

12

Planning Navigation and Page Flow

Objectives

After completing this lesson, you should be able to do the following:

- Use a JavaServer Faces (JSF) Navigation diagram to plan the pages of an application and the navigation between them
- Describe JSF Navigation
- Describe the types of JSF Navigation cases
- Create a backing bean for a JSF page



Traditional Navigation

In traditional navigation, a button or hyperlink is used to navigate to a specific page.

```
<body>
    <form action="myNewPage.html">
        <button type="button"/>
    </form>
    <a href="http://myNewPage.html">
        Go To My New Page
    </a>
</body>
```

What Is JSF Navigation?

- JSF Navigation is defined through rules.
- Rules use outcomes to determine which page is displayed in a user interface (UI) event.
- Rules can be defined as:
 - Static
 - Dynamic
- They are defined in `faces-config.xml`.

JSF Navigation: Example



JSF Navigation Rules

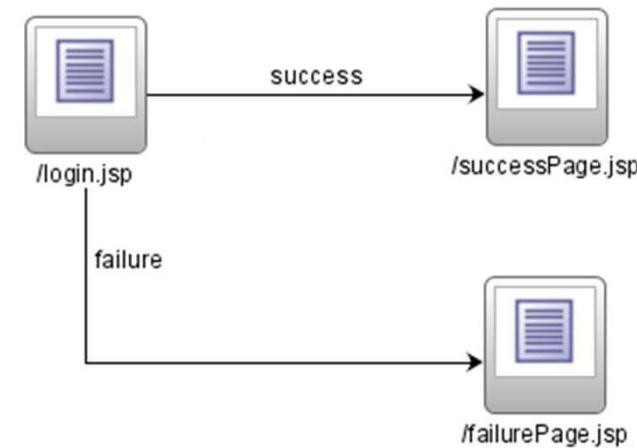
JSF Navigation Rules

- Navigation rules are stored in `faces-config.xml`.
- Navigation rules can be defined by using:
 - The `faces-config` console
 - The Pageflow diagrammer
 - The `faces-config.xml` file directly

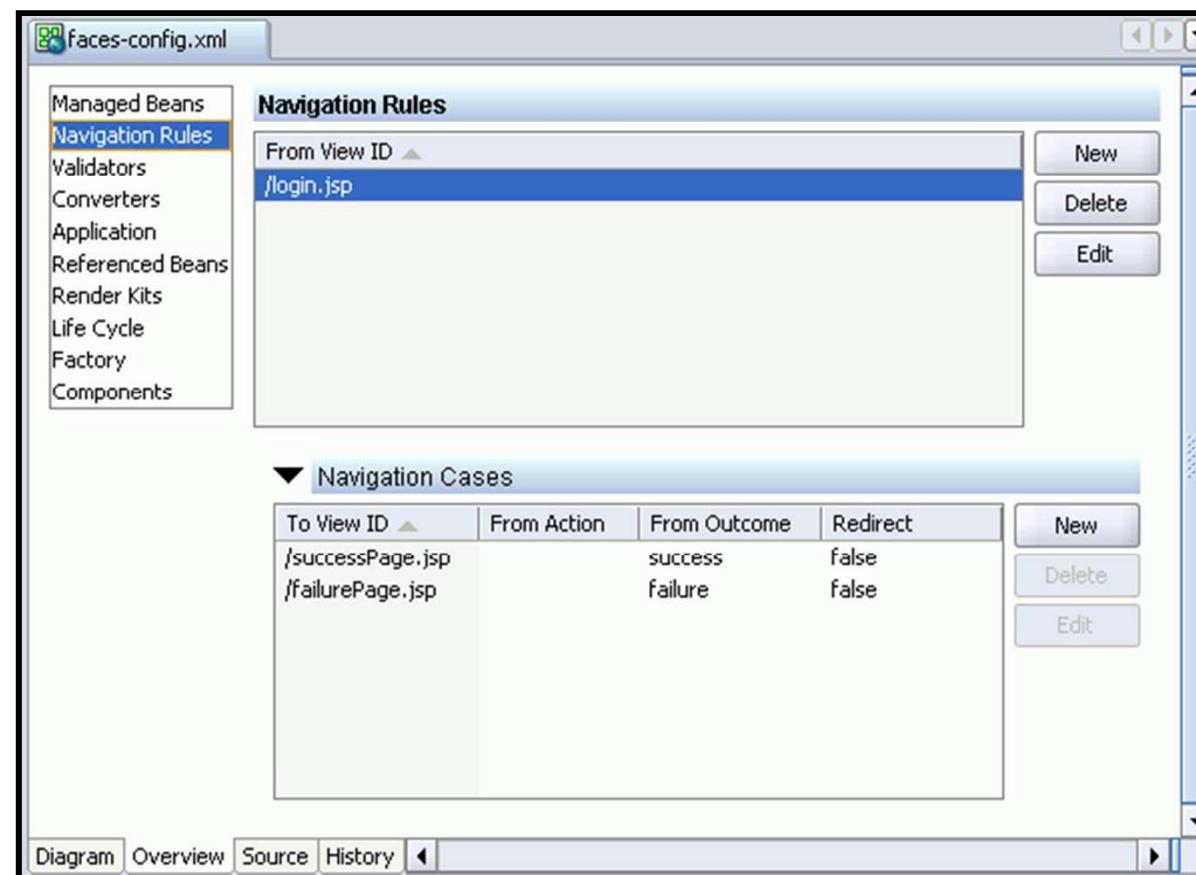
JSF Navigation Rules

Example:

- /login.jsp is the navigation rule with two cases.
- login.jsp is managed by the login backing bean.
- loginAction() is the method that returns success or failure:
 - If the method returns success, forward the user to /successPage.jsp.
 - If the method returns failure, forward the user to /failurePage.jsp.



faces-config Console

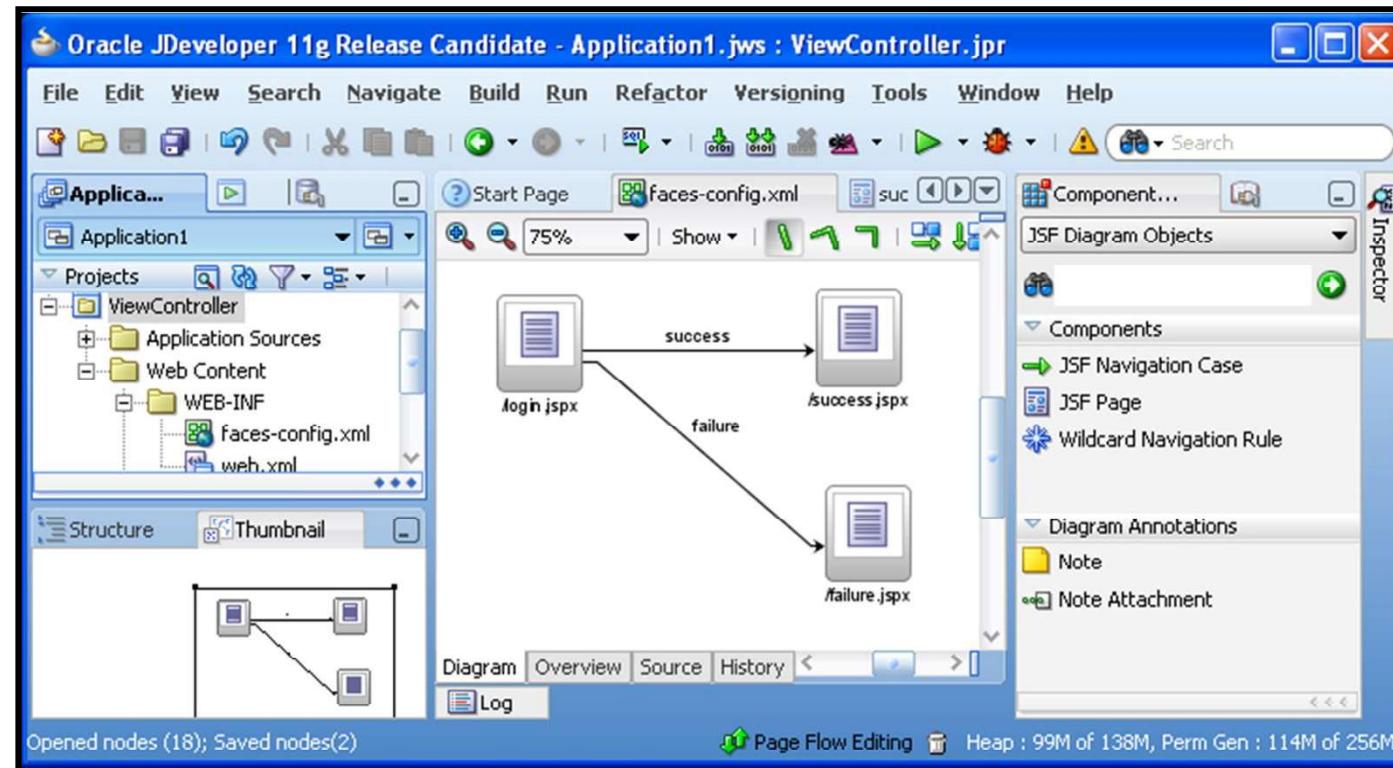


faces-config.xml

```
...
<navigation-rule>
<from-view-id>/login.jsp</from-view-id>
  <navigation-case>
    <from-outcome>success</from-outcome>
      <to-view-id>/successPage.jsp</to-view-id>
    </navigation-case>
    <navigation-case>
      <from-outcome>failure</from-outcome>
      <to-view-id>/failurePage.jsp</to-view-id>
    </navigation-case>
  </navigation-rule>
...
```

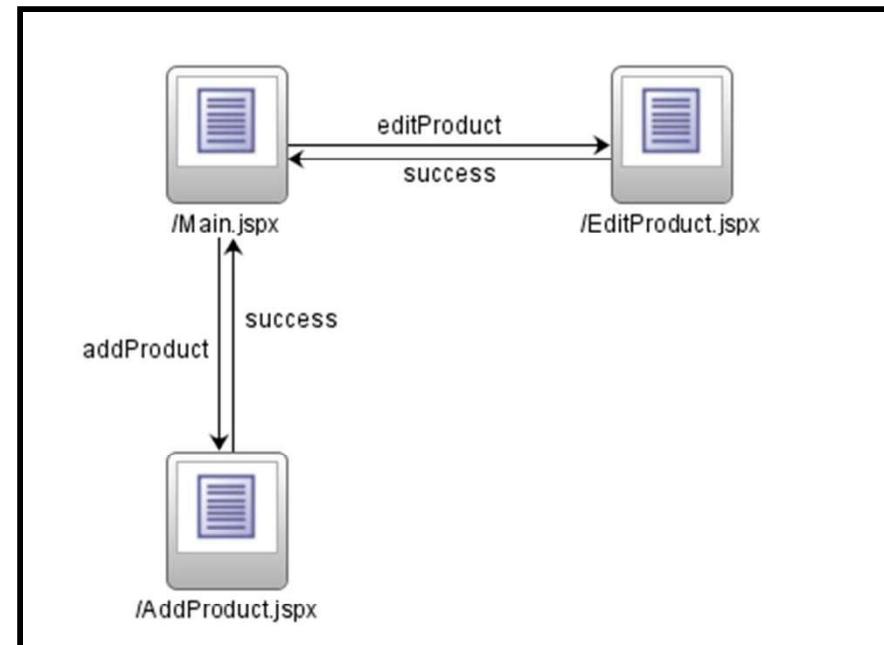
JSF Navigation Modeler

You create a JSF Navigation Diagram by dragging elements from the Component Palette. The faces-config.xml diagram is automatically updated.



JSF Navigation Diagram

With the JSF Navigation Diagram, you can visually design your application from a birds eye view.



Navigation Elements

- <navigation-rule>
- <from-view-id>
- <navigation-case>
 - <from-action>
 - <from-outcome>
 - <to-view-id>

```
<navigation-rule>
    <from-view-id>/Main.jspx</from-view-id>
    <navigation-case>
        <from-outcome>addProduct</from-outcome>
        <to-view-id>/addProduct.jspx</to-view-id>
    </navigation-case>
</navigation-rule>
```


Global rules apply to all pages in an application.

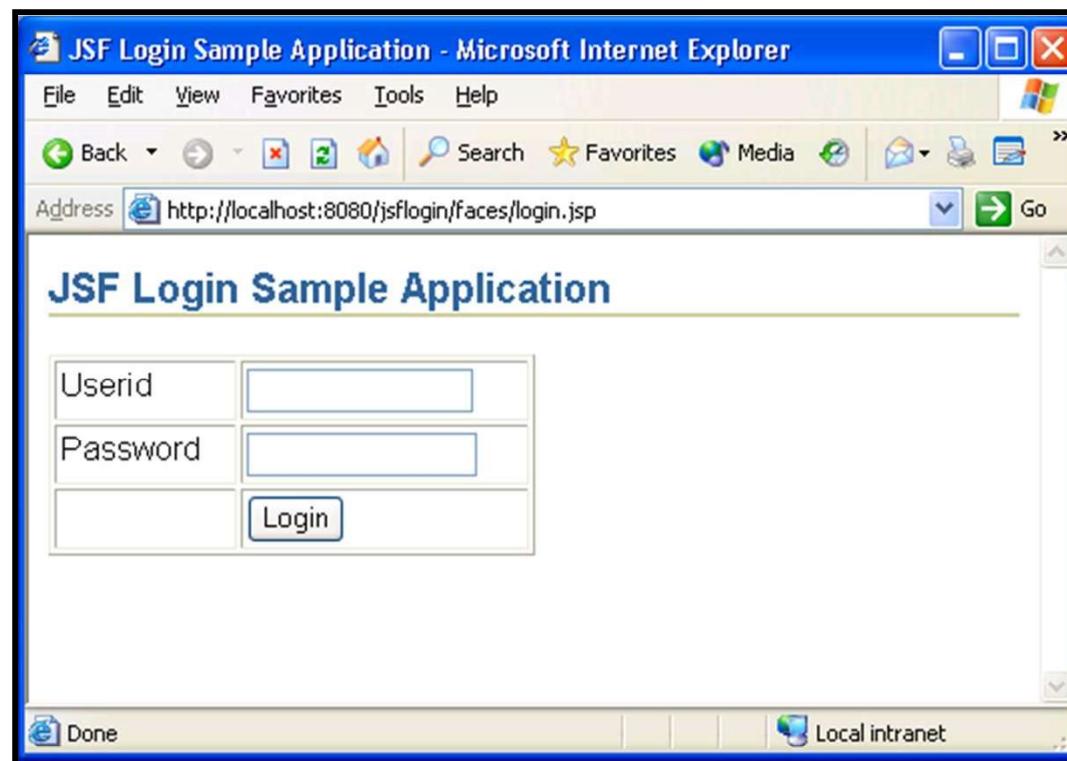
- Examples: Help page, Home page, Contact page
- Define <from-view-id> with an asterisk (*) wildcard.
- Define <from-outcome> (help).
- Define <to-view-id> (help.jsp).
- Any page in the application that returns help displays help.jsp.

Pattern-Based Rules

- Pattern-based rules are similar to global rules.
- Include a pattern for the `<from-view-id>` element:
 - `<from-view-id>/staffPages/*`
- This rule is used by all pages in the `/staffPages` path.

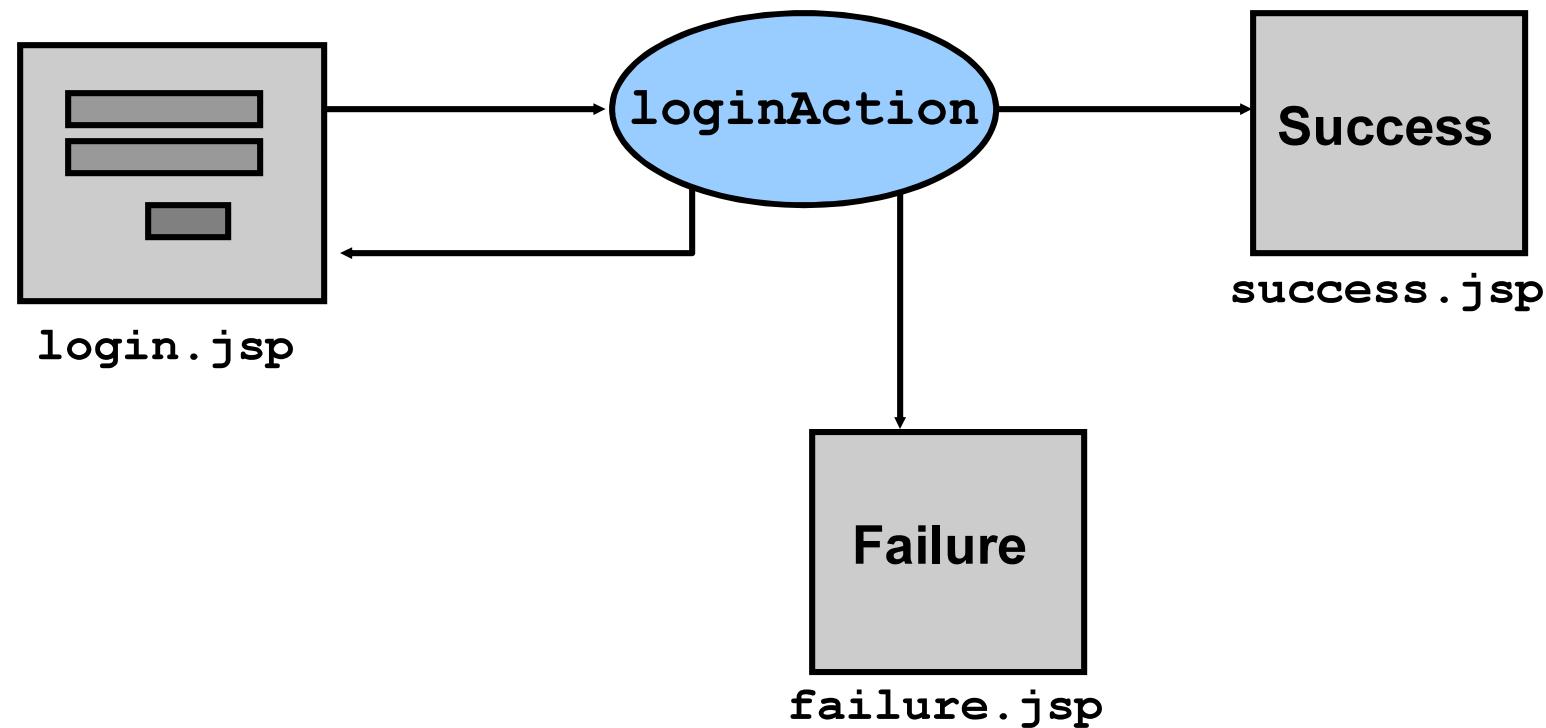
```
<navigation-rule>
    <from-view-id>/staffPages/*</from-view-id>
    <navigation-case>
        <from-outcome>help</from-outcome>
        <to-view-id>/menu/staffHelp.jsp</to-view-id>
    </navigation-case>
</navigation-rule>
```

Sample JSF login application:



JSF: Example

A simple example of a JSF Login Application:



JSF Navigation: Example

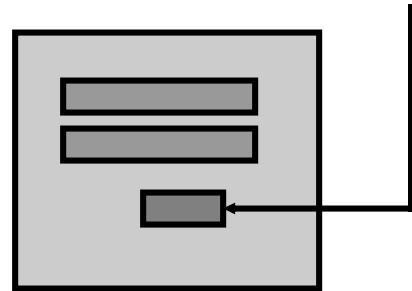
UI components and value binding to a managed bean:



JSF Navigation: Example

A command UI component bound to an action:

```
<h:commandButton action="#{Login.loginAction}" />
```



login.jsp



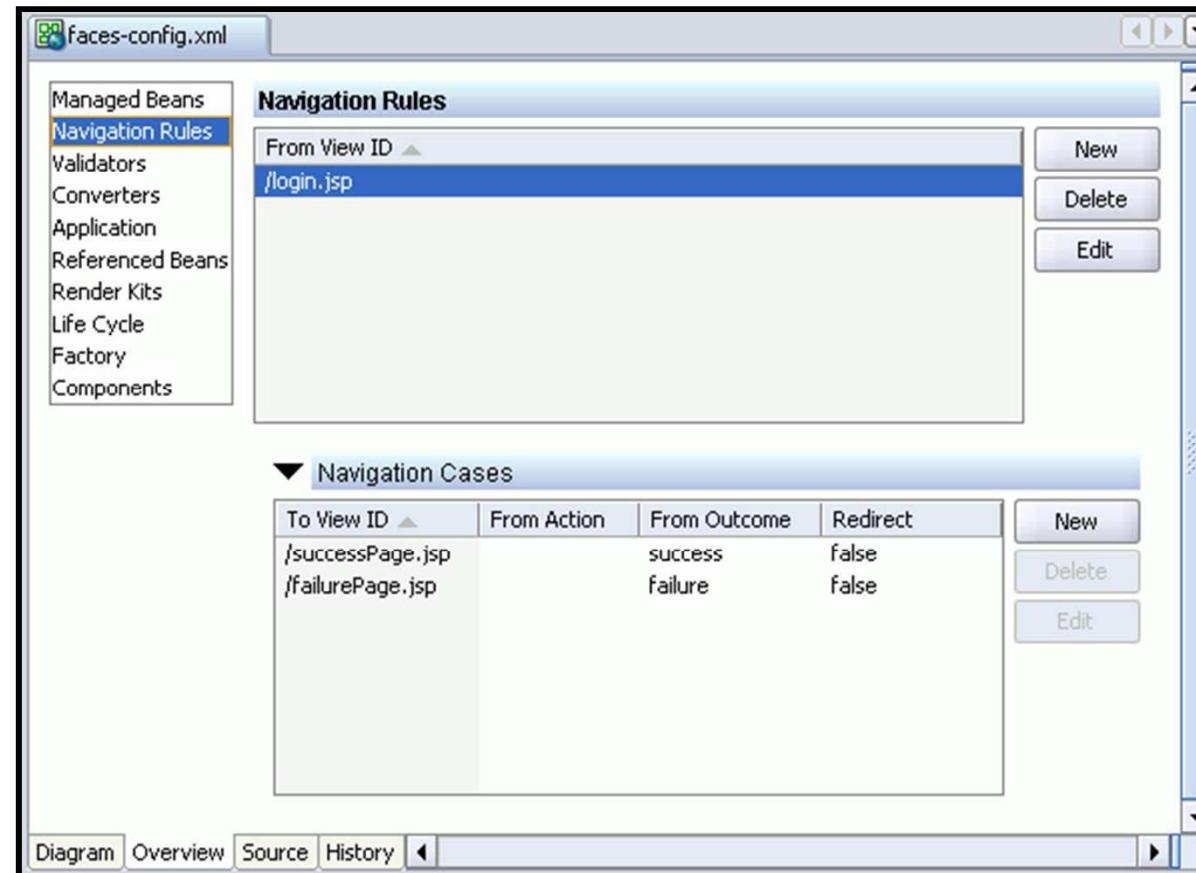
Login: A managed bean (Login.java) with an action method called loginAction()

Returns string: failure or success

JSF Navigation: Example

```
public String loginAction() {  
    String userId =  
        (String) this.getUserId().getValue();  
    String password =  
        (String) this.getPassword().getValue();  
    if (userId.equalsIgnoreCase("gary") &  
        password.equalsIgnoreCase("test")) {  
        return "success";    }  
    return "failure";  
}
```

Using the JSF Configuration Editor



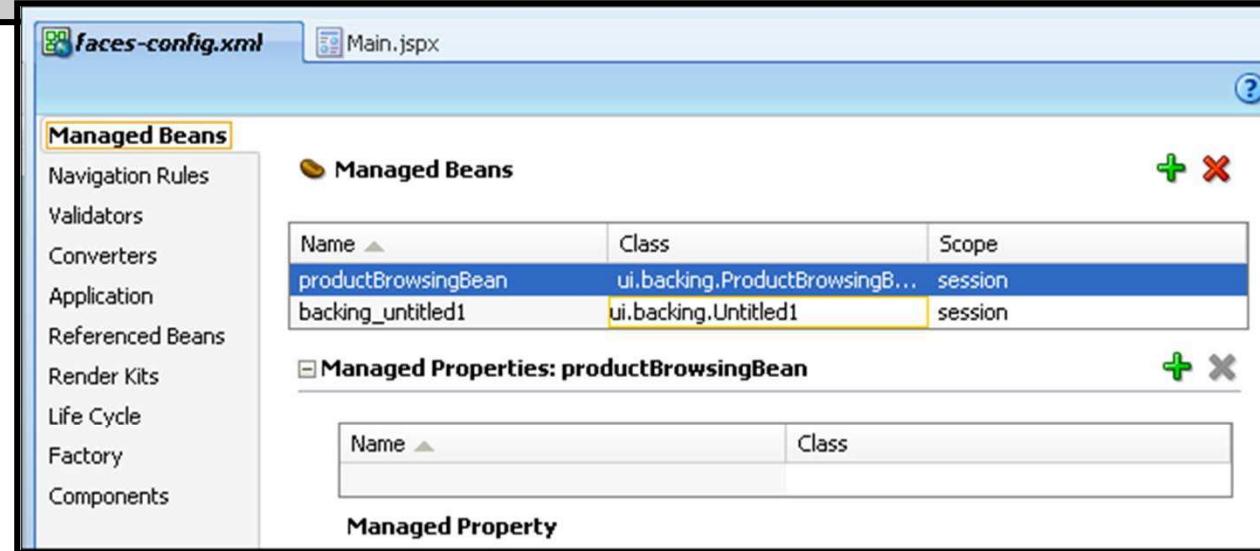
Managed Beans

- Managed beans can:
 - Store state
 - Execute a Java routine
 - Define a handler for event listeners
- They have no-argument constructors.
- Lazy initialization is performed by JavaServer Faces.
- Access scope may be none, request, application, or session.
- Backing beans:
 - Are special types of managed beans
 - Contain getter and setter methods for UI components

Creating Managed Beans

Entry in faces-config.xml:

```
<managed-bean>
    <managed-bean-name>productBrowsingBean</managed-bean-name>
    <managed-bean-class>ui.backing.ProductBrowsingBean
    </managed-bean-class>
    <managed-bean-scope> session </managed-bean-scope>
</managed-bean>
```



Managed Bean: Example

- Definition:

```
<managed-bean>
    <managed-bean-name>userbean</managed-bean-name>
    <managed-bean-class>com.oracle.sample.User</managed-bean-class>
    <managed-bean-scope>request</managed-bean-scope>
</managed-bean>
```

- Usage on a page:

```
<h:inputText value="#{userbean.name}">
```

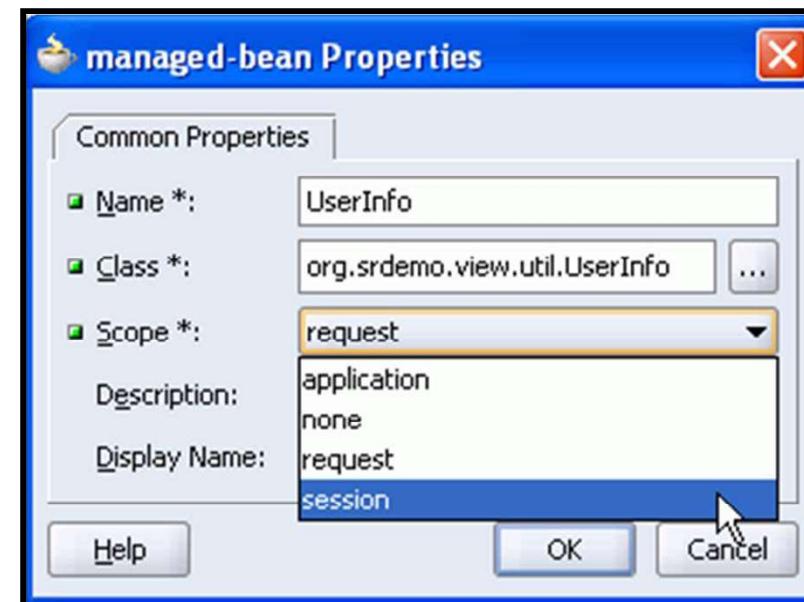
- Usage in code:

```
Application app =
FacesContext.getCurrentInstance().getApplication();
ValueBinding bind = app.createValueBinding("#{userbean}");
User user = (User)bind.getValue(ctx);
```

Setting Managed Bean Scope

Each managed bean has a scope:

- Application
- Session
- Request
- None



Relationships Between Managed Beans

Managed beans can access other beans:

Beans in scope	Access beans in scope			
	none	application	session	request
none	Yes	No	No	No
application	Yes	Yes	No	No
session	Yes	Yes	Yes	No
request	Yes	Yes	Yes	Yes

Managed Properties

- Managed bean variables that are exposed through getter and setter methods
- Configured in faces-config.xml

```
<!ELEMENT managed-property (description*, display-
    name*, icon*, property-name, property-class?,
    (map-entries|null-value|value|list-entries))>
```

- Possible values:
 - Null
 - Literal string
 - Lists and maps
 - Value binding expression (EL)

Managed Properties: Examples

- Literal string:

```
<managed-bean> ...
  <managed-property>
    <property-name>label</property-name>
    <value>Hello World</value>
  </managed-property>
```

- EL accessing a managed bean:

```
<managed-bean> ...
  <managed-property>
    <property-name>label</property-name>
    <value>#{UserInfo['firstname']}</value>
  </managed-property>
```

Managed Properties: Examples

- Populating a list:

```
<managed-bean>
...
<managed-property>
  <property-name>bookmarks</property-name>
  <list-entries>
    <value>http://otn.oracle.com/products/jev</value>
    <value>http://otn.oracle.com/products/ias</value>
  </list-entries>
</managed-property>
...
```

- Use `<map-entries>` to specify managed properties that are of the Map type.

Using the Managed Bean on the JSF Page

- Access UI components and values

Login.java managed bean:

```
...
Getter for userId
Setter for userId
...
```

successPage.jsp:

```
...
<h:outputText value="#{Login.userId.value}"
binding="#{SuccessPage.outputText1}"
id="outputText1"
/> ...
```

Summary

In this lesson, you should have learned how to:

- Use a JSF Navigation Diagram to plan the pages of an application and the navigation between them
- Describe JSF Navigation
- Define types of JSF Navigation cases
- Create a backing bean for a JSF page



Practice 12: Overview

This practice covers the following topics:

- Using a JSF Navigation Diagram to model page flow in an application
- Creating the JSF pages of the application
- Binding components to a managed bean

