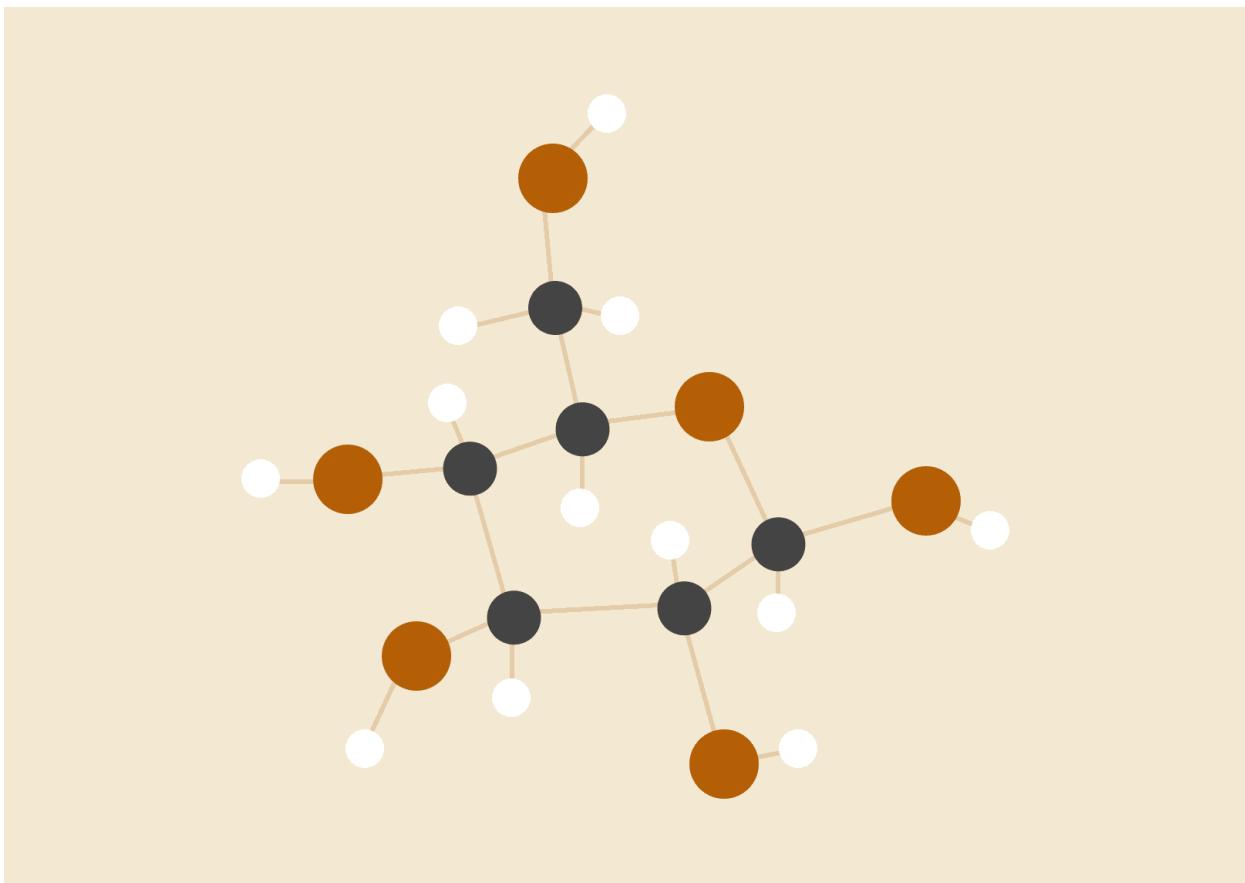


APPLICATION WEB JAVA

Comment créer un projet Maven et concevoir un framework MVC ?



Team MIATECH

21/03/2021

L3 informatique-MIAGE

INTRODUCTION

Maven est un outil de gestion de projet. Il est utilisé pour gérer la dépendance et la documentation.

Fondamentalement, il y a deux termes principaux que nous allons utiliser dans le projet **1) POM.xml, 2) Dépendance**

POM.XML est un fichier de configuration de projet. Qui gérera toutes les dépendances (Jar) du projet.

La dépendance est un JAR, ZIP de l'API qui sera utilisé dans le projet. Le même Jar sera conservé dans le pom.xml.

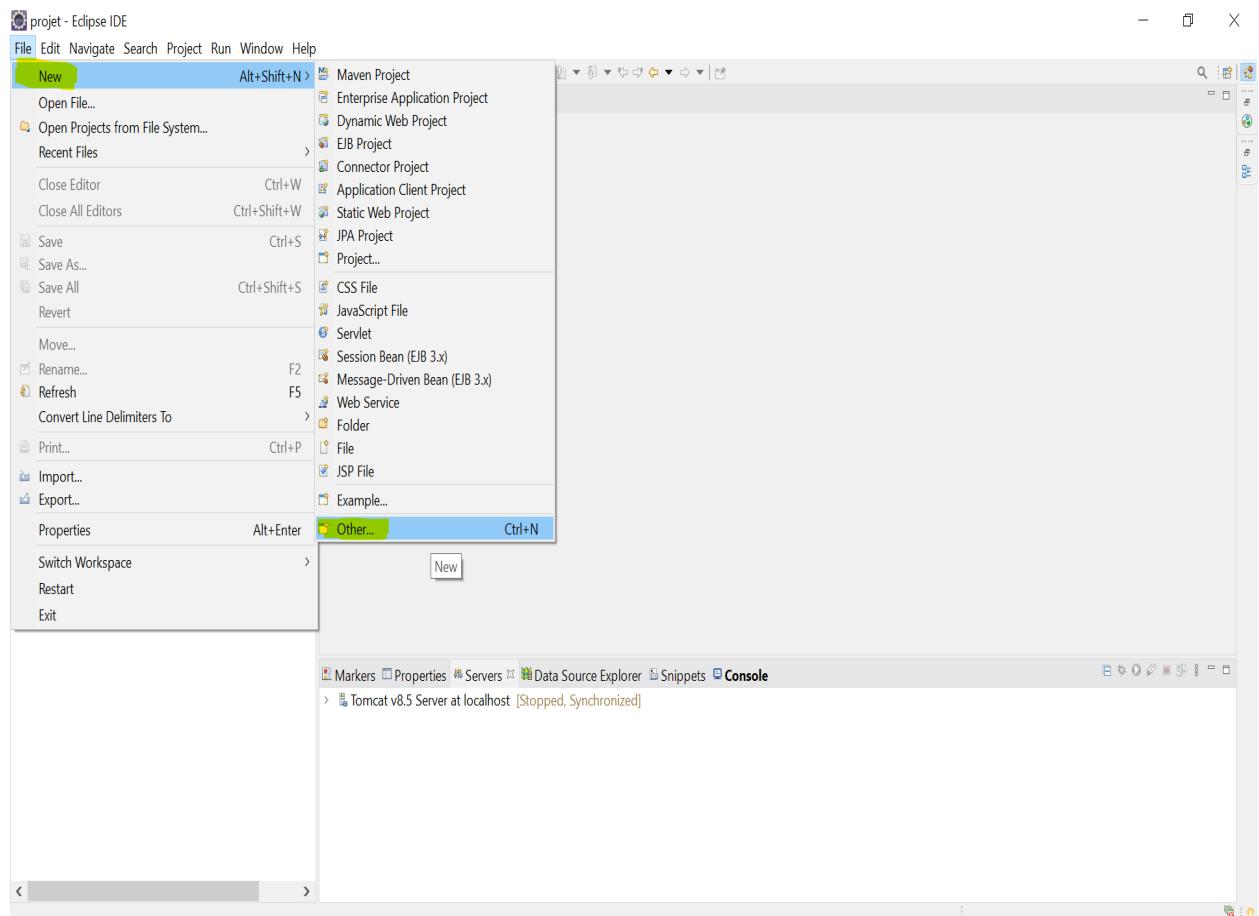
Nous allons voir comment créer un projet maven pour développer une application web en Java EE et comment concevoir l'architecture logicielle MVC.

PS : Avant de créer un projet, assurez-vous que JDK est installé sur votre machine, pour notre part on va utiliser Eclipse .

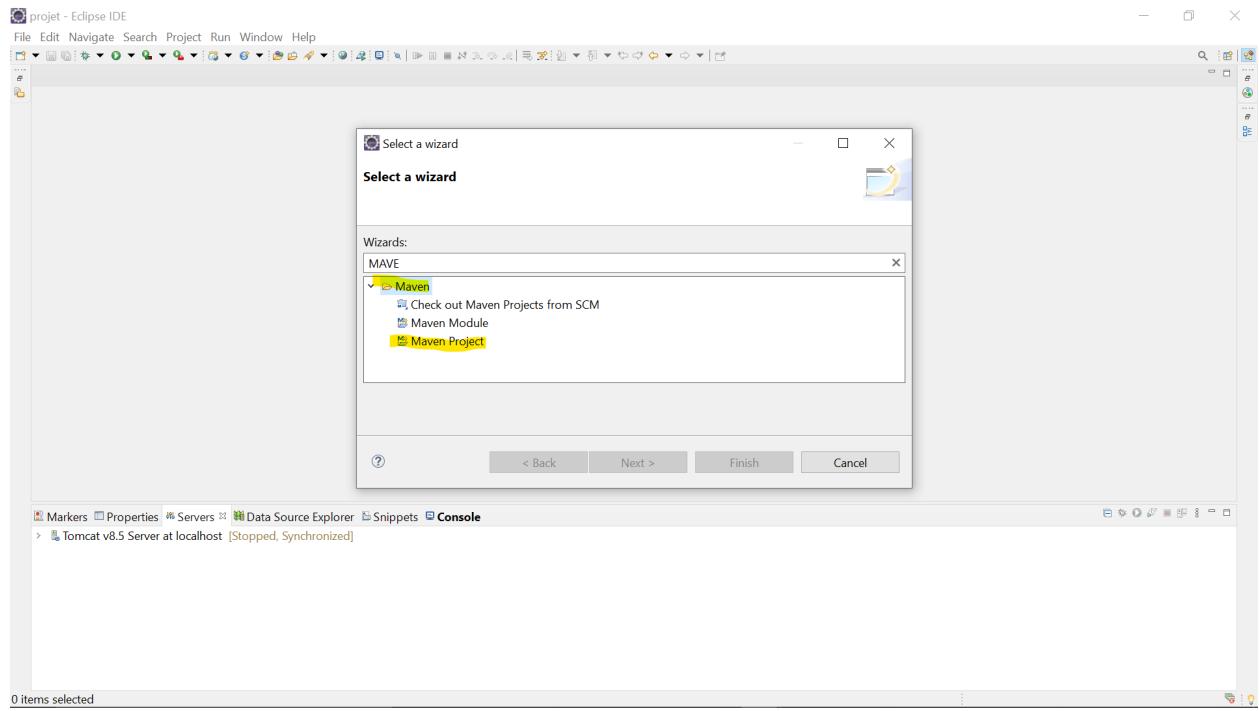
Comment créer un projet Maven et concevoir un cadre MVC

ETAPES POUR CREER UN MAVEN PROJECT

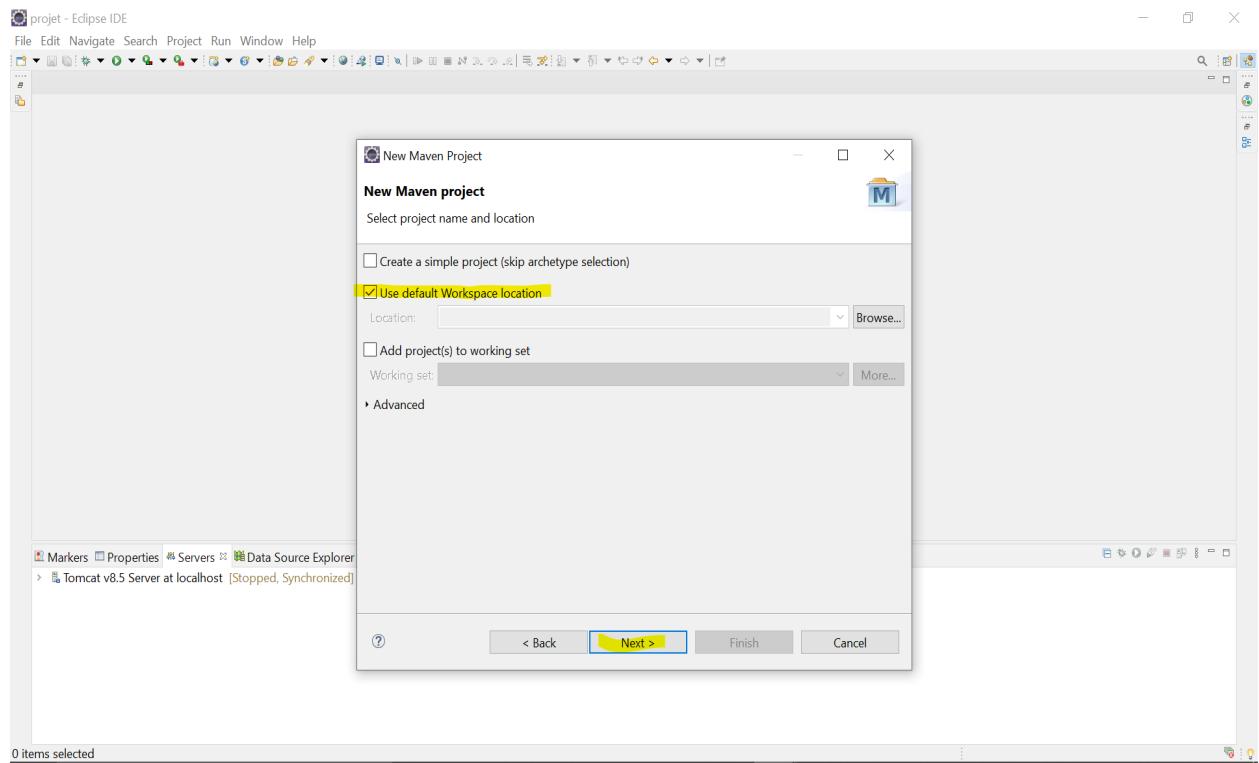
1. Lancer Eclipse
2. Pour créer un nouveau projet Maven Allez dans File ==>New ==> Other



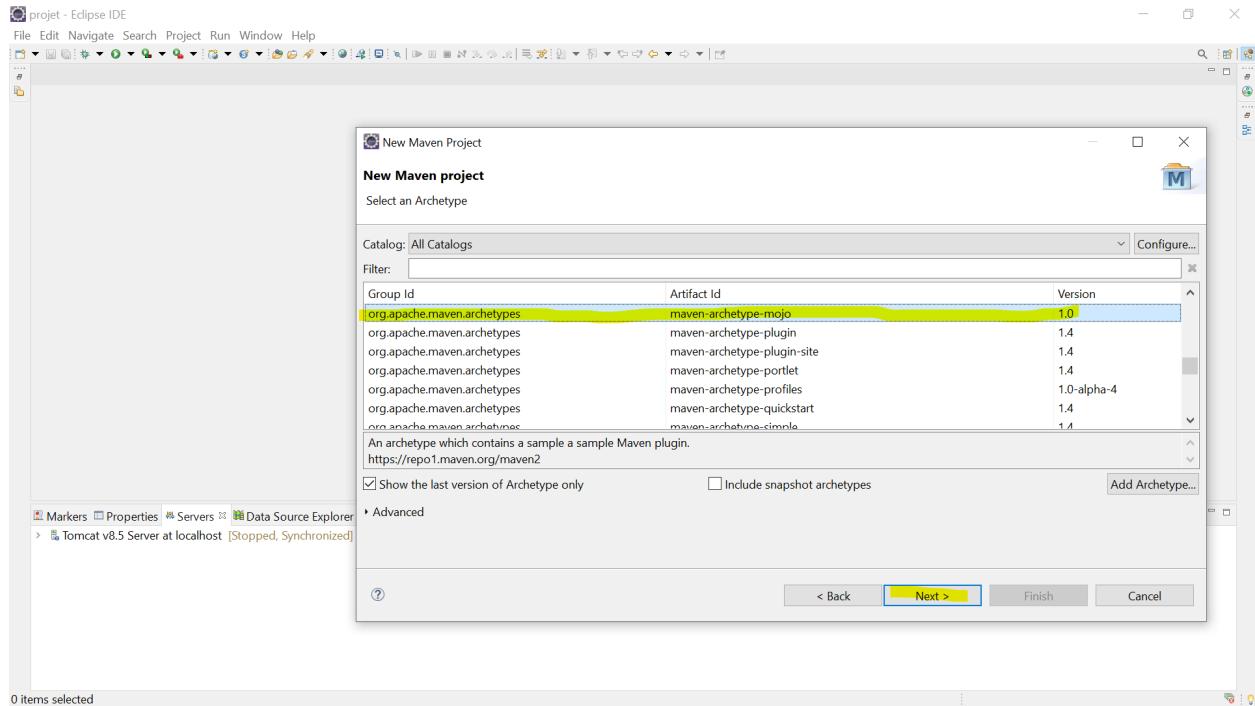
3. Recherchez «Maven» et sélectionnez «Projet Maven». Cliquez sur le bouton Suivant .



4. Sélectionnez l'espace de travail dans lequel vous souhaitez enregistrer le projet et cliquez sur le bouton Suivant .



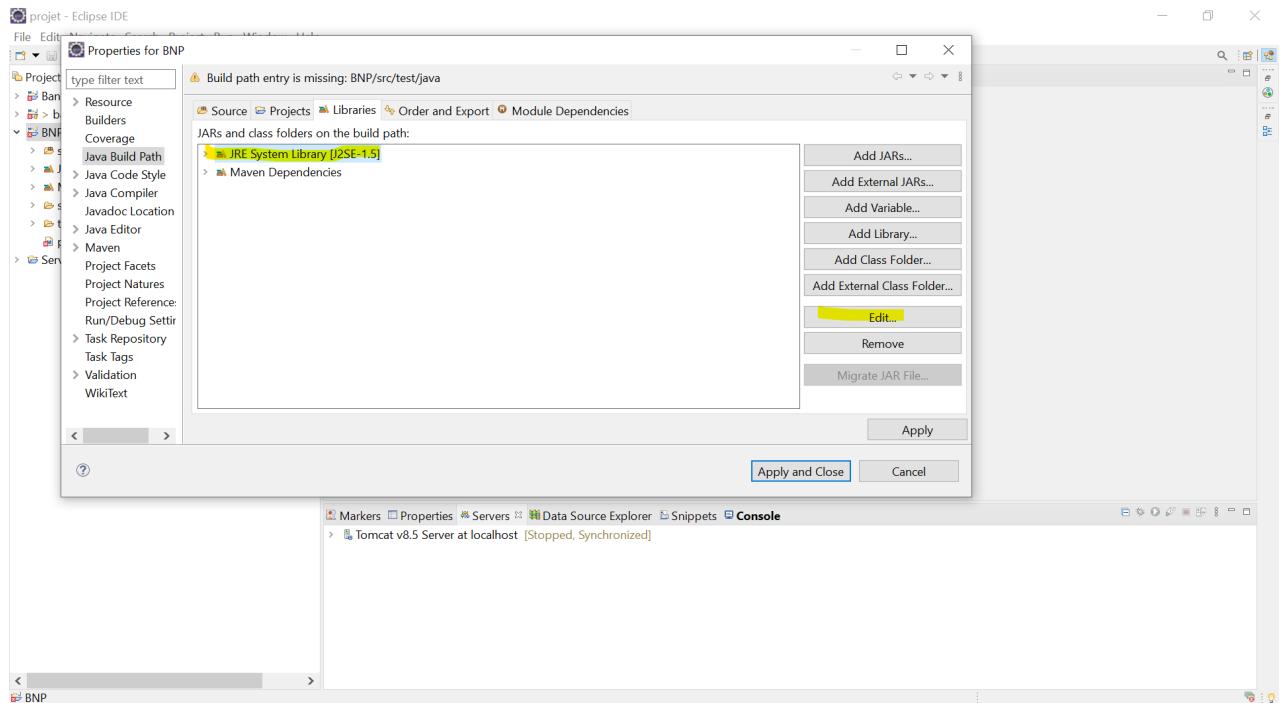
5. Sélectionnez l'archétype Maven «maven-archetype-webapp» et cliquez sur le bouton Suivant .



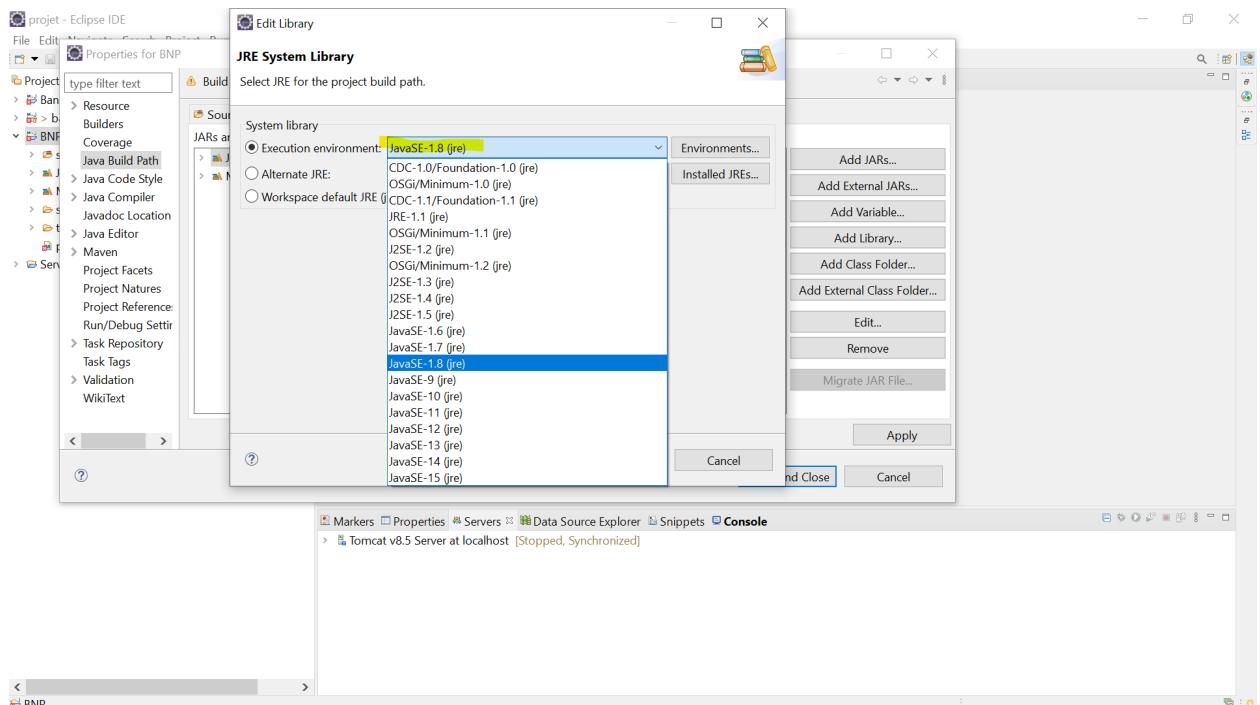
6. Entrez l' ID de groupe et le nom du projet et cliquez sur le bouton Terminer .

Comment changer la version Java d'un projet maven existant

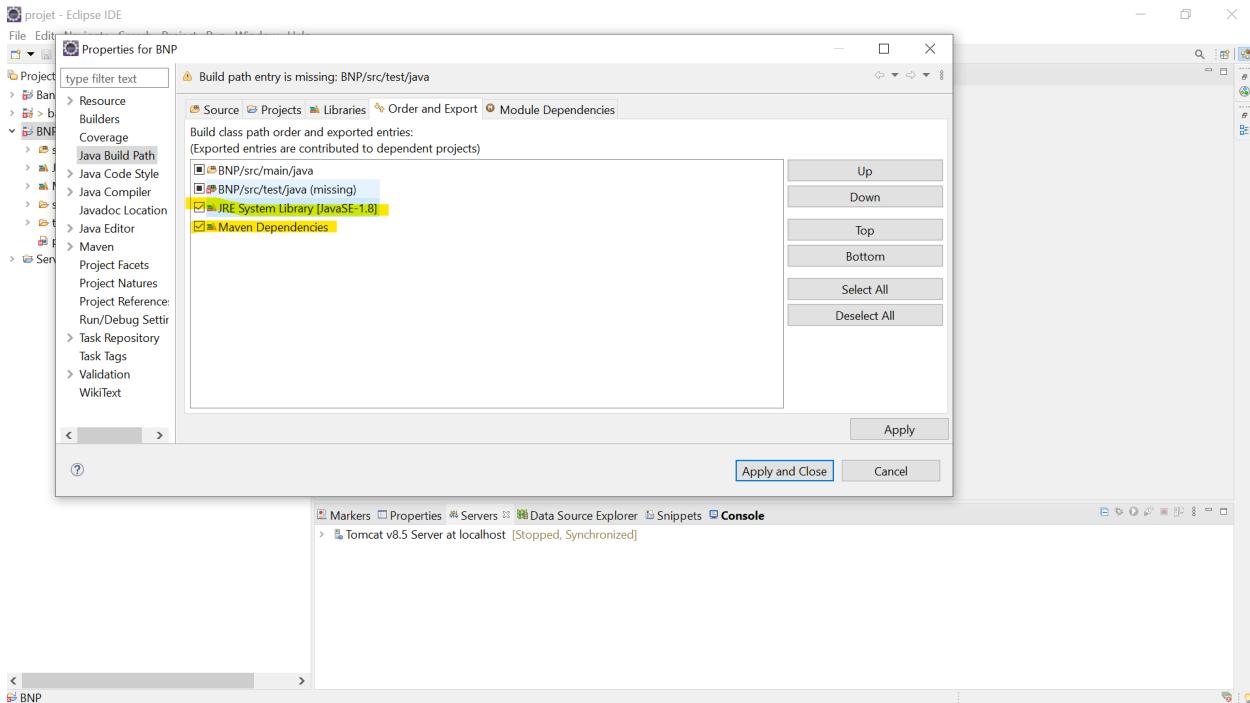
1. Cliquez avec le bouton droit sur le projet et accédez au build path ==> configure build path et Déplacez-vous vers le chemin de construction Java et cliquez sur l'onglet Bibliothèques et modifiez la «Bibliothèque système JRE»



2. Changez-le en 1.8 et cliquez pour terminer.



3.Cliquez sur l'onglet Other and Export et cochez toutes les cases non cochées et cliquez pour appliquer et fermer.



Configuration du serveur

Étape 1) Téléchargez tomcat 8.5 >> [cliquez ici](#)

Étape 2) Extraire le fichier et le copier à n'importe quel emplacement de votre machine, par exemple «lecteur C».

Étape 3) Encore une fois Basculez vers l'outil sts / Eclipse et recherchez la fenêtre du serveur. Suivez le chemin pour trouver la fenêtre du serveur

=> Windows => Afficher la vue => Autres => Serveur.

Étape 4) Cliquez avec le bouton droit de la souris dans la fenêtre du serveur et sélectionnez.

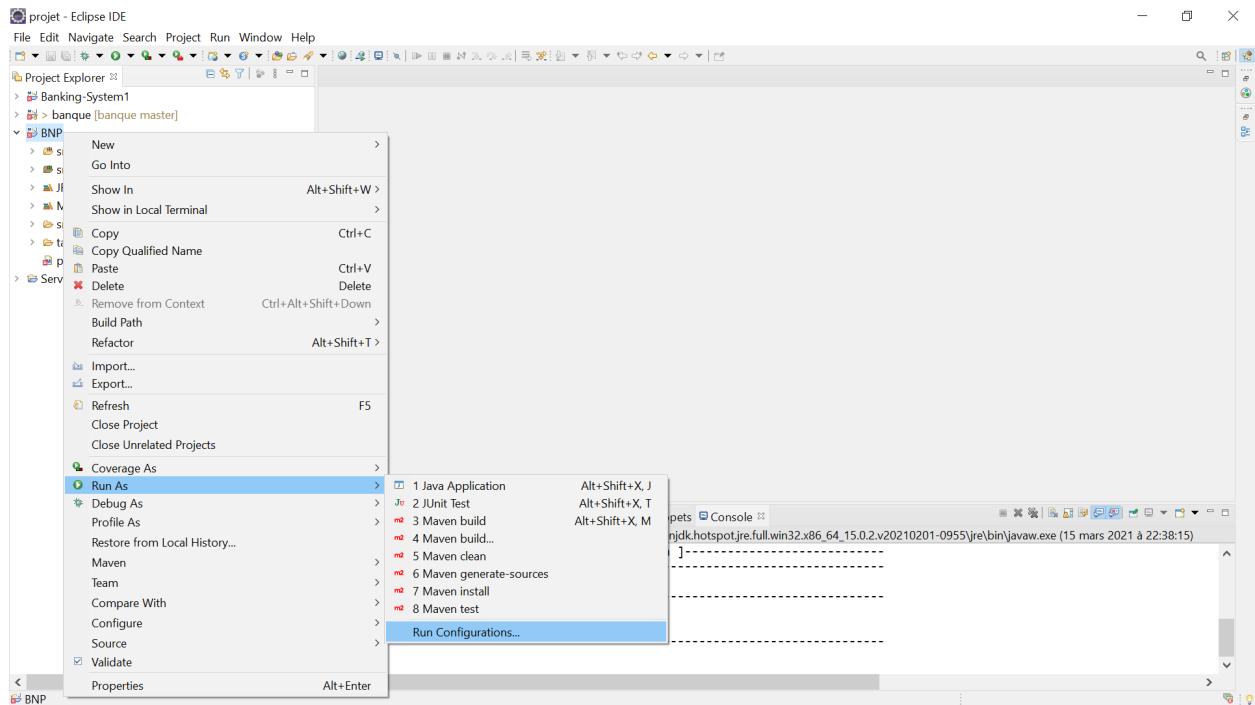
Nouveau => serveur => Apache => Tomcat 8.5 Server => Suivant.

Étape 5) Parcourez le dossier tomcat que nous avions copié dans «lecteur C».

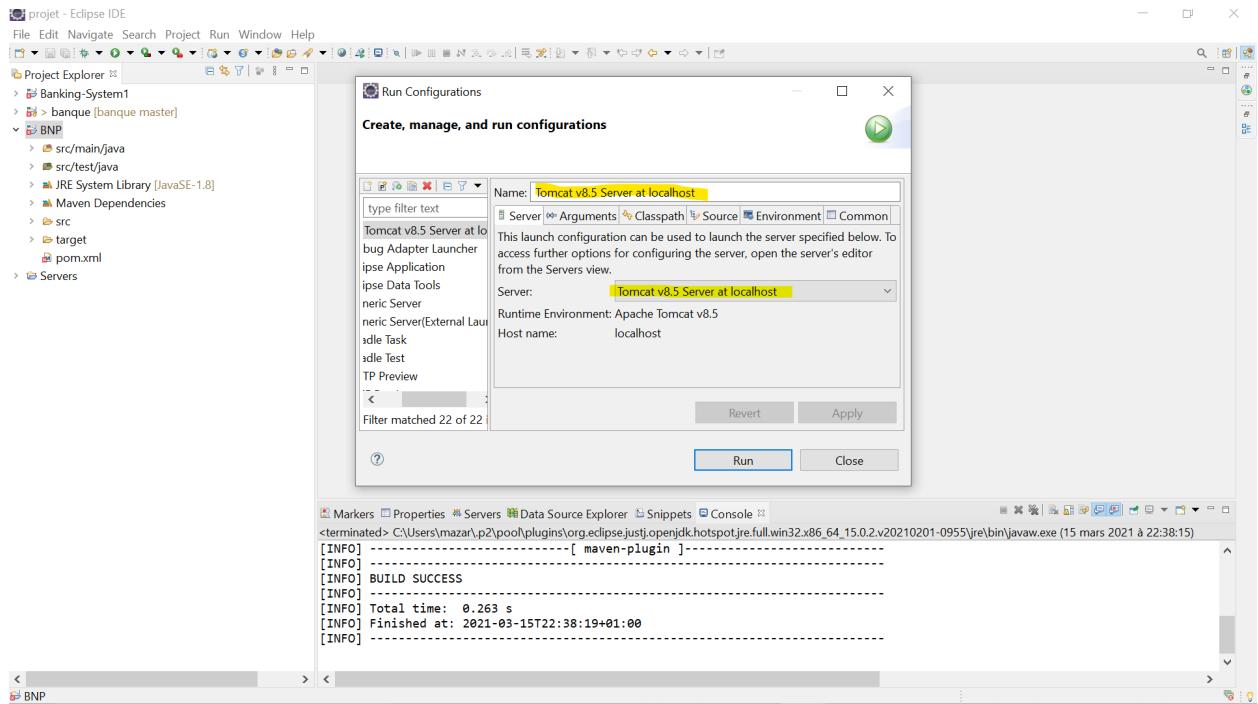
Étape 6) Cliquez sur Suivant et Terminer.

Lancez l'application

1. Cliquez avec le bouton droit sur le projet et cliquez sur Exécuter en tant que ==> Exécuter sur le serveur.



2. Sélectionnez l'option «Définir manuellement un nouveau serveur», sélectionnez la version tomcat et cliquez sur le bouton RUN.



Technologies et outils de développement d'applications Web Java

Nous allons préparer notre machine pour commencer le développement. Voyons les exigences logicielles et les technologies que nous allons utiliser dans le développement d'applications Web Java.

- *Exigence logicielle:*
 - Outil Eclipse / STS.
 - MYSQL avec MySQL Workbench.
- *La technologie*
 1. Front End: HTML, JSP, CSS.
 2. Côté serveur: Servlet.
 3. Arrière-plan: MYSQL.
 4. Serveur: Tomcat.

Faisons une brève introduction aux technologies et aux outils que nous allons utiliser pour le déploiement futur de notre application web Java

Qu'est-ce que le HTML?

HTML est utilisé pour créer un design frontal ou GUI comme les champs d'entrée, boutons, en-tête, texte, titre de page, menu déroulant, menus ETC. HTML contient des balises pour afficher les éléments. Il y a beaucoup de balises disponibles dans le HTML qui sont utilisées selon l'exigence.

Qu'est-ce que JavaScript?

JavaScript est un langage de programmation principalement utilisé pour définir la validation côté client. De nos jours beaucoup de frameworks sont disponible comme Node js, View Js, react js.

Qui prend en charge le site client ainsi que le site serveur. C'est un open source donc il est largement utilisé. Nous utilisons js dans notre projet pour créer une validation initiale.

Qu'est-ce que CSS?

CSS est utilisé pour transformer les éléments. Il aide à créer des pages HTML plus attrayantes. CSS et ses versions mises à jour comme css3 et css5 prennent en charge l'animation et certaines choses de conception avancée aussi.

Ce qui rend notre application plus attractive. Nous utilisons CSS3 pour une utilisation de base comme pour gérer les passions des divisions et pour ajouter de la couleur dans l'en-tête et le pied de page.

Qu'est-ce que Servlet?

Servlet est utilisé comme langage de programmation côté serveur. Qui est responsable de traiter la requête qui vient de la partie vue. Comme nous utilisons l'architecture MVC, la partie controller serait gérée par le seul servlet des méthodes dopost et doGet.

Qu'est-ce que MYSQL?

MYSQL est un système de gestion de base de données relationnelle. Pour exploiter la base de données, nous utilisons MYSQL workbench qui est assez simple et facile à comprendre.

Le serveur MYSQL fonctionne sur le port numéro 3306 avec le nom d'utilisateur « root » et le mot de passe « root ».

Qu'est-ce que le serveur?

Le serveur est utilisé pour déployer l'application web Java. Actuellement nous utilisons tomcat 8.5 et mettons à niveau la version en fonction des besoins.

Par défaut, Tomcat s'exécutera sur le numéro de port 8080. s'il est occupé, on. Apache Tomcat est un serveur opensource pour déployer et exécuter l'application nous pouvons le changer.

Modèle Vue Contrôleur

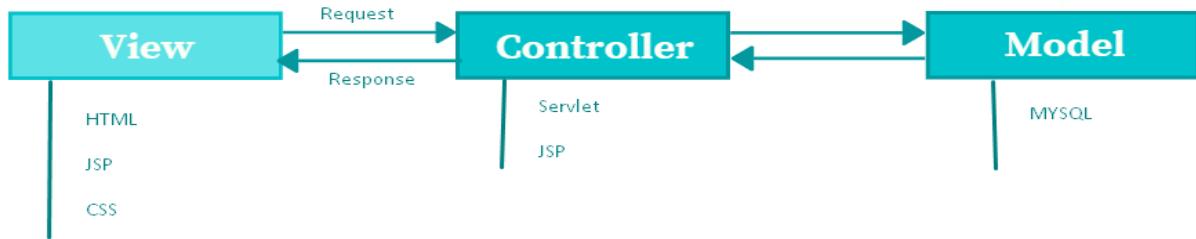
Model View Controller(MVC) est un moyen de gérer les ressources. Ici, les ressources désignent les fichiers et les technologies utilisés pour développer une application.

Ainsi MVC a divisé ces ressources en trois parties 1) Model 2) View 3) Controller.

Model: sera responsable de gérer les ressources liées à la base de données ou à l'arrière-plan. Ainsi Sous le package de model inclura toutes les classes qui contiennent le code des transactions de base de données Comme obtenir des données de base de données, Insérer des données dans la base de données, Supprimer ou Mettre à jour des données.

View : sera responsable de gérer les ressources liées à l'interface utilisateur comme HTML, CSS, Images, JSP, etc. Comme son nom l'indique, la partie front end ou view de l'application passera sous la vue.

Controller: gérera toutes les requêtes et les réponses qui sont des servlets, Lorsque User/View envoie une requête à Database/Model. Ensuite, le contrôleur recevra la requête et enverra la réponse en fonction de la demande.



Qu'est-ce que le bootstrap?

Bootstrap est un framework CSS qui fournit des classes de pré-implémentation pour concevoir des éléments Web réactifs. Des milliers de classes sont disponibles dans la bibliothèque Bootstrap. Explorez cette bibliothèque depuis le site officiel de bootstrap.

Comment intégrer le bootstrap dans un projet Java?

Il existe deux façons d'intégrer bootstrap dans le projet Java

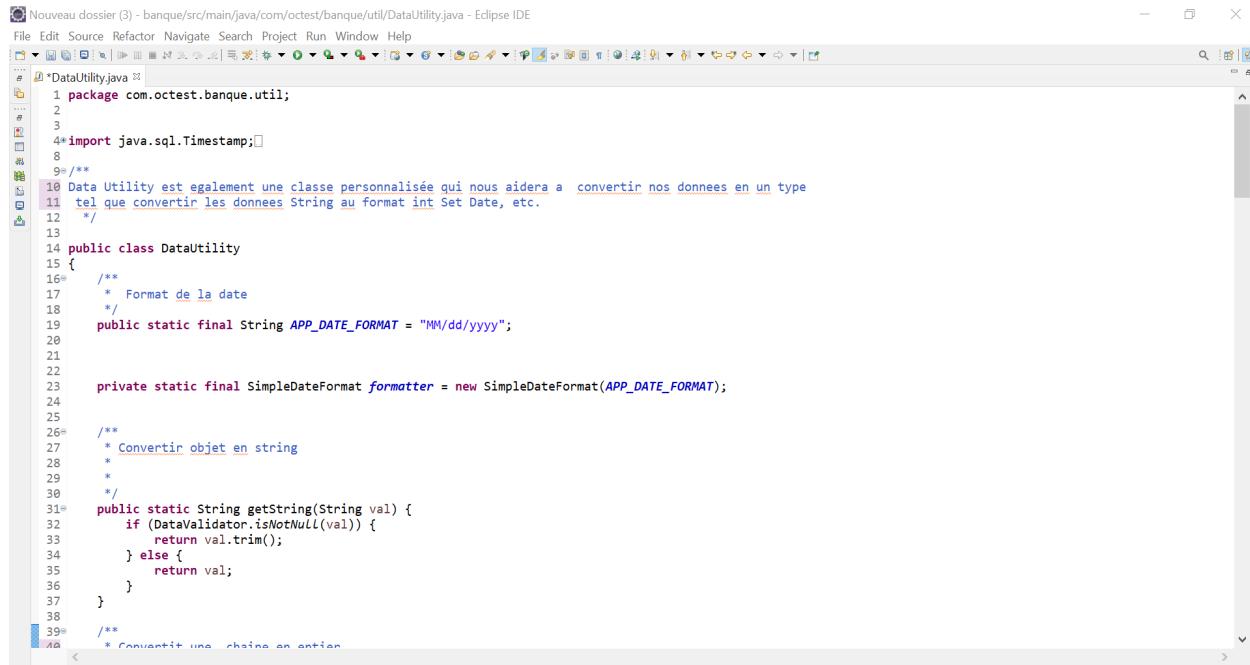
1. Téléchargez BootStrap et importez-le.
2. Transmettez les librairies Live pour utiliser bootstrap.

Comment valider les champs d'entrée en Java

Créons une classe personnalisée (`DataValidator.java`). Cette classe sera responsable de valider toutes les données qui viendront des champs d'entrées. Donc, nous allons écrire quelques méthodes qui nous aideront à valider les champs d'entrée en java.

Classe Data Utility

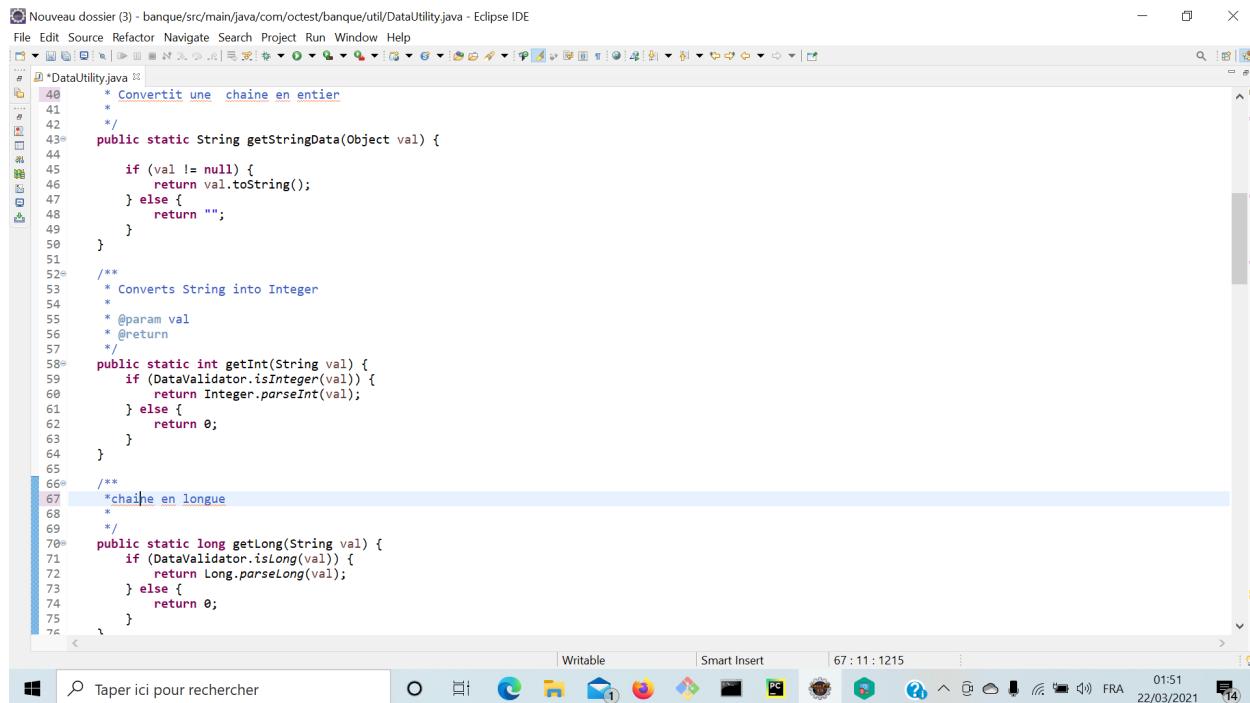
Data Utility est également une classe personnalisée qui nous aidera à convertir nos données en un type comme convertir les données string en format int Set Date, etc.



```

Nouveau dossier (3) - banque/src/main/java/com/octest/banque/util/DataUtility.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
DataUtility.java
1 package com.octest.banque.util;
2
3
4 import java.sql.Timestamp;
5
6
7 /**
8 * Data Utility est également une classe personnalisée qui nous aidera à convertir nos données en un type
9 * tel que convertir les données String au format int Set Date, etc.
10 */
11
12
13 public class DataUtility {
14
15     /**
16      * Format de la date
17      */
18     public static final String APP_DATE_FORMAT = "MM/dd/yyyy";
19
20
21
22     private static final SimpleDateFormat formatter = new SimpleDateFormat(APP_DATE_FORMAT);
23
24
25
26     /**
27      * Convertit objet en string
28      *
29      */
30
31     public static String getString(String val) {
32         if (DataValidator.isNotNull(val)) {
33             return val.trim();
34         } else {
35             return val;
36         }
37     }
38
39     /**
40      * Convertit une chaîne en entier
41      *
42      */
43     public static String getStringData(Object val) {
44
45         if (val != null) {
46             return val.toString();
47         } else {
48             return "";
49         }
50     }
51
52     /**
53      * Converts String into Integer
54      *
55      * @param val
56      * @return
57      */
58     public static int getInt(String val) {
59         if (DataValidator.isInteger(val)) {
60             return Integer.parseInt(val);
61         } else {
62             return 0;
63         }
64     }
65
66     /**
67      * chaîne en longue
68      *
69      */
70     public static long getLong(String val) {
71         if (DataValidator.isLong(val)) {
72             return Long.parseLong(val);
73         } else {
74             return 0;
75         }
76     }

```



```

Taper ici pour rechercher

```

Classe Servlet Utility

c'est une classe personnalisée qui nous aidera pendant le développement d'applications Web Java. Nous allons écrire quelques méthodes communes pour gérer la demande et la réponse des servlets ou de toute autre activité liée aux servlets Java.

Voici quelques méthodes dont nous allons discuter. Vous pouvez ajouter à nouveau sans dépendre de l'exigence.

forward (page, request, response)

Dans cette méthode, nous utilisons getRequestDispatcher (page); il aidera à transmettre une page.

redirect(page, demande, réponse)

Dans cette méthode, nous utilisons sendRedirect(page); il aide à rediriger vers une autre page.

Code source de la classe Servlet Utility

Toute la méthode de cette classe est juste utilisée pour gérer les ressources liées au servlet. Comme

- Envoyez une demande d'une page à une autre .
- Envoyer un message d'erreur.
- Pour faire avancer la page.
- Pour rediriger une page.
- Obtenez et définissez la liste.

```
ServletUtility.java ::
```

```
23
24 public class ServletUtility {
25
26     public static void forward(String page, HttpServletRequest request, HttpServletResponse response)
27             throws IOException, ServletException {
28
29         RequestDispatcher rd = request.getRequestDispatcher(page);
30         System.out.println(page);
31         rd.forward(request, response);
32     }
33
34     public static void redirect(String page, HttpServletRequest request, HttpServletResponse response)
35             throws IOException, ServletException {
36         response.sendRedirect(page);
37     }
38
39     public static void handleException(Exception e, HttpServletRequest request, HttpServletResponse response)
40             throws IOException, ServletException {
41         request.setAttribute("exception", e);
42         ServletUtility.forward(BSView.ERROR_CTL, request, response);
43         e.printStackTrace();
44     }
45
46     public static String getErrorMessage(String property, HttpServletRequest request) {
47         String val = (String) request.getAttribute(property);
48         if (val == null) {
49             return "";
50         } else {
51             return val;
52         }
53     }
54
55     public static String getMessage(String property, HttpServletRequest request) {
56         String val = (String) request.getAttribute(property);
57         if (val == null) {
58             return "";
59         } else {
60             return val;
61         }
62     }
63
64     public static void setErrorMessage(String msg, HttpServletRequest request) {
65         request.setAttribute(BaseCtl.MSG_ERROR, msg);
66     }
67
68     public static String getErrorMessage(HttpServletRequest request) {
69         String val = (String) request.getAttribute(BaseCtl.MSG_ERROR);
70         if (val == null) {
71             return "";
72         } else {
73             return val;
74         }
75     }
76
77     public static void setSuccessMessage(String msg, HttpServletRequest request) {
78         request.setAttribute(BaseCtl.MSG_SUCCESS, msg);
79     }
80
81     public static String getSuccessMessage(HttpServletRequest request) {
82         String val = (String) request.getAttribute(BaseCtl.MSG_SUCCESS);
83         if (val == null) {
84             return "";
85         } else {
86             return val;
87         }
88     }
89 }
```

```
ServletUtility.java ::
```

```
53
54
55     public static String getMessage(String property, HttpServletRequest request) {
56         String val = (String) request.getAttribute(property);
57         if (val == null) {
58             return "";
59         } else {
60             return val;
61         }
62     }
63
64     public static void setErrorMessage(String msg, HttpServletRequest request) {
65         request.setAttribute(BaseCtl.MSG_ERROR, msg);
66     }
67
68     public static String getErrorMessage(HttpServletRequest request) {
69         String val = (String) request.getAttribute(BaseCtl.MSG_ERROR);
70         if (val == null) {
71             return "";
72         } else {
73             return val;
74         }
75     }
76
77     public static void setSuccessMessage(String msg, HttpServletRequest request) {
78         request.setAttribute(BaseCtl.MSG_SUCCESS, msg);
79     }
80
81     public static String getSuccessMessage(HttpServletRequest request) {
82         String val = (String) request.getAttribute(BaseCtl.MSG_SUCCESS);
83         if (val == null) {
84             return "";
85         } else {
86             return val;
87         }
88     }
89 
```

Qu'est-ce que JDBCDataSource

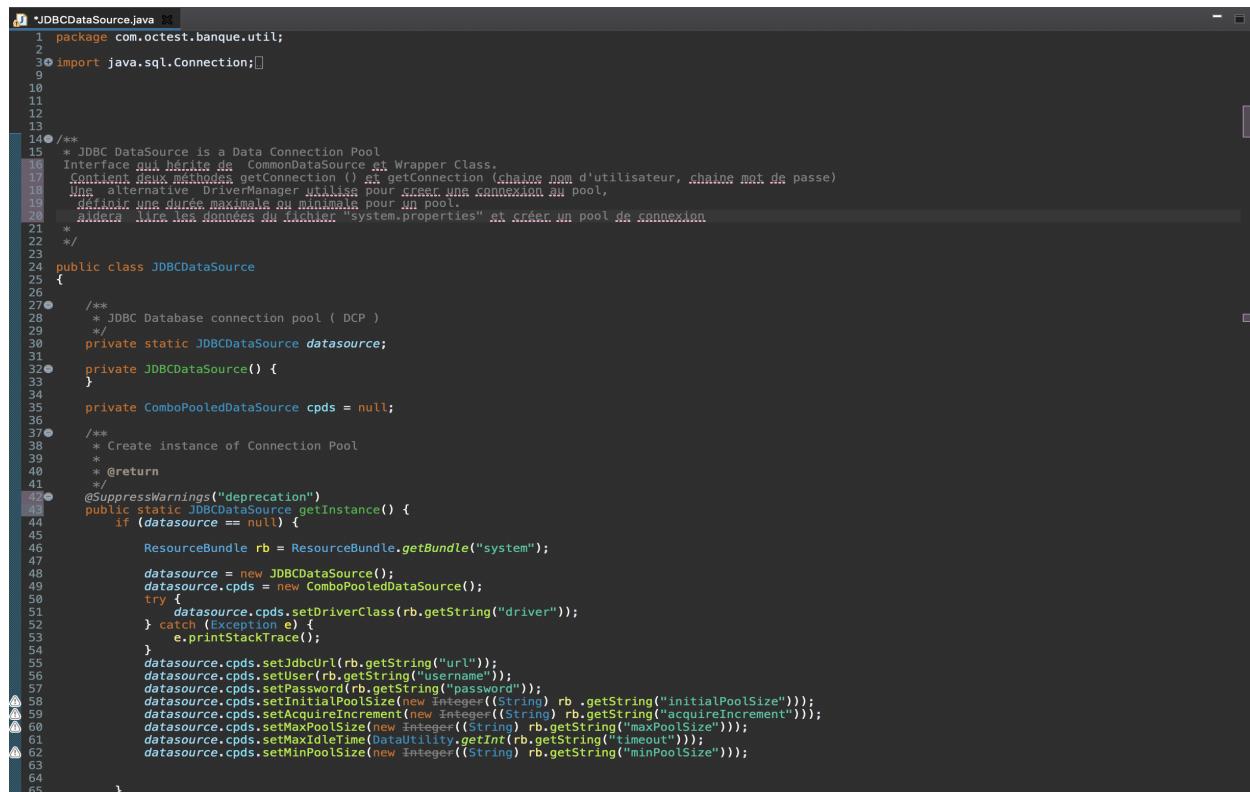
JDBC (sigle de **Java Database Connectivity**) est un logiciel qui permet à des applications informatiques écrites pour la machine virtuelle Java de manipuler des bases de données.

Qu'est-ce que le pool JDBC DataSource Connection

Un pool est un groupe de ressources qui est prêt à fournir les services. Le plus grand avantage de la mise en commun est la réutilisation des ressources qui contribuent à améliorer les performances.

JDBC DataSource Connection pool est un groupe prédefinis prêt à utiliser des objets de connexion. Si nous avons besoin de communiquer plusieurs fois avec la base de données, il n'est pas recommandé de créer un objet de connexion distinct à chaque fois. Parce que la création et la destruction d'objets de connexion à chaque fois affectent les performances de l'application. Pour surmonter ce problème de performances, nous utilisons la mise en commun datasource connection.

Avantage de la connexion JDBC DataSource est que nous pouvons utiliser la même connexion plusieurs fois qui va certainement améliorer les performances de l'application.



```
1 package com.octest.banque.util;
2
3 import java.sql.Connection;
4
5
6
7
8
9
10
11
12
13
14 /**
15 * JDBC DataSource is a Data Connection Pool
16 * Interface qui hérite de CommonDataSource en Wrapper Class,
17 * Contient deux méthodes getConnection () et getConnection (chaîne nom d'utilisateur, chaîne mot de passe)
18 * Une alternative DriverManager utilisée pour créer une connexion au pool,
19 * définir une limite maximale au maximum pour un pool.
20 * Limite max connexions au maximum "system.properties" et créer un pool de connexion
21 */
22 */
23
24 public class JDBCDataSource {
25
26
27 /**
28 * JDBC Database connection pool ( DCP )
29 */
30 private static JDBCDataSource datasource;
31
32 private JDBCDataSource() {
33 }
34
35 private ComboPooledDataSource cpds = null;
36
37 /**
38 * Create instance of Connection Pool
39 *
40 * @return
41 */
42 @SuppressWarnings("deprecation")
43 public static JDBCDataSource getInstance() {
44     if (datasource == null) {
45         ResourceBundle rb = ResourceBundle.getBundle("system");
46
47         datasource = new JDBCDataSource();
48         datasource.cpds = new ComboPooledDataSource();
49         try {
50             datasource.cpds.setDriverClass(rb.getString("driver"));
51         } catch (Exception e) {
52             e.printStackTrace();
53         }
54         datasource.cpds.setJdbcUrl(rb.getString("url"));
55         datasource.cpds.setUser(rb.getString("username"));
56         datasource.cpds.setPassword(rb.getString("password"));
57         datasource.cpds.setInitialPoolSize(new Integer((String) rb .getString("initialPoolSize")));
58         datasource.cpds.setAcquireIncrement(new Integer((String) rb.getString("acquireIncrement")));
59         datasource.cpds.setMaxPoolSize(new Integer((String) rb.getString("maxPoolsize")));
60         datasource.cpds.setMaxIdleTime(DataUtility.getInt((String) rb.getString("timeout")));
61         datasource.cpds.setMinPoolSize(new Integer((String) rb.getString("minPoolSize")));
62     }
63 }
64 }
```

Fichier System.properties

Créez un fichier de propriété pour définir le nom d'utilisateur, le mot de passe ou pour gérer les paramètres de connexion de la base de données.



```
system.properties ::  
1#Création d'un fichier de propriétés pour définir le nom d'utilisateur,  
2#le mot de passe et pour gérer les paramètres de connexion à la base de données.  
3  
4#Generic error messages  
5error.require={0} is required  
6error.integer={0} must be an integer  
7error.date={0} is invalid date  
8error.email={0} is invalid  
9error.name={0} must be an Alphabets  
10error.invalid={0} is Invalid  
11error.confirmPassword={0} is Not Matched  
12error.password={0} Must have atleast one alphanumeric and 6 char  
13  
14  
15#Database connection Parameters  
16url=jdbc:mysql://localhost:3306/banking-system?useSSL=false&useUnicode=true&useJDBCCompliantTimezoneShift=true&useLegacyDatetimeCode=false&serverTimezone=UTC  
17driver=com.mysql.cj.jdbc.Driver  
18username=root  
19password=root  
20#DATABASE=JDBC  
21#service=javaBean  
22acquireIncrement = 10  
23initialPoolSize = 10  
24maxPoolSize = 100  
25minPoolSize = 10  
26timeout = 10  
27  
28  
29  
30
```

Ajouter maven dépendance pour Mysql en **pom.xml**

```
<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/maven-v4_0_0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>com.octest.banque</groupId>
  <artifactId>banque</artifactId>
  <packaging>war</packaging>
  <version>0.0.1-SNAPSHOT</version>
  <name>Banking System Maven Webapp</name>
  <url>http://maven.apache.org</url>
<dependencies>
  <dependency>
    <groupId>junit</groupId>
    <artifactId>junit</artifactId>
    <version>3.8.1</version>
    <scope>test</scope>
  </dependency>
  <dependency>
    <groupId>mysql</groupId>
    <artifactId>mysql-connector-java</artifactId>
    <version>8.0.15</version>
  </dependency>
  <dependency>
    <groupId>javax.servlet</groupId>
    <artifactId>javax.servlet-api</artifactId>
    <version>3.1.0</version>
  </dependency>
  <dependency>
    <groupId>log4j</groupId>
    <artifactId>log4j</artifactId>
    <version>1.2.14</version>
  </dependency>
</dependencies>

```

Créez une classe **JDBCDataSource.java** sous le package utilitaire.

La méthode de la classe **JDBCDataSource** aidera à lire les données du fichier **system.properties** et à créer un pool de connexion que nous pouvons utiliser dans le projet à n'importe quel endroit.

- Obtenez la connexion
- Connexion étroite
- Restauration des transactions

```

*JDBCDataSource.java */
24 public class JDBCDataSource
25{ /**
26  * JDBC Database connection pool ( DCP )
27  */
28 private static JDBCDataSource datasource;
29
30 private JDBCDataSource() {
31 }
32
33 private ComboPooledDataSource cpds = null;
34
35 /**
36  * Create instance of Connection Pool
37  *
38  * @return
39 */
40 public static JDBCDataSource getInstance() {
41     if (datasource == null) {
42
43         ResourceBundle rb = ResourceBundle.getBundle("system");
44
45         datasource = new JDBCDataSource();
46         datasource.cpds = new ComboPooledDataSource();
47         try {
48             datasource.cpds.setDriverClass(rb.getString("driver"));
49         } catch (Exception e) {
50             e.printStackTrace();
51         }
52         datasource.cpds.setJdbcUrl(rb.getString("url"));
53         datasource.cpds.setUser(rb.getString("username"));
54         datasource.cpds.setPassword(rb.getString("password"));
55         datasource.cpds.setInitialPoolSize(new Integer((String) rb .getString("initialPoolSize")));
56         datasource.cpds.setAcquireIncrement(new Integer((String) rb.getString("acquireIncrement")));
57         datasource.cpds.setMaxPoolSize(new Integer((String) rb.getString("maxPoolSize")));
58         datasource.cpds.setMaxIdleTime(DataUtility.getInt(rb.getString("timeout")));
59         datasource.cpds.setMinPoolSize(new Integer((String) rb.getString("minPoolSize")));
60     }
61 }

```

```

*JDBCDataSource.java */
66 /**
67  * Gets the connection from ComboPooledDataSource
68  *
69  * @return connection
70  */
71 public static Connection getConnection() throws Exception {
72     return getInstance().cpds.getConnection();
73 }
74
75 /**
76  * Closes a connection
77  *
78  * @param connection
79  * @throws Exception
80  */
81 public static void closeConnection(Connection connection) {
82     if (connection != null) {
83         try {
84             connection.close();
85         } catch (Exception e) {
86         }
87     }
88 }
89
90 public static void trnRollback(Connection connection)
91     throws ApplicationException {
92     if (connection != null) {
93         try {
94             connection.rollback();
95         } catch (SQLException ex) {
96             throw new ApplicationException(ex.toString());
97         }
98     }
99 }
100 }
101

```

Créez une page d'inscription à l'aide de JSP, Servlet et MYSQL. Maintenant, nous allons mettre en place toutes les choses de base pour développer une application Web en javaEE.

Page d'inscription à l'aide de JSP, Servlet et MYSQL.

- Page d'enregistrement de conception à l'aide de JSP.
- Créez une base de données et une table à l'aide de MYSQL.
- Connecter la base de données en java à l'aide de JDBC MYSQL.

Page d'enregistrement de conception

Créer **registration.jsp** sous le dossier JSP et inclure l'en-tête.jsp et footer.jsp

UserRegistration.jsp

```
① UserRegistrationView.jsp
1 <!-- directive de page pour l'importation des différents classes dont nous avons besoins dans cette jsp -->
2 <%@page import="com.octest.banque.controller.UserRegistrationCtl"%>
3 <%@page import="com.octest.banque.util.ServletUtility"%>
4 <%@page import="com.octest.banque.util.Datautility"%>
5 <!-- directive de page indiquant le langage utilisée ainsi que le type de caractère utilisé pour la réponse HTTP -->
6 <%@ page language="java" contentType="text/html; charset=ISO-8859-1"%>
7   pageEncoding="ISO-8859-1"%>
8 <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
9<html lang="en">
10<head>
11 <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
12 <title>User Registration</title>
13 <!-- permet d'utiliser toutes les fonctionnalités ou classes de bootstrap -->
14 <link rel="stylesheet" href="https://code.jquery.com/ui/1.12.1/themes/base/jquery-ui.css">
15 <link rel="stylesheet" href="https://jqueryui.com/resources/demos/style.css">
16 <script src="https://code.jquery.com/jquery-1.12.4.js"></script>
17 <script src="https://code.jquery.com/ui/1.12.1/jquery-ui.js"></script>
18<script type="text/javascript">
19 $( function() {
20   $( "#datepicker" ).datepicker({
21     changeMonth: true,
22     changeYear: true
23   });
24 });
25 </script>
26 </head>
27<body>
28 <!-- inclure la page JSP du header -->
29 <%@ include file="Header.jsp" %>
30 <br>
31<nav
32   aria-label="breadcrumb" role="navigation">
33<ol class="breadcrumb">
34
35   <li class="breadcrumb-item active" aria-current="page">Registration</li>
36 </ol>
37 </nav>
```

```

UserRegistrationView.jsp :: 
20 <!-- inclure la page JSP du header -->
21 <%@ include file="Header.jsp" %>
22 <br>
23 <nav
24     aria-label="breadcrumb" role="navigation">
25     <ol class="breadcrumb">
26         <li class="breadcrumb-item active" aria-current="page">Registration</li>
27     </ol>
28 </nav>
29 <div col-md-5 img-thumbnail>
30     <div id="feedback"> <div class="container">
31 <div class="col-md-8">
32     <div class="form-area">
33         <form role="form" action="<%=BSView.USER_REGISTRATION_CTL%>" method="post" >
34             <jsp:useBean id="bean" class="com.octest.banque.bean.UserBean"
35                 scope="request"></jsp:useBean>
36
37             <input type="hidden" name="id" value="<%=bean.getId()%>">
38             <input type="hidden" name="createdBy" value="<%=bean.getCreatedBy()%>">
39             <input type="hidden" name="modifiedBy" value="<%=bean.getModifiedBy()%>">
40             <input type="hidden" name="createdDatetime" value="<%=DataUtility.getTimestamp(bean.getCreatedDatetime())%>">
41             <input type="hidden" name="modifiedDatetime" value="<%=DataUtility.getTimestamp(bean.getModifiedDatetime())%>">
42
43             <br style="clear:both">
44             <h3 style="margin-bottom: 15px; text-align: left;">Registration</h3>
45             <b><font color="red"> <%=ServletUtility.getErrorMessage(request)%>
46             </font></b>
47
48             <b><font color="Green"> <%=ServletUtility.getSuccessMessage(request)%>
49             </font></b>
50
51         </div>
52     </div>
53 </div>
54 </div>
55 </div>
56 <br>
57 <br>
58 <br>
59 <br>
60 <br>
61 <br>
62 <br>
63 <br>

UserRegistrationView.jsp :: 
64 <!-- inclure la page JSP du "Footer" -->
65 <%@ include file="Footer.jsp" %>
66 <br>
67 <br>
68 <br>
69 <br>
70 <br>
71 <br>
72 <br>
73 <br>
74 <br>
75 <br>
76 <br>
77 <br>
78 <br>
79 <br>
80 <br>
81 <br>
82 <br>
83 <br>
84 <br>
85 <br>
86 <br>
87 <br>
88 <br>
89 <br>
90 <br>
91 <br>
92 <br>
93 <br>
94 <br>
95 <br>
96 <br>
97 <br>
98 <br>
99 <br>
100 </body>
101 </html>

```

Base de données et installation dans MySQL

Créer une base de données

Créer une table utilisateur

The screenshot shows the MySQL Workbench interface with the following details:

- Schemas**: banking-system
- Tables**: User
- Query Editor**: SELECT * FROM `banking-system`.User;
- Result Grid** (Table Data):

ID	Login	Password	roleId	created_By	modified_by	created_datetime	modified_datetime
3	Admin123	Admin@321	1	root	root	2021-02-15 15:55:32	2021-02-15 15:55:32
5	karim	karim123	2	root	azertyui	2021-03-22 22:02:25	2021-03-22 22:02:25
6	hanane	aze123	2	root	azertyui	2021-03-22 22:02:33	2021-03-22 22:02:33
HULL	HULL	HULL	HULL	HULL	HULL	HULL	HULL

- Help Panel** on the right:

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

 - Result Grid
 - Form Editor
 - Field Types
 - Query Stats
 - Execution Plan

Créer une classe bean sous le package bean

Les Beans sont utilisés pour définir et obtenir une valeur. En d'autres termes, vous pouvez dire que ce sont des classes getter et setter pour les attributs.

```
*JDBCDataSource.java  *UserBean.java
1 package com.octest.banque.bean;
2
3 public class UserBean extends BaseBean {
4
5     /**
6      * Login of User
7      */
8     private String login;
9
10    /**
11     * Password of User
12     */
13
14    private String firstName;
15    private String LastName;
16    private String MobileNo;
17
18    public String getMobileNo() {
19        return MobileNo;
20    }
21
22    public void setMobileNo(String mobileNo) {
23        MobileNo = mobileNo;
24    }
25
26    private String password;
27    /**
28     * Confirm Password of User
29     */
30    private String confirmPassword;
31    /**
32     * Role Id of User
33     */
34    private long roleId;
35    /**
36     * Customer of User
37     */
38
39    private CustomerBean customer;
40
41
42
43
44
45
46    public CustomerBean getCustomer() {
47        return customer;
48    }
49
50    public void setCustomer(CustomerBean customer) {
51        this.customer = customer;
52    }
53
54    public String getFirstName() {
55        return firstName;
56    }
57
58    public void setFirstName(String firstName) {
59        this.firstName = firstName;
60    }
}
```

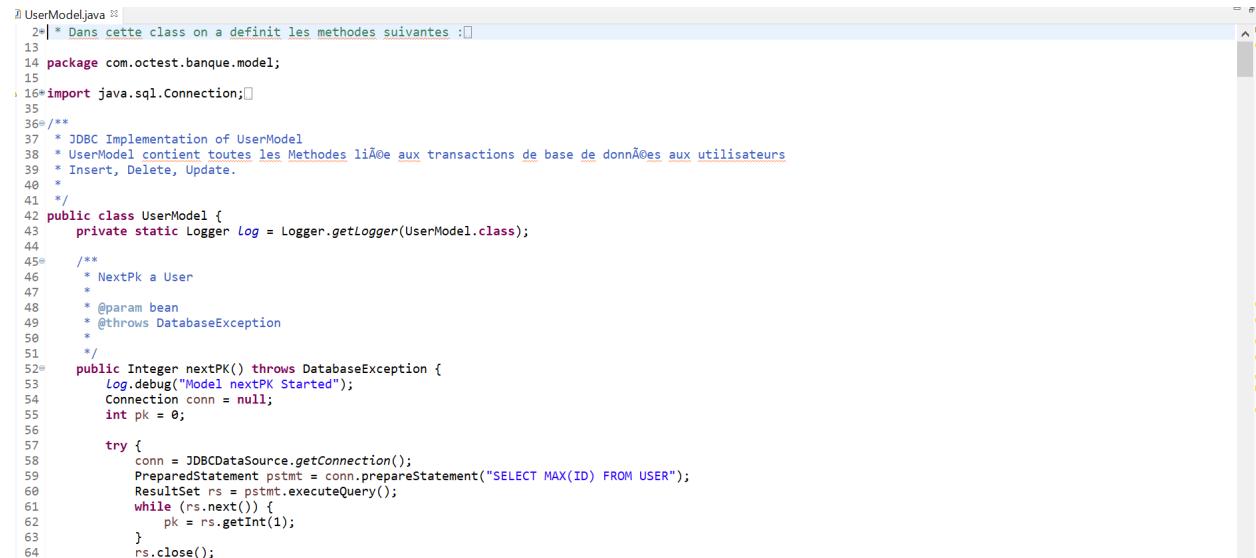
```
*JDBCDataSource.java  *UserBean.java
84@    /**
85     * @return Password Of User
86     */
87@    public String getPassword() {
88        return password;
89    }
90
91@    /**
92     * @param Password
93     *          To set User Password
94     */
95@    public void setPassword(String password) {
96        this.password = password;
97    }
98
99@    /**
100     * @return Confirm Password Of User
101     */
102@    public String getConfirmPassword() {
103        return confirmPassword;
104    }
105
106@    /**
107     * @param Confirm
108     *          PAssword To set User Confirm Password
109     */
110@    public void setConfirmPassword(String confirmPassword) {
111        this.confirmPassword = confirmPassword;
112    }
113
114
115
116@    /**
117     * @return ROLe Id Of User
118     */
119@    public long getRoleId() {
120        return roleId;
121    }
122
123@    /**
124     * @param Role
125     *          Id To set User ROLe Id
126     */
127@    public void setRoleId(long roleId) {
128        this.roleId = roleId;
129    }
130
131@    /**
132     * @return unSuccessfulLogin Of User
133     */
134
135@    public String getKey() {
136        return id + "";
137    }
138
139@    public String getValue() {
140        return login;
141    }
142 }
143
```

Crée un user model sous le package model.

UserModel sera utilisé pour contenir toutes les transactions de base de données liées à la méthode pour les utilisateurs. Comme Insérer, Supprimer, Mettre à jour. Dans ce code, nous ajoutons seulement une méthode pour enregistrer les utilisateurs

UserModel.java

Dans cette classe UserModel, nous allons utiliser les sources de données JDBC.



The screenshot shows a Java code editor with the file 'UserModel.java' open. The code is a JDBC implementation of the UserModel interface. It includes a package declaration, imports for Connection and logger, and a class definition with a constructor and a static logger. The class contains a method 'nextPK()' which retrieves the next primary key from the database. The code uses JDBC statements and result sets to perform the query. The code editor has a light blue background and syntax highlighting for Java keywords and comments.

```
2| * Dans cette class on a definit les methodes suivantes :□
13
14 package com.octest.banque.model;
15
16 import java.sql.Connection;□
35
36 /**
37  * JDBC Implementation of UserModel
38  * UserModel contient toutes les Methodes li e aux transactions de base de donn es aux utilisateurs
39  * Insert, Delete, Update.
40  *
41 */
42 public class UserModel {
43     private static Logger log = Logger.getLogger(UserModel.class);
44
45     /**
46      * NextPk a User
47      *
48      * @param bean
49      * @throws DatabaseException
50      *
51     */
52     public Integer nextPK() throws DatabaseException {
53         Log.debug("Model nextPK Started");
54         Connection conn = null;
55         int pk = 0;
56
57         try {
58             conn = JDBCDataSource.getConnection();
59             PreparedStatement pstmt = conn.prepareStatement("SELECT MAX(ID) FROM USER");
60             ResultSet rs = pstmt.executeQuery();
61             while (rs.next()) {
62                 pk = rs.getInt(1);
63             }
64             rs.close();
65         } catch (SQLException e) {
66             throw new DatabaseException("Error getting next PK: " + e.getMessage());
67         }
68     }
69 }
```

```

UserModel.java:75
75* Add a User
76*
77* @param bean
78* @throws DatabaseException
79*
80*/
81
82
83 public long add(UserBean bean) throws ApplicationException, DuplicateRecordException {
84     Connection conn = null;
85     int pk = 0;
86     UserBean existbean = findByLogin(bean.getLogin());
87     if (existbean != null) {
88         throw new DuplicateRecordException("Login Id already exists");
89     }
90     try {
91         conn = JDBCDataSource.getConnection();
92         pk = nextPK();
93         conn.setAutoCommit(false);
94         PreparedStatement pstmt = conn.prepareStatement("INSERT INTO USER VALUES(?, ?, ?, ?, ?, ?, ?, ?)");
95         pstmt.setInt(1, pk);
96         pstmt.setString(2, bean.getLogin());
97         pstmt.setString(3, bean.getPassword());
98         pstmt.setLong(4, bean.getRoleId());
99         pstmt.setString(5, bean.getCreatedBy());
100        pstmt.setString(6, bean.getModifiedBy());
101        pstmt.setTimestamp(7, bean.getCreatedDatetime());
102        pstmt.setTimestamp(8, bean.getModifiedDatetime());
103        pstmt.executeUpdate();
104        conn.commit(); // End transaction
105        pstmt.close();
106    } catch (Exception e) {
107
108        try {
109            conn.rollback();
110        } catch (Exception ex) {
111            ...
112        }
113    }
114 }

UserModel.java:120
120 */
121* Delete a User
122*
123* @param bean
124* @throws DatabaseException
125*
126*/
127
128
129 public void delete(UserBean bean) throws ApplicationException {
130
131     Connection conn = null;
132     try {
133         // Get connection:
134         conn = JDBCDataSource.getConnection();
135         conn.setAutoCommit(false); // Begin transaction
136         // Create an object for executing SQL statements
137         PreparedStatement pstmt = conn.prepareStatement("DELETE FROM USER WHERE ID=?");
138         // Setting parameters:
139         pstmt.setLong(1, bean.getId());
140         pstmt.executeUpdate();
141         conn.commit(); // End transaction
142         pstmt.close();
143
144     } catch (Exception e) {
145
146         try {
147             conn.rollback();
148         } catch (Exception ex) {
149             throw new ApplicationException("Exception : Delete rollback exception " + ex.getMessage());
150         }
151         throw new ApplicationException("Exception : Exception in delete User");
152     } finally {
153         JDBCDataSource.closeConnection(conn);
154     }
155 }

```

```

UserModel.java :: [File]
159     * Find User by Login
160     *
161     *
162     * @param login
163     *          : get parameter
164     * @return bean
165     * @throws DatabaseException
166     */
167
168    public UserBean findByLogin(String login) throws ApplicationException {
169        log.debug("Model findByLogin Started");
170        StringBuffer sql = new StringBuffer("SELECT * FROM USER WHERE LOGIN=?");
171        UserBean bean = null;
172        Connection conn = null;
173        try {
174            conn = JDBCDataSource.getConnection();
175            PreparedStatement pstmt = conn.prepareStatement(sql.toString());
176            pstmt.setString(1, login);
177            ResultSet rs = pstmt.executeQuery();
178            while (rs.next()) {
179                bean = new UserBean();
180                bean.setId(rs.getLong(1));
181                bean.setLogin(rs.getString(2));
182                bean.setPassword(rs.getString(3));
183                bean.setRoleId(rs.getLong(4));
184                bean.setCreatedBy(rs.getString(5));
185                bean.setModifiedBy(rs.getString(6));
186                bean.setCreatedDatetime(rs.getTimestamp(7));
187                bean.setModifiedDatetime(rs.getTimestamp(8));
188            }
189            rs.close();
190        } catch (Exception e) {
191            e.printStackTrace();
192            log.error("Database Exception.", e);
193            throw new ApplicationException("Exception : Exception in getting User by login");
194        } finally {
195            JDBCDataSource.closeConnection(conn);

```

Créer une servlet sous le controller package

Cette **classe RegistrationCtl** est une servlet qui recevra la demande d'effectuer l'opération et d'envoyer la réponse à la vue (registration.jsp).

RegistrationCtl.java

```

UserRegistrationCtl.java :: [File]
34 @WebServlet(name = "UserRegistrationCtl", urlPatterns = { "/UserRegistrationCtl" })
35 public class UserRegistrationCtl extends BaseCtl {
36     public static final String OP_SIGN_UP = "SignUp";
37
38     private static Logger log = Logger.getLogger(UserRegistrationCtl.class);
39
40     /**
41      * Valider les données d'entrée saisies par l'utilisateur
42      */
43     @Override
44     protected boolean validate(HttpServletRequest request) {
45         log.debug("UserRegistrationCtl Method validate Started");
46
47         boolean pass = true;
48
49         String login = request.getParameter("login");
50
51         if (DataValidator.isNull(login)) {
52             request.setAttribute("login", PropertyReader.getValue("error.require", "Login Id"));
53             pass = false;
54         }
55
56         if (DataValidator.isNull(request.getParameter("confirmPassword"))) {
57             request.setAttribute("confirmPassword", PropertyReader.getValue("error.require", "Confirm Password"));
58             pass = false;
59         }
60
61         if (DataValidator.isNull(request.getParameter("password"))) {
62             request.setAttribute("password", PropertyReader.getValue("error.require", "Password"));
63             pass = false;
64         }
65
66         if (!request.getParameter("password").equals(request.getParameter("confirmPassword"))
67             && !".".equals(request.getParameter("confirmPassword"))) {
68             ...

```

```

UserRegistrationCtl.java :: [File]
70     * ServletUtility.setErrorMessage("Confirm Password did not match", request);
71     */
72     request.setAttribute("confirmPassword",
73         PropertyReader.getValue("error.confirmPassword", "Confirm Password"));
74     pass = false;
75 }
76
77
78 Log.debug("UserRegistrationCtl Method validate Ended");
79 return pass;
80 }
81
82 /**
83 *
84 * *Remplit l'objet bean a partir des parametres de la requete
85 */
86 @Override
87 protected BaseBean populateBean(HttpServletRequest request) {
88     Log.debug("UserRegistrationCtl Method populatebean Started");
89
90     UserBean bean = new UserBean();
91
92     bean.setId(DataUtility.getLong(request.getParameter("id")));
93
94     bean.setRoleId(2L);
95
96     bean.setLogin(DataUtility.getString(request.getParameter("login")));
97
98     bean.setPassword(DataUtility.getString(request.getParameter("password")));
99
100    bean.setConfirmPassword(DataUtility.getString(request.getParameter("confirmPassword")));
101
102    populateDTO(bean, request);
103
104
105    Log.debug("UserRegistrationCtl Method populatebean Ended");
106
107

```

Comment créer une connexion à l'aide de JSP, Servlet et MYSQL?

Connexion avec JSP, Servlet et MYSQL

- Formulaire de connexion dans JSP.
- Créez une méthode de connexion dans UserModel.
- Gérez la demande de connexion dans LoginController.

Formulaire de connexion Design dans JSP

LoginView.jsp

```
1 <!-- DESING LOGIN FORM -->
2 <%@page import="com.octest.banque.util.ServletUtility"%>
3 <%@page import="com.octest.banque.util.DataUtility"%>
4 <%@ page language="java" contentType="text/html; charset=ISO-8859-1"
5    pageEncoding="ISO-8859-1"%>
6 <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
7<%@html%>
8<%@head%>
9 <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
10 <title>Login</title>
11 </head>
12<%@body%>
13 <!-- Include of the header JSP -->
14
15 <%@ include file="Header.jsp" %>
16 <br>
17<%@nav%>
18     aria-label="breadcrumb" role="navigation">
19<%@col class="breadcrumb">
20
21     <li class="breadcrumb-item active" aria-current="page">Login</li>
22 </ol>
23 </nav>
24
25
26 <div col-md-5 img="thumbnail">
27     <div id="feedback">
28         <div class="container">
29             <div class="col-md-5">
30                 <div class="form-area">
31                     <form role="form" action="<%$BSView.LOGIN_CTL%>" method="post" >
32                         <br style="clear: both">
33
34                         <jsp:useBean id="bean" class="com.octest.banque.bean.UserBean"
35                         scope="request"></jsp:useBean>
36                         <% String uri=(String)request.getAttribute("uri");%>
37
38                         <input type="hidden" name="uri" value="<%$uri%>">
39

```

```

LoginView.jsp
45
46          <h3 style="margin-bottom: 15px; text-align: left; ;">Login</h3>
47
48@         <b><font color="red"> <%=ServletUtility.getErrorMessage(request)%>
49
50         </font></b>
51@       <b><font color="Green"> <%=ServletUtility.getSuccessMessage(request)%>
52         </font></b>
53
54
55@       <div class="form-group">
56         <input type="text" class="form-control" name="login"
57             placeholder="Login Id" value="<%>DataUtility.getStringData(bean.getLogin())%>" >
58             <font color="red"><%=ServletUtility.getErrorMessage("login", request)%></font>
59
60@       <div class="form-group">
61         <input type="password" class="form-control"
62             name="password" placeholder="Password" value="<%>DataUtility.getStringData(bean.getPassword()) %>" >
63             <font color="red"><%=ServletUtility.getErrorMessage("password", request)%></font>
64
65         <input type="submit" name="operation"
66             class="btn btn-primary pull-right" value="<%>LoginCtl.OP_SIGN_IN %>" >
67             <input type="submit" name="operation"
68                 class="btn btn-primary pull-right" value="<%>LoginCtl.OP_SIGN_UP %>" >
69             <%-- <a href="<%>BSView.FORGET_PASSWORD_CTL%>"><b>Forget my password?</b></a> --%
70
71         </form>
72     </div>
73   </div>
74   <!--feedback-->
75   <br>
76   <div style="margin-top: 205px">
77   <!-- Include of the Footer JSP -->
78   <%@ include file="Footer.jsp"%>
79   </div>
80
81 </body>
82 </html>

```

la méthode Add login dans UserModel.java

Cet **UserLogin (connexion string, mot de passe string)** aidera à effectuer l'activité de connexion. Veuillez vérifier l'inscription de l'utilisateur auprès de JSP, Servlet et MYSQL pour obtenir le code complet de UserModel.java

```

    /**
     * Add login method
     *
     */
    @param login
    * : get parameter
    @param password
    * : get parameter
    @return bean
    @throws DatabaseException
    */

    public UserBean authenticate(String login, String password) throws ApplicationException {
        log.debug("Model authenticate Started");
        StringBuffer sql = new StringBuffer("SELECT * FROM USER WHERE LOGIN = ? AND PASSWORD = ?");
        System.out.println("very string"+sql);
        UserBean bean = null;
        Connection conn = null;

        try {
            conn = JDBCDataSource.getConnection();
            PreparedStatement pstmt = conn.prepareStatement(sql.toString());
            pstmt.setString(1, login);
            pstmt.setString(2, password);
            System.out.println("query string"+pstmt.toString());
            ResultSet rs = pstmt.executeQuery();
            ResultSet rs = pstmt.executeQuery();
            while (rs.next()) {
                bean = new UserBean();
                bean.setId(rs.getLong(1));
                bean.setLogin(rs.getString(2));
                bean.setPassword(rs.getString(3));
                bean.setRoleId(rs.getLong(4));
                bean.setCreatedBy(rs.getString(5));
                bean.setModifiedBy(rs.getString(6));
                bean.setCreatedDatetime(rs.getTimestamp(7));
                bean.setModifiedDatetime(rs.getTimestamp(8));
            }
        } catch (Exception e) {
            log.error("Database Exception..", e);
            throw new ApplicationException("Exception : Exception in get roles");
        } finally {
            JDBCDataSource.closeConnection(conn);
        }

        log.debug("Model authenticate End");
        return bean;
    }

```

Créer une servlet (Login Controller)

LoginCTL.java

```

1 *JDBCDataSource.java  2 LoginCtl.java  3 LoginView.jsp  4 UserModel.java
1 package com.octest.banque.controller;
2
3 import java.io.IOException;
4
5 /**
6  * Servlet implementation class LoginCtl
7 */
8
9  * Login functionality Controller. Performs operation for Authentication, and
10 * Session Creation and Give permission to access all Functionality regarding
11 * Role
12 *
13 */
14 @WebServlet(name = "LoginCtl", urlPatterns = { "/LoginCtl" })
15 public class LoginCtl extends BaseCtl {
16
17     private static final long serialVersionUID = 1L;
18
19     public static final String OP_REGISTER = "Register";
20     public static final String OP_SIGN_IN = "SignIn";
21     public static final String OP_SIGN_UP = "SignUp";
22     public static final String OP_LOG_OUT = "logout";
23     public static String HLT_URI = null;
24
25     private static Logger log = Logger.getLogger(LoginCtl.class);
26
27     /**
28      * @see HttpServlet#HttpServlet()
29      */
30     public LoginCtl() {
31         super();
32         // TODO Auto-generated constructor stub
33     }
34
35     /**
36      * Validate input Data Entered By User
37      *
38      * @param request
39      * @return
40      */
41     @Override
42     protected boolean validate(HttpServletRequest request) {
43
44         log.debug("LoginCtl Method validate Started");
45
46         boolean pass = true;
47
48         String op = request.getParameter("operation");
49
50         if (OP_SIGN_UP.equals(op) || OP_LOG_OUT.equals(op)) {
51             return pass;
52         }
53
54         String login = request.getParameter("login");
55
56         if (DataValidator.isNull(login)) {
57             request.setAttribute("login", PropertyReader.getValue("error.require", "Login Id"));
58             pass = false;
59         }
60
61         if (DataValidator.isNull(request.getParameter("password"))) {
62             request.setAttribute("password", PropertyReader.getValue("error.require", "Password"));
63             pass = false;
64         }
65     }
66
67 }

```

Welcome.jsp

Créez une page JSP bienvenue pour rediriger l'utilisateur si la connexion est réussie.

```
1 *JDBCDataSource.java      LoginCtl.java    LoginView.jsp     UserModel.java    Welcome.jsp
2
3 <!-- page d'accueil -->
4 <%@ page language="java" contentType="text/html; charset=ISO-8859-1"%>
5 <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
6 <html>
7 <head>
8 <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
9 <title>système de gestion de compte bancaire</title>
10 </head>
11 <body>
12 <!-- inclure la page JSP du header -->
13 <%@ include file="Header.jsp" %>
14 <br>
15 <nav aria-label="breadcrumb" role="navigation">
16   <ol class="breadcrumb">
17     <li class="breadcrumb-item active" aria-current="page">HelloBank</li>
18   </ol>
19 </nav>
20
21
22 <div class="container">
23   <div class="row">
24     <div class="col-lg-12">
25       <div class="jumbotron">
26         <h1 class="display-3">Choose us...Trust us!</h1>
27
28       </div>
29     </div>
30   </div>
31 </div>
32 <div style="margin-top: 289px">
33 <!-- inclure la page JSP du "Footer" -->
34 <%@ include file="Footer.jsp" %>
35 </div>
36 </div>
37 </body>
38 </html>
```

Session dans Java :

Comme son nom l'indique, la session est un intervalle de temps. Mais comment la session gère l'activité de connexion et de logout, comprenons ceci par un exemple.

Lorsqu'un utilisateur se connecte à l'application, il crée un id de session unique et lorsque le logout utilisateur de l'application de cette session est détruit ou invalidé. La session unique nous aidera à mettre les conditions selon le type d'utilisateur.

Comment définir la session en servlet

```
119     log.debug("CustomerRegistration method does started ");
120
121     HttpSession session=request.getSession(true);
122     UserBean userBean=(UserBean) session.getAttribute("user");
123     long id=userBean.getId();
124     String op=DataUtility.getString(request.getParameter("operation"));
125     CustomerModel model=new CustomerModel();
```

Récupérer des données en format table en Jsp

- Concevez une page .jsp utilisateur dans JSP.
- Créer une méthode dans UserModel pour obtenir la liste des utilisateurs de la base de données.
- Implémentez un contrôleur UserList sous le paquet contrôleur.
- Afficher la liste des utilisateurs dans userview.jsp

Concevoir une page de vue utilisateur dans JSP.

userList.jsp

```
<nav aria-label="breadcrumb" role="navigation">
<ol class="breadcrumb">
<li class="breadcrumb-item active" aria-current="page">Customer List</li>
</ol>
</nav>
<form action="<%>BSView.CUSTOMER_LIST_CTL%>" method="post">
    <div id="feedback">
        <div class="container">
            <div class="col-md-9">
                <div class="form-area">
                    <h3 style="margin-bottom: 15px; text-align: left;">Customer List</h3>
                    <div class="form-row">
                        <div class="form-group col-lg-4">
                            <input type="text" class="form-control" name="name"
                                placeholder="Name" value="<%>ServletUtility.getParameter("Name", request)%>">
                        </div>
                        <div class="form-group col-lg-4">
                            <input type="text" class="form-control" name="accNo"
                                placeholder="Account No" value="<%>ServletUtility.getParameter("accNo", request)%>">
                        </div>
                        <div class="form-group col-lg-4">
                            <input type="submit" name="operation"
                                class="btn btn-primary pull-right" value="<%>CustomerListCtl.OP_SEARCH%>&ampnbspor&ampnbsp
                                <input type="submit" name="operation"
                                class="btn btn-primary pull-right" value="<%>CustomerListCtl.OP_RESET%>">
                        </div>
                    </div>
                </div>
            </div>
        </div>
    <center>
        <b><font color="red"><%>ServletUtility.getErrorMessage(request)%></font></b>
        <b><font color="green"><%>ServletUtility.getSuccessMessage(request)%></font></b>
    </center>
    <table class="table">
        <thead class="thead-dark">
            <tr>
                <th scope="col">S.No</th>
                <th scope="col">Account No</th>
                <th scope="col">Name</th>
                <th scope="col">Email Id</th>
                <th scope="col">Phone No</th>
                <th scope="col">Address</th>
                <th scope="col">Add to Account</th>
            </tr>
        </thead>
        <tbody> <%>
```

Ajouter une méthode dans UserModel

Cette méthode de liste () obtient les données de la base de données et retourne une liste de données utilisateur.

UserModel.java

```
        }
        ArrayList list = new ArrayList();
        Connection conn = null;
        try {
            conn = JDBCDataSource.getConnection();
            PreparedStatement pstmt = conn.prepareStatement(sql.toString());
            ResultSet rs = pstmt.executeQuery();
            while (rs.next()) {
                bean = new CustomerBean();
                bean.setId(rs.getLong(1));
                bean.setAcc_No(rs.getLong(2));
                bean.setName(rs.getString(3));
                bean.setEmailId(rs.getString(4));
                bean.setMobileNo(rs.getString(5));
                bean.setAddress(rs.getString(6));
                bean.setUserId(rs.getLong(7));
                bean.setCreatedBy(rs.getString(8));
                bean.setModifiedBy(rs.getString(9));
                bean.setCreatedDatetime(rs.getTimestamp(10));
                bean.setModifiedDatetime(rs.getTimestamp(11));
                list.add(bean);
            }
            rs.close();
        } catch (Exception e) {
            log.error("Database Exception..", e);
            throw new ApplicationException("Exception : Exception in search Customer");
        } finally {
            JDBCDataSource.closeConnection(conn);
        }
        log.debug("Model search End");
        return list;
    }
```

Créer une Servlet UserListCtl

Ce servlet permet d'obtenir les données du modèle et transmettre les données à View (userList.jsp).

UserListCtl.java

```
    /**
     * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse
     *      response)
     */
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        log.debug("CustomerListCtl doGet Start");
        List list = null;

        int pageNo = 1;

        int pageSize = DataUtility.getInt(PropertyReader
            .getValue("page.size"));

        CustomerBean bean = (CustomerBean) populateBean(request);

        String op = DataUtility.getString(request.getParameter("operation"));

        String[] ids = request.getParameterValues("ids");

        CustomerModel model = new CustomerModel();
        try {
            list = model.search(bean, pageNo, pageSize);
            if (list == null || list.size() == 0) {
                ServletUtility.setErrorMessage("No record found ", request);
            }
            ServletUtility.setList(list, request);
            ServletUtility.setPageNo(pageNo, request);
            ServletUtility.setPageSize(pageSize, request);
            ServletUtility.forward(getView(), request, response);
        } catch (ApplicationException e) {
            log.error(e);
            ServletUtility.handleException(e, request, response);
            return;
        }
        log.debug("CustomerListCtl doP0st End");
    }
```