**Binding Database Table to a Table Component Using JPA**

**Create the model project**

* Click on the Create project icon in the toolbar (second button)
* Select General -> Java Application
  + Project Name: TestModelApp
  + Package: com.samples.model

**Create the Web project**

* Click on the Create project icon in the toolbar (second button)
* Select Web -> Visual Web Application
  + Project Name: TestWebApp
  + Package: com.samples.web
* Add the project TestModelApp as dependent project to Web Project
  + Right Click on the project node and select "properties"
  + In the dialog click on the "Library" node in the left
  + In the right hand side click on the add project and select the location of "TestModelApp"

**Create the Database Connection**

* Select the "Runtime" tab
* Open the Databases Node and right click on the Drivers Node and select menu item "New Driver"  
  In the resulting dialog select your MySql connextor/j JDBC driver
* Right Click on the Databases Node and select "New Connection"
* In the Name combobox select MySql (Connector/J) and then Provide the necessary connection parameters. For Ex
  + Database URL:jdbc:mysql://localhost/sample
  + User Name: sample
  + Password: sample
* Open the table node in the added connection and make sure the connection is correct.

*Note: In this walk through, it is assumed that you already have a "sample" MySQL database that has one table called "Users". The table "Users" has four columns "user\_id" (Primary Key, Type: INTEGER), userName, password, email\_address (all of type String).*

**Create the Entity Class  from Database in the Model Project**

* Right click on the project "TestModelApp" and select New -> File/Folder -> Persistence -> Entity Classes from Database
* Select Database Connection "Sample" and select "Users" as the Selected Tables
* In the next tab of the wizard click on the Create Persistence Unit and call it "samplePU"
* In the Entity Bean Tab of the Wizard provide
  + ClassName: User (IDE may suggest Users)
  + Package: com.samples.model
* Check that the persistence Unit is created correctly
  + Open Source Packages -> META-INF and double click on the Persistence.xml and check that the file has been created with persistence-unit element as shown below.

<persistence-unit name="samplePU" transaction-type="RESOURCE\_LOCAL">

<provider>

oracle.toplink.essentials.ejb.cmp3.EntityManagerFactoryProvider

</provider>

<class>com.samples.model.User</class>

<properties>

<property name="toplink.jdbc.url" value="jdbc:mysql://localhost/sample"/>

<property name="toplink.jdbc.user" value="sample"/>

<property name="toplink.jdbc.driver" value="com.mysql.jdbc.Driver"/>

<property name="toplink.jdbc.password" value="sample"/>

</properties>

</persistence-unit>

**Create the Entity Controller Class**

* In the package "com.samples.model" create a class called "UserController" and add the code

private EntityManagerFactory emf;

private EntityManager getEntityManager() {

if(emf == null){

emf = Persistence.createEntityManagerFactory("samplePU");

}

return emf.createEntityManager();

}

public Users[] getUsers() {

EntityManager em = getEntityManager();

try{

Query q = em.createQuery("select c from Users as c");

return (User[]) q.getResultList().toArray(new User[0]);

} finally {

em.close();

}

}

**Bind Entity Bean to the Table Component**

For this first we need to create a property that returns the array of User objects and then bind that array to the Table Component. This User[] array will be initialized using  JPA Entity Manager.

* Add a Property called users to the Session Bean.
  + Double  click on the SessionBean1 node in the outline window and open  it in the Java Editor.
  + Type the line private User[] users;
    - Right click on this line and select the action "Generate Code"
    - In the resulting popup menu select "Getter and Setter"
* Add a method  called updateUsers() as below that will be used to initialize the users property.

public void updateUsers(){

UsersController usersController = new UsersController();

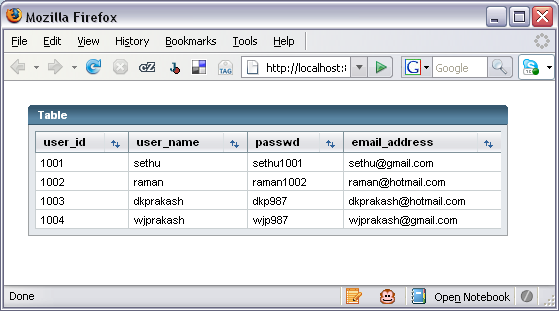
users = usersController.getUsers();

}

* Fix the needed import for UserController
* Now call  the initialization method in the init() of Sessionbean1.java

updateUsers();

* Compile the project (Note: This is a needed step)
* Open the Page1.jsp in the designer
  + Note: if the designer is already opened click on the refresh button
* Drag and drop a Table Component on to the designer.
* Click on the Table Component and select users as the binding array.
  + Adjust the columns to be displayed  if needed
* Deploy the project and the Table Component should display the data as



**Modify the UsersController to add support for Add, Delete and Update Entity**

* First add the code to UserController to add a record

public boolean addUser(User user) {

EntityManager em = getEntityManager();

try{

em.getTransaction().begin();

em.persist(user);

em.getTransaction().commit();

} finally {

em.close();

return false;

}

}

* Next  add the code to UserController to remove a record

public boolean removeUser(Users user) {

EntityManager em = getEntityManager();

try{

em.getTransaction().begin();

Users userx = em.find(Users.class, user.getUserId());

em.remove(userx);

em.getTransaction().commit();

} finally {

em.close();

return false;

}

}

* Finally add the code to UserController to update a record

public boolean updateUser(User user) {

EntityManager em = getEntityManager();

try{

em.getTransaction().begin();

User userx = em.find(User.class, user.getUserId());

userx.setUserName(user.getUserName());

userx.setPasswd(user.getPasswd());

userx.setEmailAddress(user.getEmailAddress());

em.getTransaction().commit();

} finally {

em.close();

return false;

}

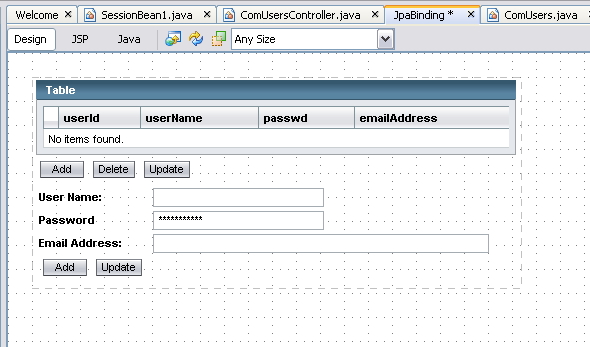
}

**Modify Page1 Page for User interaction for Add, Delete and Update Entity**

Modify the web page to add the following

* Add a GridPanel (id: mainContainer) to the designer
* Add a Table Component in to the GridPanel and bind it to the users array  (added via walk through [Binding Database Table to a Table Component Using JPA](http://www.winstonprakash.com/articles/netbeans/JPADataBinding.html))
  + For ease of arrangement  use outline (Ex. Drag Table component and drop it over GridPanel if it is outside)
* Add another grid panel (id: buttonPanel) and add three button to it.
  + Set the column count to 3 via property sheet
  + Set the properties of  first button
    - id: addButton
    - text: Add
  + Set the properties of  second button
    - id: deleteButton
    - text: Add
  + Set the properties of  third button
    - id: updateButton
    - text: Add
* Add another Grid Panel (id: addUpdatePanel) to the mainContainer Grad Panel
  + Set its columns to 2
  + Add the following to the addUpdatePanelGrid Panel
    - Label (text: UserName) TextField (id: userNameField)
    - Label (text: Password) PasswordField (id: passwordField)
    - Label (text: Email Address) TextField (id: emailAddressField)
    - Add another Grid Panel (id: addUpdateButtonPanel) and add two buttons to it.
      * Button (id: addRecordButton, text: Add)
      * Button (id: updateRecordButton, text: Update)
* Right Click on the Table Component and Select Table Layout.
  + Click on the "New" button to add a new column.
  + Using "Up" button move the newly added column to the top of the list.
  + Remove the header text.
  + Select Radio Button for component type and remove the text in Value Expression Field.

The designer should like like



* You need to write some logic to hold information about the selected radio button in the Page1.java. So add the following code to it.

private TableSelectPhaseListener tablePhaseListener =

new TableSelectPhaseListener();

public void setSelected(Object object) {

RowKey rowKey = (RowKey)getValue("#{currentRow.tableRow}");

if (rowKey != null) {

tablePhaseListener.setSelected(rowKey, object);

}

}

public Object getSelected(){

RowKey rowKey = (RowKey)getValue("#{currentRow.tableRow}");

return tablePhaseListener.getSelected(rowKey);

}

public Object getSelectedValue() {

RowKey rowKey = (RowKey)getValue("#{currentRow.tableRow}");

return (rowKey != null) ? rowKey.getRowId() : null;

}

public boolean getSelectedState() {

RowKey rowKey = (RowKey)getValue("#{currentRow.tableRow}");

return tablePhaseListener.isSelected(rowKey);

}

* Go to JSP page and add the Java Scripts to the table component.

<script>

function initAllRows() {

var table = document.getElementById("form1:table1");

table.initAllRows();

}

</script>

* Press Alt-Shift-F to fix the imports.
* Now do the following to tell the table component to keep track of selection.
  + Select the Radio Button (from outline, expand all the way down to radio button as shown below)
* Right Click on the Radio Button and bring up the "Properties Binding dialog" from the right click menu.
  + Set the properties
    - selected="#{Page1.selected}"
    - selectedValue="#{Page1.selectedValue}"
    - name="Page1.radioButton1.id"
  + In the JSP the Radio Button Tag should look like

<webuijsf:radioButton binding="#{Page1.radioButton1}" id="radioButton1"

name="#{Page1.radioButton1.id}" selected="#{Page1.selected}"

selectedValue="#{Page1.selectedValue}"/>

* Select the corresponding Table Column in the outline and set the properties  via Properties Binding dialog
  + onClick="setTimeout('initAllRows()', 0)"
  + selectId="#{Page1.radioButton1.id}"
  + In the JSP the TableColumn Tag should look like

<webuijsf:tableColumn binding="#{Page1.tableColumn5}" id="tableColumn5"

onClick="setTimeout('initAllRows()', 0)" selectId="#{Page1.radioButton1.id}">

* Select the Table Row Group in the outline and set the property via Properties Binding dialog
  + selected=#{Page1.selectedState}
  + In the JSP the Table Row Group Tag should look like

<webuijsf:tableRowGroup binding="#{Page1.tableRowGroup1}"

id="tableRowGroup1" rows="10" selected="#{Page1.selectedState}"

sourceData="#{Page1.usersDataProvider}"

sourceVar="currentRow">

* Add two  boolean variables (call them addRequest & updateRequest) to the Page1 page and set their value to false
* Double click on the addButton Button and add the following code. This code merely sets a flag to enable the addUpdatePanel.

private boolean addRequest = false;

public String addButton\_action() {

addRequest = true;

return null;

}

* Double click on the updateButton Button and add the following code. Like in addButton action handler, this code also just sets a flag to enable the addUpdatePanel.

private boolean updateRequest = false;

public String updateButton\_action() {

updateRequest = true;

return null;

}

* Double click on the deleteButton Button and add the following code. The selected rowKey is obtained from the TableRowGroup and using the index corresponding data is removed from the database using UserController.

public String deleteButton\_action() {

if (getTableRowGroup1().getSelectedRowsCount() > 0){

RowKey[] selectedRowKeys = getTableRowGroup1().getSelectedRowKeys();

Users[] users = getSessionBean1().getUsers();

int rowId = Integer.parseInt(selectedRowKeys[0].getRowId());

Users user = users[rowId];

// Remove the Entity from the database using UserController

UserController userController = new UsersController();

userController.removeUser(user);

}

return null;

}

* Now double click the addRecordButton Button and add the following code. Here we are creating a new User Entity bean from the addUpdatePanel and adding that Entity Bean data to database using UserController

public String addRecordButton\_action() {

// Create a new User Entity

User newUser = new User();

newUser.setUserName((String) userNameField.getText());

newUser.setPasswd((String) passwordField.getText());

newUser.setEmailAddress((String) emailAddressField.getText());

// Add the new Entity to the database using UserController

UserController userController = new UserController();

userController.addUser(newUser);

showAddPanel = false;

return null;

}

* Now double click the updateRecordButton Button and add the following code. Here the User  Entity bean is obtained using the selection in the table and updated with data from the addUpdatePanel data and then that Entity Bean data is used to update the database data using UserController

public String updateRecordButton\_action() {

// Create a new Users Entity

RowKey[] selectedRowKeys = getTableRowGroup1().getSelectedRowKeys();

Users[] users = getSessionBean1().getUsers();

int rowId = Integer.parseInt(selectedRowKeys[0].getRowId());

Users user = users[rowId];

user.setUserName((String) userNameField.getText());

user.setPasswd((String) passwordField.getText());

user.setEmailAddress((String) emailAddressField.getText());

// Update the database table data using UserController

UserController userController = new UserController();

userController.updateUser(newUser);

addRequest = false;

return null;

}

* Add the following code to prerender() method. Based on the addRequest or updateRequest boolean value show or hide the addUpdatePanel and also update the users data based on any modification or data addition occurred.

public void prerender() {

if(addRequest){

addUpdatePanel.setRendered(true);

addRecordButton.setRendered(true);

updateRecordButton.setRendered(false);

userNameField.setText("");

passwordField.setText("");

emailAddressField.setText("");

}else if (updateRequest){

if (getTableRowGroup1().getSelectedRowsCount() > 0){

// Get the data from the selected Entity and update the fields

RowKey[] selectedRowKeys = getTableRowGroup1().getSelectedRowKeys();

User[] users = (User[]) getSessionBean1().getUsers();

int rowId = Integer.parseInt(selectedRowKeys[0].getRowId());

Users user = users[rowId];

userNameField.setText(user.getUserName());

passwordField.setText(user.getPasswd());

emailAddressField.setText(user.getEmailAddress());

addUpdatePanel.setRendered(true);

addRecordButton.setRendered(false);

updateRecordButton.setRendered(true);

}

}else{

addUpdatePanel.setRendered(false);

}

// Refresh the users data array in the session bean to to show

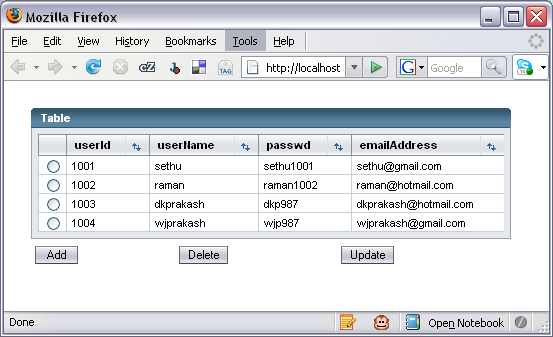
// the newly added data or modified data in the Table Component

getSessionBean1().updateUsers()

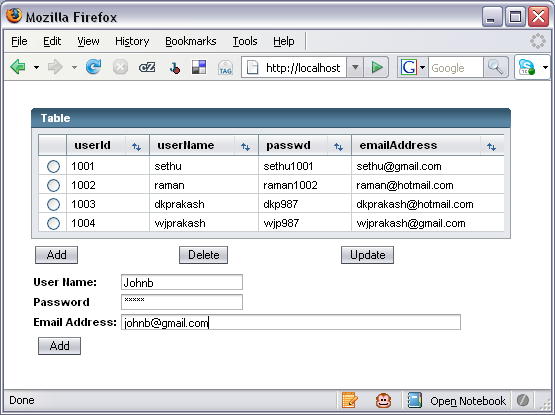
}

**Deploy the application and test**

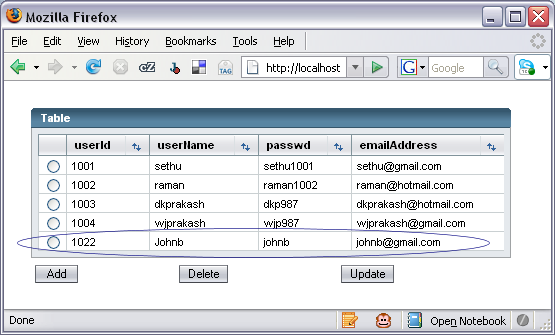
* After deploying the application it should look like



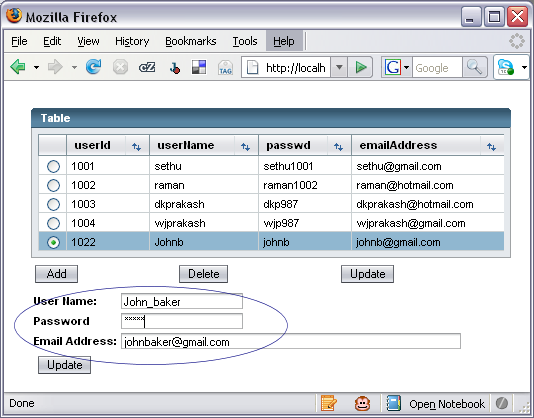
* Click on the Add button and add some values to the fields



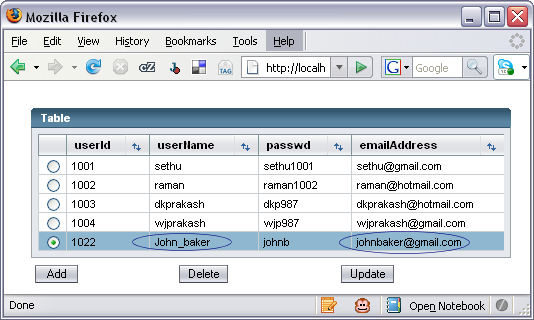
* After adding the value click on the Add button in the bottom



* The Table should be now refreshed with the newly added record. Select the row and click on the Update button and modify the values



* Click on the Update button at the bottom to see the updated values



* Now select the newly added row as show above and click on the delete button. The table should be refreshed after deleting the row as shown below