# Application Development
# DICT312

## Week 4 Part 2– Jakarta Faces

Dr. O.S Ogunleye
**Email:** olalekan.ogunleye@ump.ac.za

**Telephone: 013 002 0227**
**Office: Building 5 (Room 212)**
**Consultation( By Appointment Only)**

UNIVERSITY OF MPUMALANGA

# Objectives of this Session

By the end of this session, you should be able to:

Understand Ajax-enabled Faces applications

Understand Jakarta Faces HTML5 support

Understand Faces Flow

Understand Jakarta Faces Websocket support

Understand and work with Additional Faces component libraries

# Introduction to Jakarta Faces - Facelets

Facelets is the default Jakarta Faces view technology. Facelets are written using standard Extensible Hypertext Markup Language (XHTML), using Jakarta Faces-specific XML namespaces that provide Jakarta Faces-specific tags we can use to develop the user interface of our web applications.

JSF allows you to build web user interfaces by assembling reusable UI components on a page. Instead of thinking in terms of raw HTML and HTTP requests, you think in terms of components like data tables, input fields, and buttons. This is a stateful framework, meaning it can automatically save and restore the state of the UI across multiple requests, making it easier to manage complex forms.

# Faces-config.xml

In most cases, configuring a Jakarta Faces application is unnecessary, as it follows a convention over the configuration approach.

For some specific cases, when overriding Jakarta Faces' default error messages for example, we still need to configure Jakarta Faces via a faces-config.xml configuration file. This file is located inside the WEB-INF folder

# Standard Resource Locations

Resources are artefacts a page or Jakarta Faces component needs to render properly. Resource examples are CSS stylesheets, JavaScript files, and images.

When working with Jakarta Faces, resources can be placed in a subdirectory in a folder called resources, either in the Other Sources directory or in the META-INF Directory in Eclipse. By convention, Jakarta Faces components know they can retrieve resources from one of these two locations.

To avoid cluttering the resources directory, resources are typically placed in a subdirectory. This subdirectory is referred to from the library attribute of Faces components. For example, we could place a CSS stylesheet called styles.css in /resources/css/styles.css.

# Standard Resource Locations

In our Faces pages, we could retrieve this CSS file using the <h:outputStylesheet> tag, as follows:

```
<h:outputStylesheet library="css" name="styles.css"/>
```

The value of the library attribute must match the subdirectory where our stylesheet is located. Similarly, we could have a JavaScript file at /resources/scripts/somescript.js and an image at /resources/images/logo.png, and we could access these resources as follows:

```
<h:graphicImage library="images" name="logo.png"/>

<h:outputScript library="scripts" name="somescript.js"/>
```

Notice that in each case, the value of the library attribute matches the corresponding subdirectory name in the resources directory, and the value of the name attribute matches the resource's filename.

# Developing Application using Faces

To illustrate basic Jakarta Faces concepts, we will develop a simple application consisting of two Facelet pages and a single Contexts and Dependency Injection (CDI) named bean.

# Faces Validation

As its name implies, this tag validates that the entered value for the text field is between a minimum and maximum length. Minimum and maximum values are defined by the tag's minimum and maximum attributes. <f:validateLength> is one of the standard validators included with Jakarta Faces. Just like with the required attribute of <h:inputText>, Jakarta Faces will automatically display a default error message when a user attempts to submit a form with a value that does not validate

| Validation Tag | Description |
|---|---|
| `<f:validateBean>` | Bean validation allows us to validate named bean values by using annotations in our named beans without having to add validators to our Jakarta Faces tags. This tag allows us to fine-tune Bean Validation if necessary. |
| `<f:validateDoubleRange>` | Validates that the input is a valid `Double` value between the two values specified by the tag's `minimum` and `maximum` attributes, inclusive |
| `<f:validateLength>` | Validates that the input's length is between the values specified by the tag's `minimum` and `maximum` values, inclusive |

| Validation Tag | Description |
|---|---|
| `<f:validateLongRange>` | Validates that the input is a valid `Double` value between the values specified by the tag's `minimum` and `maximum` attributes, inclusive |
| `<f:validateRegex>` | Validates that the input matches a regular expression pattern specified in the tag's `pattern` attribute |
| `<f:validateRequired>` | Validates that the input is not empty. This tag is equivalent to setting the `required` attribute to `true` in the parent input field |

# Core Concepts of JSF

**Facelets**: This is the preferred view technology for JSF. Facelets pages are simply XML files (usually with an .xhtml extension) that look very similar to HTML but allow you to use special JSF component tags.

**Backing Beans**: These are the link between your UI and your backend business logic. A backing bean is simply a CDI bean (usually @RequestScoped or @ViewScoped) that is associated with a JSF page. It holds the data for the UI components and exposes action methods that can be triggered by buttons or links. We use the @Named annotation to make a CDI bean accessible from a JSF page.

# Core Concepts of JSF

**Expression** Language (EL): JSF uses Expression Language (e.g., #{studentController.studentList}) to bind UI components to properties and methods in the backing beans. This is how the UI reads data from your beans and sends updated values back.

**UI Components**: JSF provides a standard library of tags (e.g., <h:form>, <h:inputText>, <h:commandButton>, <h:dataTable>) that render as standard HTML. These components handle rendering, data conversion, validation, and event handling.

# Core Concepts of JSF

In this module, we will create a simple web page that displays a list of all students from our StudentService and provides a form to add new students.

# Summary of Today's Class

# Thank you