Chapter 3 Navigation

1. Introduction

➔ Navigation Handler is reponsible for selecting the next JSF page.

2. Types of Navigation

There are two types of Navigation

I. Static Navigation

➔ In a simple web application navigation is static. That is, clicking a particular button always selects a fixed JSF page for response.

➔ Static navigation is achieved through '`action`' attribute provided to each button or link.

`<h:commandButton value="Login" action="welcome"/>`

➔ The value of action is called 'outcome'. This outcome will be mapped to view-IDs. If they are not mapped then outcome are transformed into view-IDs using following steps

a. If outcome don't have any extension then extension of current view will be appended.

b. If outcome doesn't start with '/' then path of the current view will be prepended.

Ex. In /index.xhtml, the outcome is 'welcome' and it yields the target view-ID '/welcome.xhtml'.

Note: Since JSF 2.0 mapping of the view-ID is optional. Prior to JSF 2.0 one had to specify explicit navigation rules for every outcome.

ii. Dynamic Navigation

➔ To implement dynamic navigation, the '`action`' attribute must have a method expression.

Ex:

`<h:commandButton value="Login" action=#{loginController.verifyUser}/>`

Here `LoginController` references a bean class and it must have a method 'verifyUser'.

➔ Here is a example of action method

```
String verifyUser()
{
if(...)
return "success";
else
return "failure";
}
```

Depends on outcome success or failure next view will be determined.

Note: Action method may return null to indicate current view should be redisplayed.

➔ When '`action`' is a method expression then following steps are carried out

i. The specified bean is retrieved.

ii. The reference method is called and a String is returned.

iii. The outcome string is transformed into view-ID.

iv. The page corresponding to view-ID is displayed.

3. Mapping outcomes to view-IDs

➔ JSF provides mechanism for mapping logical outcome of methods to actual web pages.

➔ It is achieved by adding `navigation-rule` entries into `faces-config.xml`.

Ex

```
<navigation-rule>
    <from-view-id>/index.xhtml</from-view-id>
        <navigation-case>
            <from-outcome>success</from-outcome>
            <to-view-id>/welcome.xhtml</to-view-id>
        </navigation-case>
</navigation-rule>
```

➔ If one has action method 'logout' the application pages(i.e. many pages have logout button) then one can have all these buttons navigate to the loggedOut.xhtml page as shown below

```
<navigation-rule>
    <navigation-case>
    <from-outcome>logout</from-outcome>
    <to-view-id>/loggedOut.xhtml</to-view-id>
    </navigation-case>
</navigation-rule>
```
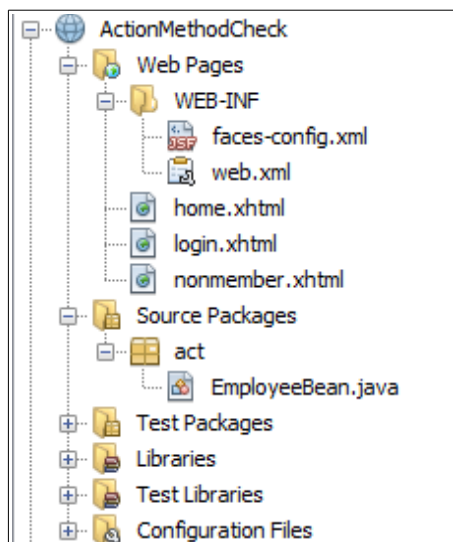
The above rule applies to all the pages because no `<from-view-id>` tag is present.

➔ One can merge navigation rules with same `<from-vieww-id>` as shown below

```
<navigation-rule>
    <from-view-id>/index.xhtml</from-view-id>
        <navigation-case>
            <from-outcome>success</from-outcome>
            <to-view-id>/welcome.xhtml</to-view-id>
        </navigation-case>
        <navigation-case>
            <from-outcome>failure</from-outcome>
            <to-view-id>/newUser.xhtml</to-view-id>
        </navigation-case>
</navigation-rule>
```

➔ Ex

Lets create ActionMethodCheck directory as shown below

web.xml

```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>
```

faces-config.xml

```xml
<?xml version="1.0"?>
<faces-config xmlns="http://java.sun.com/xml/ns/javaee"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
      http://java.sun.com/xml/ns/javaee/web-facesconfig_2_0.xsd"
   version="2.0">
   <navigation-rule>
      <navigation-case>
         <from-outcome>response1</from-outcome>
         <to-view-id>/home.xhtml</to-view-id>
      </navigation-case>
   </navigation-rule>
    <navigation-rule>
      <navigation-case>
         <from-outcome>response2</from-outcome>
         <to-view-id>/nonmember.xhtml</to-view-id>
      </navigation-case>
   </navigation-rule>
</faces-config>
```

EmployeeBean.java

```java
package act;


import javax.faces.bean.ManagedBean;
import javax.faces.bean.SessionScoped;

@ManagedBean(name="empbean")
@SessionScoped

public class EmployeeBean {
```

```java
    public EmployeeBean() {
    }

    String name;
    int empid;
    int [] empids = {111,222,333,444,555};

    public String getName() {
        return name;
    }

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

    public int getEmpid() {
        return empid;
    }

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

    public String valEmpId()
    {
        for(int i=0;i<empids.length;i++)
        {
            while(empid==empids[i])
            {

                return "home";
            }
        }
        return "nonmember";
    }
}
```

login.xhtml

```xml
<?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>Login Page</title>
    </h:head>
    <h:body>
        <h:form>
            <h:outputLabel value="Name : "/>
            <h:inputText id="name" value="${empbean.name}"/><br/>

            <h:outputLabel value="EmpID : "/>
            <h:inputText id="empid" value="${empbean.empid}"/><br/>

            <h:commandButton id="submit" value="submit" action="#{empbean.valEmpId}"/>
        </h:form>
    </h:body>
</html>
```

home.xhtml

```xml
<?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>
        Welcome ${empbean.name}
    </h:body>
</html>
```

nonmember.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>Non Member Page</title>
    </h:head>
    <h:body>
        Sorry #{empbean.name} you are not allowed !!!
    </h:body>
</html>
```

Output

1)

<div align="center">login.xhtml</div>

<div align="center">home.xhtml</div>

2)

<div align="center">login.xhtml</div>

<div align="center">nonmember.xhtml</div>

4.

4. **Redirection**

➔ One can ask JSF implementation to redirect to a new view.

➔ When redirect happens address field changes.

Using Redirect

I. Without Navigation Rules

➔ Add the string like '?faces-redirect=true' to the outcome string.

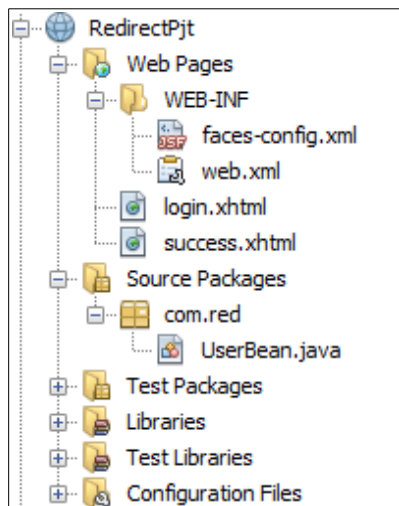Ex. <h:commandButton value="Login" action="welcome?faces-redirect=true"/>

Here when button is clicked it will directly go to welcome.xhtml.

ii. With Navigation Rule(Configuration file)

```
<navigation-case>
    <from-outcome>success</from-outcome>
    <to-view-id>/success.xhtml</to-view-id>
    <redirect/>
</navigation-case>
```


➔ Ex

Lets create RedirectPjt directory as shown below

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">
    <h:head>
        <title>Login Page</title>
    </h:head>
    <h:body>
        <h:form>
            <h:outputLabel value="Name :"/>
            <h:inputText value="#{user.name}"/><br/>
```

```
            <h:outputLabel value="Emp ID"/>
            <h:inputText value="#{user.empid}"/><br/>
            <h:commandButton id="submit" value="submit" action="success?faces-
            redirect=true"/>
            //For navigation rules
            /*<h:commandButton id="submit" value="submit" action="#{user.verify}"/>*/

        </h:form>

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

success.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>Success Page</title>
    </h:head>
    <h:body>
        Hello #{user.name}<br/>
        success redirect
    </h:body>
</html>
```

UserBean.java

```java
package com.red;

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

@Named(value = "user")
@SessionScoped
public class UserBean implements Serializable{

    public UserBean() {
    }

    public String getName() {
        return name;
    }

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

    public int getEmpid() {
        return empid;
    }

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

    String name;
    int empid;

    public String verify()
    {
        if(empid==111)
```

```
                return "success";
            else
                return "failure";
        }
}
```

faces-config.xml

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

<faces-config version="2.2"
               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-facesconfig_2_2.xsd">

    <navigation-rule>
        <navigation-case>
            <from-view-id>/login.xhtml</from-view-id>
            <from-outcome>success</from-outcome>
            <redirect/>
            <to-view-id>/success.xhtml</to-view-id>

        </navigation-case>
        <navigation-case>
            <from-view-id>/login.xhtml</from-view-id>
            <from-outcome>failure</from-outcome>
            <to-view-id>/failure.xhtml</to-view-id>
        </navigation-case>
    </navigation-rule>
</faces-config>
```

web.xml

```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>
```
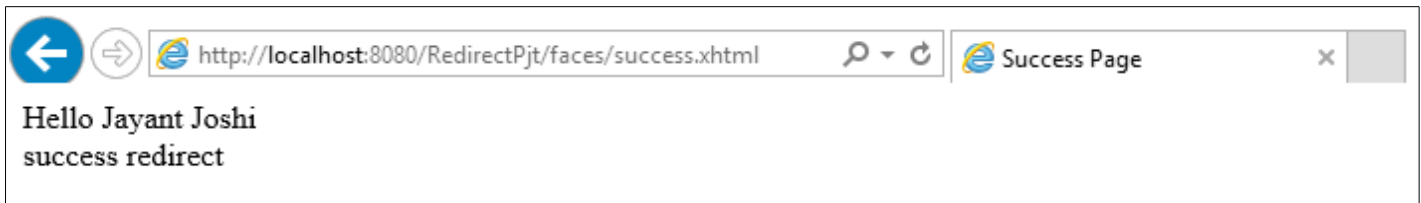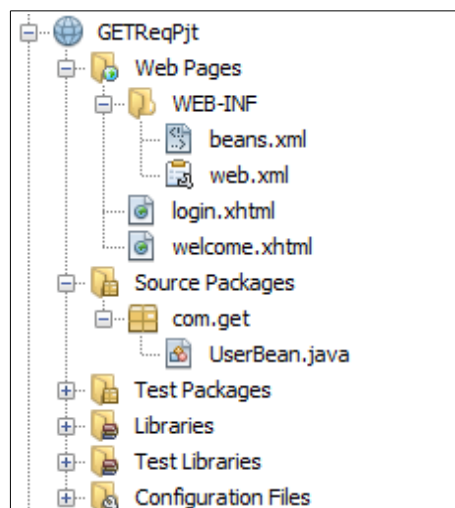
Output

Note : See the URL changes and in both the cases output will remain the same.

5. **RESTful Navigation and Bookmarkable URLs**

➔ By default, JSF application makes a sequence of POST request to the server.

➔ POST requests are neither bookmarkable nor cacheable.

➔ JSF 2.0 onward GET reqeusts too are supported.

➔ EX

Let create a GETReqPjt directory as shown below

login.xhtml

```xml
<?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:f="http://xmlns.jcp.org/jsf/core">

    <h:head>
        <title>Login Page</title>
    </h:head>
    <h:body>
        <h:form>
            Hello there <br/>
            Click here : <h:link outcome="/welcome.xhtml"  value="link">
                            <f:param name="name" value="JayJosh"/>
                         </h:link>
        </h:form>
    </h:body>
</html>
```

welcome.xhtml

```xml
<?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:f="http://xmlns.jcp.org/jsf/core">
    <f:metadata>
        <f:viewParam name="name" value="#{user.name}" />
    </f:metadata>
     <h:head>
        <title>Welcome Page</title>
    </h:head>
    <h:body>
        Hello #{user.name}
    </h:body>
</html>
```

UserBean.java

```java
package com.get;


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

@Named("user")
@SessionScoped

public class UserBean implements Serializable {

    public UserBean() {
    }

    String name;

    public String getName() {
        return name;
    }

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

```
}
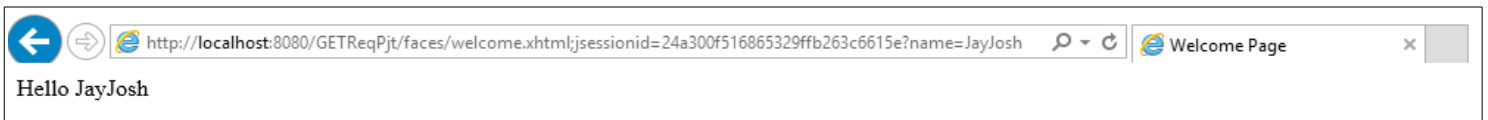```

**web.xml**

```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

login.xhtml

welcome.xhtml

6. More Navigation Rules

i. WildCards

➔ One can use wild cards in the <from-view-id> element of the navigation rule.

➔ Ex

```xml
<navigation-rule>
<from-view-id>/secure/*</from-view-id>
<navigation-case>
. . .
</navigation-case>
</navigation-rule>
```

Now this applies to all the pages that starts with the prefix `/secure`.

➔ If one use

```
<from-view-id>/*</from-view-id>
or
<from-view-id>*</from-view-id>
```

Then its means that it applies to all the pages.

ii. <u><form-action> tag</u>

➔ When different method expressions in the application have methods which returns same outcome then we can use

<form-action> to differentiate between them.

➔ Ex

```
<navigation-case>
      <from-action>#{quizBean.answerAction}</from-action>
      <from-outcome>again</from-outcome>
      <to-view-id>/again.xhtml</to-view-id>
</navigation-case>
<navigation-case>
      <from-action>#{quizBean.startOverAction}</from-action>
      <from-outcome>again</from-outcome>
      <to-view-id>/index.xhtml</to-view-id>
</navigation-case>
```

Above contents of `<from-action>` should match that of `<from-outcome>`.

➔ Here Navigation handler does not invoke the method present in #{ }. The methods will be invoked before navigation

handler kicks in. The navigation handler mere uses <form-action> as key to find matching navigation cases.

iii. <u>Conditional Navigation Cases</u>

➔ From JSF 2.0 one can supply 'if' element that activates navigation case only when the condition is fulfilled.

➔ Ex

```
<navigation-case>
<from-outcome>previous</from-outcome>
<if>#{quizBean.currentQuestion != 0}</if>
<to-view-id>/main.xhtml</to-view-id>
</navigation-case>
```

iv. <u>Dynamic Target View IDs</u>

➔ <to-view-id> element could be value expression. It will be evaluated and the result will be used as view ID.

➔ EX

```
<navigation-rule>
      <from-view-id>/main.xhtml</from-view-id>
      <navigation-case>
      <to-view-id>#{quizBean.nextViewID}</to-view-id>
      </navigation-case>
</navigation-rule>
```