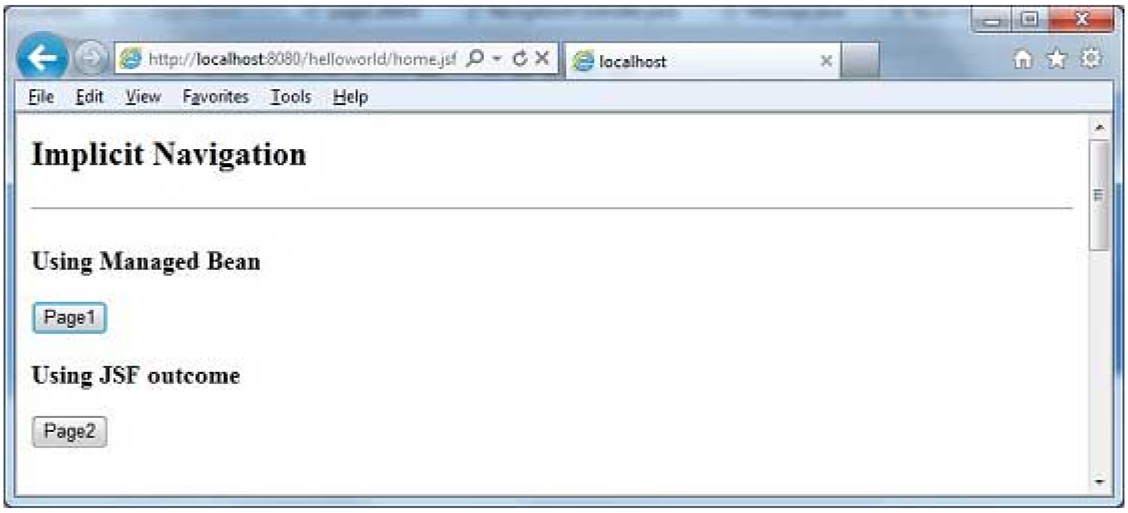
JSF - PAGE NAVIGATION

Navigation rules are those rules provided by JSF Framework which describe which view is to be shown when a button or link is clicked.

- Navigation rules can be defined in JSF configuration file named faces-config.xml.
- Navigation rules can be defined in managed beans.
- Navigation rules can contain conditions based on which resulted view can be shown.
- JSF 2.0 provides implicit navigation as well in which there is no need to define navigation rules as such.

Implicit Navigation

JSF 2.0 provides **auto view page resolver** mechanism named **implicit navigation**.In this case you only need to put view name in **action** attribute and JSF will search the correct **view** page automatically in the deployed application.


http://localhost:8080/helloworld/home.jsf
localhost
File Edit View Favorites Tools Help
Implicit Navigation

Using Managed Bean

Page1

Using JSF outcome

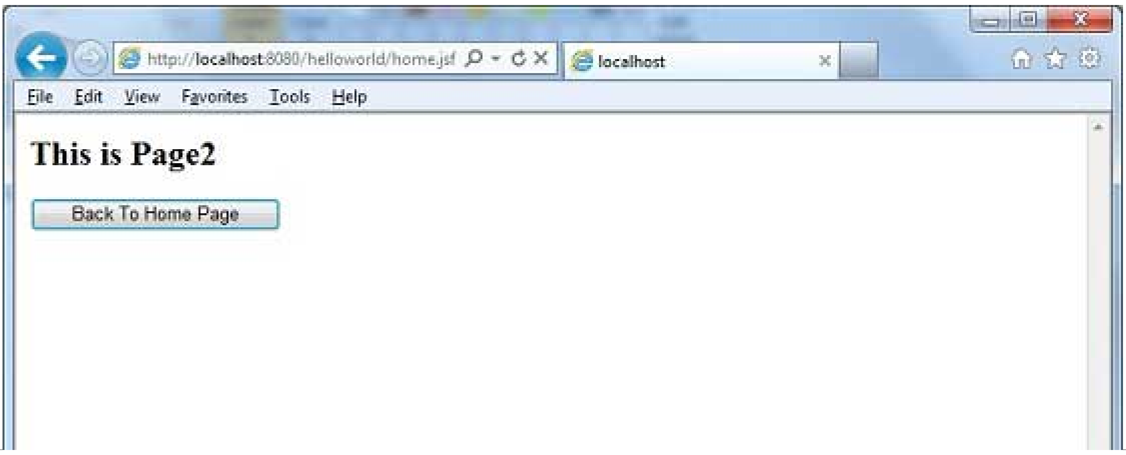
Page2

Auto navigation in JSF page

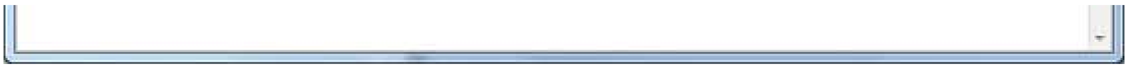
Set view name in action attribute of any JSF UI Component.

```
<h:form>
   <h3>Using JSF outcome</h3>
   <h:commandButton action="page2" value="Page2" />
</h:form>
```

Here when **Page2** button is clicked, JSF will resolve the view name, **page2** as page2.xhtml extension, and find the corresponding view file **page2.xhtml** in the current directory.


http://localhost:8080/helloworld/home.jsf
localhost
File Edit View Favorites Tools Help
This is Page2

Back To Home Page



Auto navigation in Managed Bean

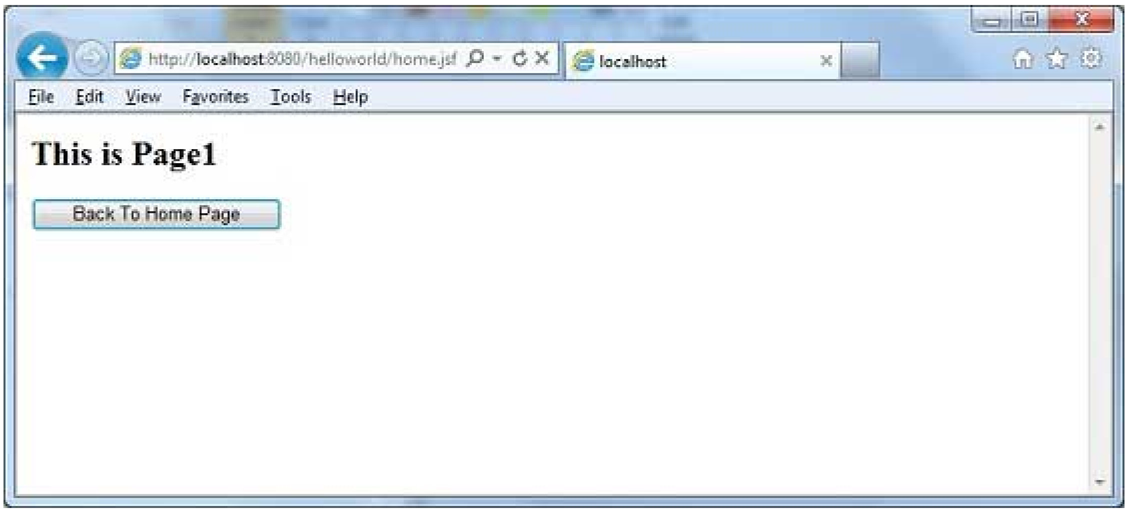
Define a method in managed bean to return a view name.

```
@ManagedBean(name = "navigationController", eager = true)
@RequestScoped
public class NavigationController implements Serializable {
   public String moveToPage1(){
      return "page1";
   }
}
```

Get view name in action attribute of any JSF UI Component using managed bean.

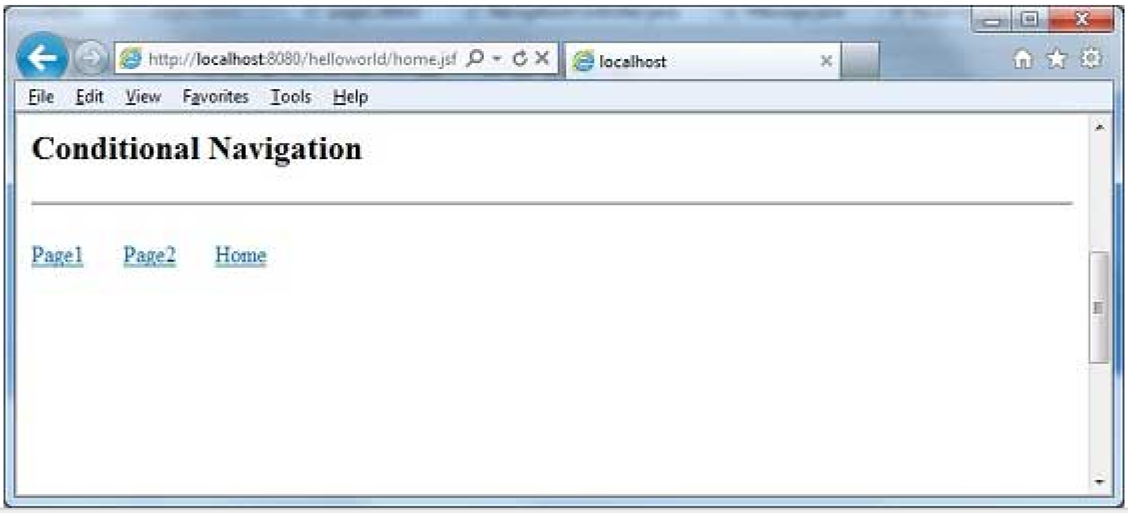
```
<h:form>
   <h3>Using Managed Bean</h3>
   <h:commandButton action="#{navigationController.moveToPage1}"
   value="Page1" />
</h:form>
```

Here when **Page1** button is clicked, JSF will resolve the view name, **page1** as page1.xhtml extension, and find the corresponding view file **page1.xhtml** in the current directory.


http://localhost:8080/helloworld/home.jsf
localhost
File Edit View Favorites Tools Help
This is Page1
Back To Home Page

Conditional Navigation

Using managed bean we can very easily control the navigation. Look at following code in a managed bean.


http://localhost:8080/helloworld/home.jsf
localhost
File Edit View Favorites Tools Help
Conditional Navigation
Page1
Page2
Home

```
@ManagedBean(name = "navigationController", eager = true)
@RequestScoped
```

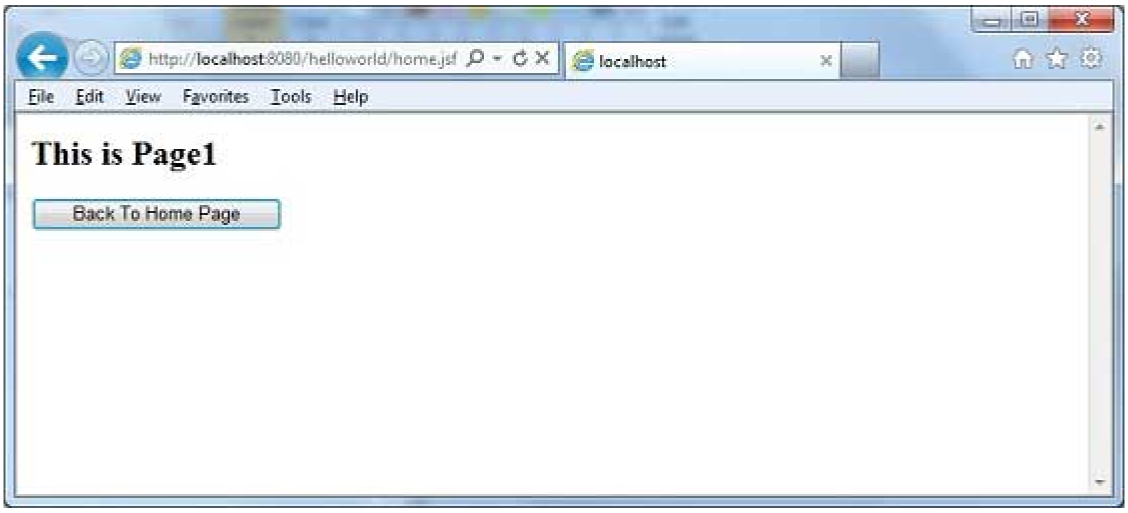
```
public class NavigationController implements Serializable {

   //this managed property will read value from request parameter pageId
   @ManagedProperty(value="#{param.pageId}")
   private String pageId;

   //condional navigation based on pageId
   //if pageId is 1 show page1.xhtml,
   //if pageId is 2 show page2.xhtml
   //else show home.xhtml
   public String showPage(){
      if(pageId == null){
         return "home";
      }
      if(pageId.equals("1")){
         return "page1";
      }else if(pageId.equals("2")){
         return "page2";
      }else{
         return "home";
      }
   }
}
```

Pass pageId as a request parameter in JSF UI Component.

```
<h:form>
   <h:commandLink action="#{navigationController.showPage}" value="Page1">
      <f:param name="pageId" value="1" />
   </h:commandLink>
   <h:commandLink action="#{navigationController.showPage}" value="Page2">
      <f:param name="pageId" value="2" />
   </h:commandLink>
   <h:commandLink action="#{navigationController.showPage}" value="Home">
      <f:param name="pageId" value="3" />
   </h:commandLink>
</h:form>
```

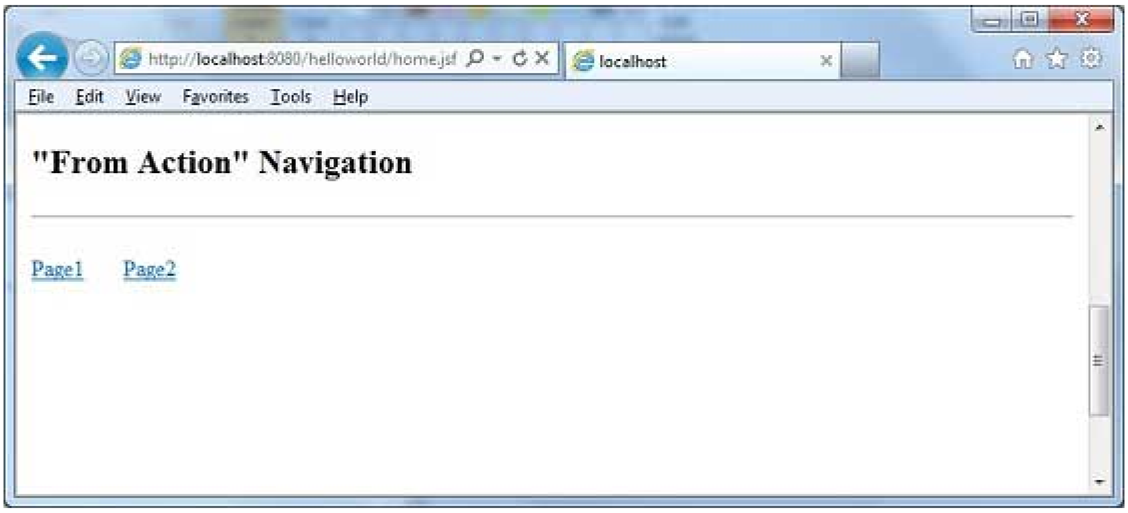
Here when "Page1" button is clicked

- JSF will create a request with parameter pageId=1
- Then JSF will pass this parameter to managed property **pageId** of **navigationController**
- Now **navigationController.showPage** is called which will return view as **page1** after checking the pageId
- JSF will resolve the view name, **page1** as **page1.xhtml** extension
- and find the corresponding view file **page1.xhtml** in the current directory


http://localhost:8080/helloworld/home.jsf
localhost
File Edit View Favorites Tools Help
Back To Home Page
This is Page1

Resolving Navigation based on from-action

JSF provides navigation resolution option even if managed bean different methods returns same view name.


http://localhost:8080/helloworld/home.jsf
localhost
File Edit View Favorites Tools Help

"From Action" Navigation

Page1 Page2

Look at following code in a managed bean.

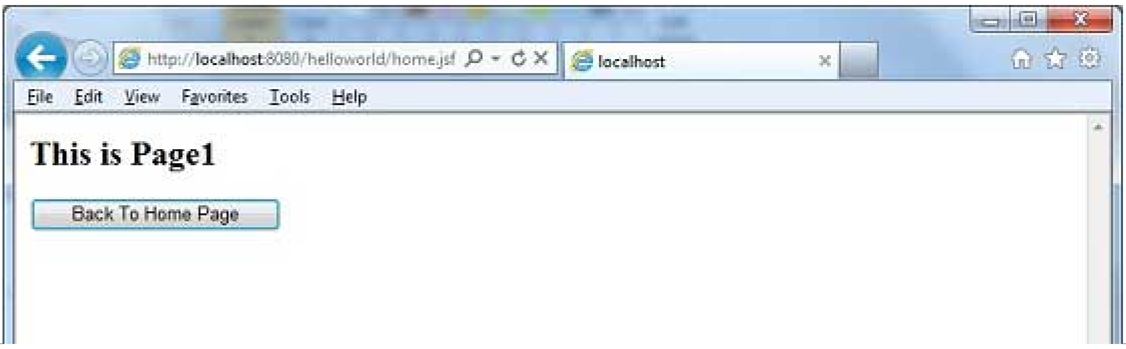
```
public String processPage1(){
   return "page";
}
public String processPage2(){
   return "page";
}
```

To resolve views, define following navigation rule in **faces-config.xml**

```
<navigation-rule>
   <from-view-id>home.xhtml</from-view-id>
   <navigation-case>
      <from-action>#{navigationController.processPage1}</from-action>
      <from-outcome>page</from-outcome>
      <to-view-id>page1.jsf</to-view-id>
   </navigation-case>
   <navigation-case>
      <from-action>#{navigationController.processPage2}</from-action>
      <from-outcome>page</from-outcome>
      <to-view-id>page2.jsf</to-view-id>
   </navigation-case>
</navigation-rule>
```

Here when **Page1** button is clicked

- **navigationController.processPage1** is called which will return view as **page**
- JSF will resolve the view name, **page1** as view name is **page** and **from-action** in **faces-config** is **navigationController.processPage1**
- and find the corresponding view file **page1.xhtml** in the current directory

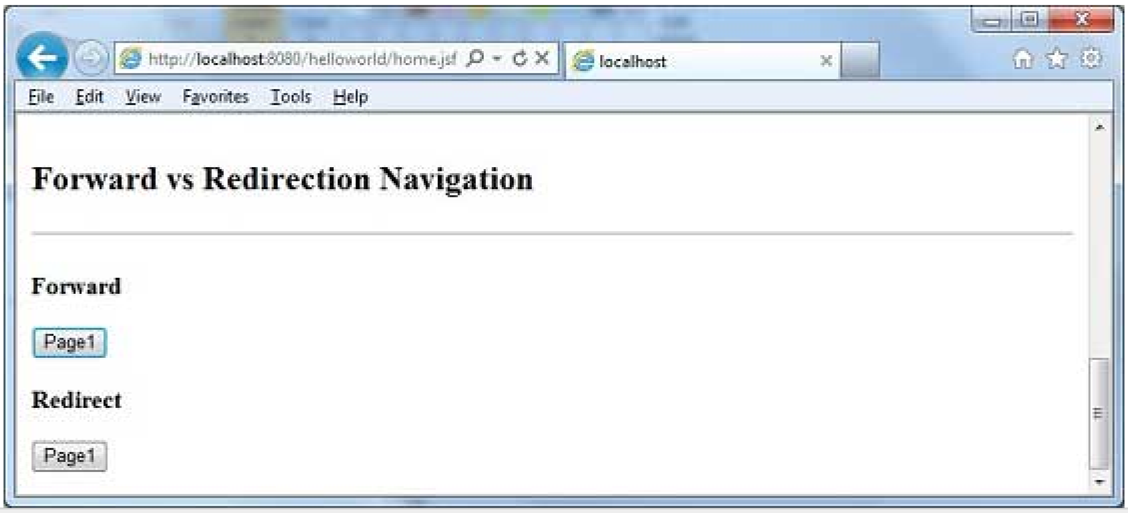

http://localhost:8080/helloworld/home.jsf
localhost
File Edit View Favorites Tools Help
Back To Home Page

This is Page1

Forward vs Redirect

JSF by default performs a server page forward while navigating to another page and the URL of the application do not changes.

To enable the page redirection, append **faces-redirect=true** at the end of the view name.


http://localhost:8080/helloworld/home.jsf
localhost
File Edit View Favorites Tools Help

Forward vs Redirection Navigation

Forward

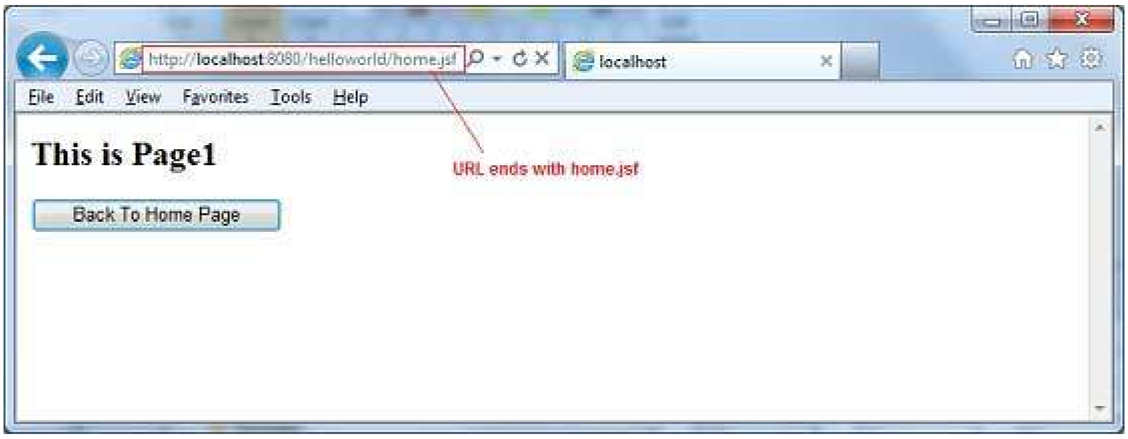
Page1

Redirect

Page1

```
<h:form>
   <h3>Forward</h3>
   <h:commandButton action="page1" value="Page1" />
   <h3>Redirect</h3>
   <h:commandButton action="page1?faces-redirect=true" value="Page1" />
</h:form>
```

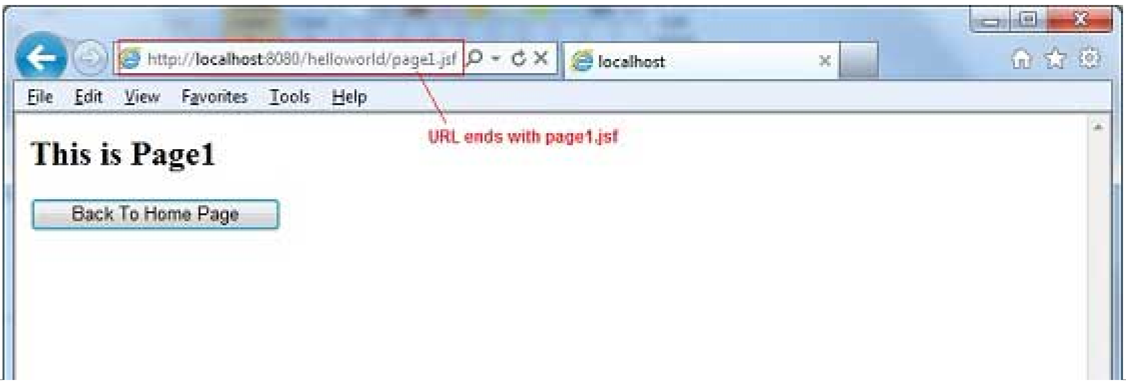
Here when **Page1** button under **Forward** is clicked


http://localhost:8080/helloworld/home.jsf
localhost
File Edit View Favorites Tools Help
URL ends with home.jsf

This is Page1

Back To Home Page

Here when **Page1** button under **Redirect** is clicked


http://localhost:8080/helloworld/page1.jsf
localhost
File Edit View Favorites Tools Help
URL ends with page1.jsf

This is Page1

Back To Home Page

Example Application

Let us create a test JSF application to test all of the above navigation examples.

Step	Description
1	Create a project with a name *helloworld* under a package *com.tutorialspoint.test* as explained in the *JSF - Create Application* chapter.
2	Create *NavigationController.java* under a package *com.tutorialspoint.test* as explained below.
3	Create *faces-config.xml* under a *WEB-INF* folder and updated its contents as explained below.
4	Update *web.xml* under a *WEB-INF* folder as explained below.
5	Create *page1.xhtml* and *page2.xhtml* and modify *home.xhtml* under a *webapp* folder as explained below.
6	Compile and run the application to make sure business logic is working as per the requirements.
7	Finally, build the application in the form of war file and deploy it in Apache Tomcat Webserver.
8	Launch your web application using appropriate URL as explained below in the last step.

NavigationController.java

```java
package com.tutorialspoint.test;

import java.io.Serializable;

import javax.faces.bean.ManagedBean;
import javax.faces.bean.ManagedProperty;
import javax.faces.bean.RequestScoped;

@ManagedBean(name = "navigationController", eager = true)
@RequestScoped
public class NavigationController implements Serializable {

   private static final long serialVersionUID = 1L;

   @ManagedProperty(value="#{param.pageId}")
   private String pageId;

   public String moveToPage1(){
      return "page1";
   }

   public String moveToPage2(){
      return "page2";
   }

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

   public String processPage1(){
      return "page";
   }

   public String processPage2(){
      return "page";
   }

   public String showPage(){
      if(pageId == null){
         return "home";
      }
      if(pageId.equals("1")){
         return "page1";
      }else if(pageId.equals("2")){
         return "page2";
      }else{
         return "home";
      }
   }

   public String getPageId() {
      return pageId;
   }

   public void setPageId(String pageId) {
      this.pageId = pageId;
   }
}
```

faces-config.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<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>
      <from-view-id>home.xhtml</from-view-id>
      <navigation-case>
         <from-action>#{navigationController.processPage1}</from-action>
         <from-outcome>page</from-outcome>
         <to-view-id>page1.jsf</to-view-id>
      </navigation-case>
      <navigation-case>
         <from-action>#{navigationController.processPage2}</from-action>
         <from-outcome>page</from-outcome>
         <to-view-id>page2.jsf</to-view-id>
      </navigation-case>
   </navigation-rule>
</faces-config>
```

web.xml

```
<!DOCTYPE web-app PUBLIC
   "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
   "http://java.sun.com/dtd/web-app_2_3.dtd" >

   <web-app>
   <display-name>Archetype Created Web Application</display-name>

   <context-param>
      <param-name>javax.faces.PROJECT_STAGE</param-name>
      <param-value>Development</param-value>
   </context-param>
   <context-param>
      <param-name>javax.faces.CONFIG_FILES</param-name>
      <param-value>/WEB-INF/faces-config.xml</param-value>
   </context-param>
   <servlet>
      <servlet-name>Faces Servlet</servlet-name>
      <servlet-class>javax.faces.webapp.FacesServlet</servlet-class>
   </servlet>
   <servlet-mapping>
      <servlet-name>Faces Servlet</servlet-name>
      <url-pattern>*.jsf</url-pattern>
   </servlet-mapping>
</web-app>
```

page1.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://java.sun.com/jsf/html">
   <h:body>
      <h2>This is Page1</h2>
      <h:form>
         <h:commandButton action="home?faces-redirect=true"
            value="Back To Home Page" />
      </h:form>
   </h:body>
</html>
```

page2.xhtml

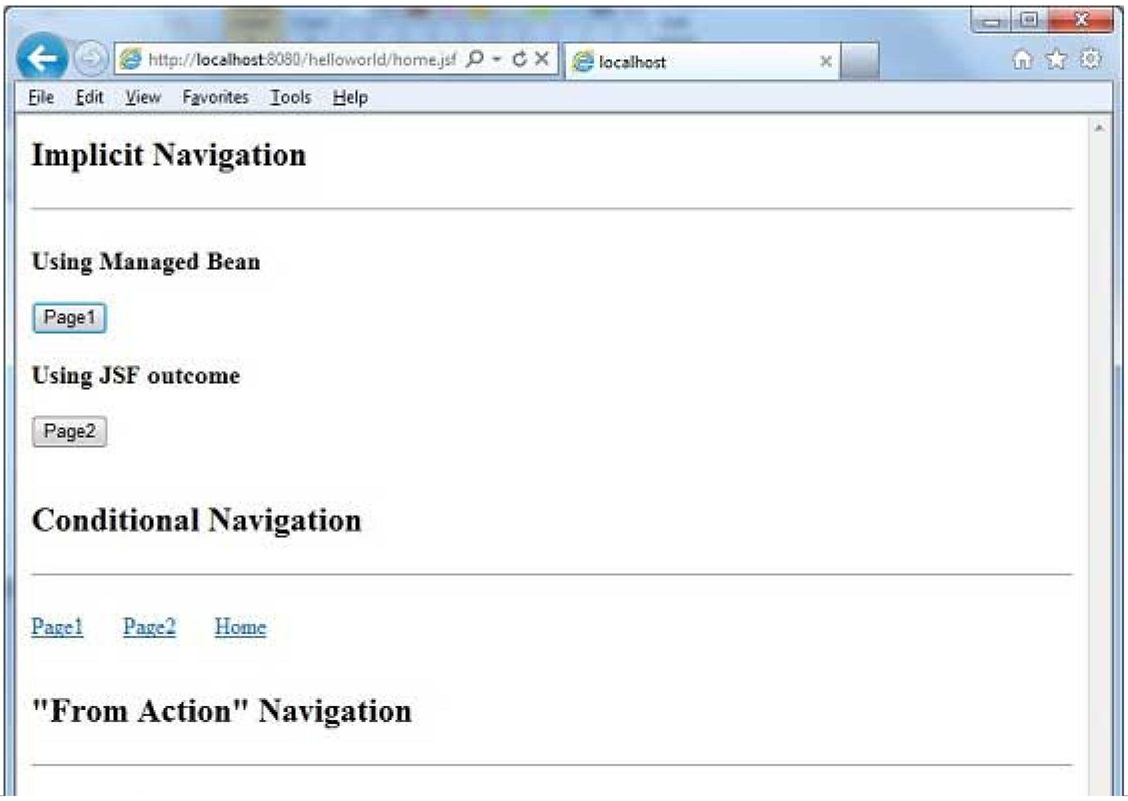
```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
   xmlns:h="http://java.sun.com/jsf/html">
   <h:body>
      <h2>This is Page2</h2>
      <h:form>
         <h:commandButton action="home?faces-redirect=true"
            value="Back To Home Page" />
      </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:f="http://java.sun.com/jsf/core"
   xmlns:h="http://java.sun.com/jsf/html">

   <h:body>
      <h2>Implicit Navigation</h2>
      <hr />
      <h:form>
         <h3>Using Managed Bean</h3>
         <h:commandButton action="#{navigationController.moveToPage1}"
            value="Page1" />
         <h3>Using JSF outcome</h3>
         <h:commandButton action="page2" value="Page2" />
      </h:form>
      <br/>
      <h2>Conditional Navigation</h2>
      <hr />
      <h:form>
```

```
         <h:commandLink action="#{navigationController.showPage}"
            value="Page1">
            <f:param name="pageId" value="1" />
         </h:commandLink>

         <h:commandLink action="#{navigationController.showPage}"
            value="Page2">
            <f:param name="pageId" value="2" />
         </h:commandLink>

         <h:commandLink action="#{navigationController.showPage}"
            value="Home">
            <f:param name="pageId" value="3" />
         </h:commandLink>
      </h:form>
      <br/>
      <h2>"From Action" Navigation</h2>
      <hr />
      <h:form>
         <h:commandLink action="#{navigationController.processPage1}"
         value="Page1" />

         <h:commandLink action="#{navigationController.processPage2}"
         value="Page2" />

      </h:form>
      <br/>
      <h2>Forward vs Redirection Navigation</h2>
      <hr />
      <h:form>
         <h3>Forward</h3>
         <h:commandButton action="page1" value="Page1" />
         <h3>Redirect</h3>
         <h:commandButton action="page1?faces-redirect=true"
         value="Page1" />
      </h:form>
   </h:body>
</html>
```

Once you are ready with all the changes done, let us compile and run the application as we did in JSF - Create Application chapter. If everything is fine with your application, this will produce following result:


http://localhost:8080/helloworld/home.jsf
localhost
File Edit View Favorites Tools Help

Implicit Navigation

Using Managed Bean

Page1

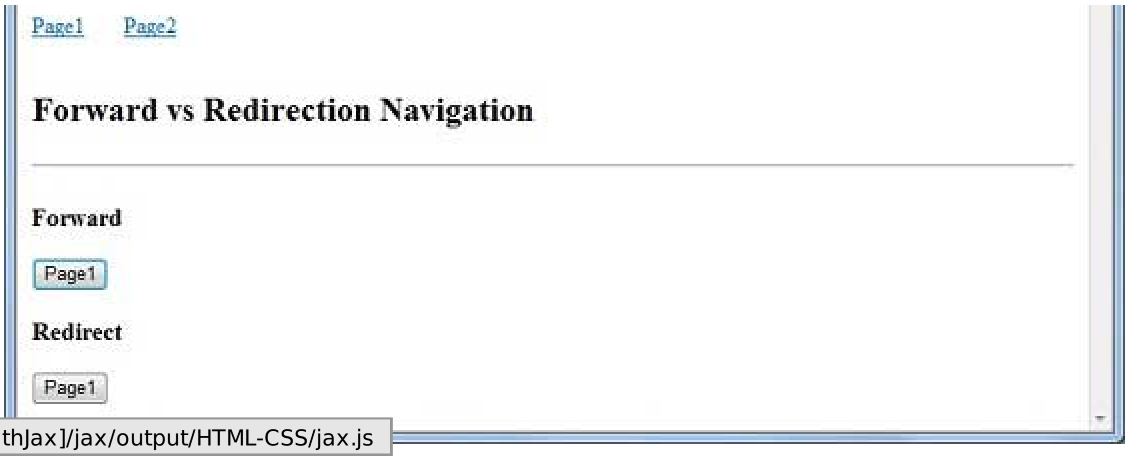
Using JSF outcome

Page2

Conditional Navigation

Page1 Page2 Home

"From Action" Navigation



Page1 Page2

Forward vs Redirection Navigation

Forward

Redirect

Loading [MathJax]/jax/output/HTML-CSS/jax.js