Index.xhtml / welcomeJSF.jsp

```
…….
  <h:body>
    <h1>Customer List</h1>
    <h:form >
        <h:panelGroup id="messagePanel" layout="block">
          <h:messages errorStyle="color: red" infoStyle="color: green"
layout="table"/>
        </h:panelGroup>
<h:panelGrid columns="3">
          <h:outputLabel value="Filter by state" for="selectState" />
<h:selectOneMenu id="selectState" label="selectedState"
value="#{custController.stateFilter}" >
          <f:selectItem itemLabel="All states" itemValue="all"/>
          <f:selectItem itemLabel="Florida" itemValue="FL"  />
          <f:selectItem itemLabel="California" itemValue="CA"/>
          <f:selectItem itemLabel="Texas" itemValue="TX"/>
          <f:selectItem itemLabel="New York" itemValue="NY"/>
          <f:selectItem itemLabel="Mississippi" itemValue="MI"/>
          <f:selectItem itemLabel="Georgia" itemValue="GA"/>
</h:selectOneMenu>
<h:commandButton value="apply filter"
action="#{custController.updateItems()}"/>
</h:panelGrid>
<h:outputText escape="false" value="No customers were found"
rendered="#{custController.items.rowCount == 0}"/>
<h:panelGroup rendered="#{custController.items.rowCount > 0}">
<h:dataTable value="#{custController.items}" var="item" border="0"
cellpadding="2" cellspacing="0" rules="all" style="border:solid 1px">
          <h:column>
            <f:facet name="header">
              <h:outputText value="Customer ID"/>
            </f:facet>
            <h:outputText value="#{item.customerId}"/>
          </h:column>
          <h:column>
            <f:facet name="header">
              <h:outputText value="Name"/>
            </f:facet>
            <h:outputText value="#{item.name}"/>
          </h:column>
          <h:column>
            <f:facet name="header">
              <h:outputText value="City"/>
            </f:facet>
            <h:outputText value="#{item.city}"/>
          </h:column>
```

Filter

Customer table

To populate the dropbox from a database table, the controller bean would call the findAll query. Suppose results are stored in stateList, then the JSF page would be:<f:selectItems var="state" value="#{custController.stateList}" itemLabel="#{state.stateName}" itemValue="#{ state.stateID}"/>

```
                <h:column>
                  <f:facet name="header">
                    <h:outputText value="State"/>
                  </f:facet>
                  <h:outputText value="#{item.state}"/>
                </h:column>
              </h:dataTable>
            </h:panelGroup>
</h:form>
</h:body>
</html>
```

---

custController, the JSF managed bean

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */
package jsf;
import java.io.Serializable;
import javax.ejb.EJB;
import javax.inject.Named;
//import javax.enterprise.context.Dependent;
import javax.enterprise.context.SessionScoped;
import javax.faces.model.DataModel;
import javax.faces.model.ListDataModel;
import session.CustomerFacade;

@Named(value = "custController")
@SessionScoped
//@Dependent
public class custController implements Serializable{

//    private entity.Customer current;
   private DataModel items = null;
   private String stateFilter="all";
   @EJB
   private session.CustomerFacade ejbFacade;

public custController() {
   }

public String updateItems() {
       items = getItems();       //update the data model
       return "index";       //return a string indicating page
                                 to display
    }
```

Variables matching the XHTML form, with the filter initialised to all states

Method called from the filter command button: regenerate the data table based on the filter selection, and return the name of the pate to display.

Fills the data table with details from the customer table. getFacade returns the customer façade, which has methods for both a finalAll query and a findByState query.

```
   public DataModel getItems() {

        if (getStateFilter().equals("all")) {
        items = new ListDataModel(getFacade().findAll());
        } else {
        items = new ListDataModel(getFacade().findByState(getStateFilter()));
              }
        return items;
   }

   public String getStateFilter() {
      return stateFilter;
   }
```

Get and set methods for stateFilter, which is the attribute for the dropdown box on the xhtml form.

```
   public void setStateFilter(String stateFilter) {
      this.stateFilter = stateFilter;
   }

private session.CustomerFacade getFacade() {
        return ejbFacade;
   }
}
```

Session bean: customer facade

```
package session;
imports …….

@Stateless
public class CustomerFacade extends AbstractFacade<Customer> {

   @PersistenceContext(unitName = "CustApp-ejbPU")
   private EntityManager em;
   private Customer entityClass;

   @Override
   protected EntityManager getEntityManager() {
      return em;    }

   public CustomerFacade() {
      super(Customer.class);     }

   public List<entity.Customer> findByState(String
state) {
        if (state.isEmpty()) {
```

Two queries need to be implemented in the session bean:

Findall is in abstract façade by default and so inherited by customer façade

This method calls the findByState named query in the entity bean, setting the parameter to state – which has been passed from the xhtml page via the custController

```
            Query query = em.createNamedQuery("Customer.findAll");
            return query.getResultList();
        } else {
            Query query = em.createNamedQuery("Customer.findByState");
            query.setParameter("state", state);
            return query.getResultList();
        }

        // the follow code shows how to implement the same query dynamically using
criteria builder
/*
        CriteriaBuilder cb = em.getCriteriaBuilder();
        CriteriaQuery cq = cb.createQuery();

        Root<Customer> customer = cq.from(Customer.class);
        cq.select(customer);
        cq.where(cb.equal(customer.get("state"),state));
        return em.createQuery(cq).getResultList();
        */
    }
}
```

Extract from the entity class:

Customer entity class

```
@Entity
@Table(name = "CUSTOMER")
@NamedQueries({
    @NamedQuery(name = "Customer.findAll", query = "SELECT c FROM Customer
c"),
…….
    @NamedQuery(name = "Customer.findByState", query = "SELECT c FROM
Customer c WHERE c.state = :state"),
})

public class Customer implements Serializable {
….
```

findAll and findByState named queries in the entity bean