

Chapter 9 Composite Component

1. Introduction

➔ Composite components are those components that are implemented using composite library. The name composite components is because new components are composed using existing components.

➔ To use composite tag library add a namespace declaration to an XHTML file as shown below

```
<html xmlns="http://www.w3.org/1999/xhtml" ...  
xmlns:composite="http://java.sun.com/jsf/composite">  
...  
</html>
```

Above one can use any prefix for namespace apart from 'composite' but it is widely used prefix.

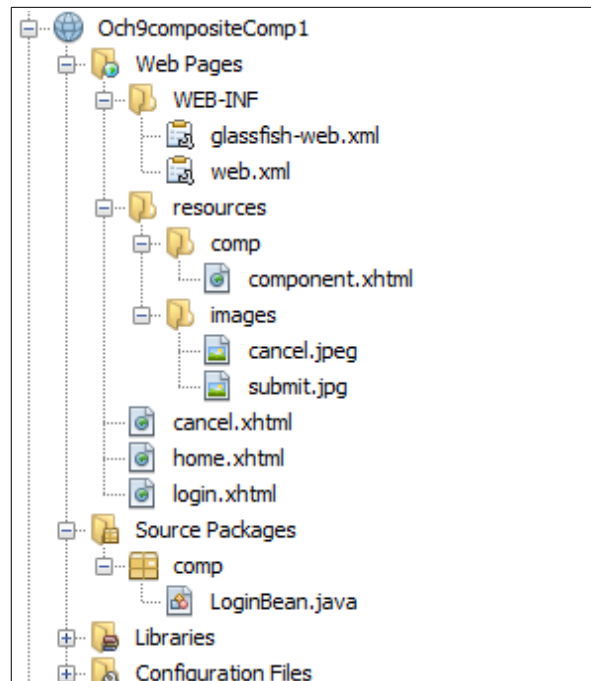
➔ Once the library is declared one can use any of the following tag belonging to composite library.

Composite Component Tags		
Tag	Description	Used In
interface	Contains other composite tags that expose a composite component's <i>attributes, action sources, value holders, editable value holders, and facets</i> .	N/A
implementation	Contains the XHTML markup that defines the component. Inside the implementation tag, the component author can access attributes with the expression <code>#{cc.attrs.attributeName}</code> .	N/A
attribute	Exposes an attribute of a component to page authors.	Interface
valueHolder	Exposes a component that holds a value to page authors.	Interface
editableValueHolder	Exposes a component that holds an editable value to page authors.	Interface
actionSource	Exposes a component that fires action events, such as a button or a link, to page authors.	Interface
facet	Declares that this component supports a facet with a given name.	Interface
extension	The component author can place this tag inside of any element in an interface. The extension tag can contain arbitrary XML.	Interface subelement
insertChildren	Inserts any child components specified by the page author.	Implementation
renderFacet	Renders a facet that was specified by the page author as a child component.	Implementation
insertFacet	Inserts a facet, specified by the page author, as a facet of the enclosing component.	Implementation

➔ Example

Here <composite:interface>, <composite:attribute> and <composite:implementation> is used.

Lets create a 'Och7compositeComp1' directory as shown below



web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="3.1" xmlns="http://xmlns.jcp.org/xml/ns/javaee"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd">
  <context-param>
    <param-name>javax.faces.PROJECT_STAGE</param-name>
    <param-value>Development</param-value>
  </context-param>
  <servlet>
    <servlet-name>Faces Servlet</servlet-name>
    <servlet-class>javax.faces.webapp.FacesServlet</servlet-class>
    <load-on-startup>1</load-on-startup>
  </servlet>
  <servlet-mapping>
    <servlet-name>Faces Servlet</servlet-name>
    <url-pattern>/faces/*</url-pattern>
  </servlet-mapping>
  <session-config>
    <session-timeout>
      30
    </session-timeout>
  </session-config>
  <welcome-file-list>
    <welcome-file>faces/login.xhtml</welcome-file>
  </welcome-file-list>
</web-app>
```

LoginBean.java

```
package comp;

import javax.inject.Named;
import javax.enterprise.context.Dependent;
import javax.faces.bean.ManagedBean;
import javax.faces.bean.SessionScoped;

@ManagedBean(name="loginbean")
@SessionScoped
public class LoginBean {

    public LoginBean() {
    }

    String fname, lname;
    int empid;

    public String getFname() {
        return fname;
    }

    public void setFname(String fname) {
        this.fname = fname;
    }

    public String getLname() {
        return lname;
    }

    public void setLname(String lname) {
        this.lname = lname;
    }

    public int getEmpid() {
        return empid;
    }

    public void setEmpid(int empid) {
        this.empid = empid;
    }

    public String action()
    {
        return "cancel";
    }

}
```

component.xhtml

```
<?xml version='1.0' encoding='UTF-8' ?>
<!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:h="http://xmlns.jcp.org/jsf/html"
      xmlns:composite="http://xmlns.jcp.org/jsf/composite">
    <h:head>
        <title>This content will not be displayed</title>
    </h:head>
    <h:body>
        <composite:interface>
            <composite:attribute name="fname" required="false"/>
        </composite:interface>
    </h:body>
</html>
```

```

        <composite:attribute name="lname" required="false"/>
        <composite:attribute name="empid" required="false"/>
    </composite:interface>
    <h:form>

        <composite:implementation>

            <h:panelGrid columns="2">

                <h:outputLabel value="First Name : "/>
                <h:inputText id="fname" value="#{cc.attrs.fname}"/>

                <h:outputLabel value="Last Name : "/>
                <h:inputText id="lname" value="#{cc.attrs.lname}"/>

                <h:outputLabel value="Emp ID : "/>
                <h:inputText id="empid" value="#{cc.attrs.empno}"/>

            </h:panelGrid>

        </composite:implementation>

    </h:form>
</h:body>
</html>

```

Note: Here we can see attributes have implemented using `cc.attrs`. Here 'cc' means composite component. Thus, `#{cc.attrs.attributeName}` accesses the `attributeName` attribute of the composite component.

login.xhtml

```

<?xml version='1.0' encoding='UTF-8' ?>
<!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:h="http://xmlns.jcp.org/jsf/html"
      xmlns:compnt="http://xmlns.jcp.org/jsf/composite/comp"
      xmlns:ui="http://xmlns.jcp.org/jsf/facelets">

    <h:head>
        <title>login page</title>
    </h:head>
    <h:body>
        <h:form>
            <compnt:component
                fname="#{loginbean.fname}"
                lname="#{loginbean.lname}"
                empno="#{loginbean.empid}"
            />

            <h:commandLink action="home">
                <h:graphicImage url="#{'resources/images/submit.jpg'}"/>
            </h:commandLink>

            <h:commandLink action="#{loginbean.action}">
                <h:graphicImage url="#{resource['/images/cancel.jpeg']}"/>
            </h:commandLink>

        </h:form>
    </h:body>
</html>

```

Note : Above we can see the component is accessed through `<compnt:component>` tag. Also note `<h:graphicImage>` tag's 'url' attribute can be used in two ways.

home.xhtml

```
<?xml version='1.0' encoding='UTF-8' ?>
<!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:h="http://xmlns.jcp.org/jsf/html">
  <h:head>
    <title>Final Page</title>
  </h:head>
  <h:body>
    Hello #{loginbean.fname} #{loginbean.lname}!!!<br/>
    Emp No : #{loginbean.empid}
  </h:body>
</html>
```

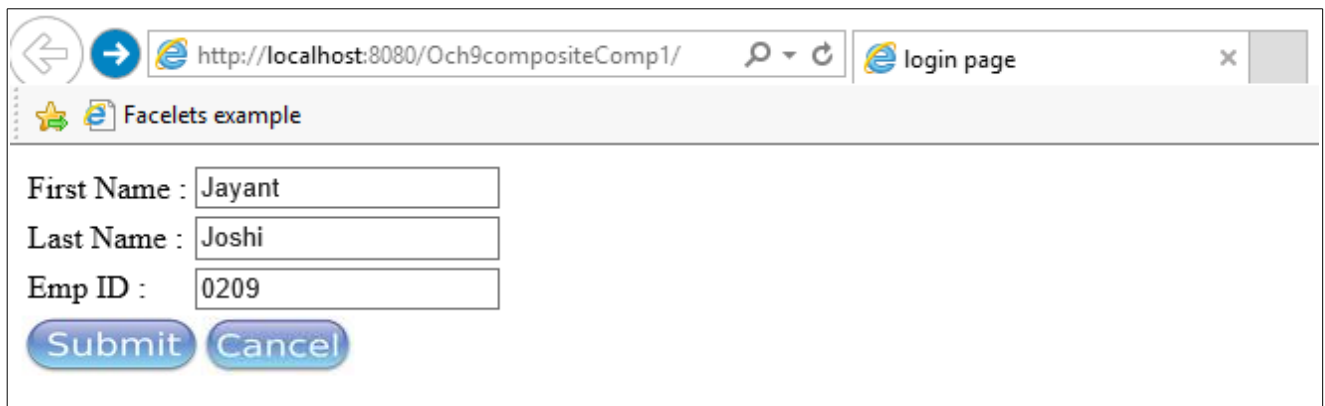
cancel.xhtml

```
<?xml version='1.0' encoding='UTF-8' ?>
<!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:h="http://xmlns.jcp.org/jsf/html">
  <h:head>
    <title>Cancel Page</title>
  </h:head>
  <h:body>
    Transaction has been canceled
  </h:body>
</html>
```

Output

1.

login.xhtml



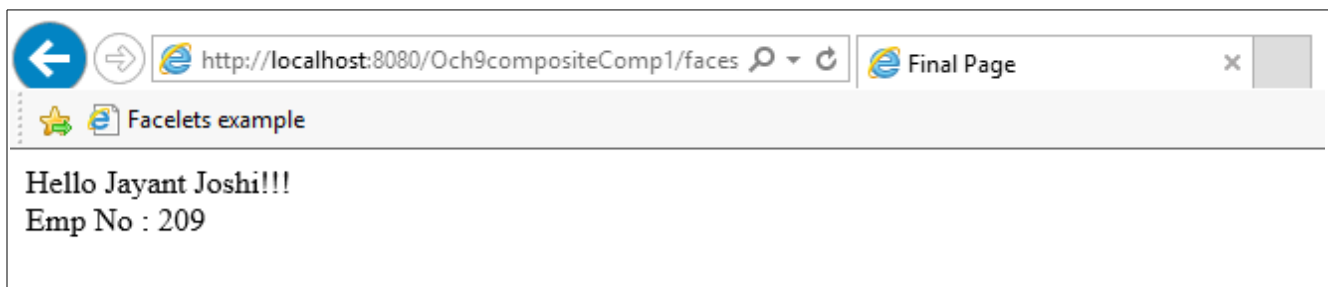
First Name : Jayant

Last Name : Joshi

Emp ID : 0209

Submit Cancel

finalPage.xhtml



Hello Jayant Joshi!!!

Emp No : 209

2.

login.xhtml

First Name :

Last Name :

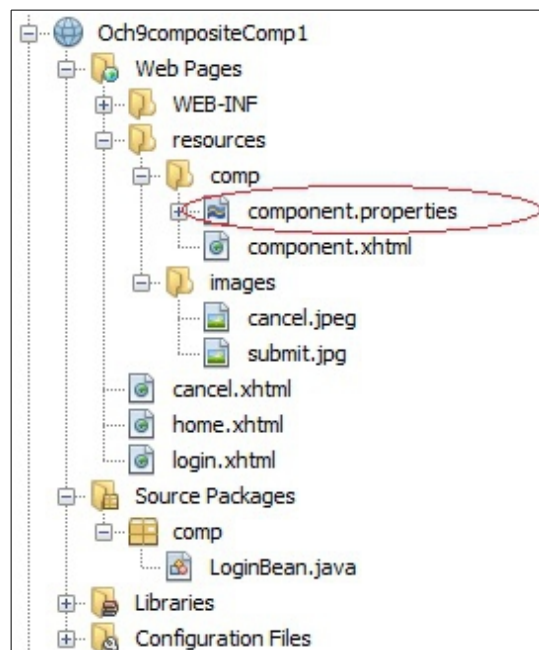
Emp ID :

cancel.xhtml

Transaction has been canceled

2. Localizing Composite Components

- ➔ Sometimes one may want to localize the component part for example to sell advertising space in one's component.
- ➔ This is achieved by creating a resource bundle with the same name as component and saved in the component folder. This bundle will be associated only with the particular component.
- ➔ This bundle will only contains key/value pairs. In previous example we had component file with name component.xhtml. Hence we should have bundle component.properties as shown below



➔ To access it we should use `#{cc.resourceBundle.key}`. In our case key is 'footer' hence we have

```
#{cc.resourceBundleMap.footer}
```

➔ Let's see if we insert it in `component.xhtml` file as shown below

component.xhtml

```
<?xml version='1.0' encoding='UTF-8' ?>
<!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:h="http://xmlns.jcp.org/jsf/html"
      xmlns:composite="http://xmlns.jcp.org/jsf/composite">
  <h:head>
    <title>This content will not be displayed</title>
  </h:head>
  <h:body>
    <composite:interface>
      <composite:attribute name="fname" required="false"/>
      <composite:attribute name="lname" required="false"/>
      <composite:attribute name="empid" required="false"/>
    </composite:interface>
    <h:form>

      <composite:implementation>

        <h:panelGrid columns="2">

          <h:outputLabel value="First Name : "/>
          <h:inputText id="fname" value="#{cc.attrs.fname}"/>

          <h:outputLabel value="Last Name : "/>
          <h:inputText id="lname" value="#{cc.attrs.lname}"/>

          <h:outputLabel value="Emp ID : "/>
          <h:inputText id="empid" value="#{cc.attrs.empid}"/>

          <h:outputText value="#{cc.resourceBundleMap.footer}"/>

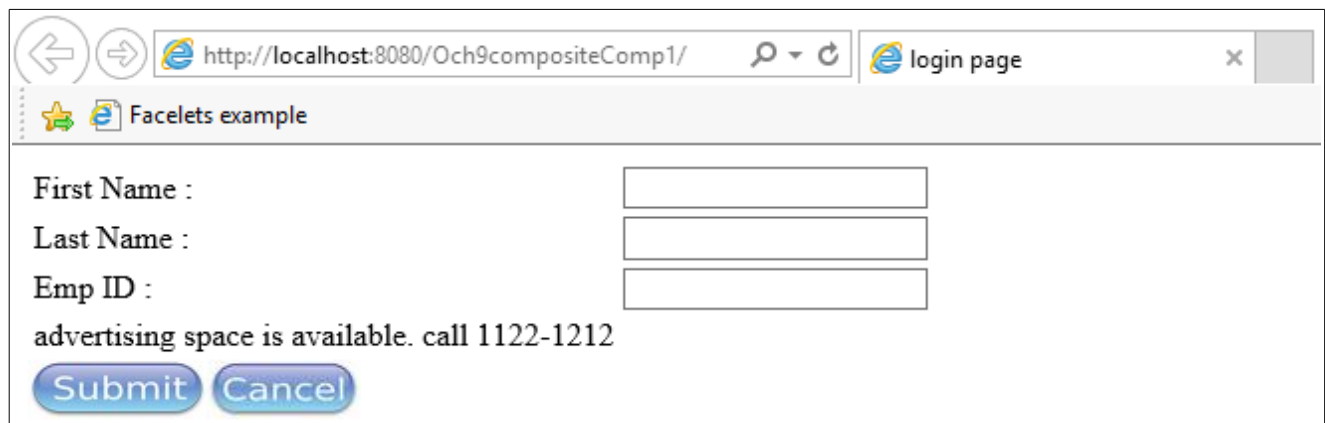
        </h:panelGrid>

      </composite:implementation>

    </h:form>
  </h:body>
</html>
```

Output

login.xhtml



First Name :

Last Name :

Emp ID :

advertising space is available. call 1122-1212

3. Adding validators to components

➔ In the previous example we have supplied inputs for composite component as

```
<compnt:component
    fname="#{loginbean.fname}"
    lname="#{loginbean.lname}"
    empno="#{loginbean.empid}"
/>
```

with this there will be one disadvantage, we cannot supply validators to the components here.

➔ To add validators we should use `<composite:editableValueHolder>` tag as shown below

```
<composite:interface>
. . . . .
<composite:editableValueHolder name="fname" />
<composite:editableValueHolder name="lname" />
<composite:editableValueHolder name="empid" />

</composite:interface>
```

No need to add anything to the `<compose:implementation>` part.

Later the view where component is used validators are added as

```
<compnt:component
    fname="#{loginbean.fname}"
    lname="#{loginbean.lname}"
    empno="#{loginbean.empid}"

<f:validateLength minimum="4" maximum="10" for="fname"/>
<f:validateLength minimum="4" maximum="10" for="lname"/>
<f:validator validatorId="com.validator.empIdValidator" for="empid"/>
</compnt:component>
```

➔ Above we can see validators for 'fname' and 'lname' specify minimum length of 4 and maximum length of 10. Validator for 'empid' is a custom validator with validatorId 'com.validator.empIdValidator'.

➔ Lets check the project 'Och9CompositeComp1' after adding validators.

component.xhtml

```
<?xml version='1.0' encoding='UTF-8' ?>
<!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:h="http://xmlns.jcp.org/jsf/html"
    xmlns:composite="http://xmlns.jcp.org/jsf/composite">
    <h:head>
        <title>This content will not be displayed</title>
    </h:head>
    <h:body>
        <composite:interface>
            <composite:attribute name="fname" required="false"/>
            <composite:attribute name="lname" required="false"/>
            <composite:attribute name="empid" required="false"/>

            <composite:editableValueHolder name="fname" />
            <composite:editableValueHolder name="lname" />
            <composite:editableValueHolder name="empid" />
        </composite:interface>
        <h:form>

            <composite:implementation>
```



```

        <h:panelGrid columns="2">

        <h:outputLabel value="First Name : "/>
        <h:inputText id="fname" value="#{cc.attrs.fname}"/>

        <h:outputLabel value="Last Name : "/>
        <h:inputText id="lname" value="#{cc.attrs.lname}"/>

        <h:outputLabel value="Emp ID : "/>
        <h:inputText id="empid" value="#{cc.attrs.empid}"/>

        <h:outputText value="#{cc.resourceBundleMap.footer}"/>

        </h:panelGrid>

    </composite:implementation>

</h:form>
</h:body>
</html>

```

Note: Above value used in 'name=fname' attribute of <composite:editableValueHolder> must match 'id=fname' of related tag inside <composite:implementation>.

Example <h:inputText id="fname" value="#{cc.attrs.fname}"/>.

login.xhtml

```

<?xml version='1.0' encoding='UTF-8' ?>
<!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:h="http://xmlns.jcp.org/jsf/html"
      xmlns:compnt="http://xmlns.jcp.org/jsf/composite/comp"
      xmlns:ui="http://xmlns.jcp.org/jsf/facelets">

    <h:head>
        <title>login page</title>
    </h:head>
    <h:body>
        <h:form>
            <compnt:component
                fname="#{loginbean.fname}"
                lname="#{loginbean.lname}"
                empno="#{loginbean.empid}"

                <f:validateLength minimum="4" maximum="10" for="fname"/>
                <f:validateLength minimum="4" maximum="10" for="lname"/>
                <f:validator validatorId="com.validator.empIdValidator" for="empid"/>
            </compnt:component>

            <h:commandLink action="home">
                <h:graphicImage url="#{'resources/images/submit.jpg'}"/>
            </h:commandLink>

            <h:commandLink action="#{loginbean.action}">
                <h:graphicImage url="#{resource['/images/cancel.jpeg']}"/>
            </h:commandLink>

        </h:form>
    </h:body>
</html>

```

empIdValidator.java

```
package comp;

import javax.faces.application.FacesMessage;
import javax.faces.component.UIComponent;
import javax.faces.context.FacesContext;
import javax.faces.validator.FacesValidator;
import javax.faces.validator.Validator;
import javax.faces.validator.ValidatorException;

@FacesValidator("com.validator.empIdValidator")
public class empIdValidator implements Validator {

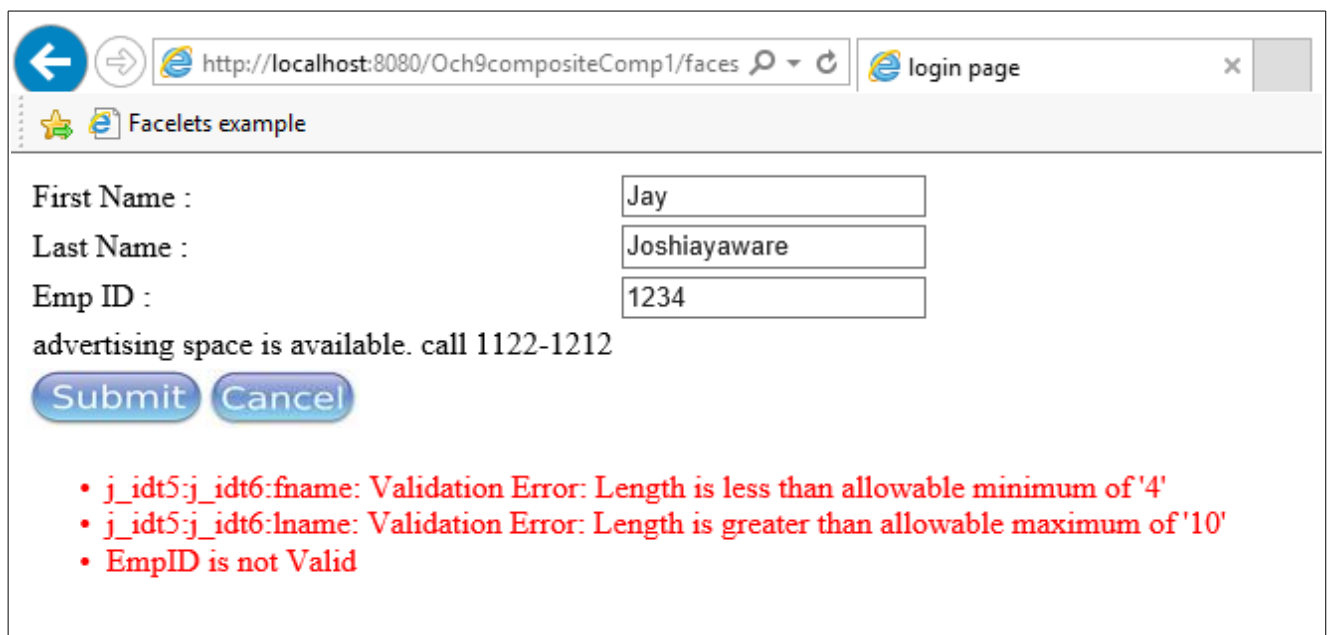
    @Override
    public void validate(FacesContext context,
        UIComponent component, Object value) throws ValidatorException {

        String [] empIds={"1111","2222","3333","4444"};
        String empId = value.toString();
        boolean match=false;
        for(int i=0; i<empIds.length; i++)
        {
            if((empId.equals(empIds[i])))
            {
                match=true;
            }
        }
        if(!match)
        {
            FacesMessage message = new FacesMessage("EmpID is not Valid");
            message.setSeverity(FacesMessage.SEVERITY_ERROR);
            throw new ValidatorException(message);
        }
    }
}
```

Output

1. Invalid Inputs

login.xhtml



The screenshot shows a web browser window with the address bar displaying `http://localhost:8080/Och9compositeComp1/faces` and the page title "login page". The browser's address bar also shows "Facelets example". The login form contains three input fields: "First Name" with the value "Jay", "Last Name" with the value "Joshiayaware", and "Emp ID" with the value "1234". Below the input fields, there is a message: "advertising space is available. call 1122-1212". At the bottom of the form, there are two buttons: "Submit" and "Cancel". Below the buttons, there are three red error messages:

- `j_idt5:j_idt6:fname: Validation Error: Length is less than allowable minimum of '4'`
- `j_idt5:j_idt6:lname: Validation Error: Length is greater than allowable maximum of '10'`
- `EmpID is not Valid`

2. Valid Inputs

login.xhtml

First Name :

Last Name :

Emp ID :

advertising space is available. call 1122-1212

home.xhtml

Hello Jayant Joshi!!!

Emp No : 3333

➔ Another way to add validator is shown below with changes in red.

Component.xhtml

```
<?xml version='1.0' encoding='UTF-8' ?>
<!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:h="http://xmlns.jcp.org/jsf/html"
xmlns:composite="http://xmlns.jcp.org/jsf/composite">
  <h:head>
    <title>This content will not be displayed</title>
  </h:head>
  <h:body>
    <composite:interface>
      <composite:attribute name="fname" required="false"/>
      <composite:attribute name="lname" required="false"/>
      <composite:attribute name="empid" required="false"/>

      <composite:editableValueHolder name="form:fname" />
      <composite:editableValueHolder name="form:lname" />
      <composite:editableValueHolder name="form:empid" />
    </composite:interface>
    <h:form id=form>
```

```

        <composite:implementation>

        <h:panelGrid columns="2">

        <h:outputLabel value="First Name : "/>
        <h:inputText id="fname" value="#{cc.attrs.fname}"/>

        <h:outputLabel value="Last Name : "/>
        <h:inputText id="lname" value="#{cc.attrs.lname}"/>

        <h:outputLabel value="Emp ID : "/>
        <h:inputText id="empid" value="#{cc.attrs.empid}"/>

        <h:outputText value="#{cc.resourceBundleMap.footer}"/>

        </h:panelGrid>

        </composite:implementation>

    </h:form>
</h:body>
</html>

```

Note: Above we can see `<h:form id="form">` has been given an id as 'form' and it becomes prefix for 'fname' and 'lname' since both of these inputs reside in 'form'. Rest login.xhtml page remains the same.

4. Exposing Action Sources

➔ We know three tags one can use inside `<composite:interface>` tags. They are

1. `<composite:actionSource>`
2. `<composite:valueHolder>`
3. `<composite:editableValueHolder>`

We have already used `<composite:editableValueSource>`. Here we are going to discuss about `<composite:actionSource>` and `<composite:valueHolder>`.

4.1. <composite:actionSource>

➔ If one want to use action Listener then he should use `<composite:actionSource>` in component file. It will be use same way `<composite:editableValueHolder>` have been used in previous example.

➔ Example

```

<composite:interface>
    . . . . .
    <composite:attribute name="submitBtnAction" method-signature="java.lang.String
                        action()"/>
    <composite:actionSource name="submitButton" targets="form:submitButton"/>
    . . . . .
</composite:interface>

and inside <composite:implementation> we have

<composite:implementation>
    <h:form id="form">
        . . . . .
        <h:commandButton id="submitButton" value="Submit"
                        action="#{cc.attrs.submitBtnAction}"/>
        </h:panelGrid>
    </h:form>
</composite:implementation>

```

Note: Above we can see that inside `<composite:interface>` we have `<composite:actionSource>` tag with attribute `target="form:submitButton"`. This is because it reside in `<h:form id="form">` tag. Also note that `name="submitButton"` attribute inside `<composite:actionSource>` tag must match with `id="submitButton"` attribute of `<h:commonButton>` tag inside `<composite:implementation>`.

4.2. <composite:valueHolder>

➔ The tag `<composite:editableValueHolder>` could be used for value Change Listeners, converter and validators(In previous example we have used it only for validators). But if one want to use only validators then we can use `<composite:valueHolder>`.

➔ Example

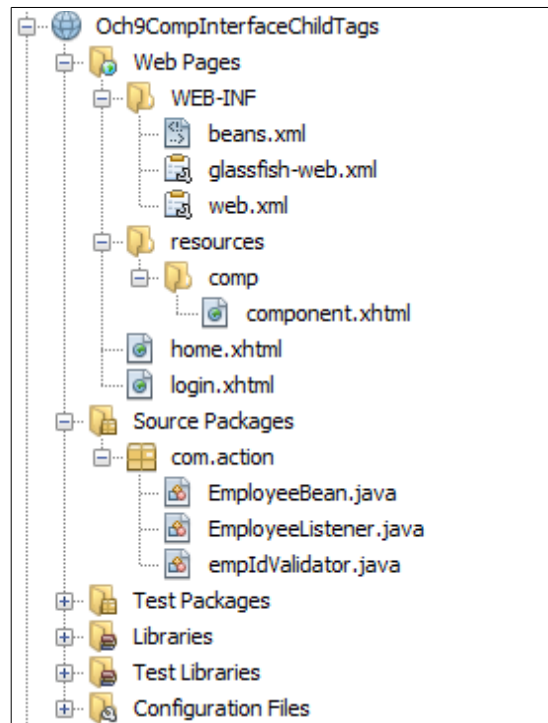
```
<composite:interface>
    . . . . .
    <composite:attribute name="empIdLabel"/>
    <composite:attribute name="empId"/>
    <composite:valueHolder name="empId" targets="form:empId"/>
    . . . . .
</composite:interface>

and inside <composite:implementation>

<composite:implementation>
    <h:form id="form">
        <h:panelGrid columns="2">
            . . . . .
            <h:inputText id="empId" value="#{cc.attrs.empId}"/>
            . . . . .
        </h:panelGrid>
    </h:form>
</composite:implementation>
```

Note: Here too attribute `name="empId"` inside `<compsite:valueHolder>` must match attribute `id="empId"` of `<h:inputText>` tag. Also attribute `target="form:empId"` has been used because it reside inside `<form id="form">` tag.

➔ Lets create directory for project 'Och9CompInterfaceChildTags' as shown below



EmployeeBean.java

```
package com.action;

import javax.inject.Named;
import javax.enterprise.context.SessionScoped;
import java.io.Serializable;
import javax.faces.event.AbortProcessingException;
import javax.faces.event.ActionEvent;

@Named(value = "employee")
@SessionScoped
public class EmployeeBean implements Serializable {

    public EmployeeBean() {
    }

    String name, empId;

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getEmpId() {
        return empId;
    }

    public void setEmpId(String empId) {
        this.empId = empId;
    }
}
```

```

    public String checkEmpId()
    {
        System.out.println("Inside bean method");
        return "home";
    }
}

```

EmployeeBean.java

```

package com.action;

import javax.inject.Named;
import javax.enterprise.context.SessionScoped;
import java.io.Serializable;
import javax.faces.event.AbortProcessingException;
import javax.faces.event.ActionEvent;

@Named(value = "employee")
@SessionScoped
public class EmployeeBean implements Serializable {

    public EmployeeBean() {
    }

    String name, empId;

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getEmpId() {
        return empId;
    }

    public void setEmpId(String empId) {
        this.empId = empId;
    }

    public String checkEmpId()
    {
        System.out.println("Inside bean method");
        return "home";
    }
}

```

EmployeeListener.java

```

package com.action;

import javax.faces.event.AbortProcessingException;
import javax.faces.event.ActionEvent;
import javax.faces.event.ActionListener;
public class EmployeeListener implements ActionListener {
    @Override
    public void processAction(ActionEvent event)
        throws AbortProcessingException {
        System.out.println("Inside listener");
    }
}

```

```

        System.out.println("Form Id by ActionListener
class:"+event.getComponent().getParent().getId());
    }
}

```

component.xhtml

```

<?xml version='1.0' encoding='UTF-8' ?>

<!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:h="http://xmlns.jcp.org/jsf/html"
      xmlns:composite="http://java.sun.com/jsf/composite">

    <composite:interface>
        <composite:attribute name="nameLable"/>
        <composite:attribute name="name"/>
        <composite:attribute name="empIdLable"/>
        <composite:attribute name="empId"/>
        <composite:valueHolder name="empId" targets="form:empId"/>
        <composite:attribute name="submitBtnAction" method-signature="java.lang.String
action()"/>
        <composite:actionSource name="submitButton" targets="form:submitButton"/>
    </composite:interface>

    <h:body>
    <composite:implementation>
        <h:form id="form">
            <h:panelGrid columns="2">
                <h:outputLabel value="#{cc.attrs.nameLable}"/>
                <h:inputText id="name" value="#{cc.attrs.name}"/>
                <h:outputLabel value="#{cc.attrs.empIdLable}"/>
                <h:inputText id="empId" value="#{cc.attrs.empId}"/>
                <h:commandButton id="submitButton" value="Submit"
                    action="#{cc.attrs.submitBtnAction}"/>
            </h:panelGrid>
        </h:form>
    </composite:implementation>

    </h:body>

</html>

```

login.xhtml

```

<?xml version='1.0' encoding='UTF-8' ?>

<!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:h="http://xmlns.jcp.org/jsf/html"
      xmlns:comp="http://java.sun.com/jsf/composite/comp"
      xmlns:f="http://xmlns.jcp.org/jsf/core">
    <h:head>
        <title>Login Page</title>
    </h:head>
    <h:body>
        <h1>Login Page</h1>
        <h:form>
            <comp:component
                nameLable="Name :"
                name="#{employee.name}"
                empIdLable="EmpID :"/>

```



```

        empId="#{employee.empId}"
        submitBtnAction="#{employee.checkEmpId}"
        <f:validator for="empId" validatorId="com.validator.empIdValidator"/>
        <f:actionListener for="submitButton" type="com.action.EmployeeListener"/>
    </comp:component>
</h:form>
</h:body>
</html>

```

home.xhtml

```

<?xml version='1.0' encoding='UTF-8' ?>

<!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:h="http://xmlns.jcp.org/jsf/html">
    <h:head>
        <title>Home Page</title>
    </h:head>
    <h:body>
        Hello #{employee.name}
    </h:body>
</html>

```

web.xml

```

<?xml version="1.0" encoding="UTF-8"?>


<web-app version="3.1" xmlns="http://xmlns.jcp.org/xml/ns/javaee"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd">
    <context-param>
        <param-name>javax.faces.PROJECT_STAGE</param-name>
        <param-value>Development</param-value>
    </context-param>
    <servlet>
        <servlet-name>Faces Servlet</servlet-name>
        <servlet-class>javax.faces.webapp.FacesServlet</servlet-class>
        <load-on-startup>1</load-on-startup>
    </servlet>
    <servlet-mapping>
        <servlet-name>Faces Servlet</servlet-name>
        <url-pattern>/faces/*</url-pattern>
    </servlet-mapping>
    <session-config>
        <session-timeout>
            30
        </session-timeout>
    </session-config>
    <welcome-file-list>
        <welcome-file>faces/login.xhtml</welcome-file>
    </welcome-file-list>
</web-app>

```

Output

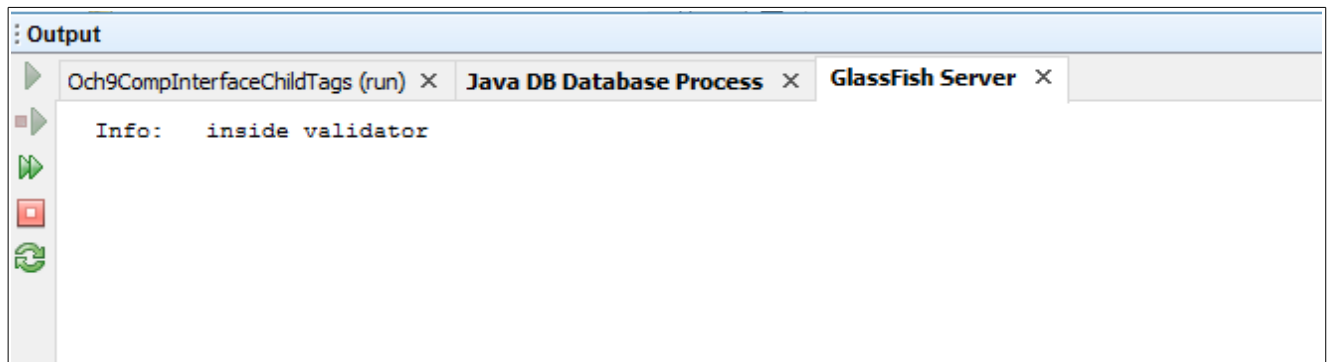
1. When 'employee ID' is not validate

login.xhtml



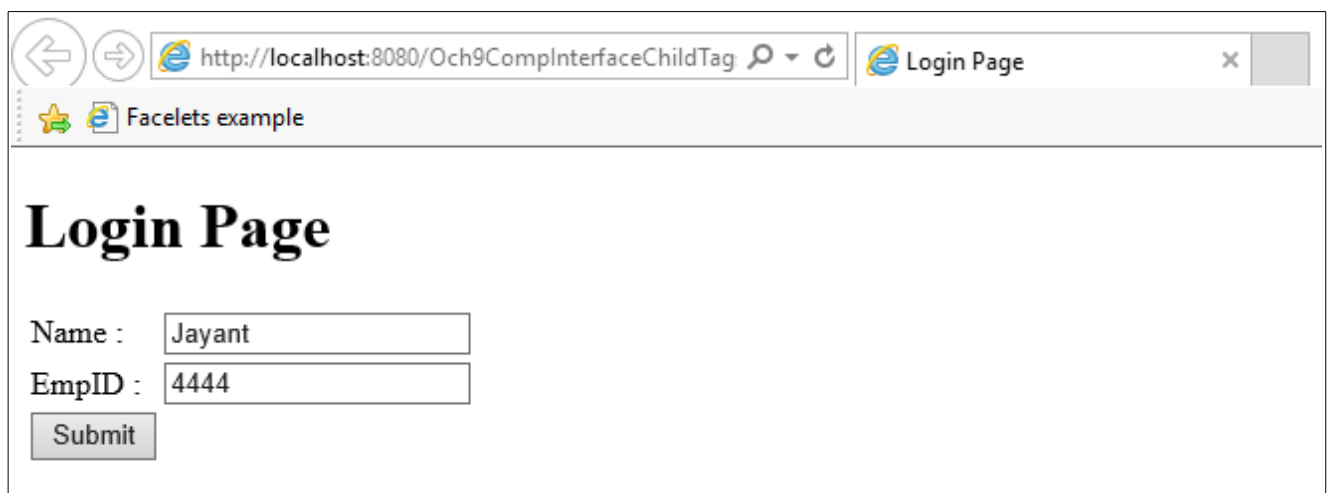
A screenshot of a web browser window. The address bar shows the URL `http://localhost:8080/Och9CompInterfaceChildTag`. The page title is "Login Page". The page content includes a heading "Login Page", two input fields labeled "Name :" and "EmpID :", and a "Submit" button. The "Name" field contains the text "Jayant" and the "EmpID" field contains "1234". Below the input fields, a red error message is displayed: "• EmpID is not Valid".

Server Console



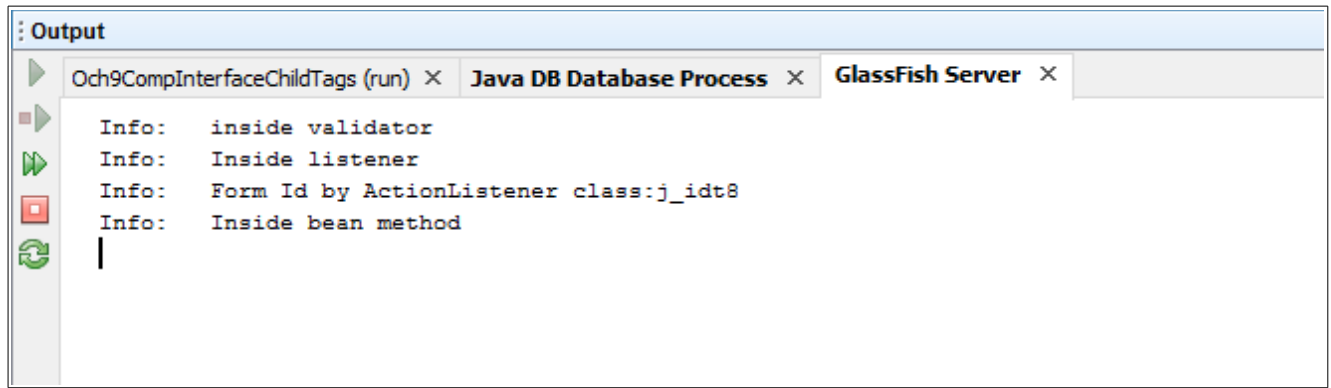
2. Valid Inputs

login.xhtml



A screenshot of a web browser window, similar to the one above. The address bar shows the URL `http://localhost:8080/Och9CompInterfaceChildTag`. The page title is "Login Page". The page content includes a heading "Login Page", two input fields labeled "Name :" and "EmpID :", and a "Submit" button. The "Name" field contains the text "Jayant" and the "EmpID" field contains "4444". There is no error message displayed.

Server Console



home.xhtml



5. Facets

➔ Adding converters, validators and listeners inside composite component is one way of adding functionality to it. Another way is through facets.

➔ Facet is used to encapsulate the set of elements specifying the details for the facet.

➔ Example

```
<composite:interface>
...
<composite:facet name="header"/>
<composite:facet name="error"/>
</composite:interface>
<composite:implementation>
...
<composite:renderFacet name="header"/>
<h:form ...>
...
</h:form>
<composite:renderFacet name="error"/>
</composite:implementation>
```

Later inside the view where composite component has been used facets are used as

```
<comp:component
...
    <f:facet name="heading" class="header">
        <h:outputText value="Login Page"/><br/><br/>
    </f:facet>

    <f:facet name="error" class="error">
        <h:messages layout="table" style="color: blue"/>
    </f:facet>
```

```

        <h:link outcome="register">Register..</h:link>
    </comp:component>

```

Above when link is clicked `outcome="register"` mentioned take control to JSF page `register.xhtml`.

6. Children <composite:insertChildren>

➔ This component is used inside `<composite:implementation>` section.

➔ They are used to have component tags placed at a specific point inside composite component.

➔ Example

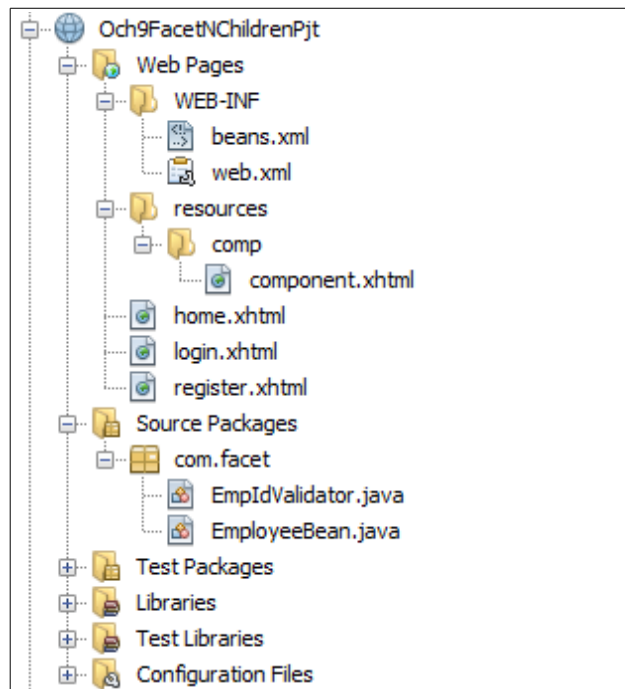
```

<cc:implementation>
    ...
<cc:insertChildren />
    ...
</cc:implementation>

```

➔ Example

Lets create project directory 'Och9FacetNChildrenPjt' as shown below



EmployeeBean.java

```

package com.facet;

import java.io.Serializable;
import javax.inject.Named;
import javax.enterprise.context.SessionScoped;

@Named(value = "employee")
@SessionScoped
public class EmployeeBean implements Serializable {

    public EmployeeBean() {

```

```

}

String name, empId;

public String getName() {
    return name;
}

public void setName(String name) {
    this.name = name;
}

public String getEmpId() {
    return empId;
}

public void setEmpId(String empId) {
    this.empId = empId;
}

public String toPage()
{
    return "home";
}
}

```

EmpIdValidator

```

package com.facet;

import javax.faces.application.FacesMessage;
import javax.faces.component.UIComponent;
import javax.faces.context.FacesContext;
import javax.faces.validator.FacesValidator;
import javax.faces.validator.Validator;
import javax.faces.validator.ValidatorException;

@FacesValidator("com.validator.empIdValidator")
public class EmpIdValidator implements Validator {

    @Override
    public void validate(FacesContext context,
        UIComponent component, Object value) throws ValidatorException {

        System.out.println("inside validator");
        String [] empIds={"1111", "2222", "3333", "4444"};
        String empId = value.toString();
        boolean match=false;
        for(int i=0; i<empIds.length; i++)
        {
            if((empId.equals(empIds[i])))
            {
                match=true;
            }
        }
        if(!match)
        {
            FacesMessage message = new FacesMessage("This EmpID is not Valid");
            message.setSeverity(FacesMessage.SEVERITY_ERROR);
            throw new ValidatorException(message);
        }
    }
}

```

component.xhtml

```
<?xml version='1.0' encoding='UTF-8' ?>
<!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:h="http://xmlns.jcp.org/jsf/html"
      xmlns:composite="http://java.sun.com/jsf/composite"
      xmlns:f="http://xmlns.jcp.org/jsf/core">
  <h:head>
    <title>Component</title>
  </h:head>
  <composite:interface>
    <composite:attribute name="nameLabel"/>
    <composite:attribute name="name"/>
    <composite:attribute name="empIdLabel"/>
    <composite:attribute name="empId"/>
    <composite:valueHolder name="empId" targets="form:empId"/>
    <composite:attribute name="submitButton"/>
    <composite:attribute name="submitAction" method-signature="java.lang.String
action()"/>
    <f:facet name="heading"/>
    <f:facet name="error"/>
  </composite:interface>
  <h:body>
    <composite:implementation>
      <composite:renderFacet name="heading"/><br/>
      <h:form id="form">
        <h:panelGrid columns="2">
          <h:outputLabel value="#{cc.attrs.nameLabel}"/>
          <h:inputText id="name" value="#{cc.attrs.name}"/>
          <h:outputLabel id="empIdLabel" value="#{cc.attrs.empIdLabel}"/>
          <h:inputText id="empId" value="#{cc.attrs.empId}"/>
          <h:commandButton id="submitButton" value="#{cc.attrs.submitButton}"
action="#{cc.attrs.submitAction}"/>
        </h:panelGrid>
      </h:form>
      <composite:renderFacet name="error"/>
      <composite:insertChildren/>
    </composite:implementation>
  </h:body>
</html>
```

login.xhtml

```
<?xml version='1.0' encoding='UTF-8' ?>
<!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:h="http://xmlns.jcp.org/jsf/html"
      xmlns:comp="http://java.sun.com/jsf/composite/comp"
      xmlns:f="http://xmlns.jcp.org/jsf/core"
      xmlns:ui="http://xmlns.jcp.org/jsf/facelets">
  <h:head>
    <title>Login Page</title>
  </h:head>
  <h:body>
    <h:form>
      <comp:component
        nameLabel="Name : "
        name="${employee.name}"
        empIdLabel="EmpID : "
```

```

        empId="${employee.empId}"
        submitButton="Submit"
        submitAction="#{employee.toPage}">

<f:validator for="empId" validatorId="com.validator.empIdValidator"/>

<f:facet name="heading" class="header">
    <h:outputText value="Login Page"/><br/><br/>
</f:facet>

<f:facet name="error" class="error">
    <h:messages layout="table" style="color: blue"/>
</f:facet>

<h:link outcome="register">Register..</h:link>
</comp:component>

</h:form>
</h:body>
</html>

```

home.xhtml

```

<?xml version='1.0' encoding='UTF-8' ?>
<!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:h="http://xmlns.jcp.org/jsf/html">
    <h:head>
        <title>Home Page</title>
    </h:head>
    <h:body>
        Hello #{employee.name}
    </h:body>
</html>

```

register.xhtml

```

<?xml version='1.0' encoding='UTF-8' ?>
<!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:h="http://xmlns.jcp.org/jsf/html">
    <h:head>
        <title>Register Page</title>
    </h:head>
    <h:body>
        This is Register page
    </h:body>
</html>

```

web.xml

```

<?xml version="1.0" encoding="UTF-8">
<web-app version="3.1" xmlns="http://xmlns.jcp.org/xml/ns/javaee"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd">
    <context-param>
        <param-name>javax.faces.PROJECT_STAGE</param-name>
        <param-value>Development</param-value>
    </context-param>

```

```


<servlet>
  <servlet-name>Faces Servlet</servlet-name>
  <servlet-class>javax.faces.webapp.FacesServlet</servlet-class>
  <load-on-startup>1</load-on-startup>
</servlet>
<servlet-mapping>
  <servlet-name>Faces Servlet</servlet-name>
  <url-pattern>/faces/*</url-pattern>
</servlet-mapping>
<session-config>
  <session-timeout>
    30
  </session-timeout>
</session-config>
<welcome-file-list>
  <welcome-file>faces/login.xhtml</welcome-file>
</welcome-file-list>
</web-app>

```

Output

1. When inputs are not valid

login.xhtml



A screenshot of a web browser window. The address bar shows the URL: `http://localhost:8080/Och9FacetNChildrenPjt/faces`. The page title is "Login Page". The browser's bookmark bar shows "Facelets example". The page content includes a "Login Page" heading, two input fields: "Name : Jayant" and "EmpID : 1234", a "Submit" button, and a blue error message: "This EmpID is not Valid". Below the error message is a blue link: "Register..".

2. Valid Inputs

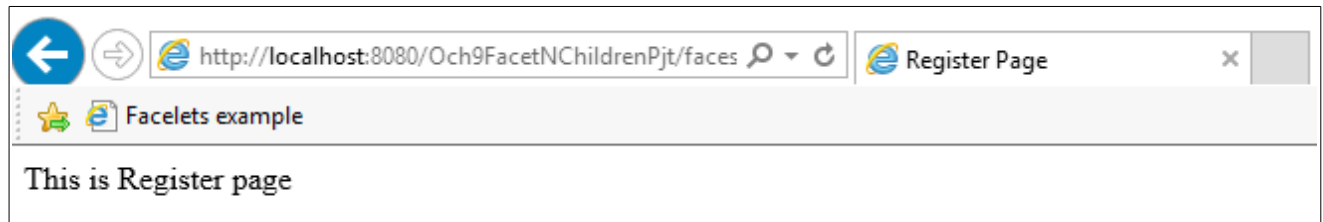
login.xhtml



A screenshot of a web browser window, similar to the one above. The address bar shows the same URL: `http://localhost:8080/Och9FacetNChildrenPjt/`. The page title is "Login Page". The browser's bookmark bar shows "Facelets example". The page content includes a "Login Page" heading, two input fields: "Name : Jayant" and "EmpID : 3333", a "Submit" button, and a blue link: "Register..".

3. Regsiter Link

register.xhtml



7. Backing Components

➔ sdf

➔ sdf