JSF Components

Components of JSF Framework

As a review, JSF is comprised of:

- Managed-beans
  - Provide the MODEL functionality
  - Can be any “Java Bean” the faces JSF tag library
- JSF Tab Libraries
  - Used to develop the VIEW in JSF
  - Two primary libraries:
    - JSF Core – View
    - JSF HTML – UI Components
- FacesServlet
  - Provides CONTROLLER functionality
  - Contains well-defined lifecycle

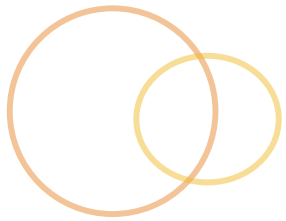
Managed Beans

- Function as model for JSF
  - Lifecycle managed by container
  - Declared in faces-config.xml file
- Have differing scopes
  - Request - life of HTTP Request (created for every request)
  - Session - life of the session (created once during session)
  - Application - life of application (created once during application, shared across sessions)
  - None - not associated with scope; temporary object

Managed Bean Bindings

- Managed bean properties bound to JSF UIComponents
  - JSF automatically get/set property data
  - JSF performs conversion using reflection and convertors
- Support most type of property types
  - Primitives, Booleans
  - Indexed properties through String[]
- If more complex type, need to assign a converter
  - Prebuilt convertors found in core tag library
  - Can create own convertors

JSF Core Tag Library



- Core tag provides customization of
  - User Interface Component tree
  - FacesServlet interactions with UI
- Included using taglib directive
  - <%@ taglib prefix="f" uri="http://java.sun.com/jsf/core" %>
  - Required to have a valid faces view

User Interface Component Tree

- A VIEW in JSF is represented in a component tree
  - Similar to the container-component structure of Swing
  - A component tree is associated with every request and response
  - Component tree is built as part of FacesServlet when request processing begins
- All JSF component trees require a top-level container
  - Component tree is modeled as a UIViewRoot
  - Contains a set of UIComponents

<f:view>

- Represents faces view component (UIViewRoot)
- One <f:view /> per JSP
- Can have subviews
  - Function as sub-containers of components
  - <f:subview> … </f:subview>
- Identified by:
  - A view id
  - Render kit
  - Locale

Core Library and FacesServlet

- FacesServlet responsible for making JSF “framework” work
- Core Library extends capabilities in three main areas:
  - Validation
  - Conversion
  - Event handling

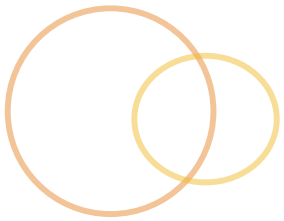
JSF HTML Tag Library

- Represents HTML User Interface components
  - Attach HTML renderers to standard UIComponent classes
  - Referenced within a <f:view> . . . </f:view>
- Each component defined in terms of:
  - Component type:
    - Input, Output, Select
    - Command
  - Render Type:
    - Text, TextArea, SelectOne
    - Button, Link

JSF HTML Components

- Components follow <h:componentTypeRenderType> format
  - <h:commandButton>
  - <h:commandLink>
  - <h:selectManyCheckbox>
  - Etc.
- Use XML syntax
  - Well-formed (open and close tags)
  - Configured through attributes
  - Many provide pass-through configuration to HTML counterpart

JSF HTML Forms

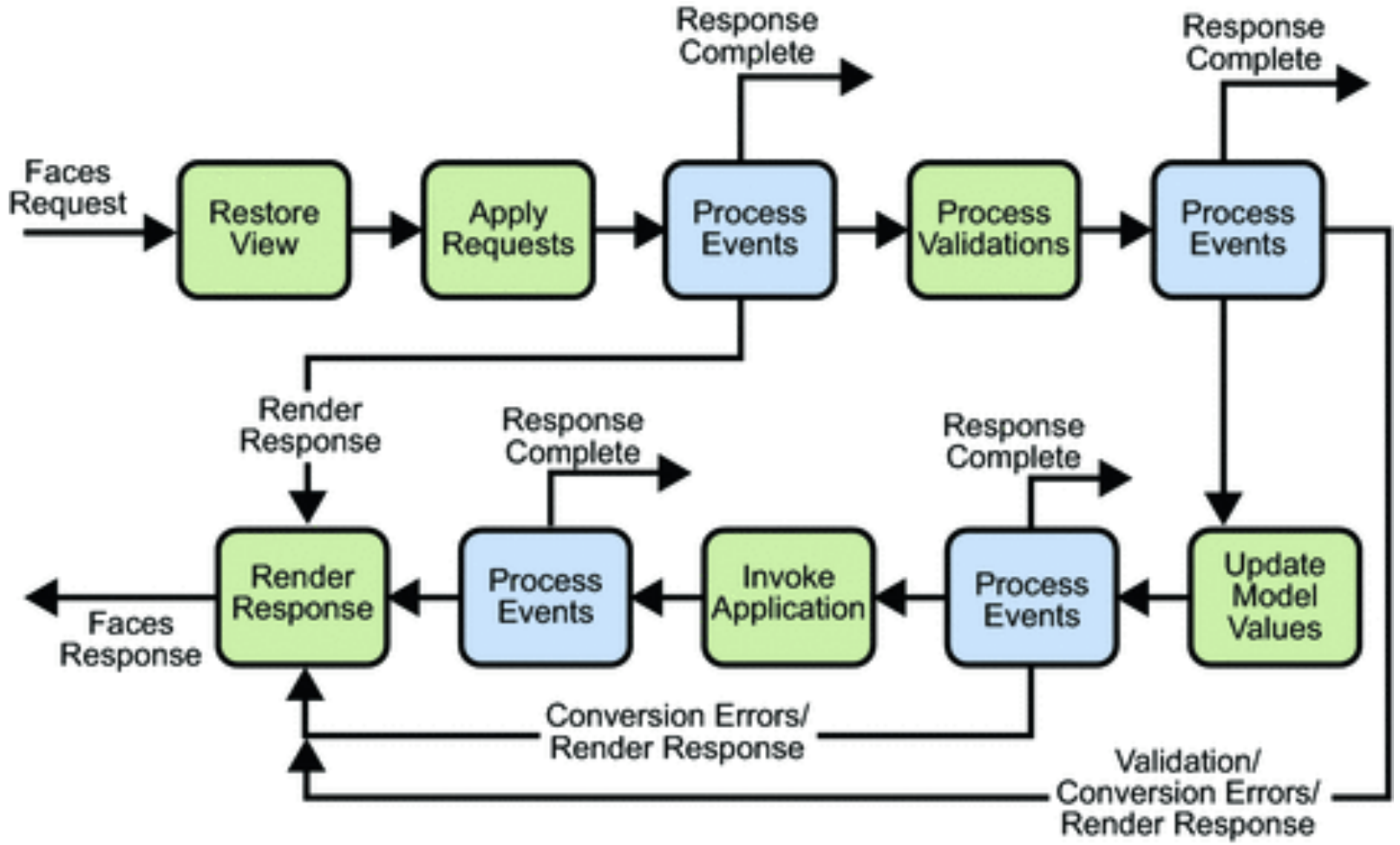


- <h:form> … </h:form>
  - JSF equivalent to HTML form
  - Contains form data; container for input components
    - Can be used to render values
    - Can be used to capture values
  - Action associated with command not form
- All forms sent using POST
  - Form data sent using a “command” component
  - <h:commandButton />
  - <h:commandLink />

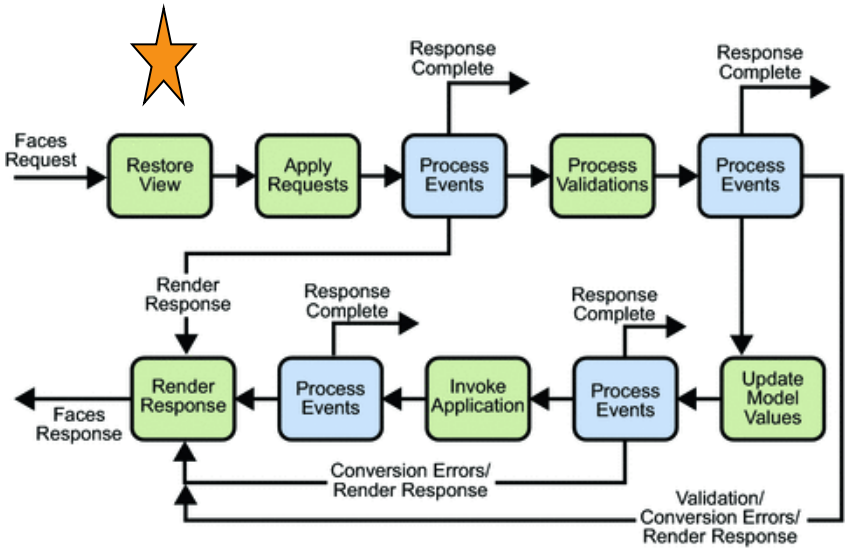
FacesServlet

- Functions as the front-controller in JSF
  - Configurable through faces-config.xml
  - Also supports extension through subclassing
- Associates a FacesContext with each request
  - Contains all per-request state data related to processing JSF request
  - Programmatically accessible by calling: FacesContext.getCurrentInstance()
  - Provides access to request, session, application contexts through getExternalContext() call
- Executes a specific lifecycle

FacesServlet Lifecycle

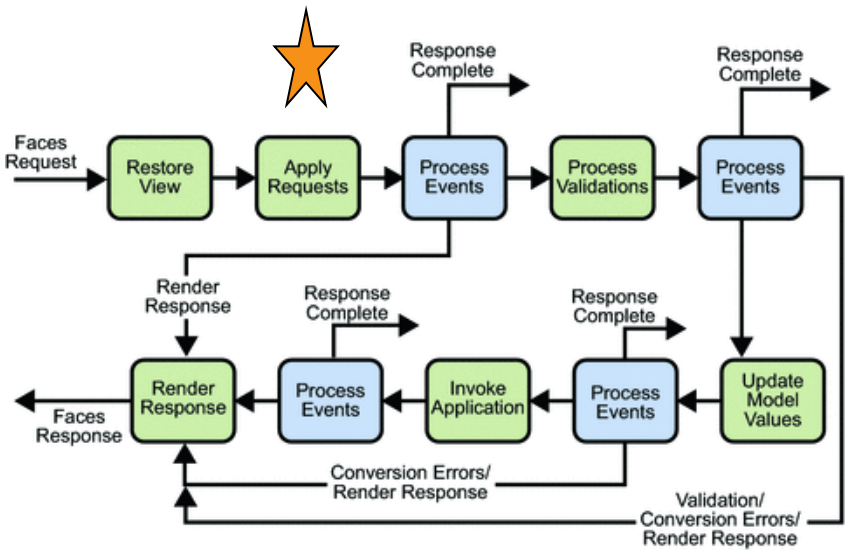


JSF Lifecycle : Restore View



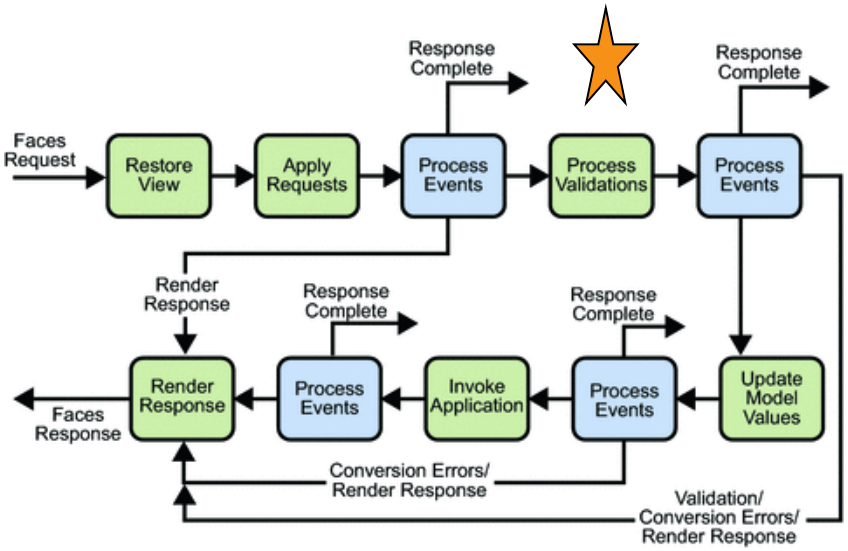
- Builds (or rebuilds) component view of a JSF page
- Associates event listeners, validators, etc. to UI components
- Saves the view in the FacesContext

JSF Lifecycle : Apply Requests



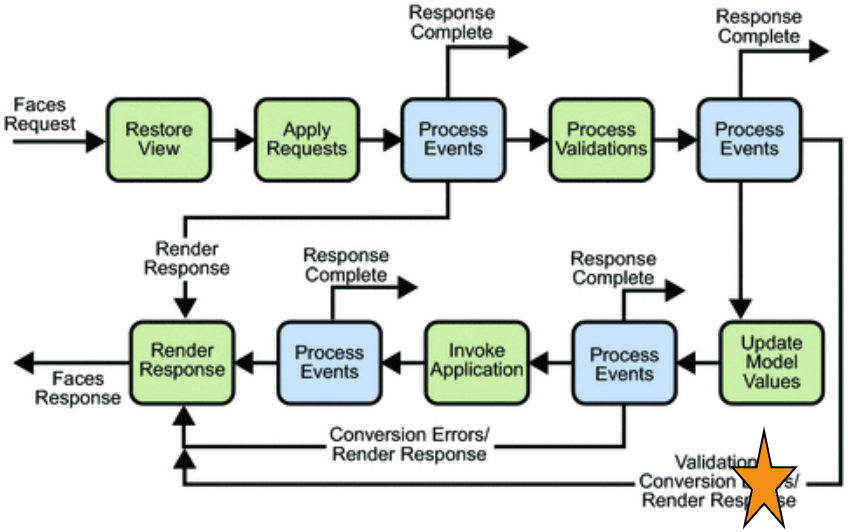
- Component “state” data needs to be updated with request parameter data
- Updating performed using a decode operation
- Performs data conversion

JSF Lifecycle : Process Validators



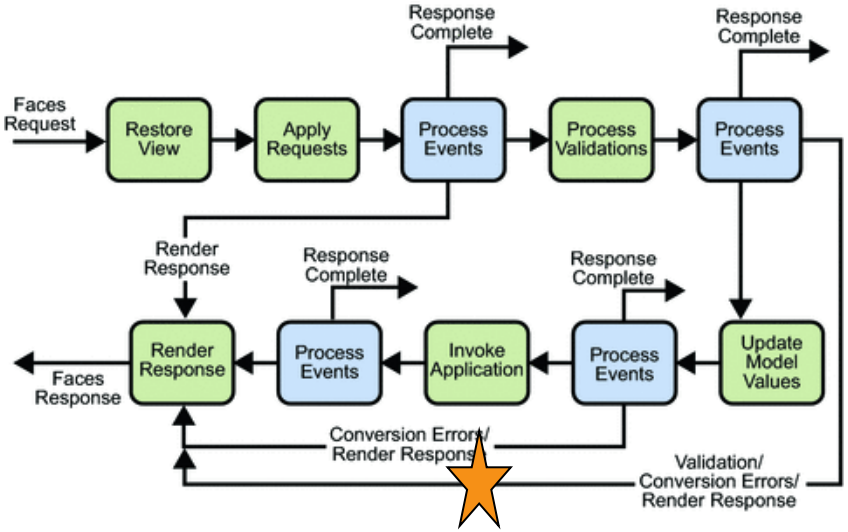
- New data needs to be validated before model is updated
- Validation occurs for where required
- Not all components may have an associated validator

JSF Lifecycle : Update Model



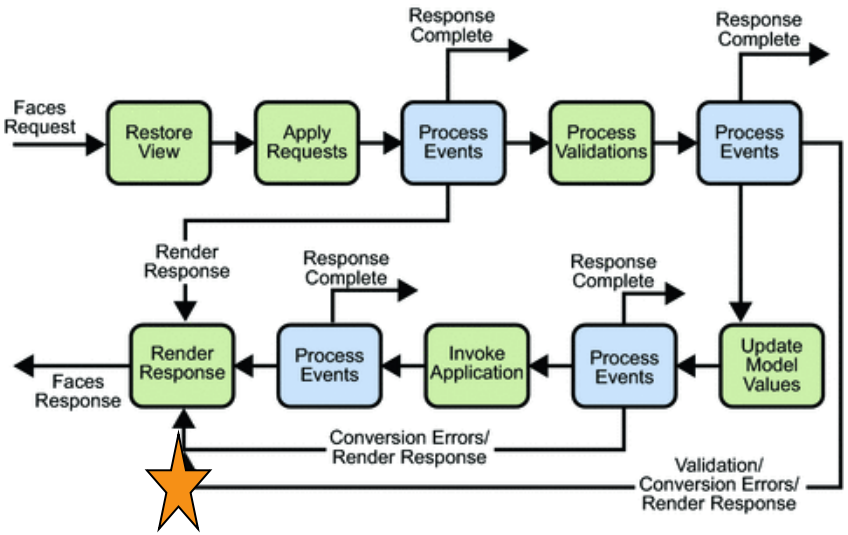
- Converted and validated data needs to be applied to model
- Data applied using set calls
- Only data associated with input components are updated

JSF Lifecycle : Invoke Application



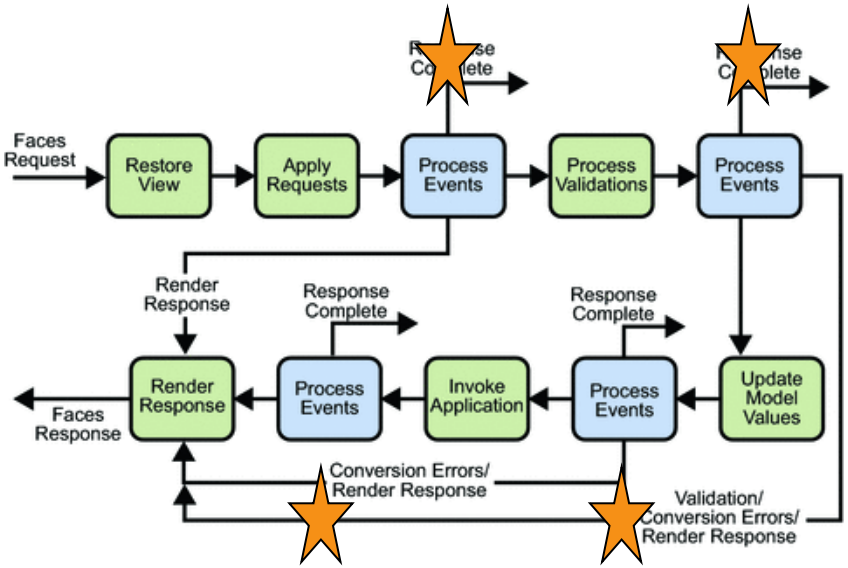
- Execute “controller” logic associated with the command components
- Calls “action” method

JSF Lifecycle : Render Response



- Delegates rendering of response to JSP
- UI tag libraries and EL are converted into HTML

JSF Lifecycle : Process Events

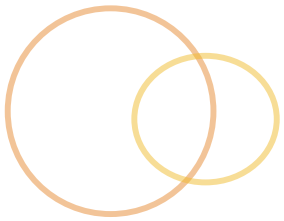


- JSF framework generates events for each phase of lifecycle
- Events are generated before a phase begins and after it completes
- Different phases have associated events
  - ActionEvent
  - ValueChangeEvent
- Event handling may effect subsequent flow

About DevelopIntelligence

- Founded in 2003
- Provides outsourced services to learning organizations in area of software development
- Represents over 35 years of combined experience, enabling software development community through educational and performance services
- Represents over 50 years of combined software development experience
- Delivered training to over 40,000 developers worldwide

Areas of Expertise



- Instruction
  - Java
  - J2EE
  - WebServices / SOA
  - Web Application Development
  - Database Development
  - Open Source Frameworks
  - Application Servers
- Courseware
  - Java Application Development
  - Java Web App Development
  - Enterprise Java Development
  - OOAD / UML
  - IT Managerial
  - Emerging Technologies and Frameworks

Contact Us

- For more information about our services, please contact us:
  - Kelby Zorgdrager
  - Kelby@DevelopIntelligence.com
  - 303-395-5340