
# JSF 2.0: Insight and Opinion

Ed Burns
Senior Staff Engineer

Roger Kitain
Staff Engineer

Sun Microsystems

TS-5979

Java

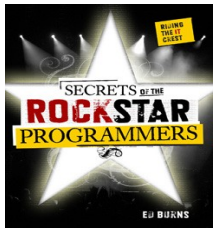
Sun microsystems

# Overall Presentation Goal

- *Inspire* Confidence in Choosing JavaServer™ Faces platform
- *Share* our vision for JavaServer Faces 2.0 application
- *Demonstrate* our progress

# Speaker Qualifications

- Ed Burns is:


  - A Senior Staff Engineer at Sun Microsystems, Inc.
  - Since inception, co-leader of the team that develops the JavaServer Faces Specification
  - Co-author of the McGraw-Hill book, JavaServer Faces, the Complete Reference
  - Author of the McGraw-Hill book: Secrets of Rock Star Programmers: Riding the IT Crest
  - Prior to JSF Ed worked on the Sun Java™ Plug-in, Mozilla Open JavaVM Interface, NCSA Mosaic
- Roger Kitain is:
  - A Staff Engineer at Sun Microsystems, Inc.
  - Since JavaServer Faces 1.1 application, co-leader of the team that develops the JavaServer™ Faces Specification
  - Prior to JavaServer Faces specification, Roger worked on early web interfaces for User Mgmt. - now known as Identity Server, internal Electronic Commerce solution for JavaSoft ™ technology

Seed and Harvest

# Agenda

- Where is JavaServer Faces techonology today?
  - Current Version Status
  - Real World Deployments
- Where is JavaServer Faces technology going?
  - How we listened
    - Discover requirements
    - Prioritize
  - What we heard
    - Pain points
    - What's missing
    - Which JavaServer Faces technology extensions are popular
    - How are people using JavaServer Faces technology
  - What we are doing about it
    - Early Draft 1
    - Beyond

# Where is JavaServer Faces Technology Today?

## Current Version Status

- Latest official specification version:
  JavaServer Faces 1.2 application Maintenance Release 1
- Latest Implementations
  - Sun Implementation (bundled in all JavaEE 5 containers):
    Mojarra 1.2_04 patch 3, in Glassfish Version 2 Update Release 1
  - Apache Implementation:
    MyFaces 1.2.2, not bundled in any container, but usable with Tomcat 6.0 or later
- Component libraries


Trinidad
Tobago



JBoss RichFaces
Project Woodstock


Oracle ADF Faces






Mojarra Scales

# Real World Deployments

JavaOne

## It's no secret, JavaServer Faces Technology is everywhere

Virgin megastores · zavvi zavvi.co.uk · 3M · BAE SYSTEMS · GARMIN · RITE AID PHARMACY With us, it's personal. · Intuit · ADP · CREDIT SUISSE · Peerflix · A Passion to Perform. Deutsche Bank · FedEx · BIG LOTS!


google: RealWorldJsfLinks

## Tool and Run-time adoption

- Every App Server except Geronimo using Sun's JSF impl
- Every Java IDE provides deep support for authoring JSF apps

555-1212.com®

Seed and Harvest

# Agenda

- Where is JavaServer Faces technology today?
  - Current Version Status
  - Real World Deployments
- Where is JavaServer Faces technology going?
  - How we listened
    - Discover requirements
    - Prioritize
  - What we heard
    - Pain points
    - What's missing
    - Which JavaServer Faces technology extensions are popular
    - How are people using JavaServer Faces technology
  - What we are doing about it
    - Early Draft 1
    - Beyond

# How we listened

- JavaServer Faces 1.0, 1.1 application: Plant the seed
  - Lots of opinions!
    - Fix or drop JavaServer Pages™ (JSP™) software
    - Too complex
- JavaServer Faces 1.2 application: Mostly planting, some harvesting
  - Incremental improvement
  - Fixed JSP software
  - A little bit easier, but not much
  - Still more opinions
    - Based on real world deployments

Y

# How we listened

- JavaServer Faces 2.0 application: Mostly harvesting, some seeding
  - Many Developer tool vendors are building on JavaServer Faces technology
    - IDE vendors
    - Extension vendors
      - UI Components, i.e. ICEFaces, RichFaces, Woodstock
      - Automated testing, i.e. JavaServer Faces Unit
      - Iterative Development aids, i.e. FacesTrace
      - Run-time enhancements, i.e. Spring Web Flow, Seam
  - Many real world deployments

# How we listened

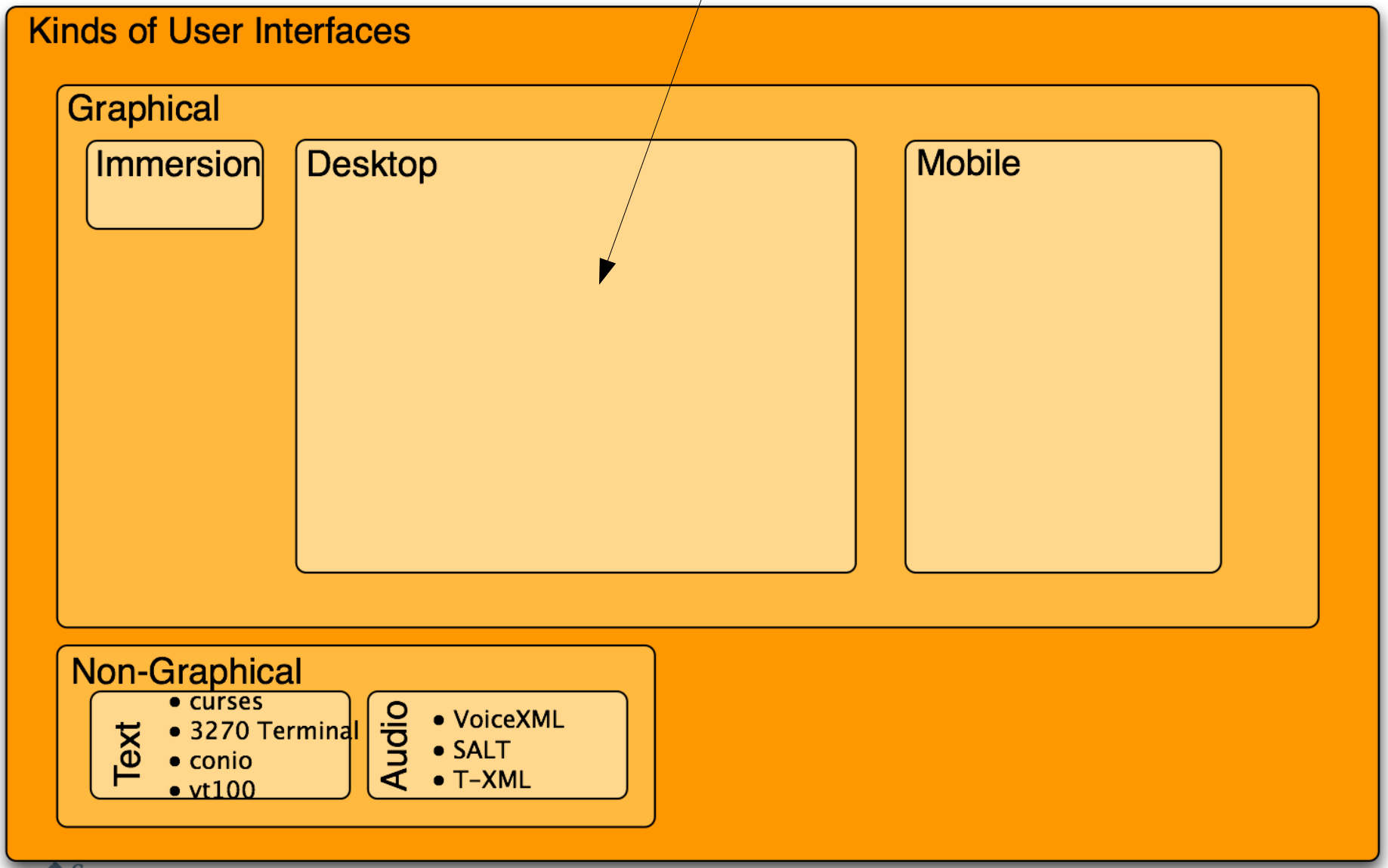
- Keep current on web development trends
  - Rails, and all that it stands for
    - Simplicity
    - Focus: CRUD
    - Convention over configuration
    - Maximize “flow state” (Mihály Csíkszentmihályi)
  - Rich Internet Applications
    - Ajax
    - Interactivity
    - Benefits of Scripting
    - Web apps that do not suck

But...

Y

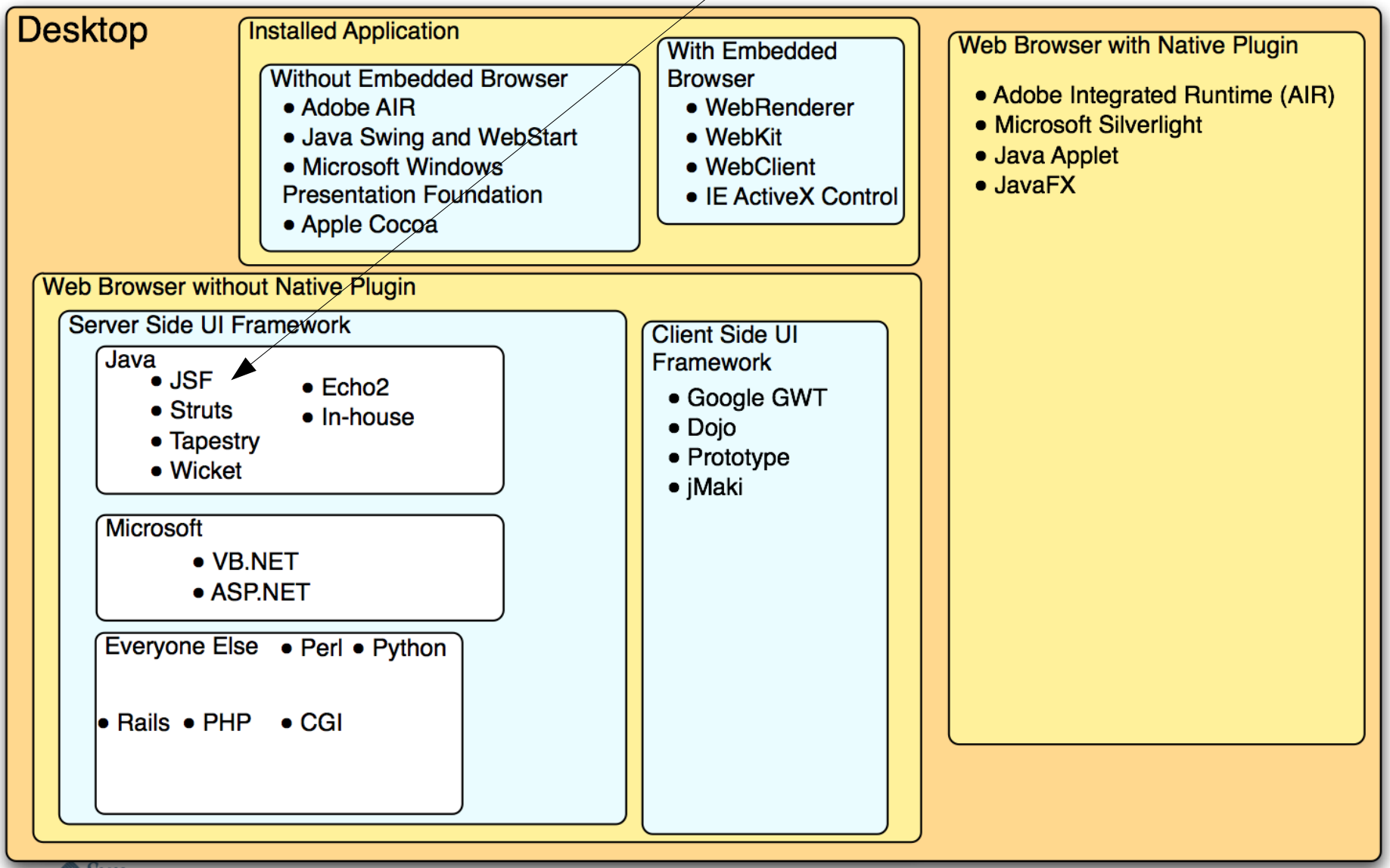
# How we listened

- JavaServer Faces components still lives mostly in this space



# How we listened

- JavaServer Faces components still lives mostly in this space



# We Hear You

# What we heard

- 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 JavaServer Faces component libraries from different vendors
- Other important goals
  - State management rewrite
  - Bookmarkable URLs
  - Zero deployment time
  - Tree traversal
  - Scopes
  - Extension prioritization
  - Better error reporting

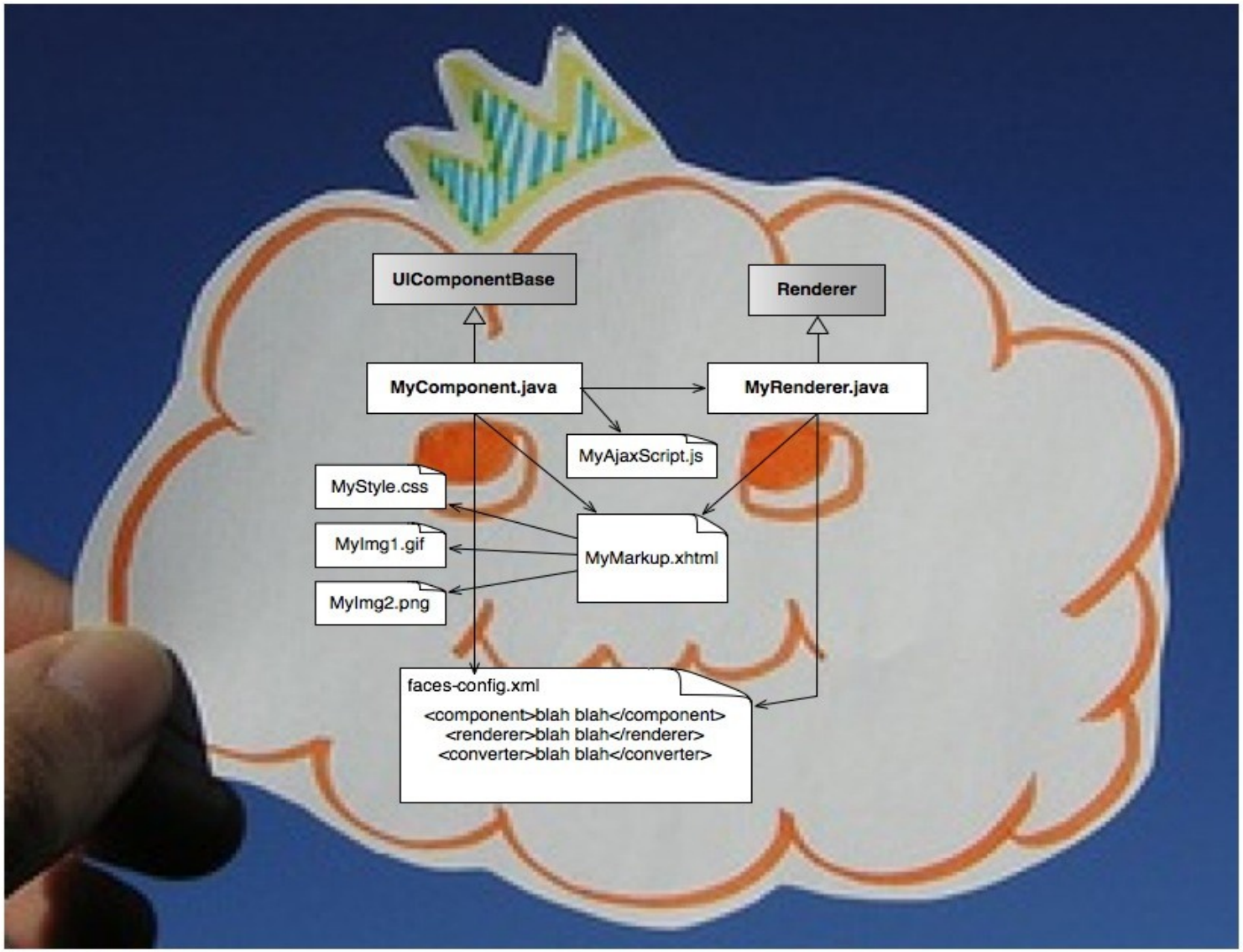
# What are we doing about it?

# What are we doing about it?

- 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 JavaServer Faces component libraries from different vendors
- Other important goals
  - State management rewrite
  - Bookmarkable URLs
  - Zero deployment time
  - Tree traversal
  - Scopes
  - Extension prioritization
  - Better error reporting

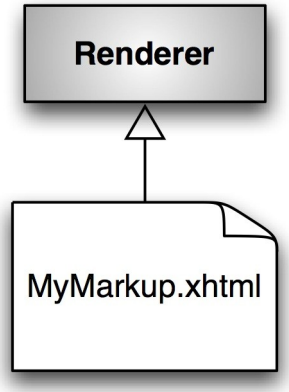
# Make components easy to develop

- This...



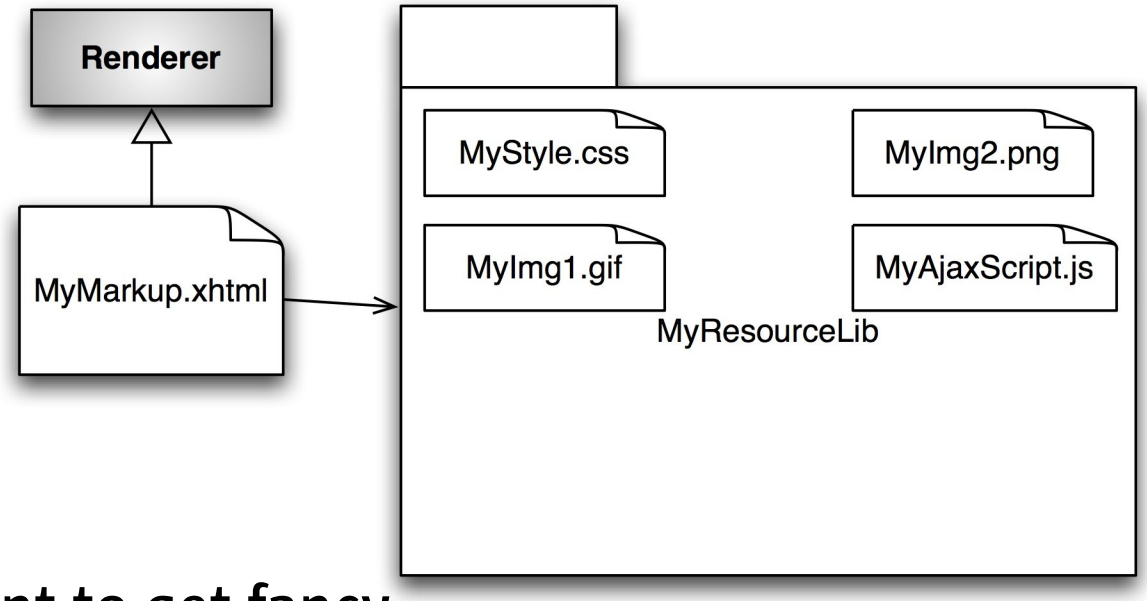
# Make components easy to develop

- Becomes this...



# Make components easy to develop

- Or maybe this...




...if you want to get fancy.

# EZComp - Details

- Inspirations
  - Facelets
  - RAILS_ENV
  - Shale-Tiger extensions
- API
  - Facelets now core part of JavaServer Faces component
  - Template based Renderers and events from JavaServer Faces component Templating
  - RAILS_ENV: Add `getProjectStage()` to `Application`

    `ProjectStage` is an enum
    - `Development`
    - `Production`
    - `SystemTest`
    - `UnitTest`

  Allows us to specify what happens during different project stages

# Demonstration

# What are we doing about it?

- 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 JavaServer Faces component libraries from different vendors
- Other important goals
  - State management rewrite
  - Bookmarkable URLs
  - Zero deployment time
  - Tree traversal
  - Scopes
  - Extension prioritization
  - Better error reporting

# Ingredients of a JavaServer Faces Component +Ajax solution

- Resource Delivery Mechanism
- Partial Tree Traversal
- Partial Page Update
- Ajaxification Capability
- Ajax Enabled Components

# Ingredients of a JavaServer Faces Component +Ajax solution

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

↑ In JSF 2.0 Spec

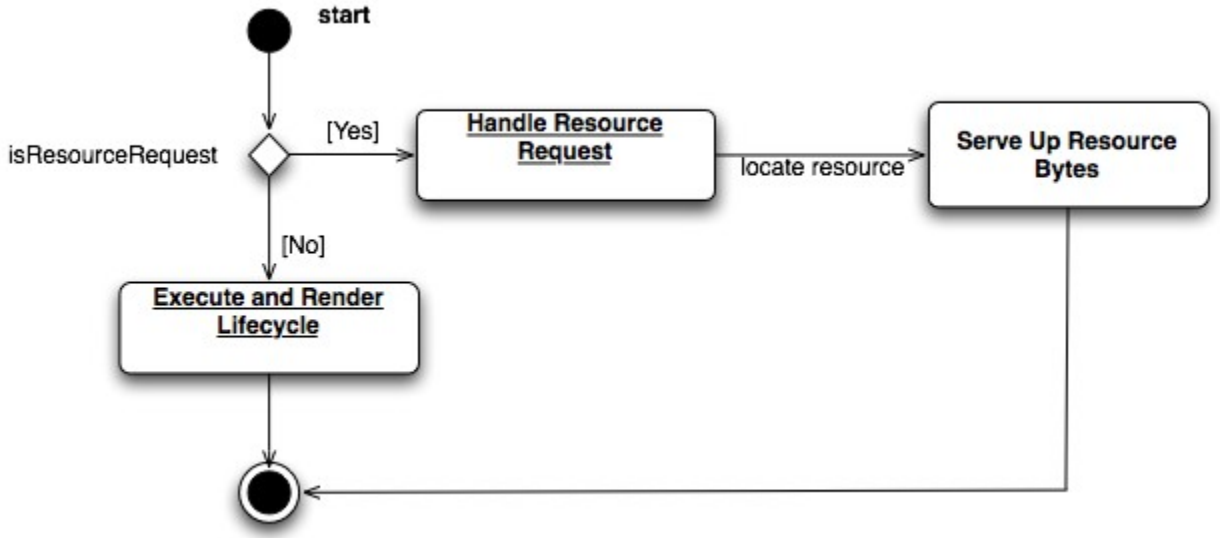
---

- Ajax Enabled Components

↓ In Component Library

# Ingredients of a JavaServer Faces component +Ajax solution

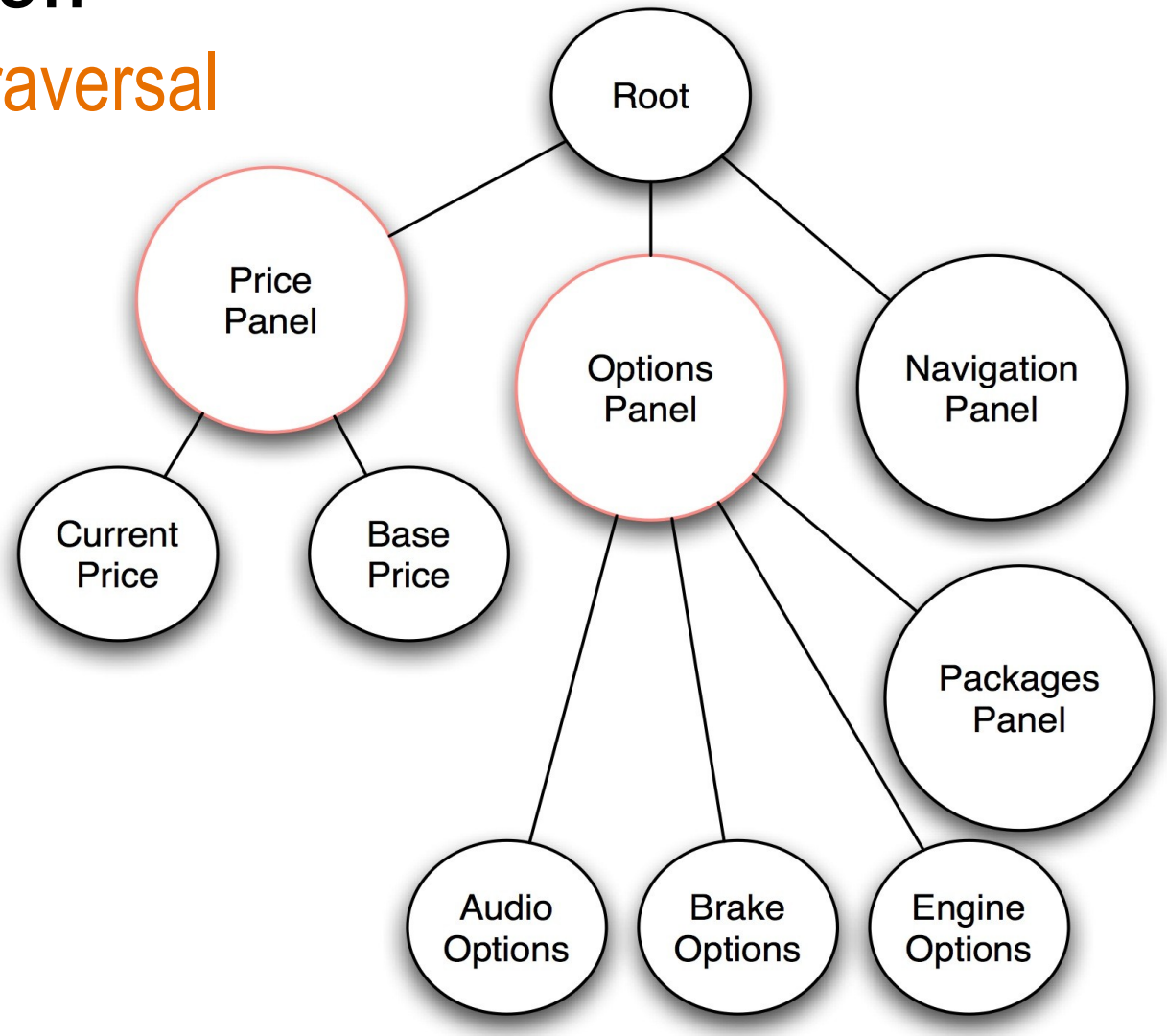
## Resource Delivery Mechanism




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

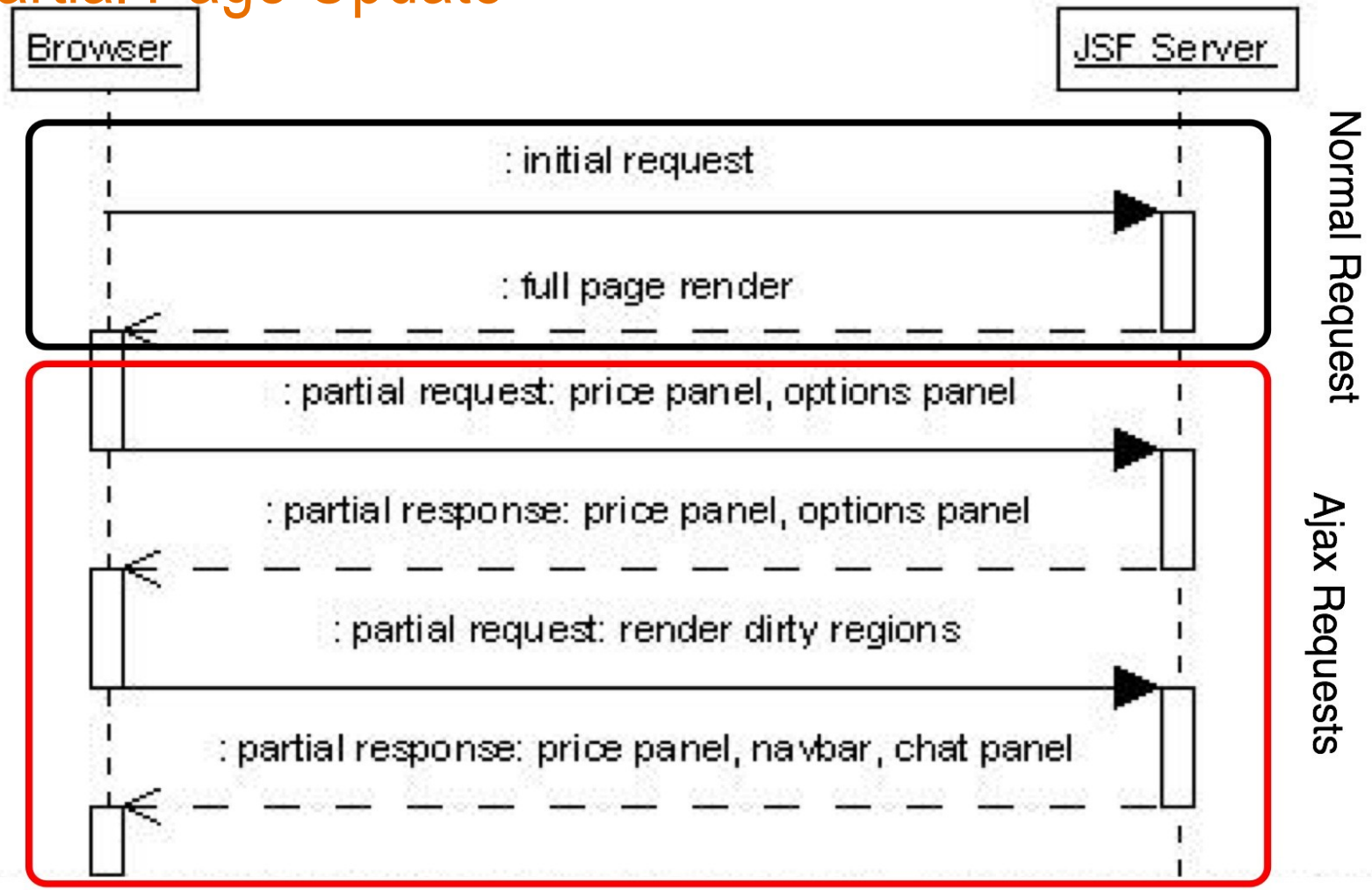
# Ingredients of a JavaServer Faces component +Ajax solution

## Partial Tree Traversal



# Ingredients of a JavaServer Faces Component +Ajax solution

## Partial Page Update



# Ingredients of a JavaServer Faces Component +Ajax solution

## Ajaxification Capability

- A way to give ajax capability to existing JavaServer Faces components without writing any JavaScript™ language
- Common approaches include
  - AjaxZone tag, enclose region to ajaxify
  - AjaxSupport tag, nest inside of component to ajaxify

# Ingredients of a JavaServer Faces Component +Ajax solution

## 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.

# Ajax - Details

- Goals:
  - Add common Ajax operations to the standard
  - Fix the “Ajax components from different libraries don't play together” problem
- JavaScript Namespacing technology
  - Registered top level  namespace with OpenAjax Alliance
  - “faces” property under top level namespace, “Ajax” property under “faces” property
- JavaScript API
  - “ajax.js”
    - House JavaScript language functions
    - Runtime must guarantee delivery of file and namespacing requirements are met

# Ajax - Details

- Inspirations
  - Weblets
  - Shale Remoting
  - jMaki
- API
  - Add `ResourceHandler` to `Application`
  - Modify `FacesServlet` to use `ResourceHandler`
  - New renderer types
    - `javax.faces.resource.Script`
    - `javax.faces.resource.Stylesheet`
    - `javax.faces.resource.Image`
  - `ELResourceResolver`
  - `@ResourceDependency` annotation
  - `UIViewRoot.addComponentResource()` method
  - `SystemEvent` facility

# Ajax - Details

- JavaScript API
  - Minimal function set (at least to start):
    - Partial Submit
    - Partial Rendering
    - Utility functions:
      - Collect/encode/return client JavaServer Faces component View State (to be used in POSTBACK or Ajax request)
      - Given JavaServer Faces componentId or clientId, return client DOM Element corresponding to outermost markup for that component
- On the radar....
  - Comet (aka reverse Ajax / Ajax push)
- Inspiration
  - ICEFaces
  - Dynamic Faces
  - RichFaces/Ajax4JavaServer Faces
  - AjaxFaces

# Demonstration

# What are we doing about it?

- 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 JavaServer Faces component libraries from different vendors
- Other important goals
  - State management rewrite
  - Bookmarkable URLs
  - Zero deployment time
  - Tree traversal
  - Scopes
  - Extension prioritization
  - Better error reporting

# View Scope

- In Java;
  - Exposed as method on UIViewRoot

    public Map<String, Object> getViewMap();
- In EL:
  - Exposed via the implicit object “viewScope”
- In the faces-config:
  - Exposed via a managed-bean-scope value of "view".
- Lifetime
  - Begins lazily the first time it is accessed
  - Ends when different UIViewRoot instance is installed via FacesContext.setViewRoot();
- Managed Bean Implications
  - “Widened” scope search for Scoped Attribute ELResolver (between request and session)
  - Managed Bean ELResolver must consider additional “view” scope when storing/instantiating managed bean instances

# “component” Scope

- In Java;
  - Exposed as method on UIComponent

    public static Map<String, Object> getCurrentComponent();
- In EL:
  - Exposed via the implicit object “component”
- Lifetime
  - Like a “this” pointer, valid during tree traversal

# Summary

- We researched
- We heard
- We are acting
  - JavaServer Faces 2.0 application is in Java Platform, Enterprise Edition 6 (Java EE 6)
  - It will work well with WebBeans
  - It will work well with Portlet 2.0
  - Java EE 6 is targeted for end of CY 2008
- “Completed” issues
  - Resource loading
  - Development lifecycle awareness
- In Progress
  - EZComp
  - Ajax

# THANK YOU

JSF 2.0: Insight and Opinion

Ed Burns
Roger Kitain

TS-5979