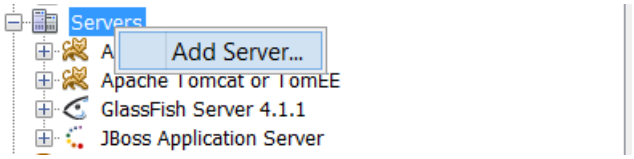
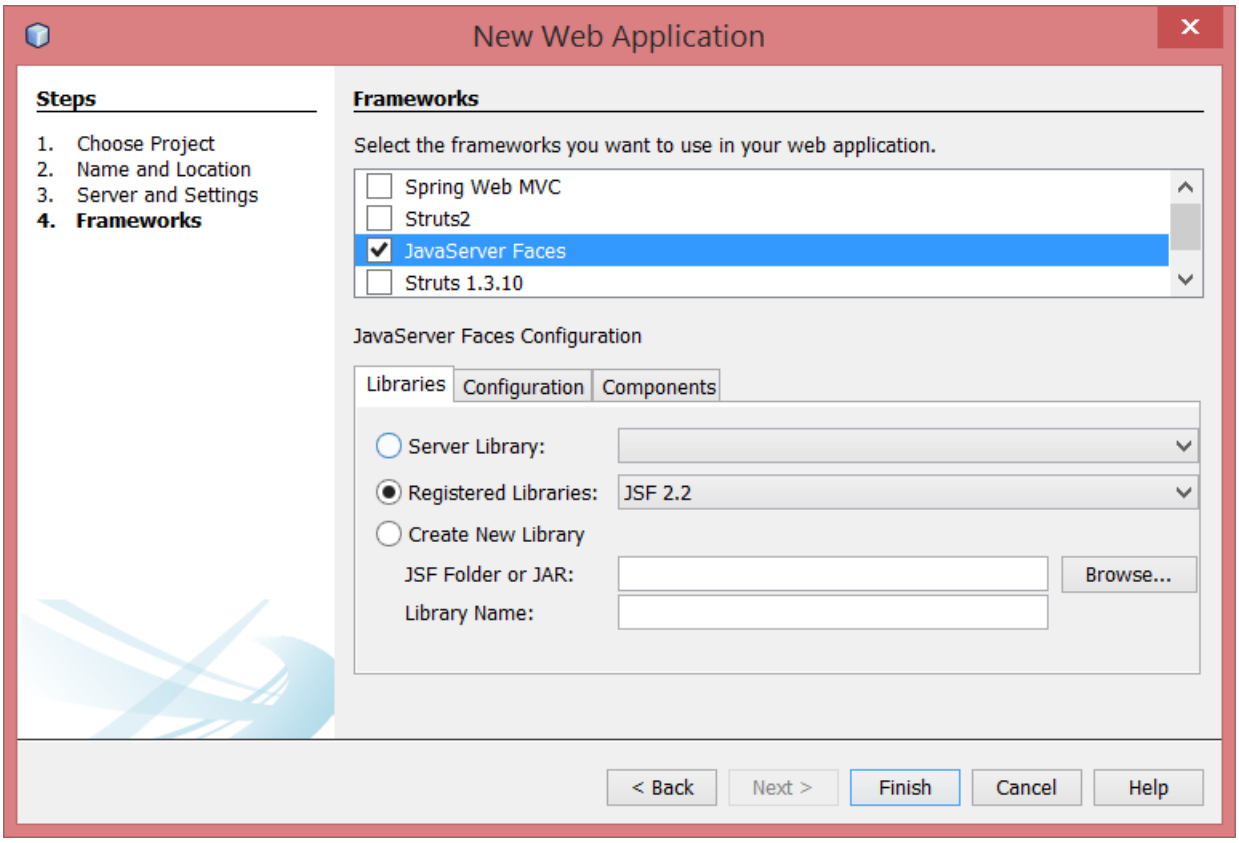
Guide: Tạo Web sử dụng JSF với 2 bảng

1. Add server Apache Tomcat và khởi động lên trước.

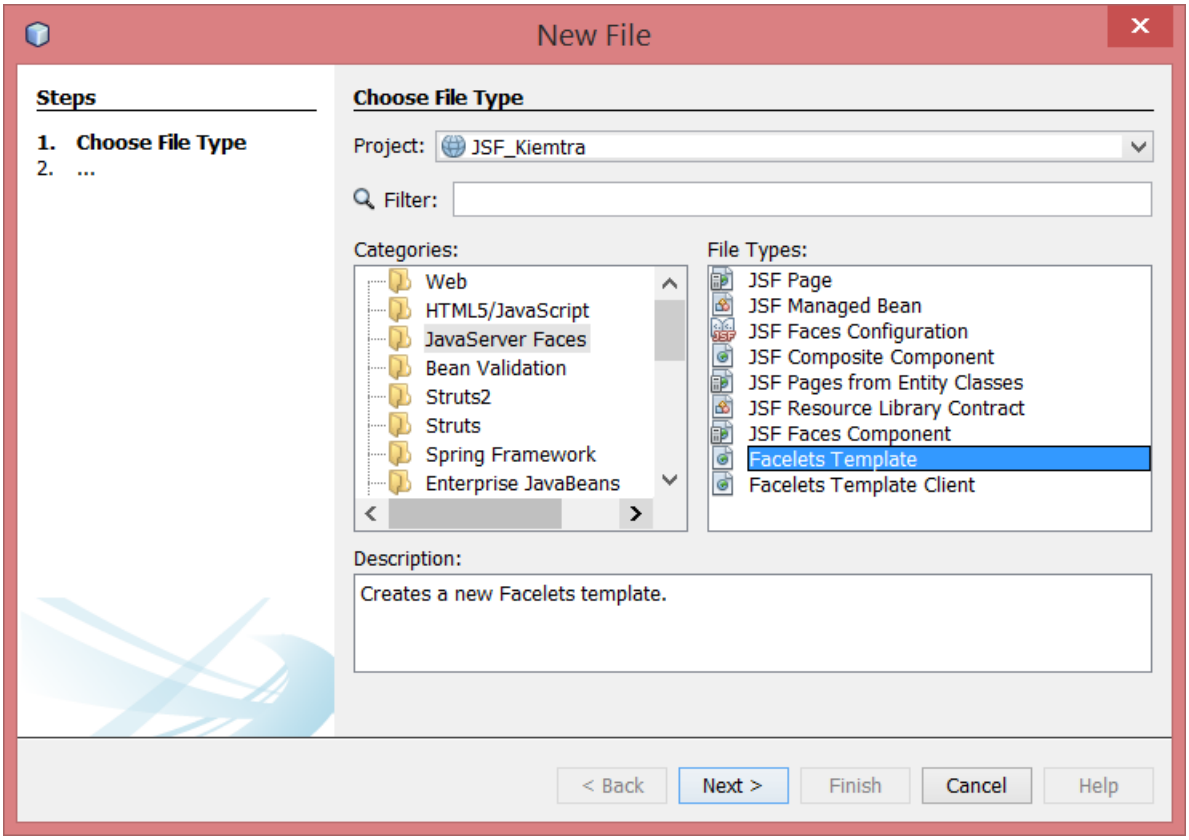


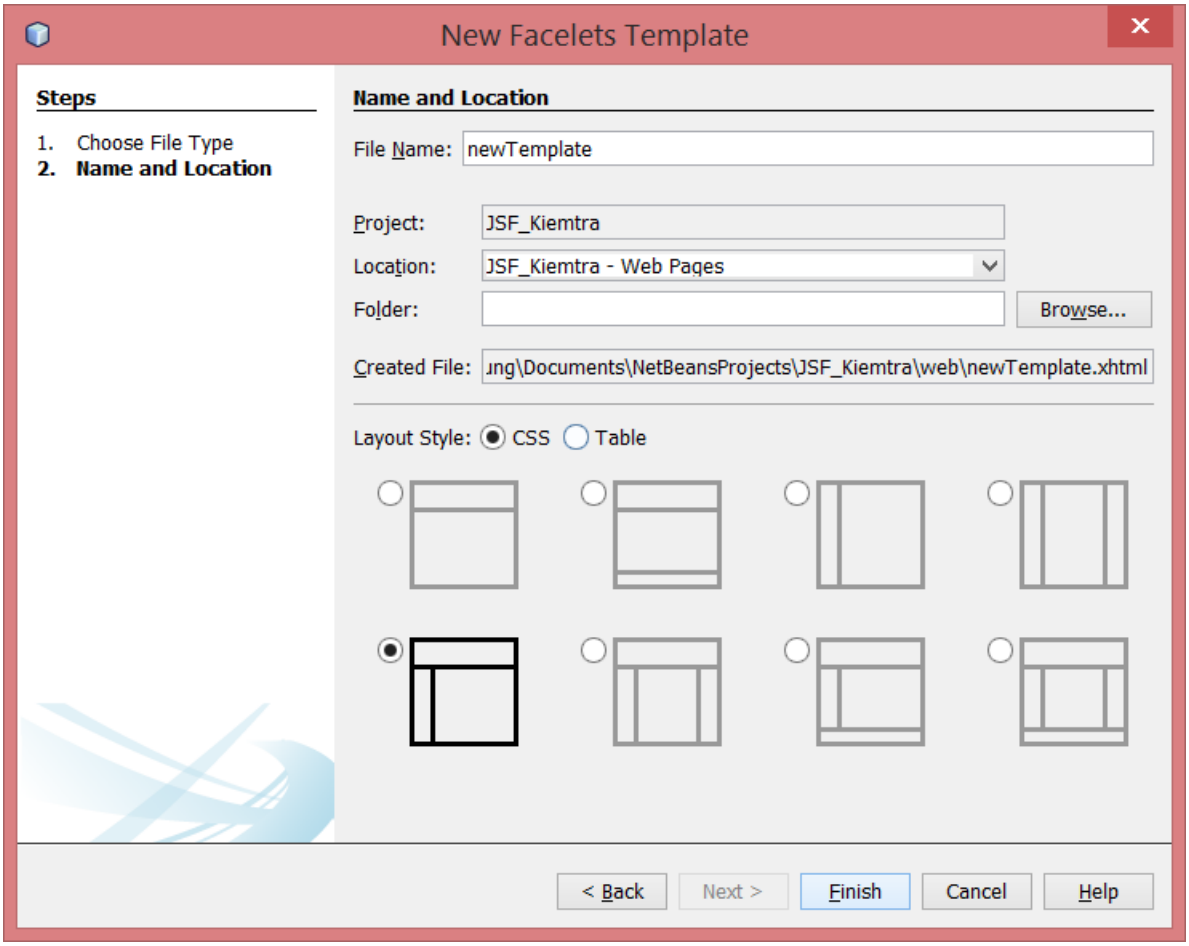
Luu ý: nếu khởi động gặp lỗi thì fix trong file: Apache Tomcat -> bin -> catalina.bat, view with Notepad++, sửa tại 2 dòng :noJuliConfig và :noJuliManager (Khoảng dòng 179) bằng cách bỏ ngoặc kép ""

2. Tạo Project JavaWeb -> WebApplication, chọn server TomCat và JavaServer Faces.

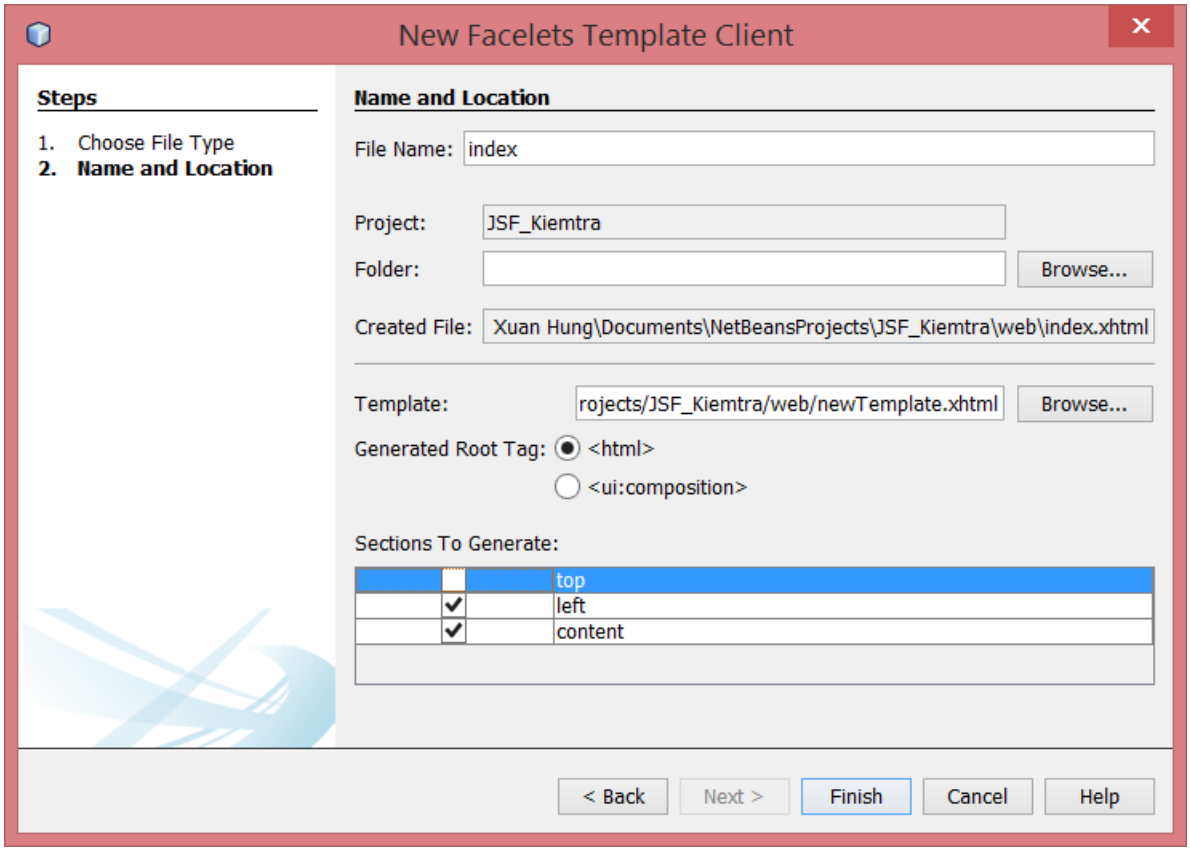


3. General class từ database về.(Note: add library sqlJDBC.jar vào, cổng thường dùng là 50239)
4. Add 1 template bằng cách right-click vào Web Pages:





5. Tiếp tục, add 1 template client vào, sử dụng chính template vừa tạo:
(note: nên xóa page index.xhtml đã có từ lúc đầu)



Chỉ √ vào phần mà trang đó trực tiếp sử dụng đến

6. Viết DAO cho các class vừa thêm.

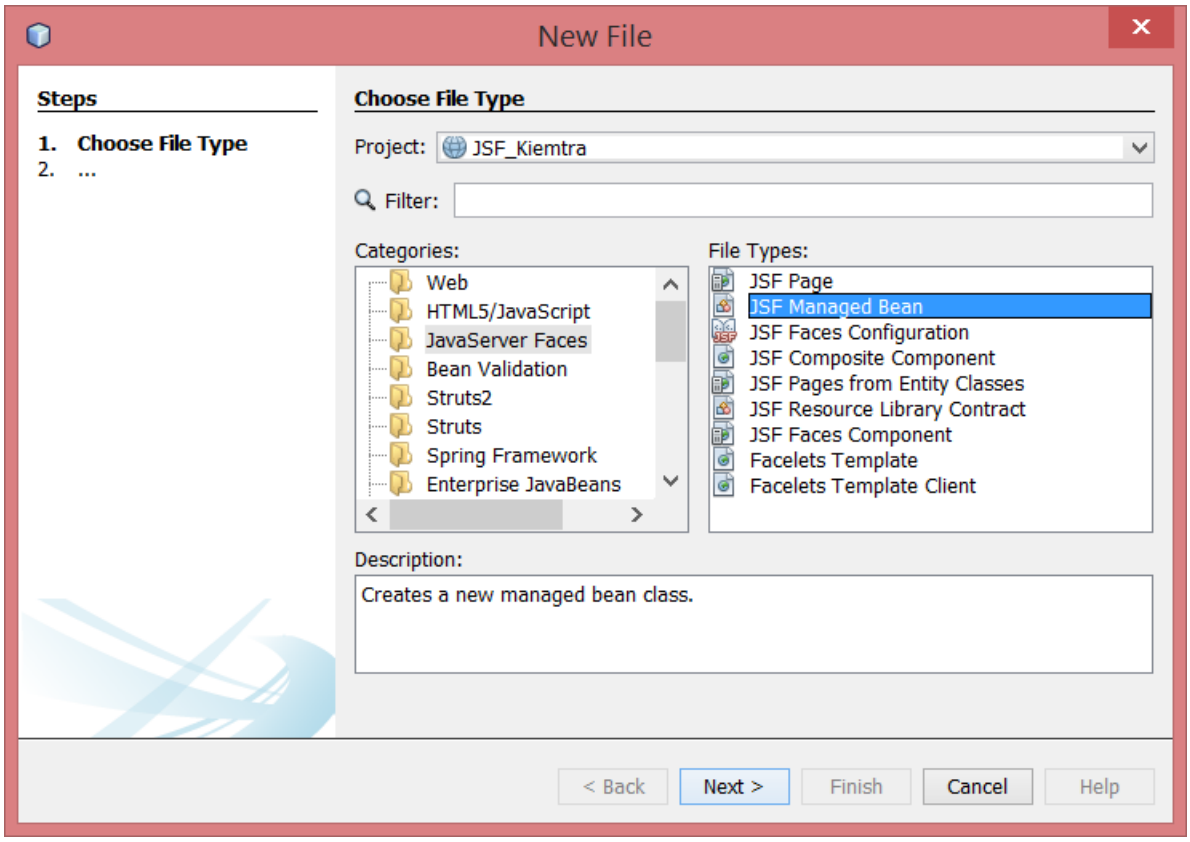
```
public List<Product> loadAllProducts(){
    EntityManager em = Persistence.createEntityManagerFactory("JSF_KiemtraPU").createEntityManager();
    String hql = String.format("select a from %s a", Product.class.getName());
    return em.createQuery(hql).getResultList();
}
public List<Product> loadAllProductsByManuID(String id){
    EntityManager em = Persistence.createEntityManagerFactory("JSF_KiemtraPU").createEntityManager();
    String hql = String.format("select a from %s a where a.manu.manuID = :id", Product.class.getName());
    return em.createQuery(hql)
            .setParameter("id", id)
            .getResultList();
}

public List<Manufacturer> loadAllManus(){
    EntityManager em = Persistence.createEntityManagerFactory("JSF_KiemtraPU").createEntityManager();
    String hql = String.format("select a from %s a", Manufacturer.class.getName());
    return em.createQuery(hql).getResultList();
}

public Product loadDetailProduct(String id){
    EntityManager em = Persistence.createEntityManagerFactory("JSF_KiemtraPU").createEntityManager();
    return em.find(Product.class, id);
}

public void updateProduct(Product p){
    EntityManager em = Persistence.createEntityManagerFactory("JSF_KiemtraPU").createEntityManager();
    Product find = em.find(Product.class, p.getPID());
    em.getTransaction().begin();
    if(find!=null){
        find.setPID(p.getPName());
        find.setCreatedDate(p.getCreatedDate());
        find.setManu(p.getManu());
    }
    em.getTransaction().commit();
```

7. Add JSF Manager Bean.



Note: sửa **@RequestScoped** => **@SessionScoped**

8. Tại đây sẽ viết các property và method để sử dụng trong các trang JSF
   Trước tiên để show dữ liệu ra ta sẽ cần **listProduct** và **listManufacturer**

```java
@ManagedBean
@SessionScoped
public class ProductBean {

    private List<Manufacturer> listManu;
    private List<Product> listProduct;
    private ProductDAO dao;
    public ProductBean() {
        dao = new ProductDAO();
        listProduct = dao.loadAllProducts();
        listManu = dao.loadAllManus();
    }

    public List<Manufacturer> getListManu() {
        return listManu;
    }

    public void setListManu(List<Manufacturer> listManu) {
        this.listManu = listManu;
    }

    public List<Product> getListProduct() {
        return listProduct;
    }

    public void setListProduct(List<Product> listProduct) {
        this.listProduct = listProduct;
    }


}
```

Các property đều phải thêm các hàm set-get thì trang JSF mới sử dụng được.

9. Show dữ liệu trong trang **index**, sử dụng **h:dataTable** hoặc **ui:repeat** để show dữ liệu dạng list, sử dụng **h:panelGrid** để show dữ liệu của 1 đối tượng.

```xml
<ui:define name="left">
    <h:form>
        <ul>
            <ui:repeat value="#{productBean.listManu}" var="it">
                <li><h:commandLink value="#{it.manuName}" action="#{productBean.loadProductByManu(it.manuID)}"/></l
            </ui:repeat>
            <li><h:commandLink value="Show all" action="#{productBean.loadProductByManu('')}"/></li>
        </ul>
    </h:form>
</ui:define>
```

```
<ui:define name="content">
    <h:form>
        <h:dataTable value="#{productBean.listProduct}" var="it" class="tableProduct" cellspacing="0">
            <h:column>
                <f:facet name="header">ProductID</f:facet>
                <h:outputText value="#{it.PID}"/>
            </h:column>
            <h:column>
                <f:facet name="header">Product name</f:facet>
                <h:outputText value="#{it.PName}"/>
            </h:column>
            <h:column>
                <f:facet name="header">Created Date</f:facet>
                <h:outputText value="#{it.createdDate}">
                    <f:convertDateTime pattern="dd-MM-yyyy"/>
                </h:outputText>
            </h:column>
            <h:column>
                <f:facet name="header">Manufacturer</f:facet>
                <h:outputText value="#{it.manu.manuName}"/>
            </h:column>
            <h:column>
                <f:facet name="header">Action</f:facet>
                <h:commandLink value="Edit"/>
                <h:commandLink value="Remove" action="#{productBean.deleteProduct(it.PID)}" style="margin-left: 10p
            </h:column>
        </h:dataTable>
    </h:form>
</ui:define>
```

10. Thêm trang **edit** bằng cách add new 1 template client, sử dụng template ban đầu. Tại đây sẽ hiện thị chi tiết sản phẩm với 1 combobox hiển thị hãng sản xuất
Để chuyển hướng từ trang **index** sang **edit**, hàm **loadDetailProduct** trong ProductBean sẽ return về "edit" để chuyển. Vì trang edit cần hiển thị 1 đối tượng nên trong ProductBean sẽ thêm 1 đối tượng Product để hiển thị

```
    private Product product;

    public Product getProduct() {
        return product;
    }

    public void setProduct(Product product) {
        this.product = product;
    }
```

Hàm **loadDetailProduct** trong ProductBean:

```
public String loadDetailProduct(String id) {
    product = dao.loadDetailProduct(id);
    return "edit";
}
```

Trong trang **index**, button **Edit** sẽ thêm action như sau:

```
<h:commandLink value="Edit" action="#{productBean.loadDetailProduct(it.PID)}"/>
```

Tương tự, nút **Remove**:

```
public void deleteProduct(String id){
    dao.deleteProduct(id);
    listProduct = dao.loadAllProducts();
}
```

```
<h:commandLink value="Remove" action="#{productBean.deleteProduct(it.PID)}" style="margin-left: 10px"/>
```

11. Tại trang **edit.xhtml**, sử dụng **h:panelGrid** để hiển thị Product:

```
<h:form>
    <h:panelGrid columns="3">
        <h:outputLabel value="Product id"/>
        <h:outputText value="#{productBean.product.PID}" id="txtID" />
        <h:message for="txtID" />

        <h:outputLabel value="Product name"/>
        <h:inputText value="#{productBean.product.PName}" id="txtName" required="true" requiredMessage="*"/>
        <h:message for="txtName"/>

        <h:outputLabel value="Created date"/>
        <h:inputText value="#{productBean.product.createdDate}" id="txtDate" required="true" requiredMessage="*">
            <f:convertDateTime pattern="dd-MM-yyyy"/>
        </h:inputText>
        <h:message for="txtDate"/>

        <h:outputLabel value="Manufacturer"/>
        <h:selectOneMenu value="#{productBean.product.manu.manuID}">
            <f:selectItems value="#{productBean.listManu}" var="it" itemLabel="#{it.manuName}" itemValue="#{it.manuID}"/>
        </h:selectOneMenu>
        <h:message for="txtManu"/>

    </h:panelGrid>
```

Trong đó thì h:**selectOneMenu** để hiển thị **combobox** với list dữ liệu hiện thị tại thẻ **f:selectItems**, còn giá trị truyền vào combobox đó sẽ set bằng thuộc tính value tại thẻ **h:selectOneMenu**.

```
<h:outputLabel value="Manufacturer"/>
<h:selectOneMenu value="#{productBean.product.manu.manuID}">
    <f:selectItems value="#{productBean.listManu}" var="it" itemLabel="#{it.manuName}" itemValue="#{it.manuID}"/>
</h:selectOneMenu>
<h:message for="txtManu"/>
```

Thêm 2 nút **Update** và **Add new**:

```
<h:commandButton value="Update" action="#{productBean.updateProduct()}"/>
<h:commandButton value="Add new" action="#{productBean.addNew()}" style="margin-left: 15px"/>
```

Các Action tương ứng trong ProductBean như sau:

```
public String updateProduct(){
    dao.updateProduct(product);
    listProduct = dao.loadAllProducts();
    return "index";
}

public String addNew(){
    product = new Product();
    return "new";
}
```

Thêm nút **Go back**:

```
<h:form>
    <h:commandLink value="Cancel" action="#{productBean.goBack()}"/>
</h:form>
```

Trong ProductBean:

```
public String goBack(){
    listProduct = dao.loadAllProducts();
    return "index";
}
```

12. Copy trang **edit.xhtml**, đổi tên thành **new.xhtml**

Sửa lại các thẻ **outputText** thành **inputText**, các thẻ để nhập đều phải thêm thuộc tính **required="true"** để validate bắt nhập hết, **requiredMessage="..."** để hiện thị thông báo lỗi tại thẻ **h:message**.

Check nếu ProductId trùng thì báo lỗi bằng cách:

Thêm thẻ sau dưới thẻ đóng **panelGrid**, trong đó thì **errorClass** là đặt class CSS cho thẻ đó.

```
<h:messages errorClass="errors" globalOnly="true"/>
```

Khi người dùng nhấn nút **Add**:

```
<h:commandButton value="Add" action="#{productBean.add()}"/>
```

Thì trong ProductBean sẽ xử lý như sau:

```
public String add(){
    FacesContext fc = FacesContext.getCurrentInstance();
    if(dao.loadDetailProduct(product.getPID())!=null){
        fc.addMessage(null, new FacesMessage("Product ID bi trung"));
        return "new";
    } else{
        dao.addnewProduct(product);
        listProduct = dao.loadAllProducts();
        return "index";
    }
}
```

Kiểm tra nếu ProductID đã tồn tại thì hiện thì lỗi và giữ nguyên trạng thái trang. còn nếu ok thì thêm vào và redirect về trang **index**.

Done!