Nom </f:facet> #{c.nom}
43       </h:column>
44       <h:column>
45         <f:facet name="header"> Prénom </f:facet> #{c.prenom}
46       </h:column>
47       <h:column>
48         <f:facet name="header"> Ville d'origine </f:facet> #{c.villeOrigine}
49       </h:column>
50       <h:column>
51         <!-- the column header -->
52         <f:facet name="header">Action</f:facet>
53         <!-- the value for each row -->
54         <h:commandLink value="Editer |"
55           action="#{personneManagedBean.LoadStudent(c.idPersonne)}" />
56         <h:commandLink value="Supprimer"
57           onclick="if (!confirm('Êtes-vous sûr de vouloir supprimer cette personne ?')) return false"
58           action="#{personneManagedBean.supprimerPersonne(c.idPersonne)}" />
59       </h:column>
60     </h:dataTable>
61   </h:form>
```

- Mettre à jour le tableau affichant les données pour y ajouter un lien pour la mise à jour et un autre pour la suppression

JSF et Base de données MySQL: Supprimer une personne

The screenshot shows a web browser window with the URL `http://localhost:8081/jsfJDBC/faces/index.xhtml`. The page title is "Exemple JDBC et Pool de Connexion". It features three input fields for "Nom:", "Prénom:", and "Ville d'origine:", followed by a "Sauvegarder" button. Below these is a table of people with columns for ID, Nom, Prénom, Ville d'origine, and Action. The table lists five people: Diop, Lo, Sall, Gueye, and Thiam. A modal dialog box titled "Message de la page Web" is open, displaying a question mark icon and the text "Êtes-vous sûr de vouloir supprimer cette personne ?". The dialog has "OK" and "Annuler" buttons.

| ID | Nom | Prénom | Ville d'origine | Action |
|----|-------|--------|-----------------|--|
| 1 | Diop | Diop | Dakar | Editer Supprimer |
| 2 | Lo | Lo | THIES | Editer Supprimer |
| 3 | Sall | Sall | Louga | Editer Supprimer |
| 4 | Gueye | Gueye | Kolda | Editer Supprimer |
| 5 | Thiam | Thiam | Diourbel | Editer Supprimer |
| 8 | Wade | Wade | Kebemer | Editer Supprimer |

The screenshot shows the same web browser window and page as the previous one, but the modal dialog is closed. The table now only contains four people: Lo, Sall, Gueye, and Thiam. The "Sauvegarder" button is still present.

| ID | Nom | Prénom | Ville d'origine | Action |
|----|-------|--------|-----------------|--|
| 2 | Lo | Lo | THIES | Editer Supprimer |
| 3 | Sall | Sall | Louga | Editer Supprimer |
| 4 | Gueye | Gueye | Kolda | Editer Supprimer |
| 5 | Thiam | Thiam | Diourbel | Editer Supprimer |
| 8 | Wade | Wade | Kebemer | Editer Supprimer |

Webography

- https://en.wikipedia.org/wiki/Java_Platform,_Enterprise_Edition
- Tutorial Java EE 7, section Servlets:
<https://docs.oracle.com/javaee/7/index.html>
- <http://www.servletworld.com/> : nombreux tutoriaux et exemples
- <http://www.kodejava.org/browse/8.html> : idem, nombreux exemples
- Google
- Youtube