**ITL-3 Practical 7**

**Q:** Design reusable components of the form using taglib.

**A:**

As you build JSF pages for your application, some pages may become complex and long, making editing complicated and tedious. Some pages may always contain a group of components arranged in a very specific layout, while other pages may always use a specific group of components in multiple parts of the page. And at times, you may want to share some parts of a page or entire pages with other developers. Whatever the case is, when something changes in the UI, you have to replicate your changes in many places and pages. Building and maintaining all those pages, and making sure that some sets or all are consistent in structure and layout can become increasingly inefficient.

Instead of using individual UI components to build pages, you can use page building blocks to build parts of a page or entire pages. The building blocks contain the frequently or commonly used UI components that create the reusable content for use in one or more pages of an application. Depending on your application, you can use just one type of building block, or all types in one or more pages. And you can share some building blocks across applications. When you modify the building blocks, the JSF pages that use the reusable content are automatically updated as well. Thus, by creating and using reusable content in your application, you can build web user interfaces that are always consistent in structure and layout, and an application that is scalable and extensible.

ADF Faces provides the following types of reusable building blocks:

Page fragments: Page fragments allow you to create parts of a page. A JSF page can be made up of one or more page fragments. For example, a large JSF page can be broken up into several smaller page fragments for easier maintenance. For details about creating and using page fragments, see [Section 10.3, "Using Page Fragments."](https://docs.oracle.com/cd/E16162_01/web.1112/e16181/af_reuse.htm#CACHFJDJ)

Page templates: By creating page templates, you can create entire page layouts using individual components and page fragments. For example, if you are repeatedly laying out some components in a specific way in multiple JSF pages, consider creating a page template for those pages. When you use the page template to build your pages, you can be sure that the pages are always consistent in structure and layout across the application.

The page template and the declarative component share much of the functionality. The main difference is that the page template supports ADF Model and ADF Controller using a page template model. Using the value attribute, you can specify which object to use as the bindings inside of the page template. If the value is a page template model binding, ADF Model page bindings may be used, and you may use the ADF page definition to determine which view to include.

For details about creating and using page templates, see [Section 10.4, "Using Page Templates,"](https://docs.oracle.com/cd/E16162_01/web.1112/e16181/af_reuse.htm#CACCFCJC) and [Section 10.4.3, "How to Create JSF Pages Based on Page Templates."](https://docs.oracle.com/cd/E16162_01/web.1112/e16181/af_reuse.htm#CACEHCBA)

Declarative components: The declarative components feature allows you to assemble existing, individual UI components into one composite, reusable component, which you then declaratively use in one or more pages. For example, if you are always inserting a group of components in multiple places, consider creating a composite declarative component that comprises the individual components, and then reusing that declarative component in multiple places throughout the application. Declarative components can also be used in page templates.

The declarative component is deployed as part of an ADF library JAR file. It features its own TLD file that allows you to put the component in your own namespace. The declarative component allows you to pass facets into the component and also any attributes and method expressions. Inside of the declarative component, the attributes and facets may be accessed using EL expressions It has a relatively low overhead as it does not involve ADF Model or ADF Controller, which also means that it does not have support for ADF Model transactions or ADF Controller page flows.

Note that you should not reference individual components inside of a declarative component, and individual components within a declarative component should not reference external components. The reason is that changes in the declarative component or in the consuming page could cause the partial triggers to no longer work. For details about creating and using declarative components, see [Section 10.5, "Using Declarative Components."](https://docs.oracle.com/cd/E16162_01/web.1112/e16181/af_reuse.htm#CACBFGFC)

Page templates, page fragments, and declarative components provide consistent structure and layout to the pages in an application. These building blocks can not only be reused in the same application, but also can be shared across applications. When update a building block, all the instances where it is used is automatically updated.

Page templates are data-bound templates that support both static areas that do not change and dynamic areas where they change during runtime. You can use page fragments to build modular pages. For instance, you can create page fragments for the header, footer, and company logo and reuse these fragments throughout the application. You can use declarative components when you have several components that always used in a group. By creating a declarative component, you can add it to the tag library and be able to drag and drop the declarative component from the JDeveloper Component Palette.

Page templates, declarative components, and regions implement the javax.faces.component.NamingContainer interface. At runtime, in the pages that consume reusable content, the page templates, declarative components, or regions create component subtrees, which are then inserted into the consuming page's single, JSF component tree. Because the consuming page has its own naming container, when you add reusable content to a page, take extra care when using mechanisms such as partialTargets and findComponent(), as you will need to take into account the different naming containers for the different components that appear on the page. For more information about naming containers, see [Section 4.5, "Locating a Client Component on a Page."](https://docs.oracle.com/cd/E16162_01/web.1112/e16181/af_arch.htm#BABJDABH)

If you plan to include resources such as CSS or JavaScript, you can use the af:resource tag to add the resources to the page. If this tag is used in page templates and declarative components, the specified resources will be added to the consuming page during JSP execution. For more information, see [Section 10.6, "Adding Resources to Pages."](https://docs.oracle.com/cd/E16162_01/web.1112/e16181/af_reuse.htm#BABHJJHE)

If you are not using an ADF task flow to navigate a portion of the page, you should not be using regions, but instead use one of the other compound components. Among the compound components, you should use a page template if you need to use bindings inside of your compound component and they differ from the bindings of the host page. You should use a declarative component if you do not need bindings for your page and do not need to use a bounded task flow as part of your page.

**Reusable Components Use Cases and Examples**

The File Explorer application uses a fileExploreorTemplate to provide a consistent look and feel to all the pages in the application. The facets of the file provide working area to place different types of information. The template defines an appCopyright facet that is used to display copyright information for every page.The main page of the File Explorer application not only uses the page template, but also uses page fragments to contain the content for the individual facets of the template. The header.jspx page fragment contains the menu commands for the application.If you have several components that works as a group and repeats in several places, you can define a declarative component to group these components together. Once you have created the component, you can use this declarative component like any other component. For example, you may use several inputText components to denote first name, last name, and email address. Since this three inputText components will be used repeatedly in your application, you can create a declarative component for them.

**Additional Functionality for Reusable Components**

You may find it helpful to understand other ADF features before you implement your reusable components. Following are links to other functionality that are related to reusable components.

* For more information about customization, see the "[Customizing Applications with MDS](https://docs.oracle.com/cd/E16162_01/web.1112/e16182/customize.htm#ADFFD2077)" chapter of the Oracle Fusion Middleware Fusion Developer's Guide for Oracle Application Development Framework.
* For more information on using the Quick Start Layouts to provide a pre-configured layout, see [Section 9.2.3, "Using Quick Start Layouts."](https://docs.oracle.com/cd/E16162_01/web.1112/e16181/af_orgpage.htm#CACFIFJA)
* For information about using model parameters and ADF Model data bindings, see the "[Using Page Templates](https://docs.oracle.com/cd/E16162_01/web.1112/e16182/web_getstarted.htm#ADFFD564)" section of the Oracle Fusion Middleware Fusion Developer's Guide for Oracle Application Development Framework
* For information about packaging a page template into an ADF Library JAR for reuse, see the "[Reusing Application Components](https://docs.oracle.com/cd/E16162_01/web.1112/e16182/reusing_components.htm#ADFFD1845)" chapter of the Oracle Fusion Middleware Fusion Developer's Guide for Oracle Application Development Framework.