Java Server Faces

M. en C. José Asunción Enríquez Zárate

JSF 2.X Feature

- Facelets is the principle view declaration language.
- A comprehensive composite component model
- AJAX request processing and partial page updates
- Partial State Saving
- Broader support for system events
- HTTP GET now fully supported in the JSF lifecycle.
- New scopes, and support for custom scopes
- Annotations to simplify configuration.
- Standardized resource loading is provided.
- Exception handling has been improved

Java Server Faces (JSF)

- JSF pages are text-based documents that describe a user interface.
- JSF uses an object-oriented, component-based architecture to create user interfaces.
- Application developers do not create JSPs or Servlet classes.
- JSF 2.x uses a page-templating technology called Facelets.
  - Source files for Facelets typically end with the .xhtml extension.

Goals of JSF

- The goals for the JSF framework include:
  - Addressing common web application requirements
  - Managing UI component state across requests and page navigation
  - Encapsulating markup differences across clients
  - Supporting complex form processing
  - Supporting the strong type event model
  - Providing capability for data validation, conversion, and error handling.

Benefits of JSF

- As a component-based web application framework, JSF offers a clean separation between behavior and presentation.
  - Reduces the amount of browser-specific knowledge required to develop web applications
  - Provides an object-oriented approach to UI development; similar to Swing
  - Maintains state across HTTP requests
  - Leverages JSP knowledge by using JSP-like tags and the
  - unified Expression Language (EL)
  - Enables advanced type conversion and validation

Benefits of JSF

- As a component-based web application framework, JSF offers a clean separation between behavior and presentation.
  - Provides page templating based on Java EE standards
  - Is designed to be extended, enables creation of custom components
  - Supports internationalization
  - Supports AJAX

HelloWordJSF Example

```
<?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>
            Hello World JSF !
        </title>
    </h:head>
    <h:body>
        <h:outputText value="Hola mundo desde JSF!"/>
    </h:body>
</html>
```

A JSF component object is created for this page.

JSF Application Elements

web.xml

Faces-config.xml

Faces Servlet

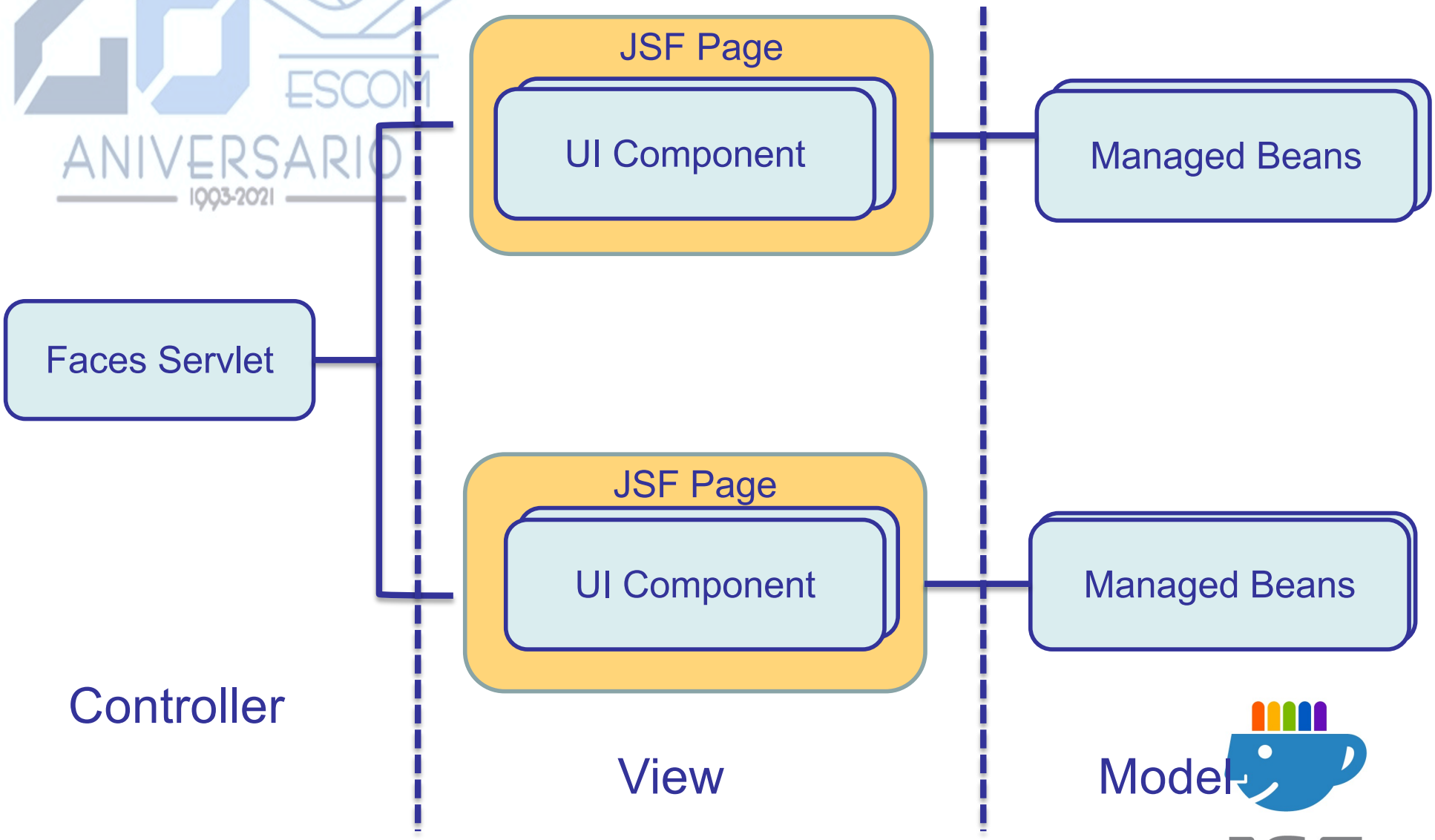
JSF (.xhtml)

Managed Beans (.java)

Model (.java)

libraries

MVC in JSF Applications



MVC in JSF Applications

- Controller.
  - The JSF framework provides a built-in front controller component javax.faces.webapp.FacesServlet, witch manages the JSF request processing life cycle
- View.
  - JSF pages use a View Description Language (VDL) and JSF tag libraries to lay out the user interface. It is important to note that JSF maintains a server-side view of the components represented as the UI view on the client. This state management is one of the most important aspects of JSF.
- Model
  - Application objects, such as CDI beans, provide the data properties and functions.

JSF Forms

- Forms use the JSF HTML tag library.
- The JSF implementation parses form data and makes it available to CDI beans.
- The managed bean properties are associated with form fields through the Expression Language.

```
<h:form id="frmDatos">
    <h:outputLabel
        for="txtNombre"
        value="Escribe tu nombre"/>
    <br/>
    <h:inputText
        id="txtNombre"
        value="#{helloWorld.nombre}"
        required="true"/>
    <h:message for="txtNombre" />
    <br/>
    <h:commandButton id="btnEnviar" value="Enviar"
                     action="#{helloWorld.saludar()}"/>
</h:form>
```

JSF Page Tags

- Facelets have no scriptlets or Java code. For the application developer, tags are all that is needed for views.
- Any Java code will be in a controller or model class.
- Tag libraries are enabled on a page with a namespace declaration:

```
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://xmlns.jcp.org/jsf/html">
```

JSF Tag Libraries

Prefix	Description	Namespace URI http://java.sun.com
h	JSF HTML tags. Very similar to many HTML tags. Use whenever available.	xmlns:h="http://xmlns.jcp.org/jsf/html"
f	Core Faces tags. Primarily modifies or enhances the behavior of the “h” tags.	xmlns:f="http://xmlns.jcp.org/jsf/core"
ui	Facelet Templating tags.	xmlns:ui="http://xmlns.jcp.org/jsf/facelets"

Managed Beans

- Managed beans are JavaBeans objects that encapsulate the application data properties and actions (methods).
  - User action processing logic
  - Data validation logic
- The JSF user interface interacts with the managed beans via the unified Expression Language.

Managed Beans and Forms

- Managed beans are often used to maintain form data and handle form submission.
  - Input form elements are associated with bean properties through the use of the EL.
  - Form input elements have a value attribute:

```
<h:inputText
    id="txtNombre"
    value="#{helloWorld.nombre}"
    required="true"/>
```

Managed Beans and Forms

- JSF uses the setters and getters for bean properties.
  - When displaying a form, the getter is used.
  - When submitting a form, the setter is used.
- Form submission is associated with a Java method using an action attribute:

```
<h:commandButton id="btnEnviar" value="Enviar"
                 action="#{helloWorld.saludar()}"/>
```

JSF Deployment Descriptor

```
<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>
```

The load-on-startup value indicates that the container must load and initialize the servlet as soon as the application is deployed.

The pattern */faces/**. The servlet will use to translate and execute Facelet pages

JSF Application Life Cycle Phases

Create/Restore View	Convert Facelet to the UIComponent tree on initial view. Restore the previous UIComponent tree on postback.
Apply Request Values	Apply data conversion (for example, String->int).
Process Validations	Apply JSF and Bean validation
Update Model Values	Copy UIComponent data to managed beans (via EL).
Invoke Application	Execute action methods.
Render Response	Read data from backing beans (via EL). Generate and send output.

Introduction to Facelets

- Is used to implement a flexible user interface
- Replaces JSP as JSF's default view technology
- Enables true Model-View-Controller (MVC) separation by disallowing the inclusion of Java code in markup pages
- Provides a way to declare a JSF view using standard XHTML syntax
- Provides an extensible tag library

Features of Facelets

- Uses XHTML for building pages
- Supports multiple tag libraries
  - The JSF core and user interface tag libraries
  - The HTML Renderer Kit tag library
  - Subset of the JSTL tag libraries
- Supports unified EL and compile-time EL validation
- Supports templating for UI components and pages
  - Template file gives main layout and defines sections
  - Client file uses template and replaces sections
  - Can pass JSF data to included pages
- Supports composite components
  - Simplified model for creating reusable custom components

JSF Page Evaluation

- When the JSF page is first requested (HTML Request), the following events occur:
  - The UI components form a tree.
  - Each JSF tag identifies a component.
    - The renderer produces the markup (encoding).
  - Non-JSF content is passed through to the renderer.
  - Page is rendered (HTML).
- When the form is posted back (HTML POST), the following events occur:
  - The POST request is received.
  - Each component inspects the request parameters for a chance for decoding data

HTML Tags

- HTML Tags can be grouped in the following categories:
- Inputs (input...)
- Outputs (output..., graphicImage)
- Commands (commandButton and commandLink)
- GET Requests (button, link, outputLink)
- Selections (checkbox, listbox, menu, radio)
- HTML pages (head, body, form, outputStylesheet, outputScript)
- Layouts (panelGrid, panelGroup)
- Data table (dataTable and column)
- Errors and messages (message, messages)

Java Server Faces

M. en C. José Asunción Enríquez Zárate