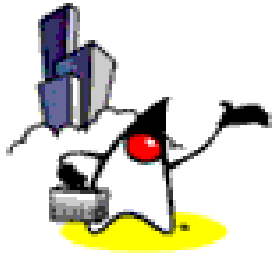
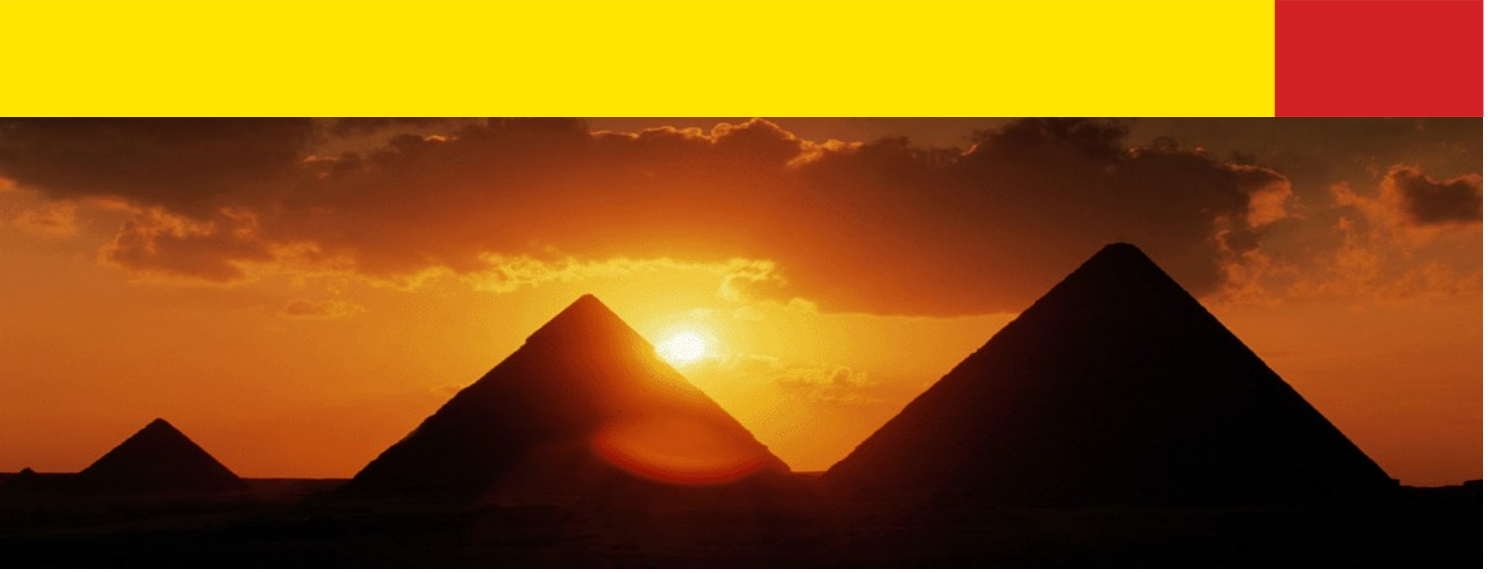
JavaServer Faces (JSF) Overview

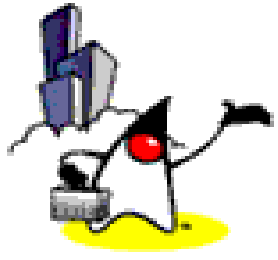


Topics

- Evolution of Web Application Framework
- What is and why JSF?
- JSF design goals
- Quick overview on JSF architecture, concepts, and features
- Developer roles (in Web app development)
- Managed Beans
- Page navigation
- JSF Echo system
- JSF components
- JSF 2.0

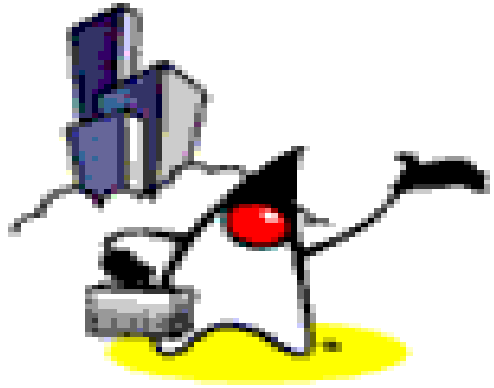


Evolution of Web Application Design Architecture



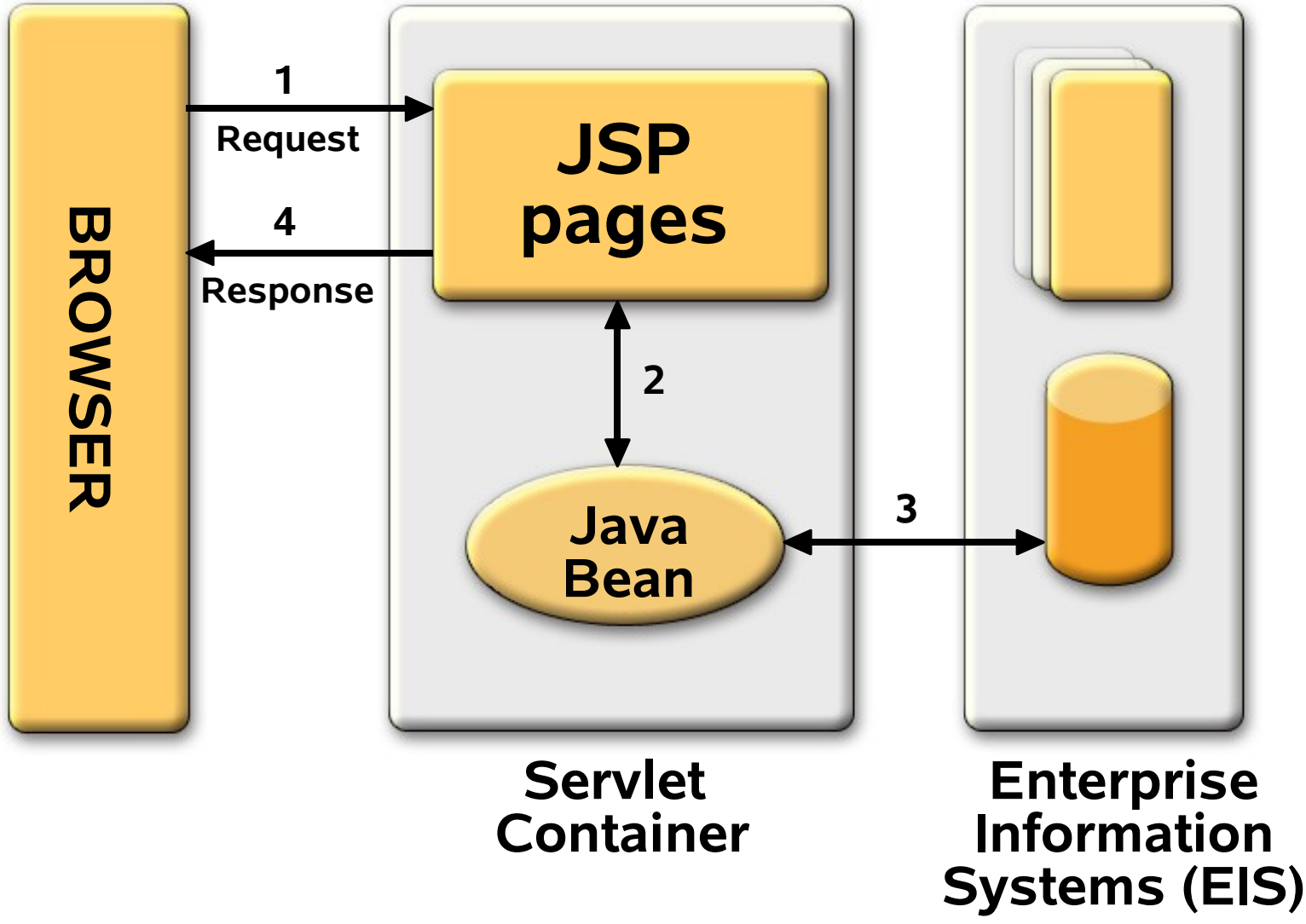
Evolution of MVC Architecture

1. No MVC
2. MVC Model 1 (Page-centric)
3. MVC Model 2 (Servlet-centric)
4. Web application frameworks
   - Struts
5. Standard-based and component-based Web application framework
   - JavaServer Faces (JSR-127)

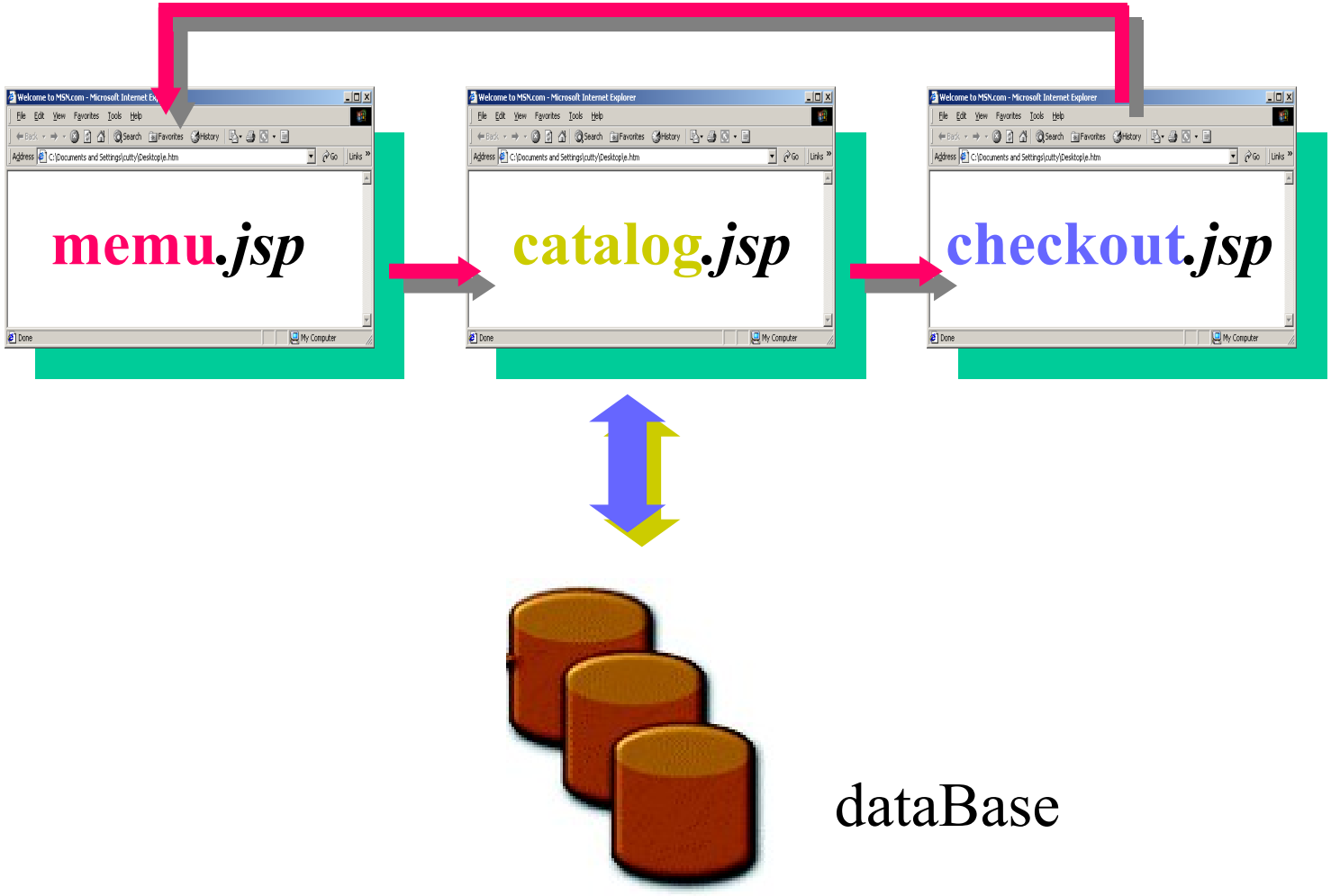


Model 1 (Page-Centric Architecture)

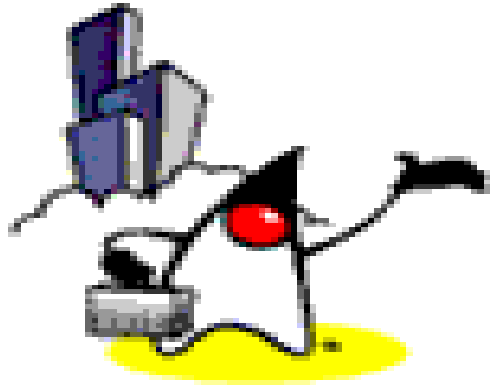
Model 1 Architecture (Page-centric)



Page-centric Architecture

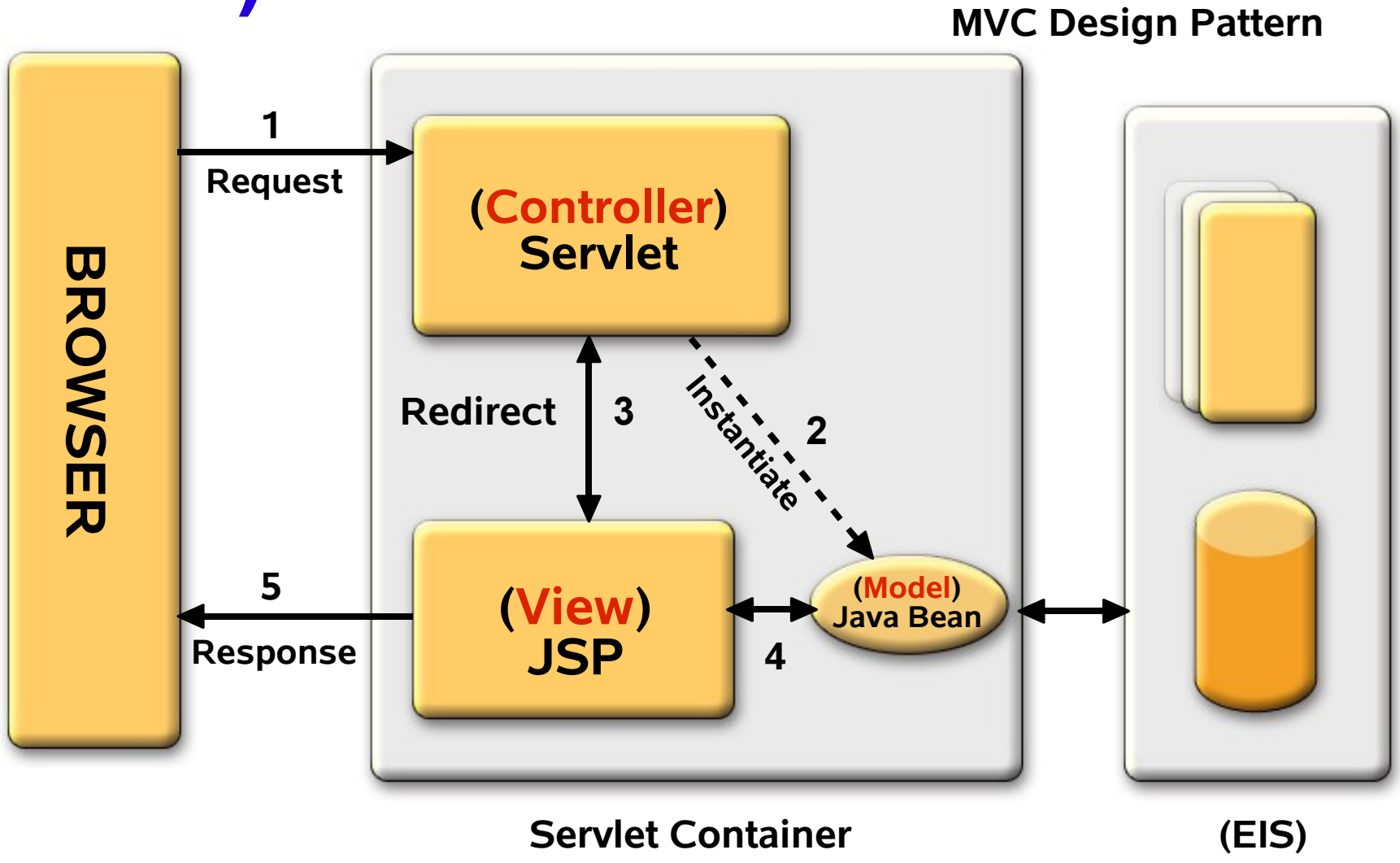


page–centric catalog application



Model 2 (Servlet-Centric Architecture)

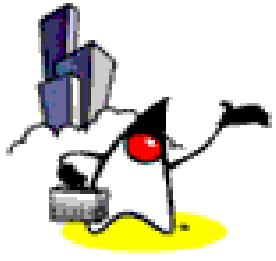
Model 2 Architecture (Servlet-centric)



Why Model 2 Architecture (instead of Model 1)?

- What if you want to present different JSP pages depending on the data you receive?
  - JSP technology alone even with JavaBeans and custom tags (Model 1) cannot handle it well
- Solution
  - Use Servlet and JSP together (Model 2)
  - Servlet handles initial request, partially process the data, set up beans, then forward the results to one of a number of different JSP pages

What is & Why JSF?



JavaServer™ Faces (JSF) Framework Is…

A server-side user interface (UI) component framework for Java™ technology-based web applications.

Drag-and-drop UI components to build a web Application.

What is JSF?

- Next generation Web application framework based on component model
  - UI Components
  - Events-based interaction model
  - Back-end-data integration
- Designed to be leveraged by tools (as opposed to be used by developers directly)
  - Example: NetBeans Visual Web Pack (VWP), ICEfaces
  - You can still create JSF application by writing JSP pages yourself

Why JSF?

- Higher abstraction for Web application development
  - Event-driven programming model (as opposed to HTTP request/response programming model)
- MVC for web applications
- Extensible Component and Rendering architecture
  - Support for client device independence
- Standard
- Huge vendor and industry support

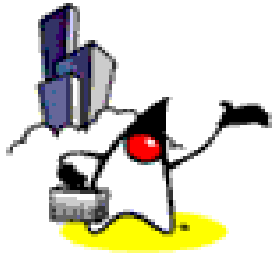
Why JSF? (Continued)

- Offers finer-grained separation of behavior and presentation than JSP
  - Component-specific event handling
  - UI elements as stateful objects on the server
- UI-component and Web-tier concepts without limiting you to a particular view technology (markup language)
  - Can work with any presentation technology including JSP
  - Facelet is getting popular

Why JSF?

- JSP and Servlet
  - No built-in UI component model
- A few words on Struts first
  - I am not saying you should not use Struts
  - Struts and JSF can be used together
- Struts
  - No built-in UI component model
  - No built-in event model for UI components
  - No built-in state management for UI components
  - No built-in support of multiple renderers (Struts is more or less tied up with HTML)

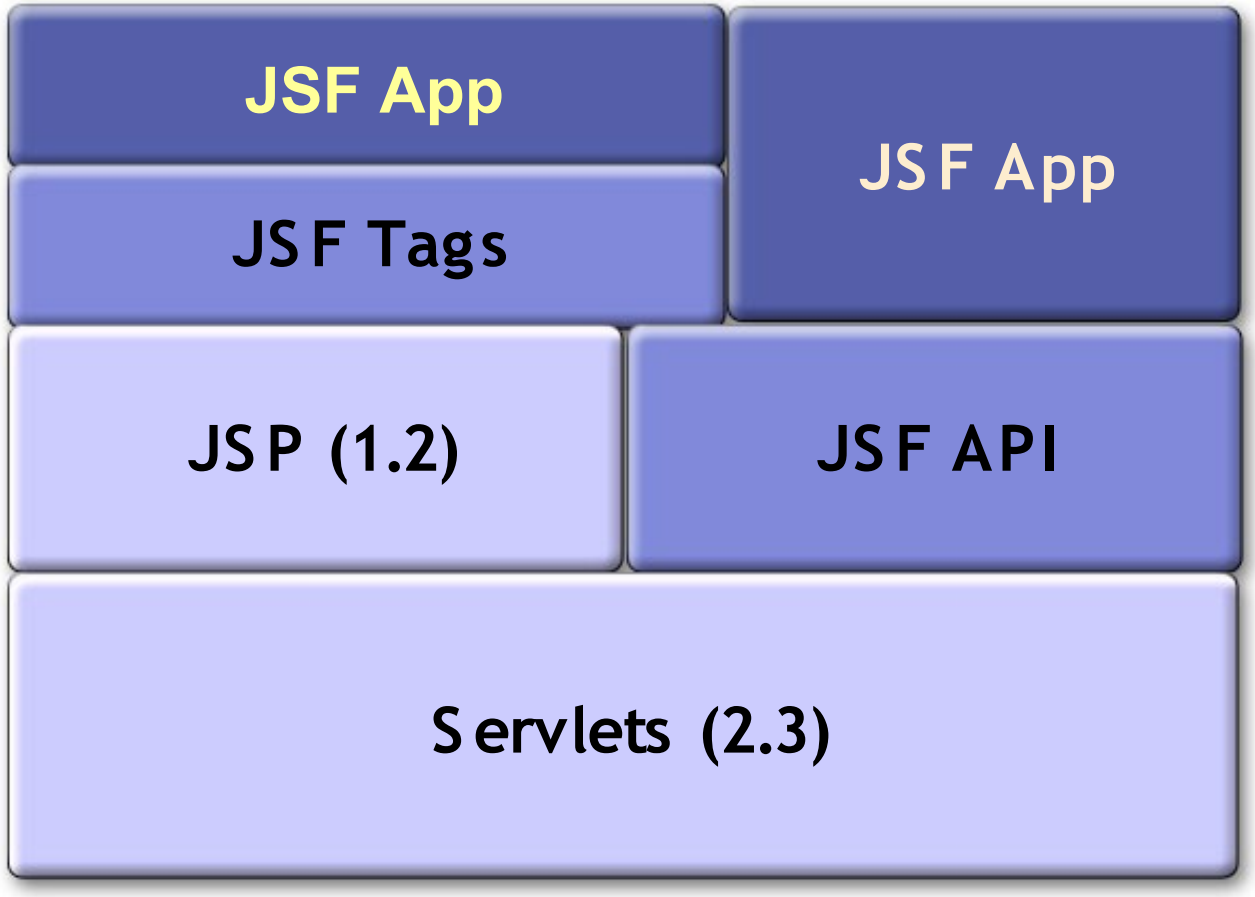
JSF Design Goals

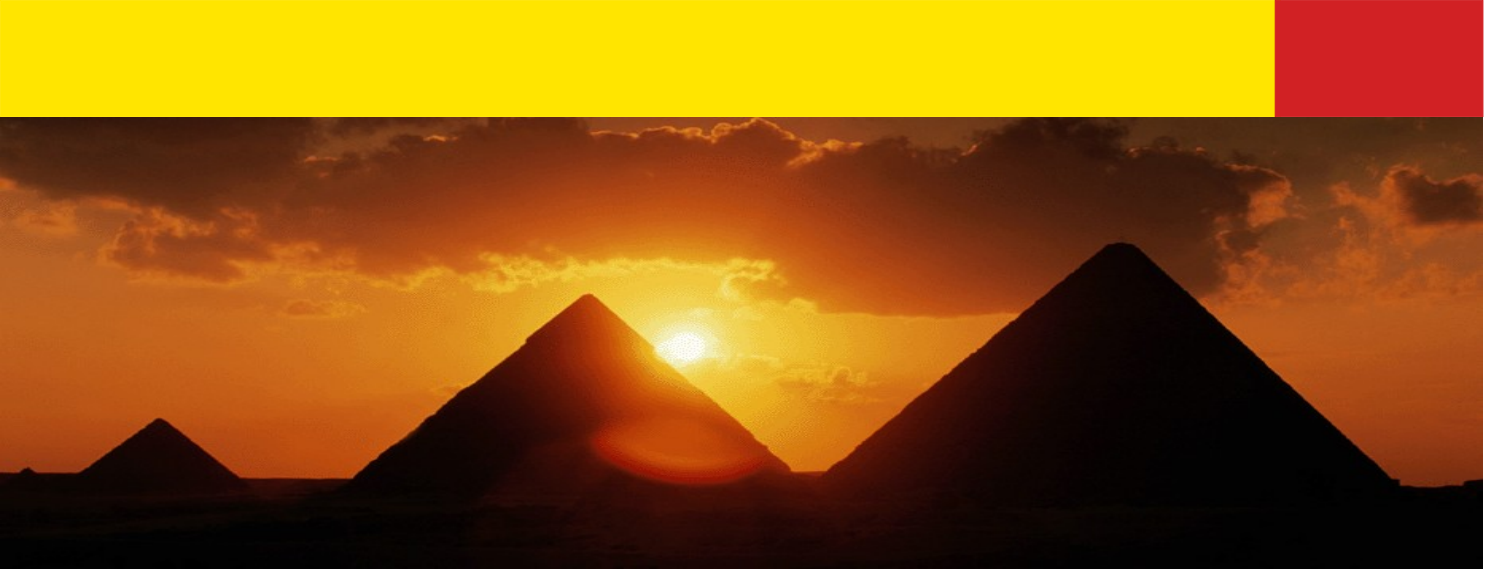


JavaServer Faces Must Be ...

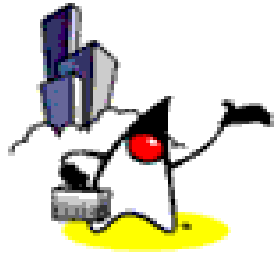
- Tool friendly
- Client device / protocol neutral
- Usable with JavaServer Pages (JSP)
- Usable without JSP
- Useful in the context of HTML and today's browsers
- Extensible
  - Facelets, Seam, etc.

How the JSF Specification Fits In

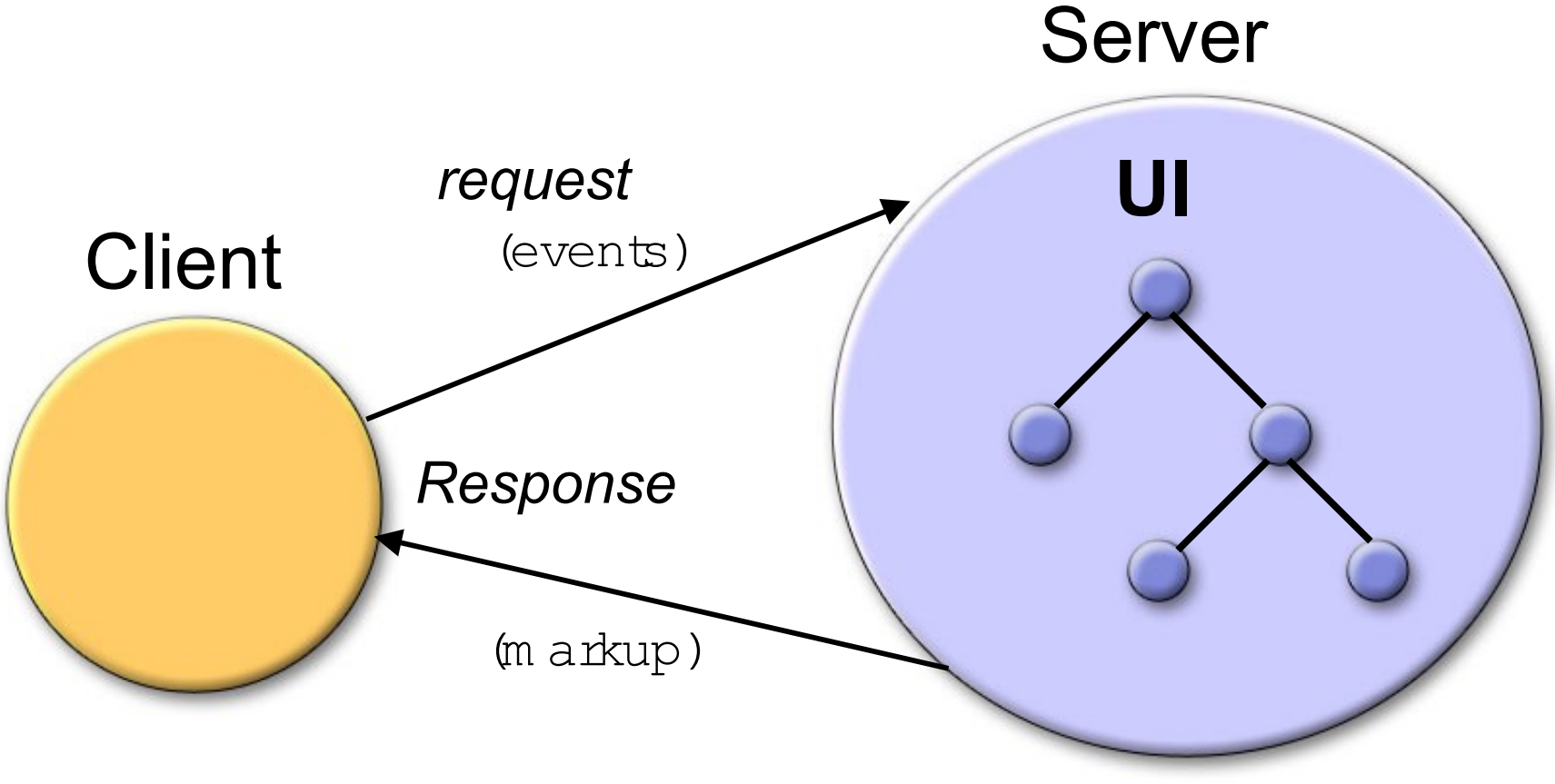




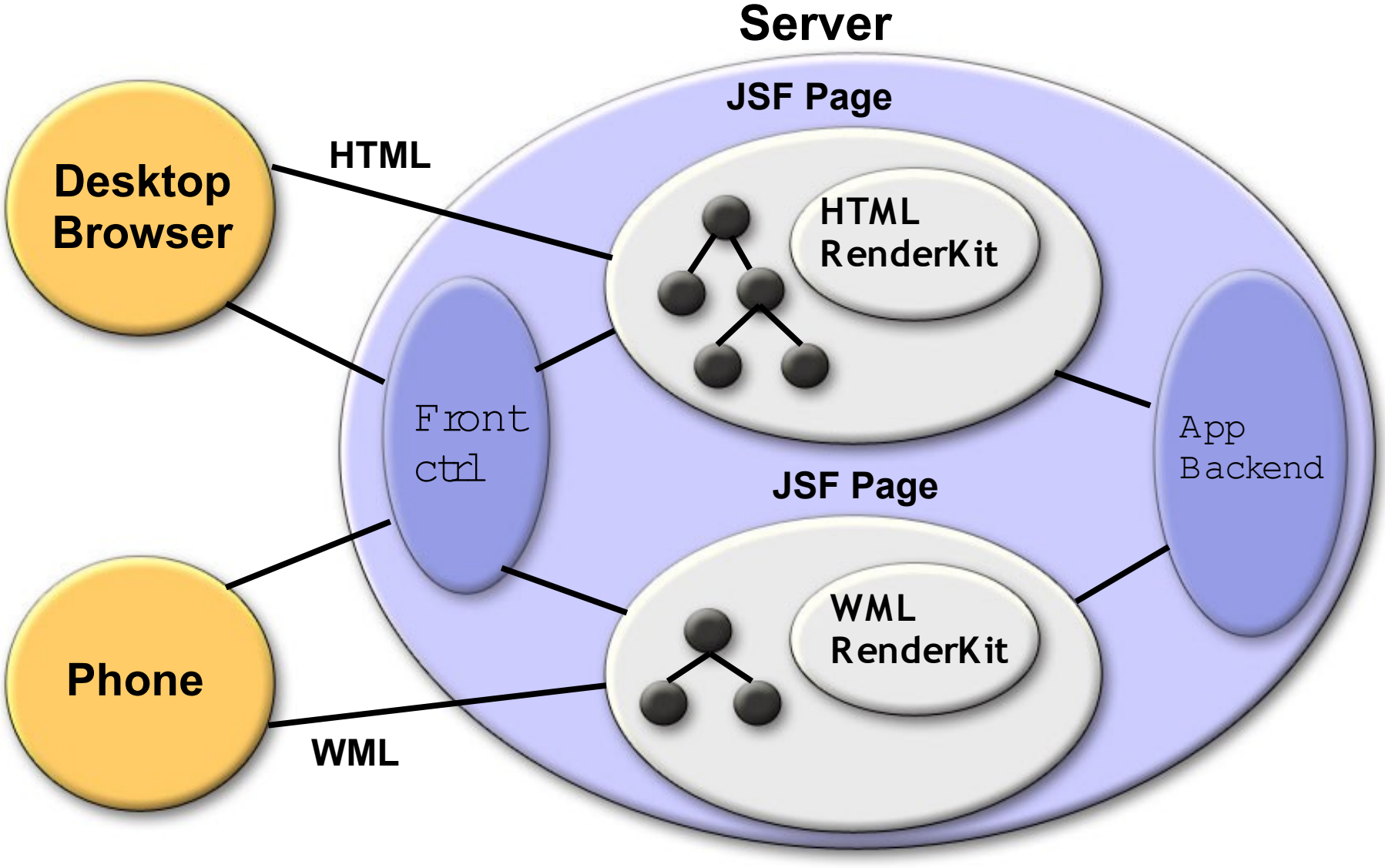
Quick Overview on JSF Architecture, Concept, & Features



JSF is a UI Framework for Java Web Applications



JSF Architecture



Important Basic Capabilities

- Extensible UI component model
- Flexible rendering model
- Event handling model
- Validation framework
- Basic page navigation support
- Internationalization
- Accessibility

Key JSF Concepts

- UIComponent
  - Render-independent characteristics
  - Base class with standard behaviors
- Standard UIComponent Subclasses:
  - UICommand, UIForm, UIGraphic, UIInput, UIOutput, UIPanel, UISelectBoolean, UISelectMany, UISelectOne
- FacesEvent
  - Base class for request and application events
- Validator
  - Base class for standard and application defined validators

Key JSF Concepts

- Converter
  - Plug-in for String-Object conversion
- FacesContext
  - Servlet request, response, session
  - JSF request, response trees
  - Model reference expression evaluators

    Syntax similar to the expression language of the JSP Standard Tag Library (JSTL) 1.x

    Primary interface between components and the data provided by (or to) the application

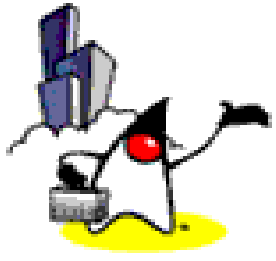
Key JSF Concepts

- Renderer
  - Converts components to and from a specific markup language
  - Supports render-dependent attributes on components
  - May support more than one component type
- RenderKit
  - Library of Renderers
  - Extensible at runtime
  - Basic HTML RenderKit is part of the specification

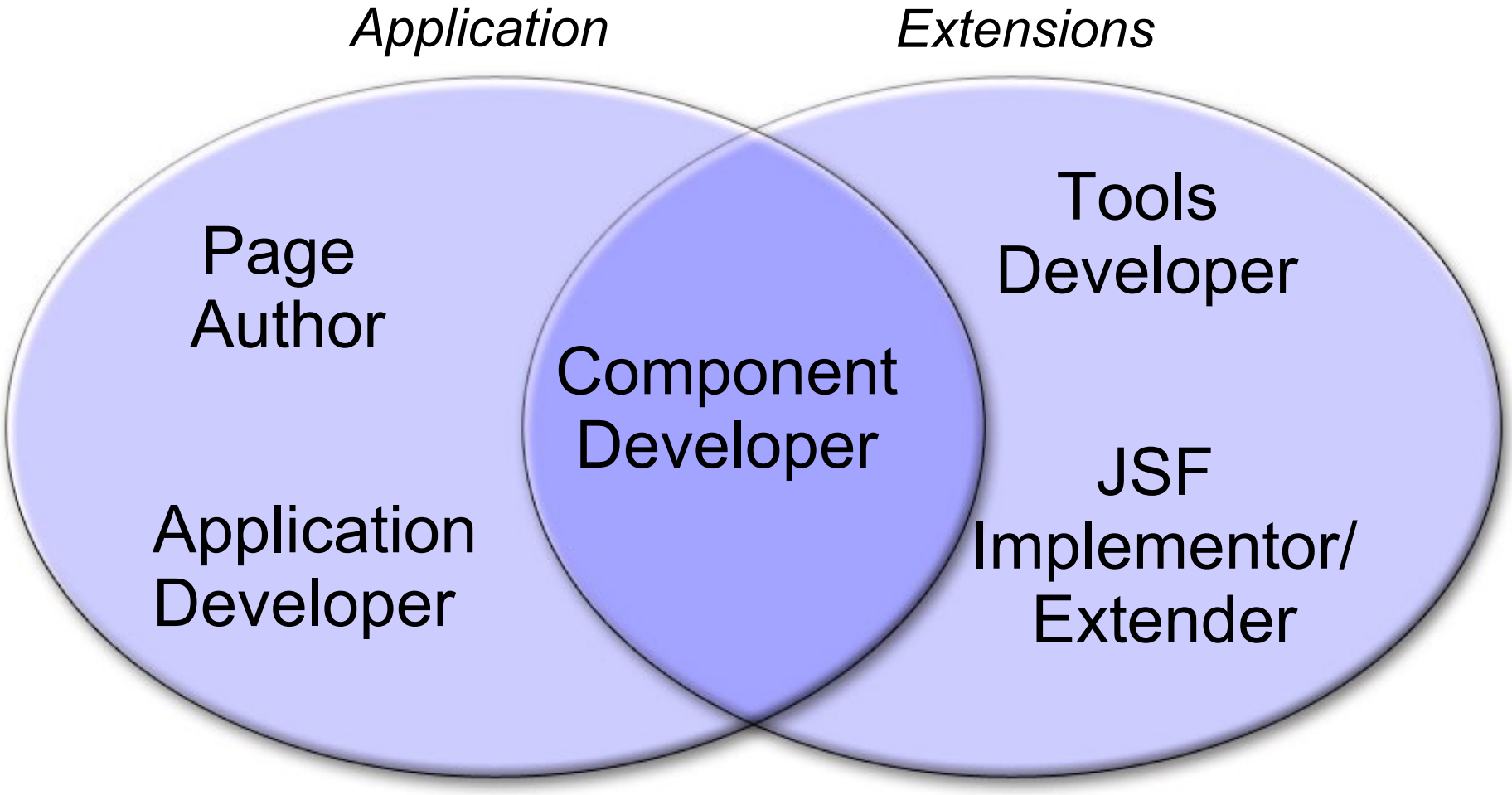
Relationship to Other JSRs

- JSF is based on:
  - Servlet 2.3 (JSR-53)
  - JSP 1.2 (JSR-53)
- JSF must be synergistic with:
  - JSTL 1.0 (JSR-52)
  - Portals (JSR-168)
- JSF is not part of J2EE 1.4 standard yet
  - Will be considered for J2EE 5.0
  - It is included in J2EE 1.4 SDK, however

Developer Roles



JSF Developer Roles



Roles Definition

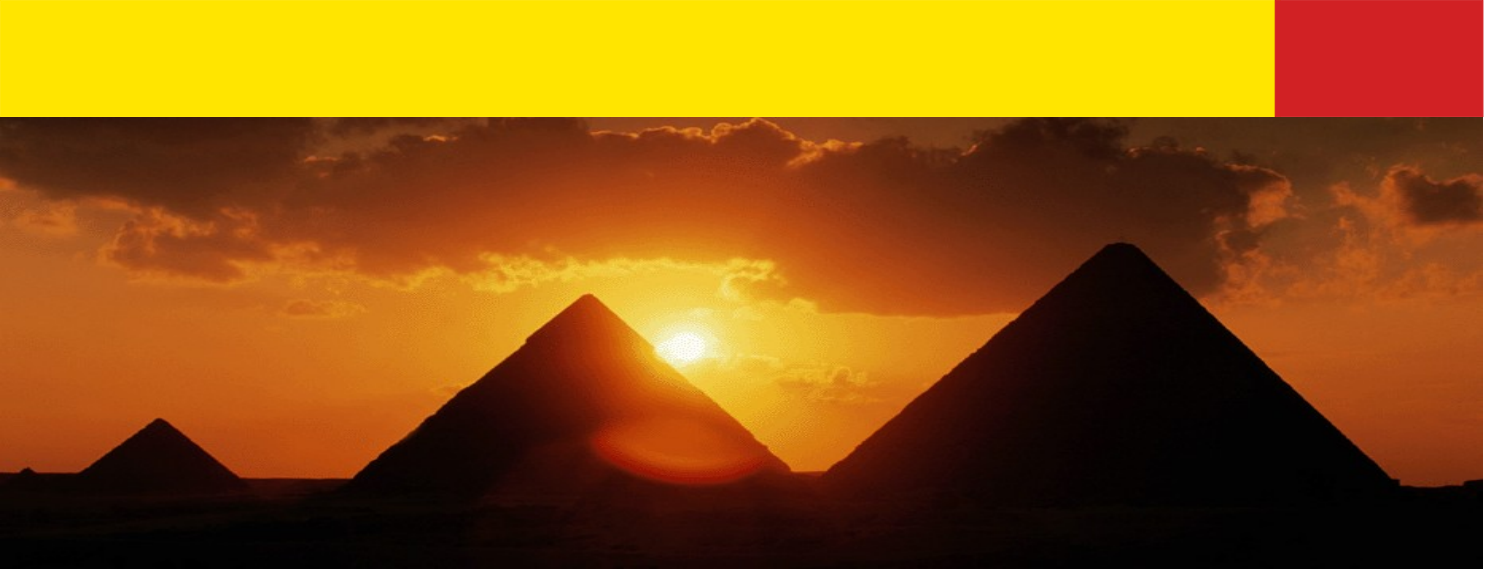
- Page Author – Creates the user interface of a web application
  - Familiar with markup language(s) to be used
  - Assembler of prebuilt components
  - Uses “Drag and drop” IDE like Sun Java Studio Creator
- Component Writer – Creates reusable components, renderers, and libraries
  - Components – Render-independent properties
  - Renderers – Render-dependent properties

Roles Definition

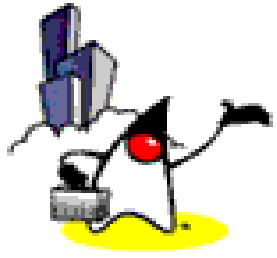
- Application Developer – Creates the server-side functionality of a web application not directly related to the user interface
  - Business logic components implemented in standard J2EE ways (EJBs, JavaBeans, Connectors)
  - Persistence tier components implemented in standard J2EE ways (EJBs, JDBC, Connectors)
  - Model data exposed to user interface via JavaBean programming model
  - Validator, Convertor, Event handler

Roles Definition

- Tool Provider – Creates tools to assist page authors, component writers, and application developers
  - GUI-oriented page development tools
  - IDEs to facilitate creation of components
  - Application generators (from high level description)
  - Web application frameworks that utilize JSF components for their user interface
  - Example: Sun Java Studio Creator
- JSF Implementor – Provides runtime environment to execute JSF webapps
  - J2EE SDK 1.4



Important Built-in Classes



UIViewRoot

- *UIViewRoot* is a *UIComponent* that represents the root of the *UIComponent* tree.
- Serves as the root of the component tree, and as a place to hang per-view *PhaseListeners*

UIViewRoot

- *UIViewRoot* is a *UIComponent* that represents the root of the *UIComponent* tree.
- Serves as the root of the component tree, and as a place to hang per-view *PhaseListeners*
  - We will talk about PhaseListeners in the JSF life-cycle presentation

FacesContext

- Contains all of the per-request state information related to the processing of a single JavaServer Faces request, and the rendering of the corresponding response.
- It is passed to, and potentially modified by, each phase of the request processing lifecycle

PhaseListener

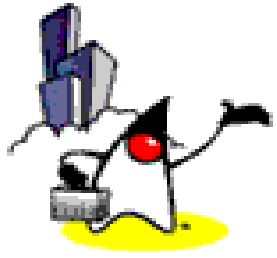
- An interface implemented by objects that wish to be notified at the beginning and ending of processing for each standard phase of the request processing lifecycle
- You can provide your own implementation of PhaseListener and plug it into the application for custom request handling
  - Ajax request handling
- Before and after each phase handling
  - “around” semantics

Application Class

- Represents a per-web-application singleton object
- Maintains application wide objects
  - set of supported locales
  - converters
  - validators
- Serves as a factory for creating components, converters, and validators

```
public abstract UIComponent createComponent(String
    componentType)  throws FacesException
```

Application Configuration



Application Configuration File

- XML file for configuring resources required at application startup time
  - navigation rules, converters, validators, render kits
- Usually named as faces-config.xml
- A <faces-config> tag must enclose all of the other declarations

```
<faces-config>
   ....
</faces-config>
```

faces-config.xml of guessNumber

```
<?xml version="1.0"?>

<!--
 Copyright 2003 Sun Microsystems, Inc. All rights reserved.
 SUN PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
-->

<!DOCTYPE faces-config PUBLIC
  "-//Sun Microsystems, Inc.//DTD JavaServer Faces Config 1.0//EN"
  "http://java.sun.com/dtd/web-facesconfig_1_0.dtd">

<faces-config>

  <application>
    <locale-config>
      <default-locale>en</default-locale>
      <supported-locale>de</supported-locale>
      <supported-locale>fr</supported-locale>
      <supported-locale>es</supported-locale>
    </locale-config>
  </application>
```

faces-config.xml of guessNumber

```
<navigation-rule>
  ...
   <from-view-id>/greeting.jsp</from-view-id>
  ...
</navigation-rule>

<navigation-rule>
  ...
  <from-view-id>/response.jsp</from-view-id>
  ...
</navigation-rule>

<managed-bean>
  ...
  <managed-bean-name>UserNumberBean</managed-bean-name>
  ...
<managed-bean>

</faces-config>
```

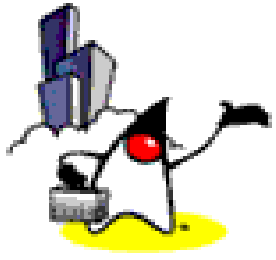
Application Configuration File

- You can have more than one application configuration file
- There are three ways that you can make these files available to the application]
  - A resource named /META-INF/faces-config.xml in any of the JAR files in the Web application's /WEB-INF/lib directory
  - A context init parameter, javax.faces.application
  - A resource named faces-config.xml in the /WEB-INF/ directory of your application (most common)

Application Class

- When an application starts up, the JSF implementation creates a single instance of the Application class
- Is automatically created for each application
- FacesContext.getApplication()

Backing Bean (Model Object) Management



What are Backing Beans?

- Server-side objects associated with UI components used in the page
- Define UI component properties, each of which is bound to
  - a component's value or
  - a component instance
- Can also define methods that perform functions associated with a component, which include validation, event handling, and navigation processing.

Why Backing Beans?

- Separation of Model from View (MVC)
  - Model handles application logic and data: Backing Beans are Model objects
  - View handles presentation: UI components

How to Specify Backing Beans in JSP page?

- A page author uses the JavaServer Faces expression language (JSF EL) to bind a component's value or its instance to a backing bean property
  - JSF EL is in the form of "#{...}"
- A page author also uses the JSF EL to refer to the backing-bean methods that perform processing for the component

Example: Binding Component Value to Backing Bean in greeting.jsp

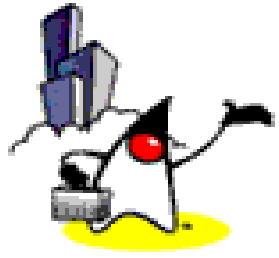
```
<h:inputText id="userNo" value="#{UserNumberBean.userNumber}"
             validator="#{UserNumberBean.validate}"/>
```

- userNo component's value is bound to the UserNumberBean.userNumber backing-bean property

UserNumberBean in faces-config.xml

```
<managed-bean>
   <description>
     The "backing file" bean that backs up the guessNumber webapp
   </description>
   <managed-bean-name>UserNumberBean</managed-bean-name>
   <managed-bean-class>guessNumber.UserNumberBean</managed-bean-class>
   <managed-bean-scope>session</managed-bean-scope>
   <managed-property>
     <property-name>minimum</property-name>
     <property-class>int</property-class>
     <value>0</value>
   </managed-property>
   <managed-property>
     <property-name>maximum</property-name>
     <property-class>int</property-class>
     <value>10</value>
   </managed-property>
</managed-bean>
```

Page Navigation



Define Page Navigation

- Application developer responsibility
  - Navigation rules are defined in the application configuration file
- Navigation rules
  - Determine which page to go to after the user clicks a button or a hyperlink

Navigation Rule 1 for guessNumber Example (V1)

```
<navigation-rule>
  <description>
     The decision rule used by the NavigationHandler to
     determine which view must be displayed after the
     current view, greeting.jsp is processed.
  </description>
  <from-view-id>/greeting.jsp</from-view-id>
  <navigation-case>
     <description>
        Indicates to the NavigationHandler that the response.jsp
        view must be displayed if the Action referenced by a
        UICommand component on the greeting.jsp view returns
        the outcome "success".
     </description>
    <from-outcome>success</from-outcome>
    <to-view-id>/response.jsp</to-view-id>
  </navigation-case>
</navigation-rule>
```

Navigation Rule 2 for guessNumber Example (V1)

```
<navigation-rule>
 <description>
     The decision rules used by the NavigationHandler to
     determine which view must be displayed after the
     current view, response.jsp is processed.
 </description>
 <from-view-id>/response.jsp</from-view-id>
 <navigation-case>
     <description>
         Indicates to the NavigationHandler that the greeting.jsp
         view must be displayed if the Action referenced by a
         UICommand component on the response.jsp view returns
         the outcome "success".
     </description>
     <from-outcome>success</from-outcome>
     <to-view-id>/greeting.jsp</to-view-id>
 </navigation-case>
</navigation-rule>
```

Navigation Rule

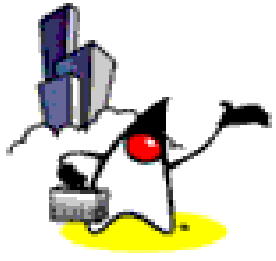
- <navigation-rule>
  - defines how to get from one page (specified in the from-tree-id element) to the other pages of the application
  - can contain any number of <navigation-case> elements
- <navigation-case>
  - defines the page to open next (defined by to-tree-id) based on a logical outcome (defined by from-outcome)

Where can Outcome come from?

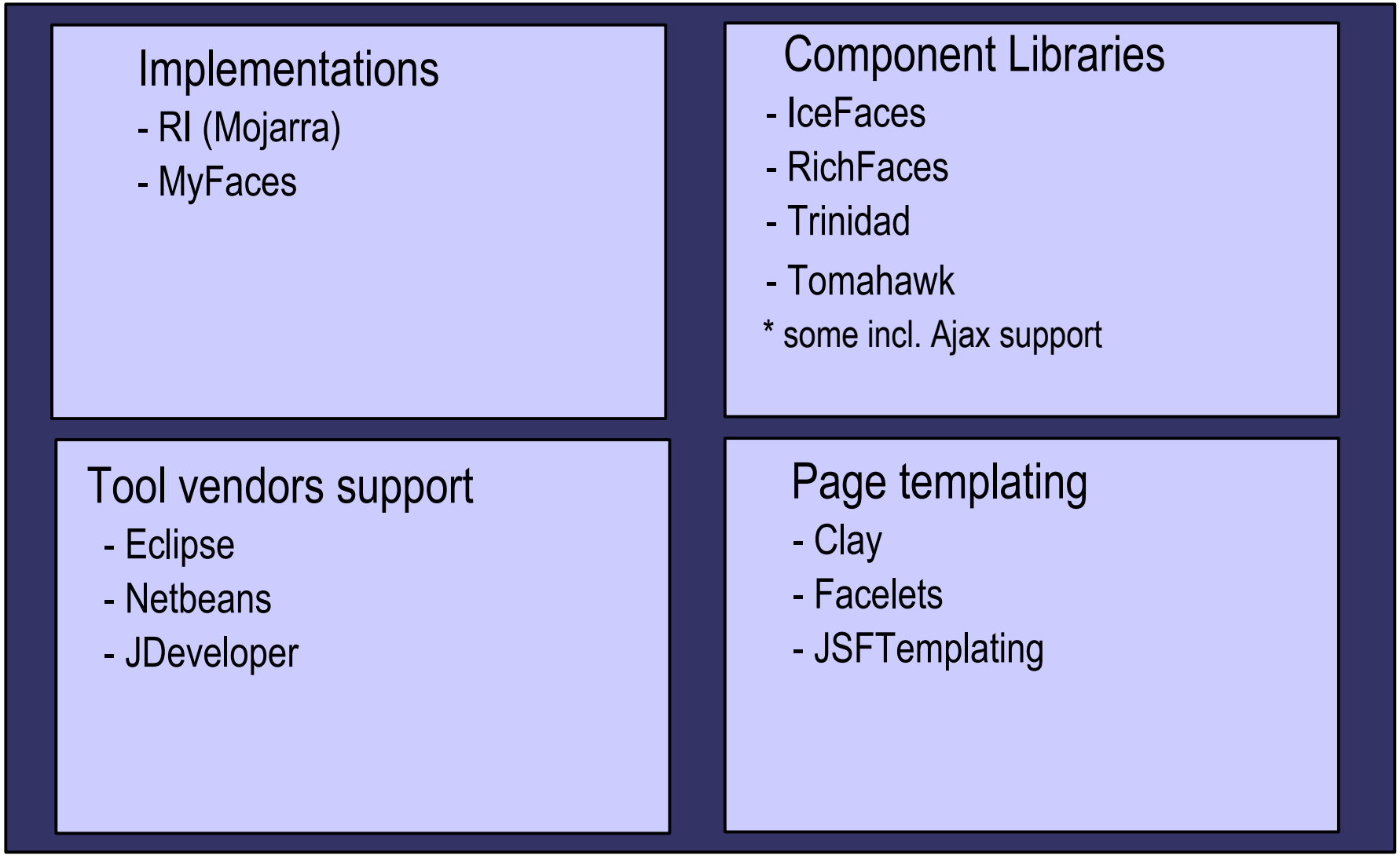
- Outcome can be defined by the action attribute of the UICommand component that submits the form
- “action” attribute can be a string or action method (#{<BackingBean>.<Method>})

```
<h:commandButton id="submit"
                 action="success" label="Submit" />
<h:commandButton
                 action="#{carstore.buyCurrentCar}"
                 value="#{bundle.buy}" />
```

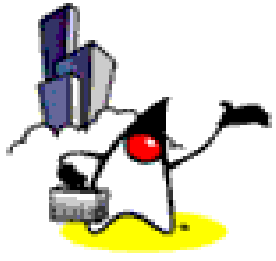
JSF Echo System



JSF Ecosystem



JSF Components



JSF Components

- JSF is a Component Framework
- Traditional OO benefits of encapsulation
- Build complex things from aggregations of simple things
- Treat a component as a “black box”
- UI Components know how to
  - encode themselves to a client device (rendering)
  - decode their “value” from the incoming request
  - ask for their data to be converted
  - ask for their data to be validated
  - ask for their data to be pushed to the model tier

Third Party Components

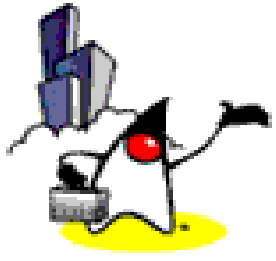
- Component libraries may offer Rich components as:
  - Trees
  - Tabbed Pane
  - Auto-complete Ajax
  - Layout Managers

- Converters and Validators
  - ZIP Code validator
  - E-mail validator

Third Party Components

- There are many good component libraries available:
  - Woodstock (from Sun)
  - IceFaces
  - RichFaces
  - Apache Trinidad
  - Apache Tomahawk
  - BusinessObjects
  - ChartFX

JSF 2.0



JSF 2.0

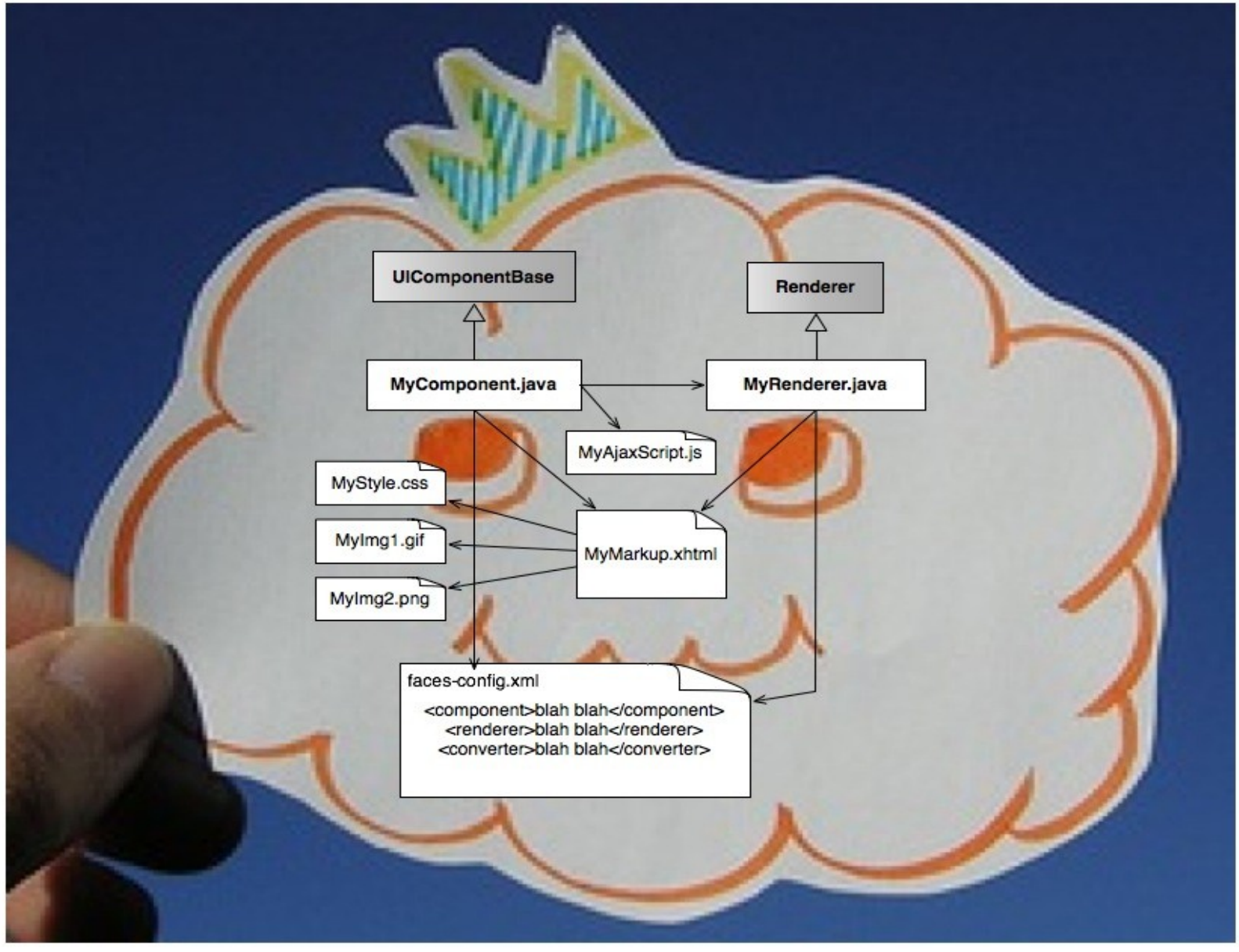
- Top Five Goals
  1) Make custom components much easier to develop
  2) Ajax support
  3) Page description language (PDL)
  4) Reduce the configuration burden
  5) Provide for better compatibility between JSF component libraries from different vendors
- Other important goals
  - State management rewrite
  - Bookmarkable URLs
  - Zero deployment time
  - Better Error Reporting

Java EE 6

- Servlet 3.0 and JSF 2.0 will be in the FCS release of GlassFish v3
- In early access form after v3 Prelude release

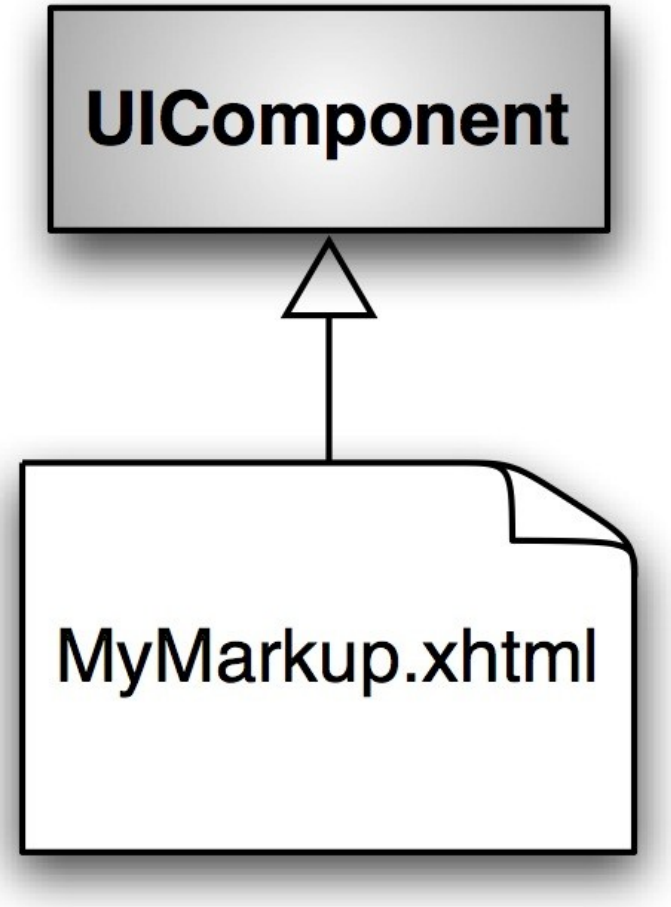
Easy Component Development

This:



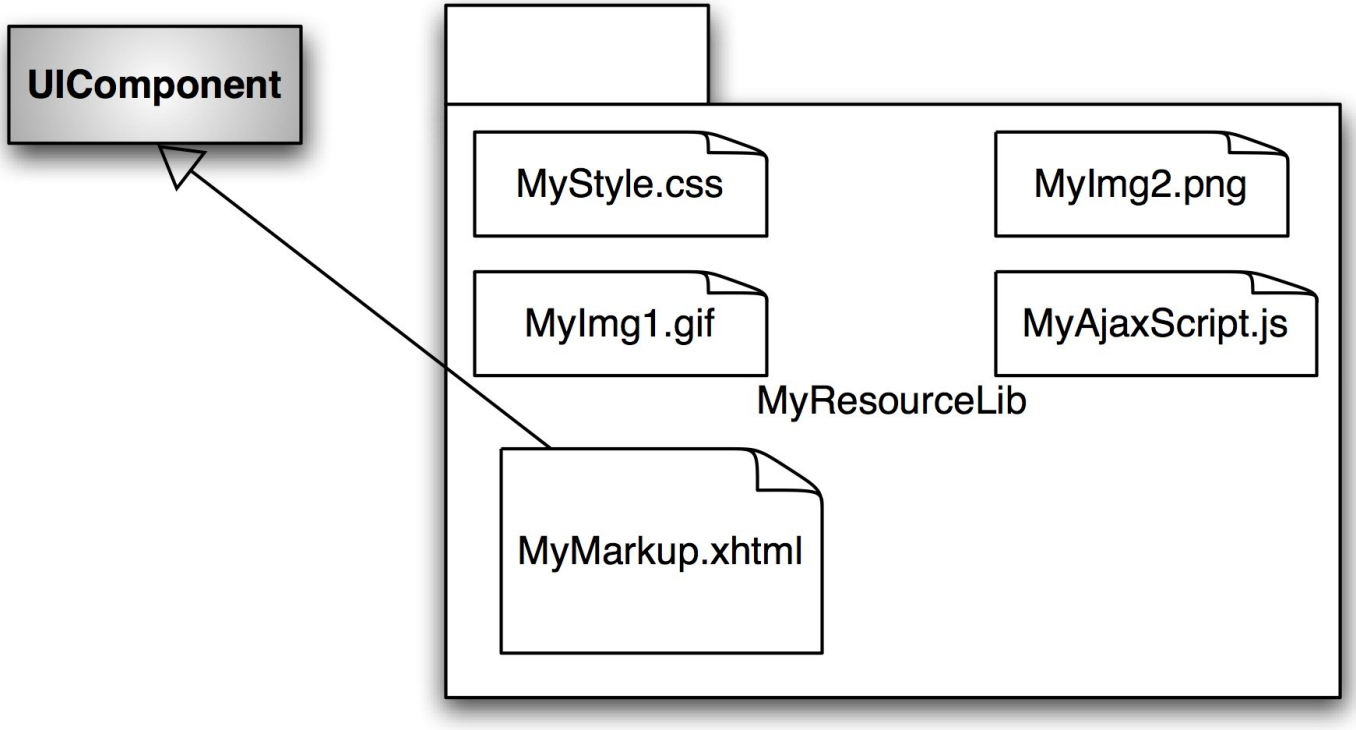
Easy Component Development

Becomes This:



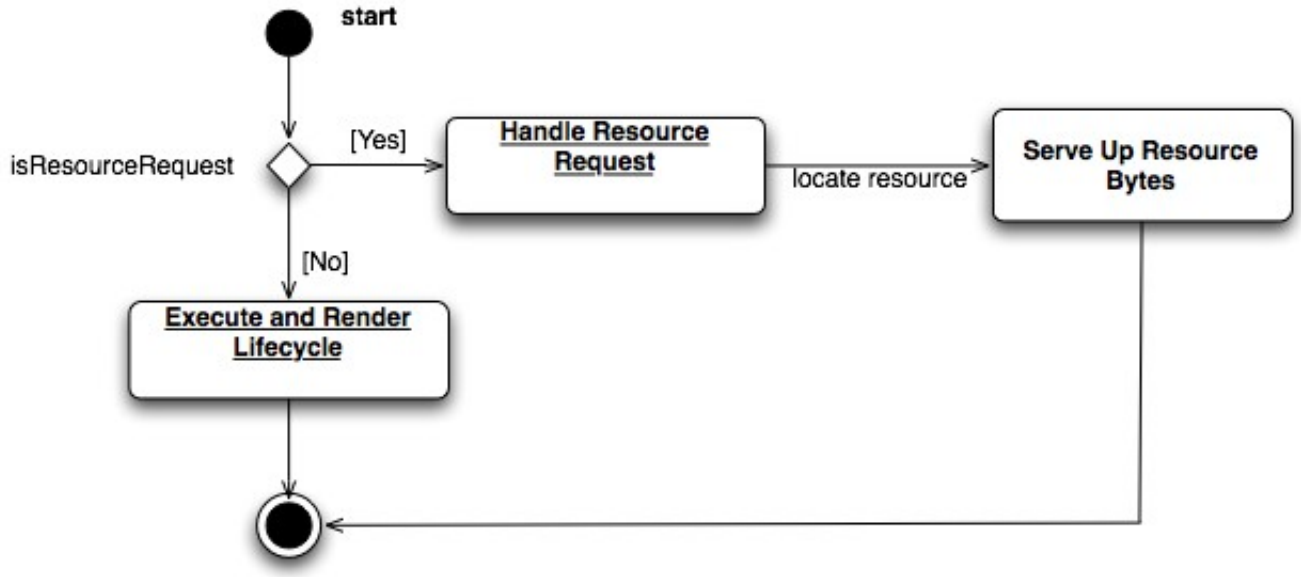
Easy Component Development

Or Even This:



... If you want to get fancy

Resource Delivery



- Delivers static resources to the user-agent in response to HTTP GET requests
- Includes support for localized, versioned resources and resource libraries

Easy Component Development

- Inspirations
  - Facelets
  - JSFTemplating
- API
  - Facelets now core part of JSF
  - Template based Renderers and events from JSFTemplating

Ajax and JSF

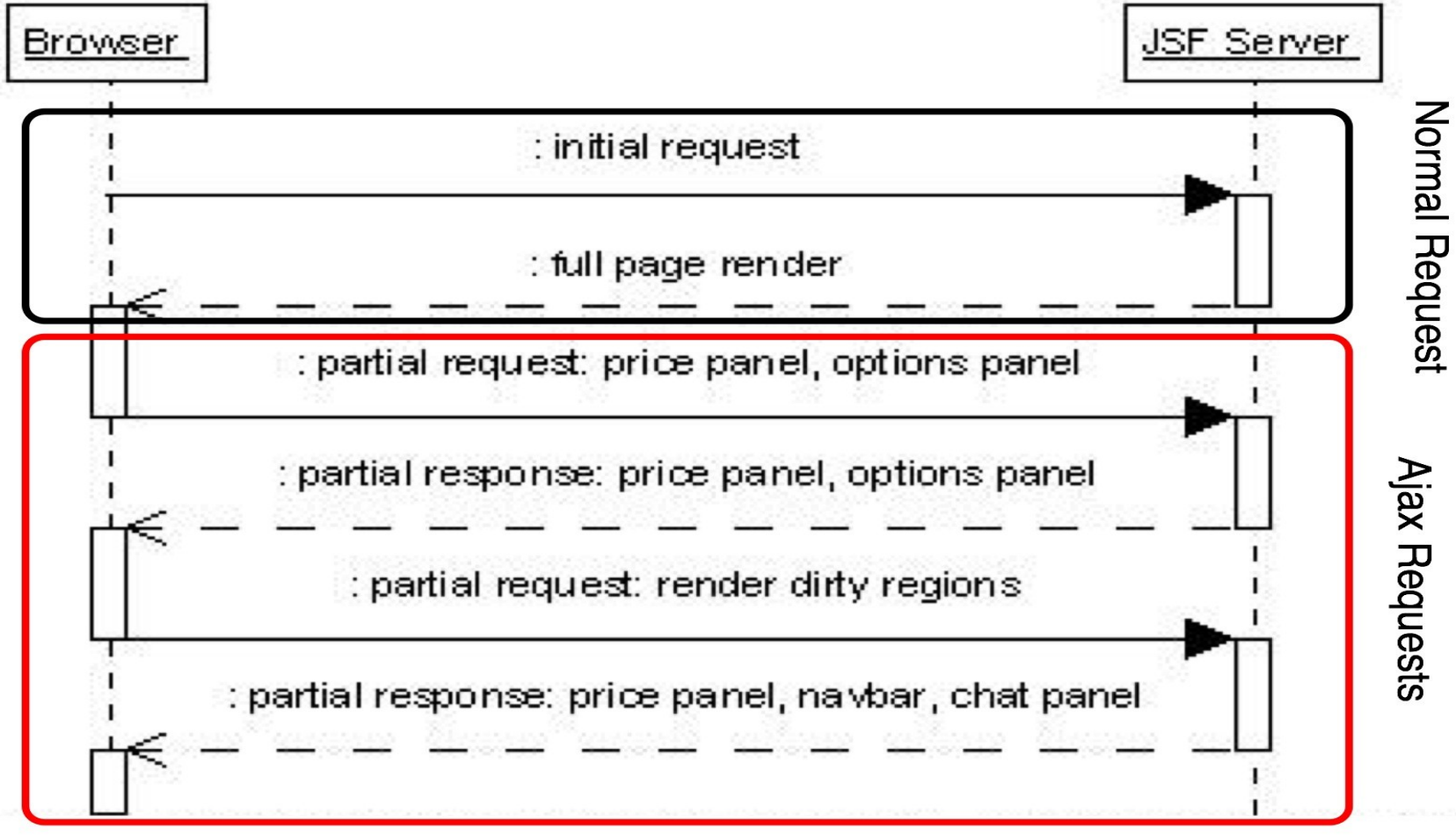
- Resource Delivery Mechanism
- Partial Tree Traversal
- Partial Page Update
- Ajaxification Capability

↑ In JSF 2.0 Spec

- Ajax Enabled Components

↓ In Component Library

JSF Ajax – Partial Page Update



Ajax Enabled Components

- Such components always build on top of the previous ingredients
- Current offerings are tightly coupled to their specific implementation of the previous ingredients.
- By standardizing the foundations upon which these components build, we can guarantee interoperability between them.

JSF - Ajax

- JavaScript API
  - Minimal function set (at least to start):
    - Partial Submit
    - Partial Rendering
    - Utility functions:
      - Collect/encode/return client JSF View State (to be used in POSTBACK or Ajax request)
      - Given JSF componentId or clientId, return client DOM Element corresponding to outermost markup for that component
- On the radar....
  - Comet
- Current JSF & Ajax solutions
  - ICEFaces
  - Dynamic Faces
  - RichFaces/Ajax4JSF
  - AjaxFaces

Glassfish Specific JSF - Groovy

- Use Groovy to create any JSF artifact
  - Managed Beans
  - Renderer
  - PhaseListener
  - ActionListener (application level)
  - Renderer
  - ELResolver
  - Component
  - Converter
  - Validator
- Further instructions here:
- http://blogs.sun.com/rlubke/entry/groovy_mojarra

Passion!