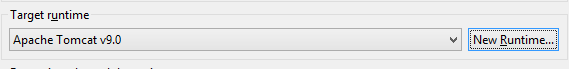
# Data Centric RAD

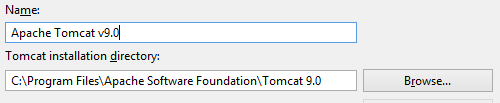
## Lab 10 JSF with Database

### Setting up a JSF Project

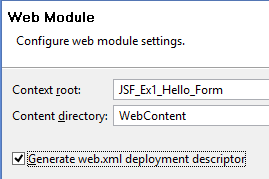
* Open Eclipse Java EE IDE for Web Developers.
* Select *File/New/Other/Web/Dynamic Web Project*.
* Enter a project name and select the Target runtime as shown:



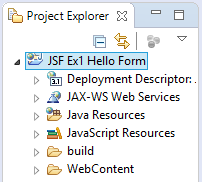
* + If there are no options, press *New Runtime*
  + Click on Apache Tomcat v9.0 and press *Next*
  + Browse to the Apache Tomcat installation directory and press *Finish*



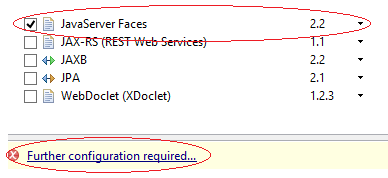
* Press *Next* until the *Web Module* screen appears, select the *Generate web.xml deployment descriptor* and press *Next*.



* Press *Finish* on the *JSF Capabilities* window.
* The Dynamic Web Application project has now been created.



* Goto <http://repo1.maven.org/maven2/org/glassfish/javax.faces/2.2.9/> and download the following jar file [javax.faces-2.2.9.jar](http://repo1.maven.org/maven2/org/glassfish/javax.faces/2.2.9/javax.faces-2.2.9.jar) to the *WebContent\WEB-INF\lib* in the project.
* Download the MySQL driver from <https://dev.mysql.com/downloads/connector/j/>. Extract the jar file and place it in *WebContent\WEB-INF\lib* in the project.
* Select *Project/Properties/Project Facets*.
* Select *JavaServerFaces 2.2* and then click *Further configuration required…*



* The *Modify Faceted Project* window now opens. Change the *JSF Implementation Library Type* to *Disable Library Configuration* and press *OK*, then press *OK* on the *Properties* window.



* Download MySQL JDBC Driver from <https://dev.mysql.com/downloads/connector/j/> unzip it and put the Jar file in the WEB-INF/lib folder of your application.
* Create a new file called *context.xml* in the *WebContent/META-INF/* folder and add the following to it:

<Context>

<Resource name="jdbc/employeesdb14"

auth="Container" type="javax.sql.DataSource"

maxActive="20" maxIdle="5" maxWait="10000"

username="root" password=""

driverClassName="com.mysql.jdbc.Driver"

url="jdbc:mysql://localhost:3306/employeesdb14"/>

</Context>

* Add the following to the *web.xml* file in *WebContent/WEB-INF/*:

<resource-ref>

<description>EmployeesDB Datasource</description>

<res-ref-name>jdbc/employeesdb14</res-ref-name>

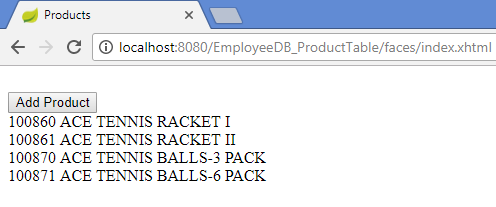
<res-type>javax.sql.DataSource</res-type>

<res-auth>Container</res-auth>

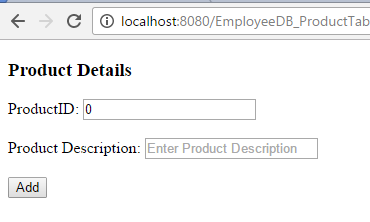
</resource-ref>

### Part 1 Product Details

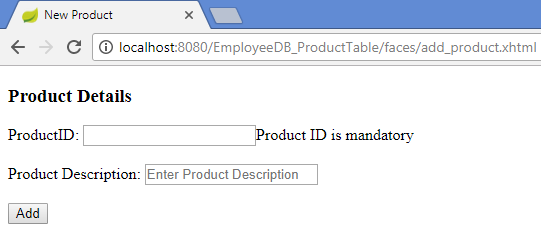
* Download and import *employeesDB14.sql* from Moodle.
* Write a JSF/JDBC application that queries the *product* table in the *employees* database and displays the first 4 Products on a web page *(index.xhtml)*.



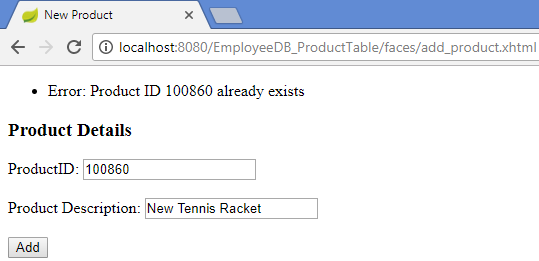
* The web page should have a button called *Add Product* that allows the user to enter details of a new product and add this product to the database.



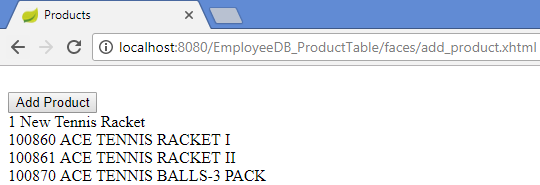
* If no Product ID is entered an error message must be displayed beside the empty field.



* If a Product ID is entered that already exists, an error message must be displayed.



* If a Product is successfully entered, the user should be brought back to the main page where the first 4 products are shown.



## NOTES:

* The database should only be accessed when the view (*index.xhtml*) loads.
* All database access should be performed in a DAO.
* A Controller class should handle user interaction and interface with the DAO.
* Displaying the first 4 products can be hardcoded e.g.:

#{productController.products[0].productID}

#{productController.products[0].description}

<br/>

#{productController.products[1].productID}

#{productController.products[1].description}

<br/>

#{productController.products[2].productID}

#{productController.products[2].description}

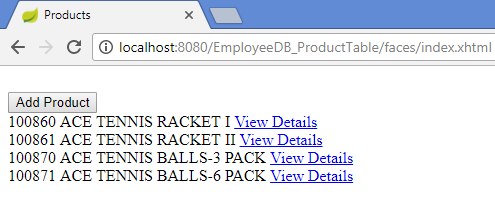
<br/>

#{productController.products[3].productID}

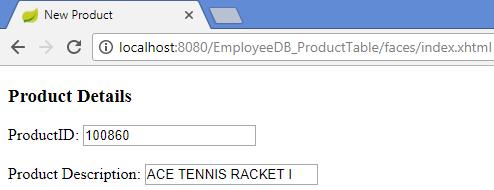
#{productController.products[3].description}

### Part 2 Product Details in separate view

* Update Part 1 so that the list of the first 4 products looks as follows:



* When the View Details is clicked, the Product ID and Description of the appropriate product is shown in a view called *view\_product.xhtml*.



## NOTES:

* The “View Details” is a h:commandLink that calls a method in the Product Controller which in turn calls a method in the DAO which gets the details of the appropriate product.