Java Server Faces
&
PrimeFaces

Anil Joseph

Created By Anil Joseph(anil.jos@gmail.com)

Agenda

Understanding JSF	JSF Features and Architecture	JSF Page Lifecycle	Ajax in JSF	Primefaces
Primefaces Core Components	Primefaces Styles & Themes	Primefaces Data Components	Navigation	Validation, Converters and Error Handling
Ajax Support	Event Handling	Custom UI Components	Composite Components	Configuration and Deployment

Created By Anil Joseph(anil.jos@gmail.com)

Software

JDK 1.7 or higher

Eclipse IDE for Java EE Developers

Apache Tomcat 8.0 or higher

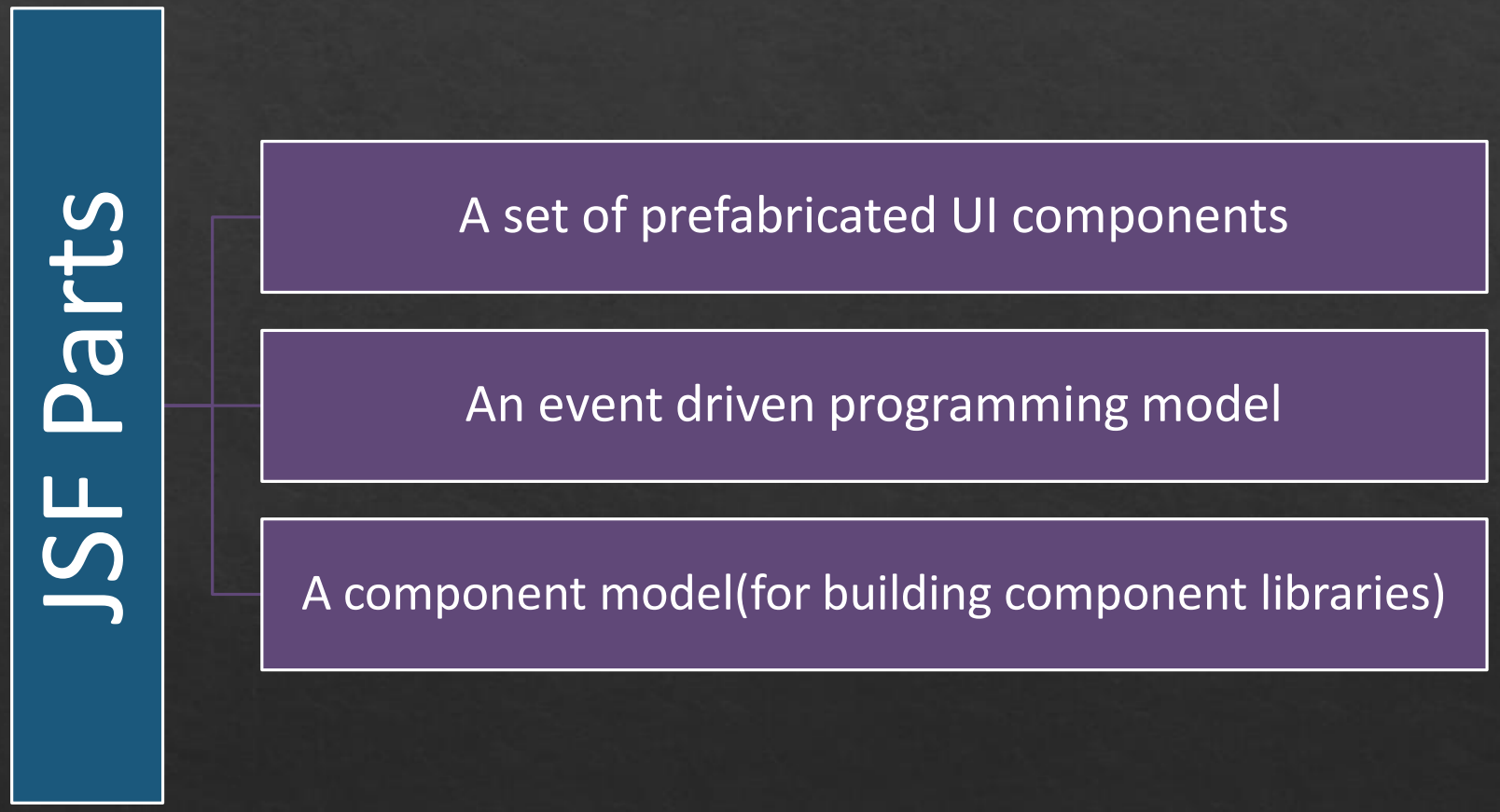
Apache MyFaces (http://myfaces.apache.org/)

PrimeFaces (http://primefaces.org/)

Java Server Faces

- JSF is a ***component-based framework*** for building the user interface of a web application.
- It is the view layer in the Java EE standard.
- Based on the *Model-View-Controller* architecture.

Java Server Faces



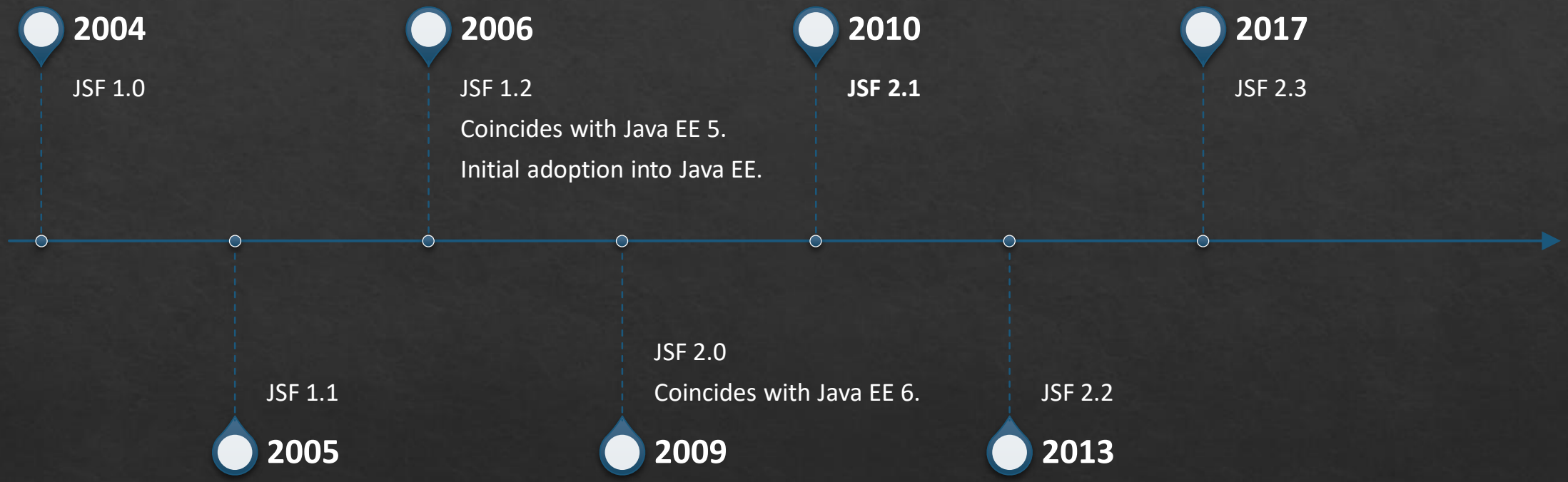
Created By Anil Joseph(anil.jos@gmail.com)

Benefits of JSF

- JSF provides a built-in UI component library, which handles most of the user interface management.
- Offers a clean separation between behavior and presentation.
- Manages the component state
- JSF provides a validation framework
- Robust event handling mechanism.
- Allows rendering of UI to any client format. The default is HTML.
- JSF also supports internationalization and accessibility
- Supports AJAX
- Offers multiple, standardized vendor implementations

Created By Anil Joseph(anil.jos@gmail.com)

JSF Version



Created By Anil Joseph(anil.jos@gmail.com)

JSF Implementations

Oracle Mojarra

- Implementation from Oracle

Apache MyFaces

- Opensource from Apache

PrimeFaces

- Ajax framework with JSF components

ICEfaces

- open-source JSF extension framework

JBoss RichFaces

- Ajax-enabled JSF components for layout, file upload, forms, inputs and many other features.

Created By Anil Joseph(anil.jos@gmail.com)

JSF Components(Based on MC)

FacesServlet

The front controller in JSF that handles all requests and defines the its lifecycle.

Implementation class: ***javax.faces.webapp.FacesServlet***

Views

The presentation layer.

In JSF 1.1 & 1.2, the views are JSP pages

In JSF 2.0 onwards they are facelets(.xhtml) file.

Model

Represents the state

Handles the application logic, Data access, etc.

Created By Anil Joseph(anil.jos@gmail.com)

JSF Components

Managed Beans

- Specialized JavaBeans that collect values from UI components and implement event listener methods.
- They can also hold references to UI components.

UI Components

- A stateful object, maintained on the server, that provides specific functionality for interacting with an end user.
- UI components are JavaBeans with properties, methods, and events.

Created By Anil Joseph(anil.jos@gmail.com)

JSF Components

Renderer	• Responsible for displaying a UI component and translating a user's input into the component's value.
Validators	• Responsible for ensuring that the value entered by a user is acceptable. • One or more validators can be associated with a single UI component.
Converters	• Converts a component's value to and from a string for display. • A UI component can be associated with a single converter.
Events and Listeners	• JSF uses the JavaBeans event/listener model. • UI Components generate events, and listeners can be registered to handle those events.

Created By Anil Joseph(anil.jos@gmail.com)

JSF Components

Messages	• Information that's displayed back to the user.
Navigation Rules	• The ability to move from one page to the next. • JSF has a powerful navigation system that's integrated with specialized event listeners.
Configuration Files	• A XML based file to configure various JSF Features and components

Created By Anil Joseph(anil.jos@gmail.com)

JSF Tag Libraries

Core	Html	Facelets
• *http://java.sun.com/jsf/core* • The core library contains the tags that are independent of HTML rendering. • Validation, Converters, Ajax etc	• *http://java.sun.com/jsf/html* • This library contains the tags to render HTML • Input, Output, table etc	• http://java.sun.com/jsf/facelets

Created By Anil Joseph(anil.jos@gmail.com)

JSF Tag Libraries

Composite components	• http://java.sun.com/jsf/composite
Passthrough	• http://xmlns.jcp.org/jsf/passthrough • HTML5 Support
JSLT Core	• http://java.sun.com/jsp/jstl/core
JSLT Functions	• http://java.sun.com/jsp/jstl/functions

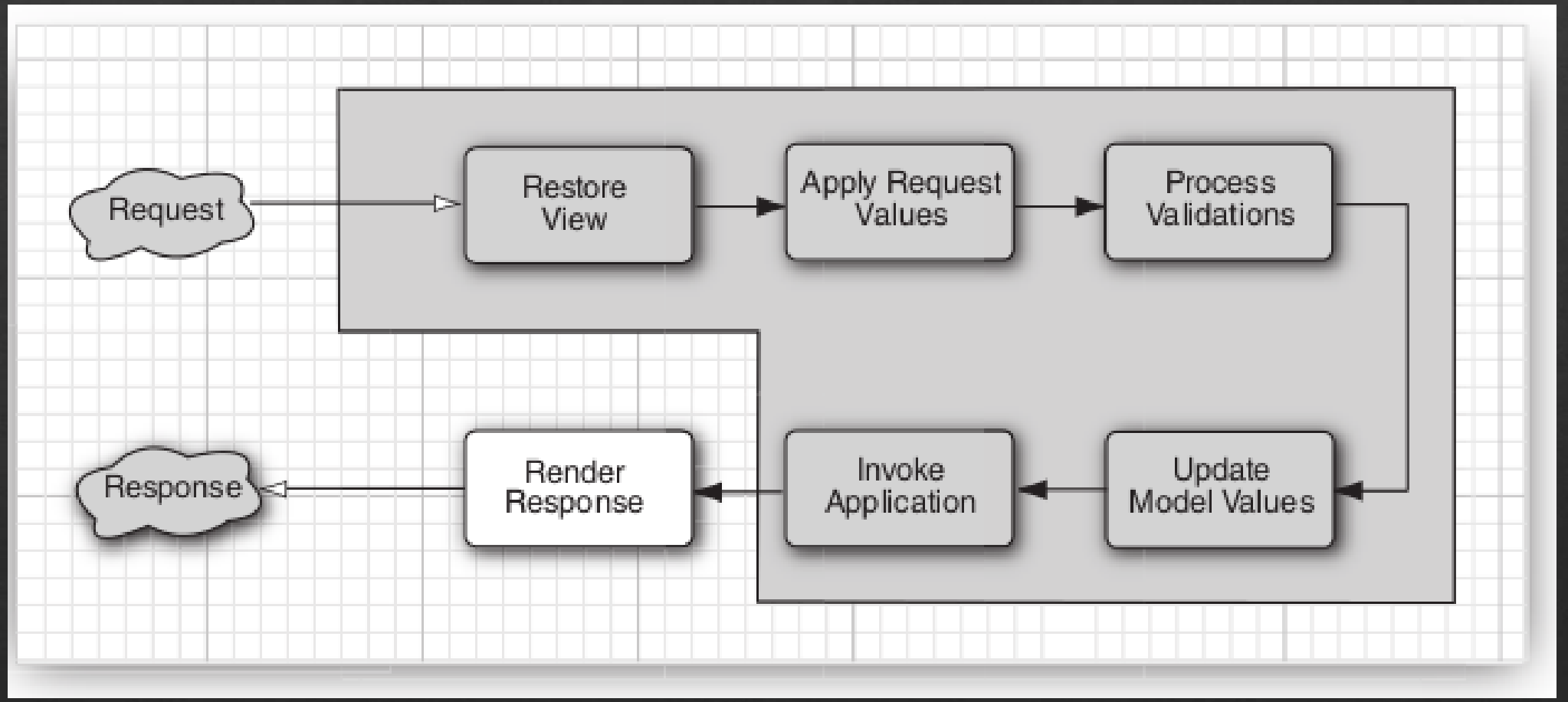
Created By Anil Joseph(anil.jos@gmail.com)

Request Life Cycle

- The lifecycle consists of two main phases
  - **Execute**
  - **Render**
- Execute phase
  - The application view is built or restored.
  - The request parameter values are applied.
  - Conversions and validations are performed for component values.
  - Managed beans are updated with component values.
  - Application logic is invoked.
- Render phase
  - The requested view is rendered as a response to the client.

Created By Anil Joseph(anil.jos@gmail.com)

Lifecycle Stages



Created By Anil Joseph(anil.jos@gmail.com)

FacesContext

- The FacesContext holds all contextual information necessary for processing a request and generating a response.
- Accessing Context
  - FacesContext.getCurrentInstance()

Created By Anil Joseph(anil.jos@gmail.com)

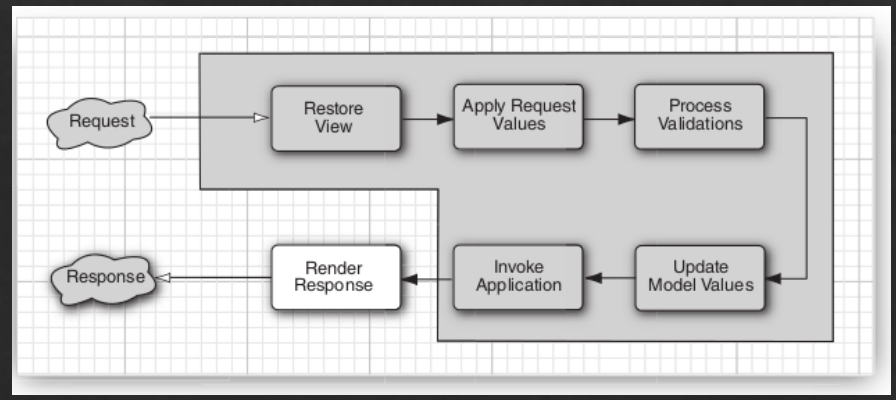
FacesContext

- Holds the current component tree and application configuration objects.
- Provides generic access to the external context.
- Allows to alter the sequential execution of life-cycle phases.
  - renderResponse
  - renderComplete
- Gives access to locale of the current client request.
- Holds a queue of messages(error) during the life-cycle.

Created By Anil Joseph(anil.jos@gmail.com)

Restore View

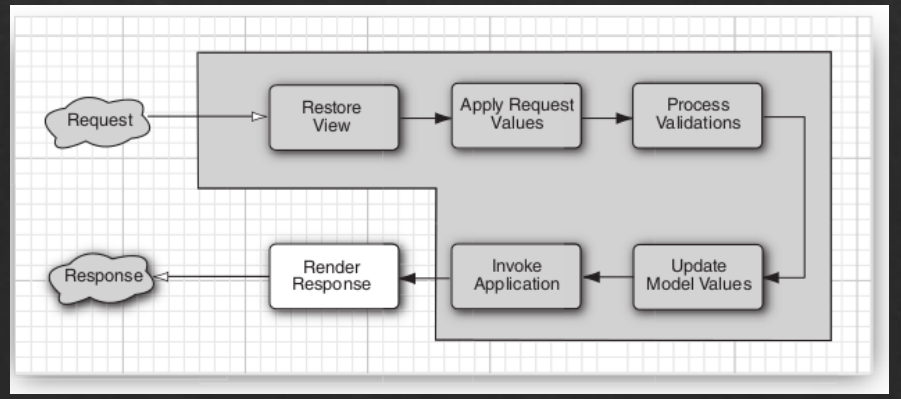
- During this phase, JSF builds the view of the page, wires event handlers and validators to components in the view, and saves the view in the ***FacesContext*** instance
- If the request for the page is an ***initial request***, the lifecycle advances to the ***Render Response phase***.
- If the request for the page is a ***postback***, the view is restored from the FacesContext and proceeds to the ***Apply Request Values*** stage.



Created By Anil Joseph(anil.jos@gmail.com)

Apply Request Values

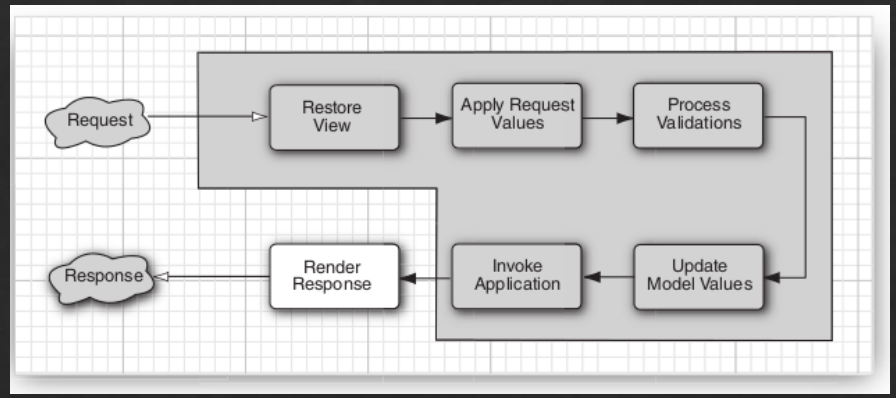
- In this stage each component in the tree extracts its new value from the request parameters.
  - The method invoked is ***processDecodes***()
  - The mechanism is called decoding
  - The value is then stored locally on each component.
- If any decode methods or event listeners invokes ***renderResponse***, then it proceeds to the ***Render Response phase***.
- If a components has set the immediate attribute to true, the it processes the
  - Validations
  - Conversions
  - Events



Created By Anil Joseph(anil.jos@gmail.com)

Process Validations

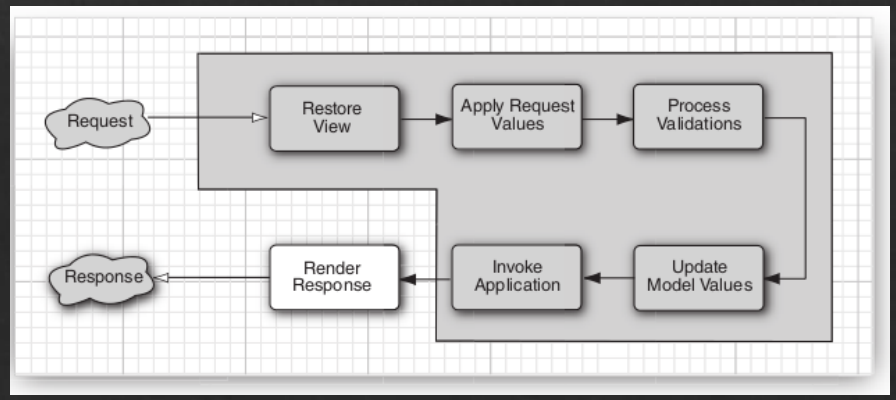
- Processes all validators registered on the components in the tree.
  - The method invoked is ***processValidators***.
- If a validation fails
  - An error messages is added to the FacesContext
  - Lifecycle advances directly to the ***Render Response phase***
  - In effect the same page with error messages would be displayed.



Created By Anil Joseph(anil.jos@gmail.com)

Update Model Values

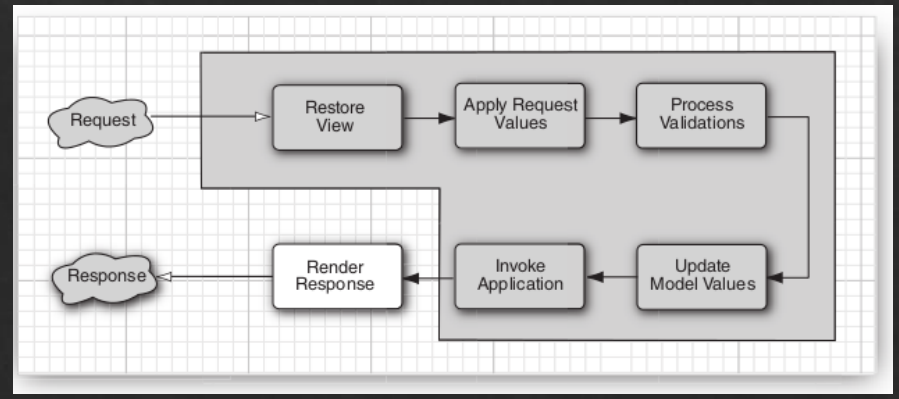
- In this stage, JSF sets the corresponding server-side object properties to the components' local values.
- If the local data cannot be converted to the types specified by the bean properties, the lifecycle proceeds to the ***Render Response phase.***



Created By Anil Joseph(anil.jos@gmail.com)

Invoke Application

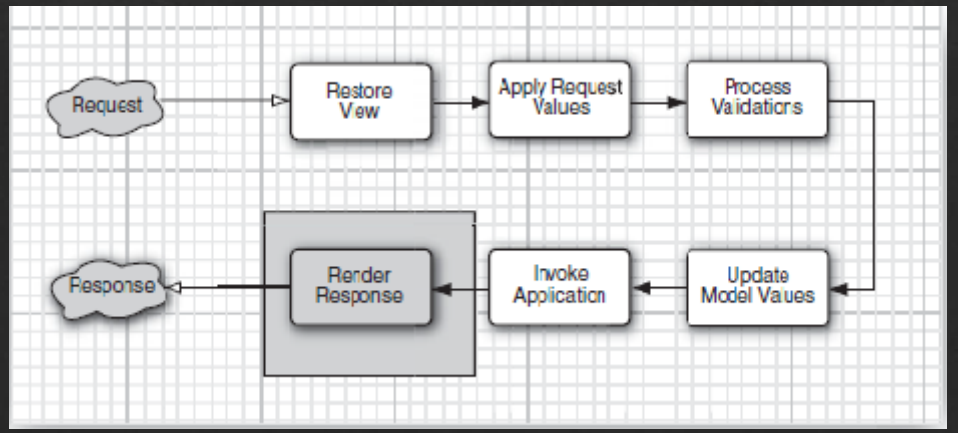
- During this phase, the JSF handles any application-level events, such as submitting a form or linking to another page.
- Finally, the JSF transfers control to the ***Render Response phase***.



Created By Anil Joseph(anil.jos@gmail.com)

Render Response

- During this phase, JSF builds the view and delegates authority to the appropriate resource for rendering the pages.
- In case of an initial request
  - the components that are represented on the page will be added to the component tree
- In case of a postback request
  - the components are already added to the tree.
- Finally the state of the response is saved so that subsequent requests can access it.



Created By Anil Joseph(anil.jos@gmail.com)

Managed Beans

- Managed Beans store the state of web pages.
- Bean creation and manipulation is under the control of the JSF implementation
- JSF does the following
  - Creates and discards beans as needed (hence the term “managed beans”)
  - Reads bean properties when displaying a web page
  - Sets bean properties when a form is posted.

Created By Anil Joseph(anil.jos@gmail.com)

Bean Scopes

Scopes define the lifetime of a bean.

Application Scope	@ApplicationScope Available for the lifetime of the application.
Session Scope	@SessionScope Available for a session(user).
Request Scope	@RequestScope Available for a single request.
View Scope	@ViewScope Available while the same JSF page is redisplayed Useful for Ajax applications
Custom Scope	@CustomScope Application defined scope

Created By Anil Joseph(anil.jos@gmail.com)

CDI Beans

- JSF pioneered the concept of “managed beans” in web applications.
- CDI(“Contexts and Dependency Injection”) defines a more flexible model for beans.
- CDI beans are managed by the application server.
- CDI specifies mechanisms for
  - injecting beans, intercepting
  - decorating method calls
  - firing and observing events
- Recommended to use CDI beans, if the application is hosted in a application server.

Created By Anil Joseph(anil.jos@gmail.com)

Facelets

- Facelets is a powerful but lightweight page declaration language that is used to build JSF 2.0 views
- Facelets are the default view technology of JSF 2.0.
- It provides a better and easy view technology than JSP's.
- Based on pure XML templates.

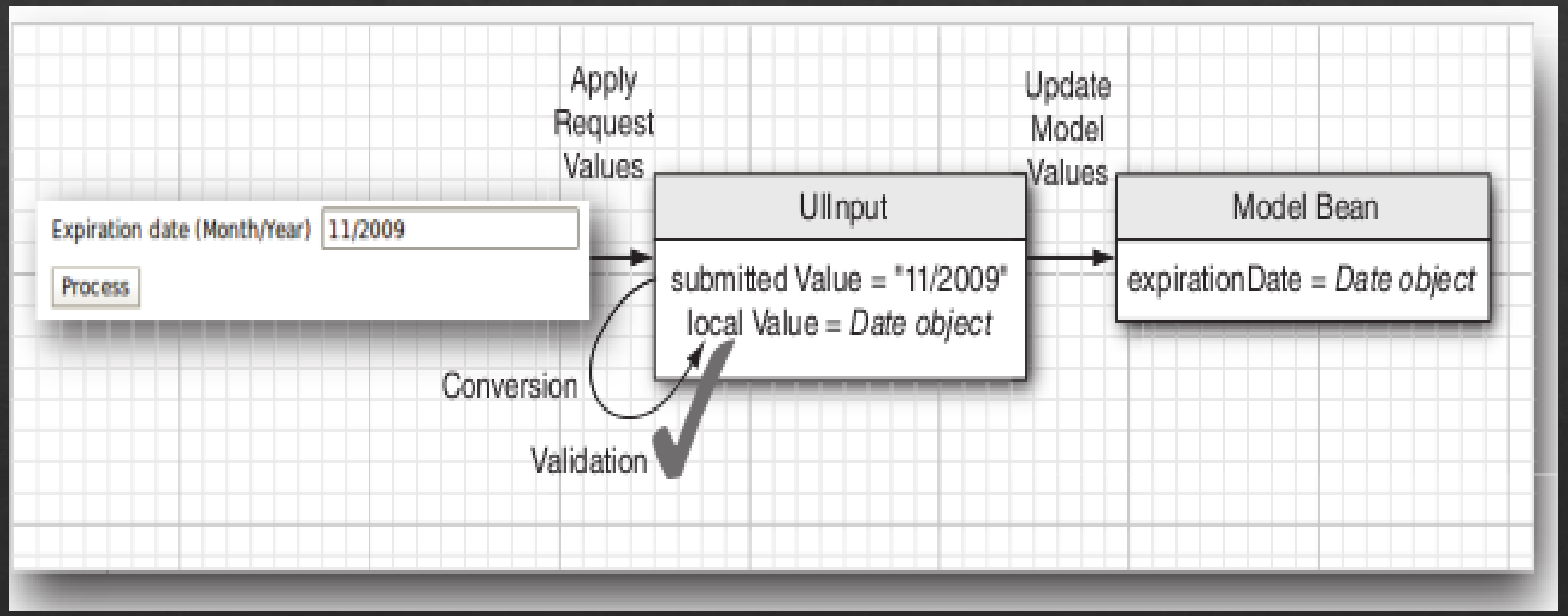
Created By Anil Joseph(anil.jos@gmail.com)

Converters & Validators

- A *conversion* process transforms the incoming strings to the arbitrary types of the web application.
- The validation process verifies the data submitted against rules defined by the application.
- Conversions & Validations are executed at the ***Process Validations*** phase.

Created By Anil Joseph(anil.jos@gmail.com)

Converters & Validators



Created By Anil Joseph(anil.jos@gmail.com)

Standard Converters

- f:convertNumber
  - Converts strings to numbers and vice-versa
  - Attributes
    - type: *number (default), currency, or percent*
    - maxFractionDigits, minFractionDigits
    - integerOnly
    - currencyCode: *USD, EUR etc*
    - currencySymbol
    - groupingUsed: *true, false*
- f:convertDateTime
  - Converts strings to datetime and vice-versa
  - Attributes
    - type: date (default), time, or both
    - pattern: Formatting pattern, as defined in ***java.text.SimpleDateFormat***

Created By Anil Joseph(anil.jos@gmail.com)

Standard Validators

- f:validateDoubleRange
  - DoubleRangeValidator
  - A double value within an optional range
- f:validateLongRange
  - LongRangeValidator
  - A long value within an optional range
- f:validateLength
  - LengthValidator
  - A String with a minimum and maximum number of characters

Created By Anil Joseph(anil.jos@gmail.com)

Standard Validators

- f:validateRequired
  - RequiredValidator
  - The presence of a value
- f:validateRegex
  - RegexValidator pattern
  - A String against a regular expression
- f:validateBean
  - BeanValidator
  - Specifies validation groups for bean validators
  - JSR 303

Created By Anil Joseph(anil.jos@gmail.com)

Bean Validation(JSR 303)

- @Null, @NotNull
  - Check that a value is null or not null.
- @Min, @Max
  - Check that a value is at least or at most the given bound.
  - The type must be one of int, long, short, byte and their wrappers, BigInteger, BigDecimal.
- @DecimalMin, @DecimalMax
- @Digits
  - Check that a value has, at most, the given number of integer or fractional digits.
  - Applies to int, long, short, byte and their wrappers, BigInteger, BigDecimal, String.

Created By Anil Joseph(anil.jos@gmail.com)

Bean Validation(JSR 303)

- @AssertTrue, @AssertFalse
  - Check that a Boolean value is true or false.
- @Past, @Future
  - Check that a date is in the past or in the future.
- @Size
  - Check that the size of a string, array collection, or map is at least or at most the given bound.
- @Pattern
  - A regular expression and optional compilation flags.

Created By Anil Joseph(anil.jos@gmail.com)

Custom Converters & Validators

- A converter must implement the ***javax.faces.convert.Converter*** interface.
- The ***Converter*** interface has two methods:
  - Object getAsObject(FacesContext context, UIComponent component, String newValue)
  - String getAsString(FacesContext context, UIComponent component, Object value)
- ConverterException to be thrown, if conversion fails
- @FacesConverter annotation to configure a class as a converter implementation.

Created By Anil Joseph(anil.jos@gmail.com)

Custom Converters & Validators

- A validator must implement the interface ***javax.faces.validator.Validator*** interface.
- The interface has a single method
  - void validate(FacesContext context, UIComponent component, Object value)
- On the event of validation errors, throw the ***ValidatorException***.
- @FacesValidator annotation to configure a class as a validator implementation.

Created By Anil Joseph(anil.jos@gmail.com)

AJAX

- JSF 2.0 integrates Ajax into its lifecycle.
- This integration lets us handle Ajax requests in the same manner as other requests.
- Behaviors like conversions, validations, events handling work in the same way.

Created By Anil Joseph(anil.jos@gmail.com)

AJAX

- The JSF 6-phase life cycle is divided into two parts
  - E**xecute**
  - **Render**.
- During the execute part, for every component on the server it
  - Converts and validates the component's value (if the component is an input)
  - Pushes valid input values to the model (if the component is wired to a bean property)
  - Executes actions and action listeners (if the component is an action):
- The render part of the life cycle
  - Renders components on the client

Created By Anil Joseph(anil.jos@gmail.com)

JSF Ajax

- To use Ajax with JSF, consider the following
  - Associate a component and an event with an Ajax request.
  - Identify components to execute on the server during the Ajax request.
  - Identify components to render after the Ajax request(once the response is generated).

Created By Anil Joseph(anil.jos@gmail.com)

Ajax Tag

- Ajax is part of the core library.
  - xmlns:f=*http://java.sun.com/jsf/core*
- f:ajax
  - The tag adds ajax behavior to a JSF component.
- Attributes
  - event
    - The event that triggers the Ajax request
  - execute
    - A space-separated list of components that JSF executes on the server during the Ajax call.
  - render
    - A space-separated list of components that JSF renders on the client after the Ajax call returns from the server.
  - onerror
    - A JavaScript function that JSF calls if the Ajax call results in an error.

Created By Anil Joseph(anil.jos@gmail.com)

PrimeFaces

- PrimeFaces is an open source JSF component suite with various extensions.
- Rich set of components (HtmlEditor, Dialog, AutoComplete, Charts and many more).
- Built-in Ajax based on standard JSF 2.0 Ajax APIs.
- Lightweight, one jar, zero-configuration.
- Ajax Push support via websockets.
- Mobile UI kit to create mobile web applications for handheld devices.
- Skinning Framework with 35+ built-in themes and support for visual theme designer tool.

Created By Anil Joseph(anil.jos@gmail.com)

Setup

- PrimeFaces only requires a JAVA 5+ runtime and a JSF 2.x implementation as mandatory dependencies.
- Single jar file to be added to the web application.
  - primefaces-6.0.jar
  - primefaces-{version}.jar
- PrimeFaces namespace is *http://primefaces.org/ui*
  - xmlns:p=*"http://primefaces.org/ui"*

Created By Anil Joseph(anil.jos@gmail.com)