J2EE-JSP-Servlet-JSF-JDBC

2015

Ce Tp a pour objectif de concevoir une application Web Dynamique reposant sur la technologie JSF. Elle permettra d'appréhender les concepts et servlet, jdbc, dao et jsp

Mise en Oeuvre D'une application J2E

Table des matières

1. Introduction

Ce Tp déroule en 4 étapes comme suit :

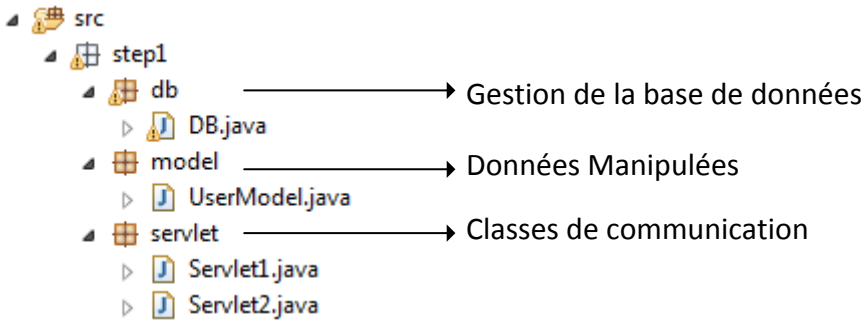
- **Step 1** : prise en main des servlet, envoie de données http POST et GET, connexion simple à une base de données
- **Step 2** : Prise en main des servlets, des JSP et des java bean réutilisation de la gestion de la base de données précédemment créer.
- **Step 3** : Réalisation d'un DAO Data Access Object permettant de gérer la persistance de plusieurs objets.
- **Step 4** : Prise main de JSF, des java Bean «Model », et java Bean « Controler », réutilisation du DAO.

2. STEP 1 : Servlet et JDBC

Objectif : Créer des objets Servlet et classe Java utilisant le JDBC permettant d'enregistrer des données sur une base de données

2.1. Création d'un projet Web Dynamic

2.1.1.Créer Un projet Web Dynamic possédant la structure suivante :



2.1.2.A quoi sert l'organisation en Paquage d'un projet ?

2.2. Créer la classe DB.java comme ci-dessous

```
public class DB {
	private static final String DB_HOST="db-tp.cpe.fr";
	private static final String DB_PORT="3306";
	private static final String DB_NAME="binome32";
	private static final String DB_USER="binome32";
	private static final String DB_PWD="binome32";
	private Connection connection;

	public DB() {
		try {
			// Chargement du Driver, puis établissement de la connexion
			Class.forName("com.mysql.jdbc.Driver");

			//create connection
			connection =
	java.sql.DriverManager.getConnection("jdbc:mysql://"+DB_HOST+":"+DB_PORT+"/"+DB_NAME
, DB_USER, DB_PWD);
		} catch (SQLException e) {
			e.printStackTrace();
		} catch (ClassNotFoundException e) {
			e.printStackTrace();
		}
	}

	public ArrayList<UserModel> getData(){

		//return value
		ArrayList<UserModel> userList=new ArrayList<UserModel>();

		// Création de la requête
		java.sql.Statement query;
		try {

		//TODO récupérez l'ensemble des paramètres de tous les utilisateurs de la
		 table ((`surname`, `lastname`, `age`, `login`, `pwd`)

			connection.close();

		} catch (SQLException e) {
			e.printStackTrace();
		}
```

```
		return userList;
	}

	public void addUser(UserModel user) {

		// Création de la requête
		java.sql.Statement query;
		try {
			//Creation de l'élément de requète
			query = connection.createStatement();

	//TODO créez la requête permettant d'ajout un utilisateur avec ts ces paramètres
	//((`surname`, `lastname`, `age`, `login`, `pwd`)
			connection.close();
		} catch (SQLException e) {
			e.printStackTrace();
		}
	}
}
```

2.2.1. A quoi sert les propriétés des attributs suivant :

- Private ?
- Static ?
- Final ?

2.2.2.Pourquoi est-il préférable de déclarer les constantes de chaines de caractères comme cela ?

2.3.Créer la classe UserModel.java comme suit :

```
public class UserModel {
	private String lastname;
	private String surname;
	private int age;
	private String login;
	private String pwd;

	public UserModel(String lastname,String surname,int age,String login,String pwd) {
		this.lastname = lastname;
		this.surname = surname;
		this.age = age;
		this.login = login;
		this.pwd = pwd;
	}
	public String getLastname() { return lastname; }
	public void setLastname(String lastname) { this.lastname = lastname; }
	public String getSurname() { return surname; }
	public void setSurname(String surname) { this.surname = surname; }
	public int getAge() { return age;}
	public void setAge(int age) {this.age = age;}
	public String getLogin() {   return login;  }
	public void setLogin(String login) {this.login = login;   }
	public String getPwd() {return pwd; }
	public void setPwd(String pwd) {this.pwd = pwd;}

	@Override
	public String toString() {
	  return
"[SURNAME]:"+this.getSurname()+",[LASTNAME]:"+this.getLastname()+",[AGE]:"+this.getAge()+",
[LOGIN]:"+this.getLogin()+",[PWD]:"+this.getPwd();
	}
}
```

2.3.1.Pourquoi est-il utile de surcharger la méthode toString() ?

2.4.Créer la classe Servlet1.java comme suit :

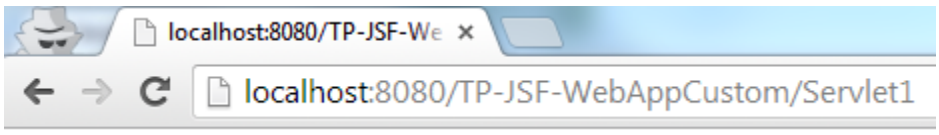
```
/**
 * Servlet implementation class Servlet1
 */
@WebServlet("/Servlet1")
public class Servlet1 extends HttpServlet {
	private static final long serialVersionUID = 1L;
	private DB db;

    /**
     * @see HttpServlet#HttpServlet()
     */
    public Servlet1() {
        super();
        db=new DB();
    }

	/**
	 * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)
	 */
	protected void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {

	//TODO Lors de l'appel de http get utiliser les classes précédement créer pour
		récupérer la liste des utilisateurs
	//TODO écrire la liste des utilisateurs dans le flux de réponse HttpServletResponse

	}

	/**
	 * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)
	 */
	protected void doPost(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
	}

}
```

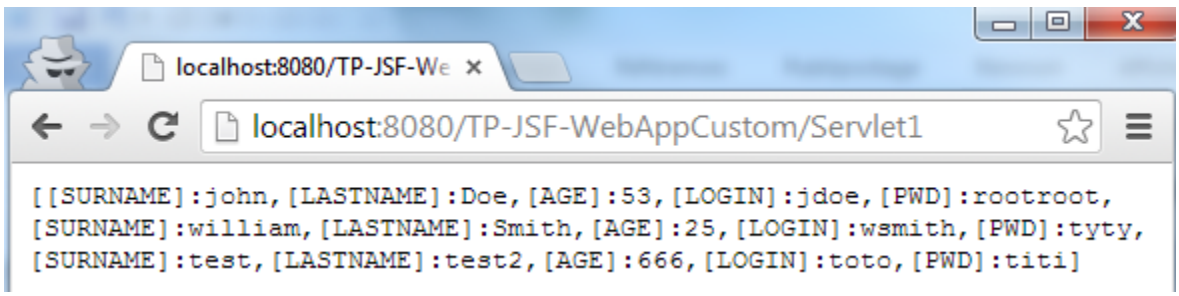
2.4.1. Pourquoi l'objet DB.java est-il créer dans le constructeur de la servlet ?

2.4.2.A quoi sert la ligne de code suivante : `@WebServlet("/Servlet1")` existe –il une description alternative ?

2.5.Démarrer votre projet Web Dynamique et test l'appel d'un http Get sur l'URL /Servlet1

Résultat attendu :





2.5.1.Que se passe-t-il si vous effectuer plusieurs appel ?

2.5.2.Proposer une amélioration

2.6.Créer la classe Servlet2.java comme suit :

```
/**
 * Servlet implementation class Servlet2
 */
@WebServlet("/Servlet2")
public class Servlet2 extends HttpServlet {
	private static final long serialVersionUID = 1L;
	private DB db;

    /**
     * @see HttpServlet#HttpServlet()
     */
    public Servlet2() {
        super();
        db=new DB();
    }

	/**
	 * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)
	 */
	protected void doGet(HttpServletRequest request, HttpServletResponse response)
			throws ServletException, IOException {
		//Crée un utilisateur depuis les informations transmises
		//ATTENTION ERREUR SI LES INFOS TRANSMISES SONT INEXACTE
		UserModel user=new UserModel(request.getParameter("lastname"),
		request.getParameter("surname"),Integer.valueOf(request.getParameter("age")),
		request.getParameter("login"), request.getParameter("pwd"));

		//Demande à DB d'ajouter l'utlisateur
		db.addUser(user);

	}

	/**
	 * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)
	 */
	protected void doPost(HttpServletRequest request, HttpServletResponse response)
		throws ServletException, IOException {
		//redirection sur le doGet car même action
		doGet(request, response);
	}

}
```

2.6.1.Tester l'ajout d'un utilisateur en http GET

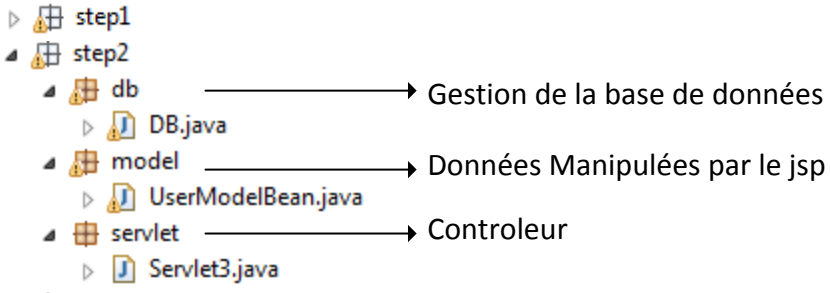
2.6.2. Comment feriez-vous un test en http POST ?

3. STEP 2 : JDBC, JavaBean et JSP

Objectif : Créer des objets JSP utilisant des JavaBean permettant d'enregistrer un utilisateur dans la base de données.

3.1. Mise à jour du projet Web Dynamic précédemment créé.

3.1.1.Modifier le projet Web Dynamic afin qu'il possède la structure suivante :



3.2. Réutiliser la classe DB.java précédemment créée

3.3. Créer la classe UserModelBean.java comme suit :

```
//contrainte BEAN implements Serializable
public class UserModelBean implements Serializable{
	private String lastname;
	private String surname;
	private int age;
	private String login;
	private String pwd;

	//Contrainte BEAN constructeur sans paramètre
	public UserModelBean() {
	}

	public String getLastname() {return lastname;}
	public void setLastname(String lastname) {this.lastname = lastname;}
	public String getSurname() { return surname;}
	public void setSurname(String surname) {this.surname = surname;}
	public int getAge() { return age;}
	public void setAge(int age) {this.age = age;}
	public String getLogin() {return login;}
	public void setLogin(String login) {this.login = login;}
	public String getPwd() {return pwd;}
	public void setPwd(String pwd) {this.pwd = pwd;}

	@Override
	public String toString() {
		return
	"[SURNAME]:"+this.getSurname()+",[LASTNAME]:"+this.getLastname()+",[AGE]:"+this.getA
	ge()+",[LOGIN]:"+this.getLogin()+",[PWD]:"+this.getPwd();
	}
}
```

3.3.1.Quelles sont les contraintes liées aux JavaBean ? A quoi servent-elles ?

3.4.Créer la classe Servlet3.java comme suit :

```
/**
 * Servlet implementation class Servlet3
 */
@WebServlet("/Servlet3")
public class Servlet3 extends HttpServlet {
	private static final long serialVersionUID = 1L;
	private DB db;

    /**
     * @see HttpServlet#HttpServlet()
     */
    public Servlet3() {
        super();
    }

    @Override
    public void init() throws ServletException {
       super.init();

       //Vérifie si DB existe dans l'espace de mémoire partagé entre les servlet
       // si oui on les récupére, si non on le crée et on l'ajoute dans l'espace de mémoire
       //partagé entre les servlet

       if(getServletContext().getAttribute("BD")!=null){
              db=(DB)getServletContext().getAttribute("BD");
       }else {
              db=new DB();
              getServletContext().setAttribute("BD",db);
       }
    }

       /**
        * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)
        */
       protected void doGet(HttpServletRequest request, HttpServletResponse response)
                     throws ServletException, IOException {
              //Nothing to do
       }

       /**
        * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)
        */
       protected void doPost(HttpServletRequest request, HttpServletResponse response)
                     throws ServletException, IOException {
           UserModelBean user=(UserModelBean)request.getSession().getAttribute("myUser");

       //TODO Sauvegarder l'utilisateur user (créer dans la page jsp et stocké dans la
       mémoire session) dans la base de données

       //TODO Rediriger la page courante vers la page /step2/display.jsp


       }

}
```

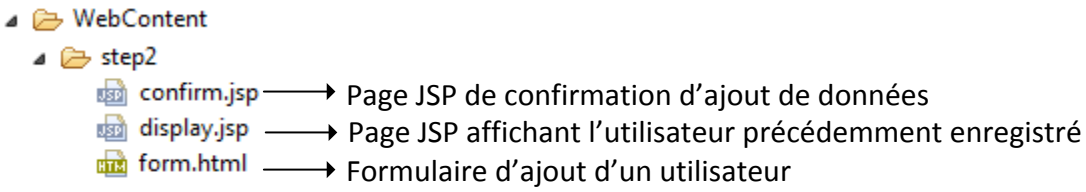
3.4.1.Quelle est l'utilité de la méthode init() et la classe Servlet3 ?

3.4.2.Que permet d'effectuer la ligne suivante ?

```
UserModelBean user=(UserModelBean)request.getSession().getAttribute("myUser");
```

3.5.Créer les éléments de la vue

3.5.1.Modifier le projet Web Dynamic afin d'avoir une arborescence comme suit :



3.6.Créer la classe form.html comme suit :

```
<!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>Step2: Form</title>
</head>
<body>
         <form action="confirm.jsp" method="post">
       <h5>    Put your lastname <input type="text" name="lastname" /> </h5>
       <h5>    Put your surname <input type="text" name="surname" /> </h5>
               <h5>Put your age <input type="text" name="age" /> </h5>
               <h5>Put your login <input type="text" name="login" /> </h5>
               <h5>Put your pwd <input type="text" name="pwd" /> </h5>
               <h5><input type="submit" value="GO!"> </h5>
       </form>
</body>
</html>
```

3.6.1. A quoi sert la ligne de code suivante ?

```
<form action="confirm.jsp" method="post">
```

3.7.Créer la classe confirmation.jsp comme suit :

```
<%@ 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>Step2: Confirmation</title>
       <jsp:useBean id="myUser" scope="session" class="step2.model.UserModelBean" />
          <jsp:setProperty name="myUser" property="surname" />
          <jsp:setProperty name="myUser" property="lastname"/>
          <jsp:setProperty name="myUser" property="age"/>
          <jsp:setProperty name="myUser" property="login"/>
          <jsp:setProperty name="myUser" property="pwd"/>

</head>
<body>
         <form action="../Servlet3" method="post">
              Save DATA ?
              <input type="submit" value="GO!">
       </form>
</body>
</html>
```

3.7.1.Que fait la ligne de code suivant :

```
<jsp:useBean id="myUser" scope="session" class="step2.model.UserModelBean" />
```

3.7.2.Comment le fichier .jsp permet-il de récupérer les propriétés entrées dans form.html précédemment ?

3.7.3.Que se passera – t - il lors d'un clic sur le bouton GO ?

3.8.Créer la classe display.jsp comme suit :

```
<%@ 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>Step2: Display</title>
        <jsp:useBean id="myUser" scope="session" class="step2.model.UserModelBean" />
           <jsp:setProperty name="myUser" property="surname" />
           <jsp:setProperty name="myUser" property="lastname"/>
           <jsp:setProperty name="myUser" property="age"/>
           <jsp:setProperty name="myUser" property="login"/>
           <jsp:setProperty name="myUser" property="pwd"/>

</head>
<body>
        <h1> <jsp:getProperty name="myUser" property="surname" />  <jsp:getProperty
name="myUser" property="lastname" />  </h1>
        <h2> AGE:<jsp:getProperty name="myUser" property="age" /></h2>
        <h2> Login:<jsp:getProperty name="myUser" property="login" /></h2>
        <h2> pwd:********</h2>
</body>
</html>
```

3.8.1.Comment la page JSP fait-elle pour afficher les propriétés de l'utilisateur ?

3.8.2.Que se serait-il passé si le scope du bean «myUser » avait été *request* dans confirmation.jsp ?

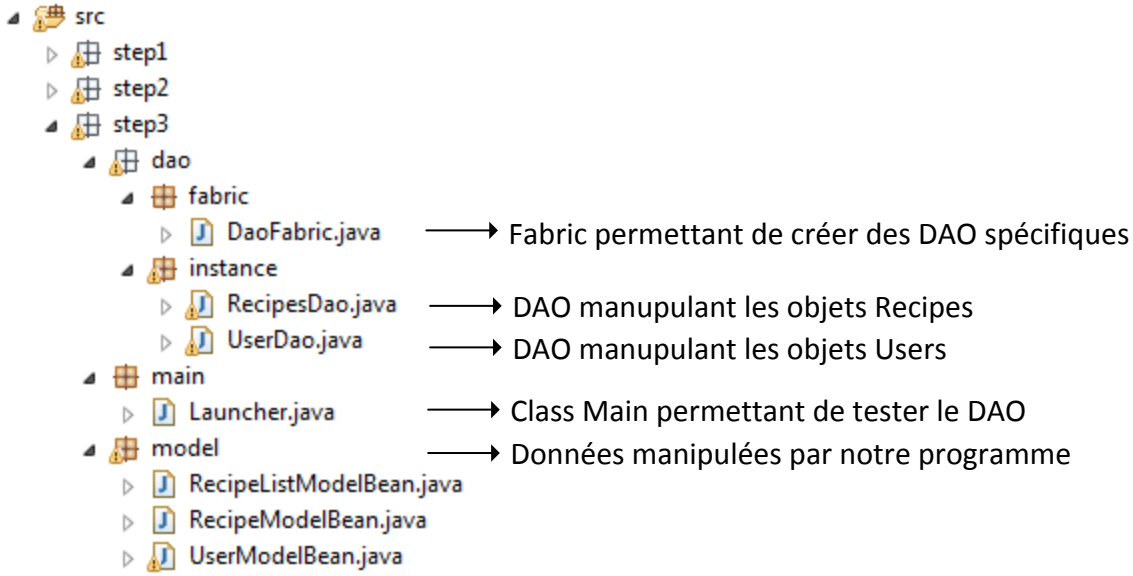
4. STEP 3 : Data Access Object

Objectif : Créer un DAO permettant d'accéder à la base de données. La construction et l'organisation de cette objet permet facilement de manipuler différent type de donner dans l'ensemble du programme

Référence : http://www.oracle.com/technetwork/java/dataaccessobject-138824.html

4.1.Mise à jour du projet Web Dynamic précédemment créé.

4.1.1.Modifier le projet Web Dynamic afin qu'il possède la structure suivante :



4.2.Créer la classe DaoFabric.java comme suit :

```
public final class DaoFabric {

	// L'utilisation du mot clé volatile permet, en Java version 5 et supérieur,
	// permet d'éviter le cas où "Singleton.instance" est non-nul,
	// mais pas encore "réellement" instancié.
	// De Java version 1.2 à 1.4, il est possible d'utiliser la classe
	// ThreadLocal.
	private static volatile DaoFabric instance = null;

	private static final String DB_HOST = " db-tp.cpe.fr ";
	private static final String DB_PORT = "3306";
	private static final String DB_NAME = "binome32";
	private static final String DB_USER = " binome32";
	private static final String DB_PWD = " binome32";

	private DaoFabric() {
		super();
		try {
			// Chargement du Driver, puis établissement de la connexion
			Class.forName("com.mysql.jdbc.Driver");
		} catch (ClassNotFoundException e) {
			e.printStackTrace();
		}
	}

	/**
	 * Méthode permettant de renvoyer une instance de la classe Singleton
	 *
	 * @return Retourne l'instance du singleton.
```

```
	 */
	public final static DaoFabric getInstance() {
		// Le "Double-Checked Singleton"/"Singleton doublement vérifié" permet
		// d'éviter un appel coûteux à synchronized,
		// une fois que l'instanciation est faite.
		if (DaoFabric.instance == null) {
			// Le mot-clé synchronized sur ce bloc empêche toute instanciation
			// multiple même par différents "threads".
			synchronized (DaoFabric.class) {
				if (DaoFabric.instance == null) {
					DaoFabric.instance = new DaoFabric();
				}
			}
		}
		return DaoFabric.instance;
	}

	public UserDao createUserDao() {
		UserDao userDao = new
		UserDao(this.DB_HOST,this.DB_PORT,this.DB_NAME,this.DB_USER,this.DB_PWD);
		return userDao;
	}

	public RecipesDao createRecipesDao(){
		RecipesDao receipesDao = new
		RecipesDao(this.DB_HOST,this.DB_PORT,this.DB_NAME,this.DB_USER,this.DB_PWD);
		return receipesDao;
	}

}
```

4.2.1. Pourquoi la méthode getInstance() est t il en static ? Quel intérêt présente cette propriété ?

4.2.2. Quelle est l'utilité de la propriété static de l'attribut instance ?

4.3. Créer la classe UserDao.java comme suit :

```
public class UserDao {

	private Connection connection;
	private String dB_HOST;
	private String dB_PORT;
	private String dB_NAME;
	private String dB_USER;
	private String dB_PWD;
	public UserDao(String DB_HOST,String DB_PORT, String DB_NAME,String DB_USER,String
			DB_PWD) {
		dB_HOST = DB_HOST;
		dB_PORT = DB_PORT;
		dB_NAME = DB_NAME;
		dB_USER = DB_USER;
		dB_PWD = DB_PWD;
	}
	public void addUser(UserModelBean user) {
		// Création de la requête
		java.sql.Statement query;
		try {
		// create connection
			connection = java.sql.DriverManager.getConnection("jdbc:mysql://"
				+ dB_HOST + ":" + dB_PORT + "/" + dB_NAME, dB_USER, dB_PWD);

			//TODO A l'image de DB.java créer une réquète permettant d'ajout
			 l'utilisateur à la base de données

			connection.close();
```

```
		} catch (SQLException e) {
			e.printStackTrace();
		}
	}
public ArrayList<UserModelBean> getAllUser(){
		//return value
		ArrayList<UserModelBean> userList=new ArrayList<UserModelBean>();
		try {
		// create connection
		connection = java.sql.DriverManager.getConnection("jdbc:mysql://"
				+ dB_HOST + ":" + dB_PORT + "/" + dB_NAME, dB_USER, dB_PWD);

		//TODO A l'image de DB.java créer une réquète permettant de récupérer
		l'ensemble des utilisateurs contenu dans la base et de les placer dans une
		liste

			connection.close();

		} catch (SQLException e) {
			e.printStackTrace();
		}
		return userList;
	}
}
```

4.4.Créer la classe RecipeDao.java

4.4.1. En vous inspirant de la classe UserDao.java créer la RecipeDao.java permettant d'ajouter une recette (`title`, `description`, `expertise`, `duration`, `nbpeople`,`type`) et de récupérer l'ensemble des recettes disponibles dans la base de données.
Cette classe devra respecter la structure suivante :

```
public class RecipesDao {

	//TODO

	public RecipesDao(String DB_HOST,String DB_PORT, String DB_NAME,String
DB_USER,String DB_PWD) {

		//TODO

	}

	public void addRecipe(RecipeModelBean recipe) {

		//TODO

	}

	public ArrayList<RecipeModelBean> getAllRecipes() {

		//TODO

	}

}
```

4.5.Créer les modèles de données RecipeModelBean.java, UserModelBean.java.

```
public class UserModelBean implements Serializable{
	private String lastname;
	private String surname;
	private int age;
	private String login;
	private String pwd;

	public UserModelBean() {
	}

	public UserModelBean(String lastname,String surname,int age,String login,String pwd)
	{
		this.lastname = lastname;
		this.surname = surname;
		this.age = age;
		this.login = login;
		this.pwd = pwd;
	}

	public String getLastname() {return lastname;}
	public void setLastname(String lastname) { this.lastname = lastname;}
	public String getSurname() { return surname;}
	public void setSurname(String surname) {this.surname = surname;}
	public int getAge() { return age;}
	public void setAge(int age) {this.age = age;}
	public String getLogin() {return login;}
	public void setLogin(String login) {this.login = login;}
	public String getPwd() {return pwd;}
	public void setPwd(String pwd) {this.pwd = pwd;}
		@Override
	public String toString() {
	return
	"[SURNAME]:"+this.getSurname()+",[LASTNAME]:"+this.getLastname()+",[AGE]:"+this.getA
ge()+",[LOGIN]:"+this.getLogin()+",[PWD]:"+this.getPwd();
	}
}
```

```
package step3.model;

public class RecipeModelBean {
	private String title;
	private String description;
	private int expertise;
	private int nbpeople;
	private int duration;
	private String type;

	public RecipeModelBean() {}

	public RecipeModelBean(String title,String description,int expertise,int
duration,int nbpeople,String type) {
		this.title = title;
		this.description = description;
		this.expertise = expertise;
		this.duration = duration;
		this.nbpeople = nbpeople;
		this.type = type;
	}
	public String getTitle() {return title;}
	public void setTitle(String title) {this.title = title;}
	public String getDescription() {return description;}
	public void setDescription(String description) {this.description = description;}
```

```
	public int getExpertise() {return expertise;}
	public void setExpertise(int expertise) {this.expertise = expertise;}
	public int getNbpeople() {return nbpeople;}
	public void setNbpeople(int nbpeople) {this.nbpeople = nbpeople;}
	public String getType() {return type;}
	public void setType(String type) {this.type = type;}
	public int getDuration() {return duration;}
	public void setDuration(int duration) {this.duration = duration;}

	public String toString() {
		//TODO
	}

}
```

4.5.1.A quoi servent les doubles constructeurs de chaque classe ?

4.6.Tester votre DAO.

4.6.1.Créer la classe Launcher.java

```
public class Launcher {

	public static void main(String[] args) {
		UserDao userDao=DaoFabric.getInstance().createUserDao();
		RecipesDao recipesDao=DaoFabric.getInstance().createRecipesDao();

		UserModelBean user1=new UserModelBean("Doe", "John",55, "jdoe", "pwd");
		RecipeModelBean recipe1=new RecipeModelBean("Fish Salad", "bla bla bal bla",
					5, 180, 10, "salad");
		RecipeModelBean recipe2=new RecipeModelBean("Fresh Meat", "bla bla bal bla",
					1, 20, 1, "meat");

		userDao.addUser(user1);
		recipesDao.addRecipe(recipe1);
		recipesDao.addRecipe(recipe2);

		List<UserModelBean> userList=userDao.getAllUser();
		List<RecipeModelBean> recipeList=recipesDao.getAllRecipes();

		for(UserModelBean userTmp:userList){
			System.out.println("User added:"+userTmp);
		}

		for(RecipeModelBean recipeTmp:recipeList){
			System.out.println("User added:"+recipeTmp);
		}
	}

}
```

4.6.2. Pourquoi est-il possible d'appeler la méthode getInstance() de la classe DaoFabric ?

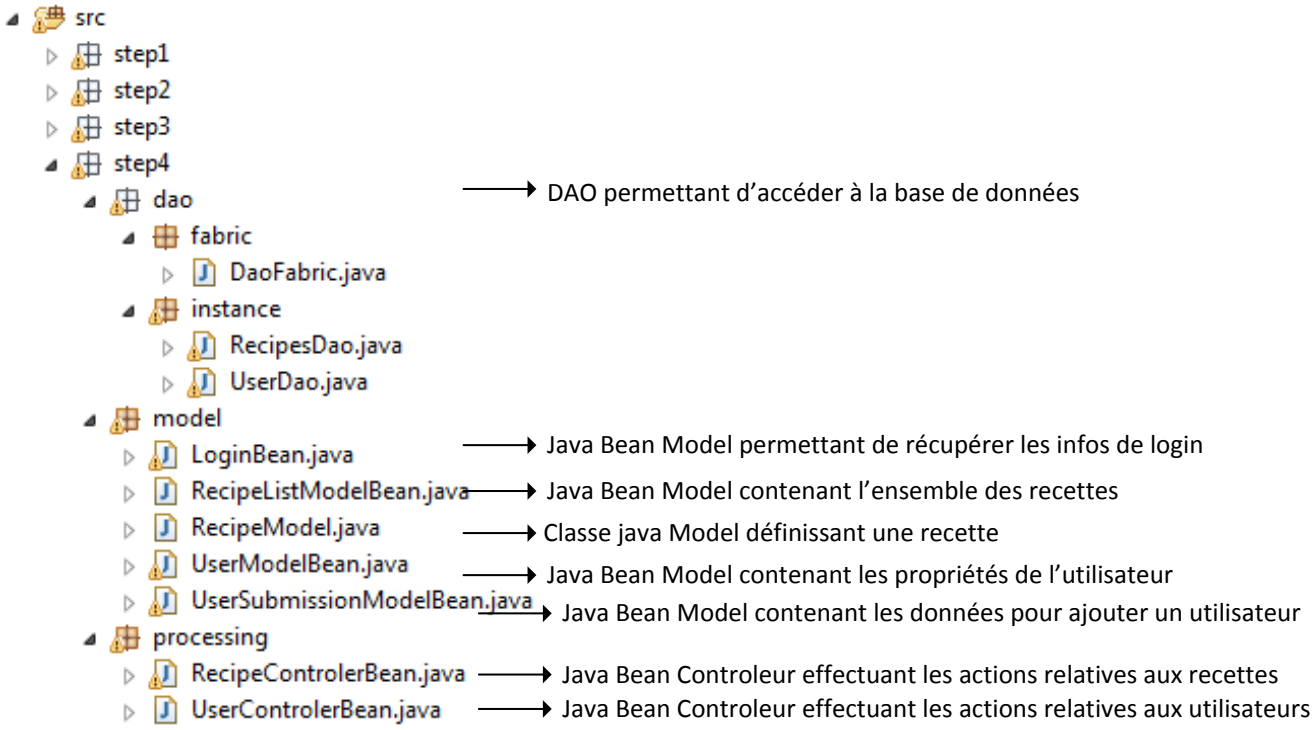
4.6.3. Que ce passe-t-til si j'exécute en parallèle 5 Launcher.java ? Combien d'objet DaoFrabic sera créé ? Combien de UserDao ?

5. STEP 3 : JSF, ManagedBean et DAO

Objectif : Créer l'ensemble des briques logicielles permettant de mettre en place un projet JSF complet comportant les modèles (JavaBean Model), vues (.xhtml,.html) et les contrôleurs (JavaBean contrôleurs, Dao)

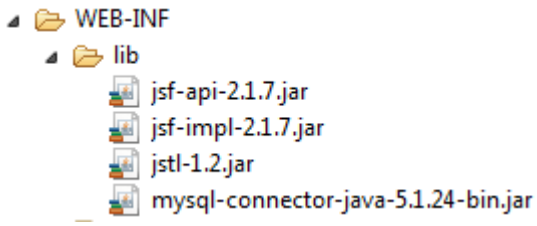
5.1.Mise à jour du projet Web Dynamic précédemment créé.

5.1.1.Modifier le projet Web Dynamic afin qu'il possède la structure suivante :



5.2.Configuration du projet pour JSF

5.2.1. Modifier votre projet en ajoutant les librairies suivantes



5.2.2.Associer ces librairies à votre projet web dynamic (Projet->propriétés-> Java Build PAth→onglet Libraries → bouton Add JARS…)

5.2.3.Modifier votre web.xml comme suit :

```
<servlet>
  <servlet-name>Faces Servlet</servlet-name>
  <servlet-class>javax.faces.webapp.FacesServlet</servlet-class>
</servlet>
<servlet-mapping>
  <servlet-name>Faces Servlet</servlet-name>
  <url-pattern>*.jsf</url-pattern>
</servlet-mapping>
```

```
  <context-param>
    <param-name>javax.faces.PROJECT_STAGE</param-name>
    <param-value>Development</param-value>
  </context-param>
```

5.3.DAO : reprendre le DAO précédemment créé

5.4. Création de loginBean.java

```
@ManagedBean
@RequestScoped
public class LoginBean implements Serializable{
	private String login;
	private String pwd;

	public LoginBean() { }

	public String getLogin() {return login;}

	public void setLogin(String login) {this.login = login;}

	public String getPwd() {return pwd;}

	public void setPwd(String pwd) {this.pwd = pwd;}

}
```

5.4.1.A quoi sert l'annotation @managedBean

5.4.2.A quoi sert l'annotation @RequestScoped ? Selon vous pourquoi une telle persistance ?

5.5. Création de UserModelBean.java

5.5.1.Modifier UserModelBean précédemment créé comme suit :

```
@ManagedBean
@SessionScoped
//contrainte BEAN implements Serializable
public class UserModelBean implements Serializable{
	private String lastname;
	private String surname;
	private int age;
	private String login;
	private String pwd;

	//Contrainte BEAN constructeur sans paramètre
	public UserModelBean() {
	}

	....IDEM....


}
```

5.5.2.A quoi sert l'annotation @SessionScoped ? Selon vous pourquoi une telle persistance ?

5.6. Création de UserSubmissionModelBean.java

Créer la classe comme suit :

```
@ManagedBean
@RequestScoped //Durée de vue uniquement lors d'une requète
//même propriétés que UserModelBean mais portée différente
public class UserSubmissionModelBean extends UserModelBean{
	public UserSubmissionModelBean() {
	}
	}
```

5.6.1.A quoi sert cette classe selon vous ?

5.7.Création de RecipeModel.java

Reprenez votre classe RecipeModelBean.java et renommez-la en RecipeModel

5.8.Création de RecipeListModelBean.java

Créer la classe comme suit :

```
@ManagedBean
@RequestScoped
public class RecipeListModelBean {
	private List<RecipeModel> recipeList;

	public RecipeListModelBean() {
		recipeList=new ArrayList<RecipeModel>();
	}

	public void addRecipeList(RecipeModel recipe){
		this.recipeList.add(recipe);
	}

	public List<RecipeModel> getRecipeList() {
		return recipeList;
	}

	}
```

5.8.1.A quoi va servir cette classe ?

5.8.2.Pourquoi la classe RecipeModel.java n'est pas un JavaBean ?

5.9.Création de RecipeControlerBean.java

Créer la classe comme suit :

```
@ManagedBean
@ApplicationScoped

public class RecipeControlerBean {
	private RecipesDao recipeDao;

	public RecipeControlerBean() {
		this.recipeDao=DaoFabric.getInstance().createRecipesDao();
	}
	public void loadAllRecipe(){
		ArrayList<RecipeModel> list = this.recipeDao.getAllRecipes();
		RecipeListModelBean recipeList=new RecipeListModelBean();
		for(RecipeModel recipe:list){
			recipeList.addRecipeList(recipe);
		}
		//récupère l'espace de mémoire de JSF
		ExternalContext externalContext =
				FacesContext.getCurrentInstance().getExternalContext();
		Map<String, Object> sessionMap = externalContext.getSessionMap();
		//place la liste de recette dans l'espace de mémoire de JSF
		sessionMap.put("recipeList", recipeList);
	}
}
```

5.9.1.A quoi va servir cette classe

5.9.2.A quoi sert l'annotation @ApplicationScoped ? Selon vous pourquoi une telle persistance ?

5.10. Création de UserControlerBean.java

Créer la classe comme suit :

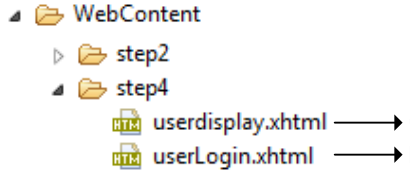
```
@ManagedBean
@ApplicationScoped // Utilisation de application scope afin d'offrir un point d'entrée
unique à l'ensemble des clients
public class UserControlerBean {
	private UserDao userDao;

	public UserControlerBean() {
		this.userDao=DaoFabric.getInstance().createUserDao();
	}

	public String checkUser(LoginBean loginBean){
		UserModelBean user = this.userDao.checkUser(loginBean.getLogin(),
loginBean.getPwd());
		if( user!=null){

			//récupère l'espace de mémoire de JSF
			ExternalContext externalContext =
FacesContext.getCurrentInstance().getExternalContext();
			Map<String, Object> sessionMap = externalContext.getSessionMap();

			//place l'utilisateur dans l'espace de mémoire de JSF
			sessionMap.put("loggedUser", user);

			//redirect the current page
			return "userdisplay.xhtml";
		}else{

			//redirect the current page
			return "userLogin.xhtml";
		}
	}

	public void checkAndAddUser(UserSubmissionModelBean userSubmitted){

		//Vérifier les propriétés de l'utilisateur
		//TODO

		//ajout de l'utilisateur à la base de données
		this.userDao.addUser(userSubmitted);
	}

}
```

5.10.1. A quoi va servir cette classe

5.10.2. A quoi sert l'annotation @ApplicationScoped ? Selon vous pourquoi une telle persistance ?

5.11. Créer les éléments de la vue

5.11.1. Modifier le projet Web Dynamic afin d'avoir une arborescence comme suit :



Page JSF affichant les données utilisateurs chargées depuis la DB

Page JSF permettant d'envoyer les infos de login

5.11.2. Créer la classe userLogin.xhtml

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:ui="http://java.sun.com/jsf/facelets"
      xmlns:h="http://java.sun.com/jsf/html"
      xmlns:f="http://java.sun.com/jsf/core">
<head>
<title>Step2: Form</title>
</head>
<body>
      <h1>Login Page</h1>
      <h:form>
            <table border="0">
                  <tr>
                        <td>Your Login</td>
                        <td><h:inputText id="userLogin" value="#{loginBean.login}"
                                    required="true">
                              </h:inputText></td>
                  </tr>
                  <tr>
                        <td>Your Password</td>
                        <td><h:inputSecret value="#{loginBean.pwd}" /></td>
                  </tr>
                  <tr>
                        <td></td>
                        <td>
                          <h:commandButton value="OK"
                              action="#{userControlerBean.checkUser(loginBean)}"/>
                        </td>
                  </tr>
            </table>
      </h:form>
</body>
</html>
```

5.11.3. Quelle est la portée de LoginBean ?

5.11.4. Que se passe-t-il lors de l'exécution du bouton OK ?

5.11.5. Que se passe-t-il si le couple login/pwd n'est pas bon ?

5.11.6. Créer la classe userdisplay.xhtml

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:ui="http://java.sun.com/jsf/facelets"
      xmlns:h="http://java.sun.com/jsf/html"
      xmlns:f="http://java.sun.com/jsf/core">
<head>
<title>Step2: Form</title>
</head>
<body>
      <h1>User Display</h1>
      <h:form>
            <table border="0">
                  <tr>
                        <td>Your Login</td>
                        <td><h:outputText value="#{loggedUser.login}"/>
                          </td>
                  </tr>
                  <tr>
                        <td>Your surname</td>
                        <td><h:outputText value="#{loggedUser.surname}"/>
                          </td>
                  </tr>
                  <tr>
                        <td>Your lastname</td>
                        <td><h:outputText value="#{loggedUser.lastname}"/>
```

```
                                </td>
                        </tr>
                        <tr>
                                <td>Your age</td>
                                <td><h:outputText value="#{loggedUser.age}"/>
                                </td>
                        </tr>
                </table>
        </h:form>
</body>
</html>
```

5.11.7. Quelle est la portée de LoggedUser?

5.11.8. Quel élément a créé cet Objet ?

5.12. Tester votre application JSF

5.12.1. Rappeler la fonction de chaque classe et objet du paquage step4