True Abstraction
JavaServer™ Faces 2.0 Composite Components

Ed Burns <http://ridingthecrest.com/>
Presentation Goals

- Demonstrate the usage of composite components
- Share the vision for composite components
- Show how to build a composite component
Agenda: All About Composite Components

- **Beginning**
  - JSF 1.0 Vision and Reality
  - JSF 2.0 Vision
  - DEMO: A taste of the future
- **Middle**
  - DEMO: Building a simple composite Component
  - Terminology
  - Core Concepts
- **End**
  - How to Think About Composite Components
Speaker Qualifications: Ed Burns

- Senior Staff Engineer at Sun Microsystems, Inc.
- Since inception, co-leader of the team that develops the JavaServer™ Faces (JSF) Specification
- 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
Beginning
Create a market for re-usable JSF components, allowing developers to easily create UI's for web applications by combining off-the-shelf components from multiple vendors using nice GUI tools.
JavaServer Faces: Motivation

- Eliminate burden on developers
  - Today must create/maintain own frameworks
- Boost Tools, 3rd Party Component support
  - easier to leverage single framework
- Improve GUI quality
  - tools & framework do the hard stuff
JSF 1.X Component Reality

- Good
  > Very active and healthy component market
  > Very good IDE support

- Bad
  > Components not easy enough to build
  > Component vendors had to invent stuff because the spec didn't solve
    - Ajax
    - Resource Loading
    - Library Ordering Precedence
JSF 2.0 Vision for Components

This...
Make components easy to develop

Becomes this...

![Diagram showing relationship between UIComponent and MyMarkup.xhtml]
Make components easy to develop

- Or maybe this...

... if you want to get fancy
- “Pay as you go” complexity
JSF 2.0 Component Vision

- multi-select components on a JSF page
- press a “componentize” button
- you get a wizard that lets you choose how to expose the content of this component to the page author
- the component appears in a palette.
Demo: Login Panel from Admin GUI
How do we enable this vision?

- Reduce the number of artifacts required
- Provide a way to bundle associated resources with the component
- Do it all dynamically, while the app is deployed
- Allow the composite component to be a real component
  - attached objects
  - children components
  - facets
  - ajax capable
- The “old way” still works.
How do we solve the “multi-component-vendors in a single page” problem?

Look at what each vendor had to invent and standardize it

<table>
<thead>
<tr>
<th>Source of difficulty</th>
<th>Old Way</th>
<th>Standard Way</th>
</tr>
</thead>
<tbody>
<tr>
<td>Resource Loading</td>
<td>PhaseListener, Filter, etc.</td>
<td>Built into the Lifecycle</td>
</tr>
<tr>
<td>Ajax</td>
<td>•Proprietary .js file for innerHTML updating</td>
<td>•jsf.ajax JavaScript Object with four functions</td>
</tr>
<tr>
<td></td>
<td>•Propretary .js file for innerHTML updating</td>
<td>•Standard XML Ajax Response Format</td>
</tr>
<tr>
<td></td>
<td>•XML Ajax Response Format</td>
<td></td>
</tr>
<tr>
<td>Library Loading Precedence</td>
<td>Rename Jar files for alphabetical sort order</td>
<td>&lt;ordering&gt;, &lt;absolute-ordering&gt; elements</td>
</tr>
</tbody>
</table>
Demo: Building your first composite component
Coming to Terms with Composite Components

- **Composite Component**
  - A tree of *UIComponent* instances, rooted at a *top level component* that can be thought of and used as a single component in a view.

- **Using Page**
  - The page in which a composite component is used.

- **Composite Component Tag Instance**
  - The tag that places an instance of a composite component in the using page.

- **Defining Page**
  - The markup page, usually Facelets that contains the composite component declaration and definition.
Coming to Terms with Composite Components

- **Top Level Component**
  - The UIComponent instance implicitly created by the run-time to serve as the root for the composite component tree.

- **Inner Component**
  - Any UIComponent inside the defining page or any pages used by that defining page.
Other Composite Component Concepts

- Method Expressions DEMO
- Runtime Metadata
- Resources
- Per-component ResourceBundle
- Custom top level components, also from Script
End
How to think about Composite Components

- Building components should not be scary or hard
- Make everything private and expose only what the page author needs to see
- Just do it!