



About Faces: The JavaServer™ Faces Framework

Edward Burns

Senior Staff Engineer
Sun Microsystems Inc.



Agenda – JavaServer™ Faces

- Is powerful
 - > Expressive power
 - > Integration power
- Is easy to use
 - > Inside tools
 - > “by hand”
- Has market and mindshare
 - > Industry support
 - > Adoption

Faces: Expressive Power

**“Simple things should be simple.
Complex things should be
possible.”**

Alan Kay



Faces: Expressive Power

Model-View-Controller (MVC)

- We all talk about it, but do our tools really support it?
 - > Model: managed-beans
 - > View: JSP (or other templating) pages
 - > Controller: Supporting framework
- Sometimes it's better to break the rules
 - > Prototyping: put everything in one page
 - > Throwaway projects

Faces: Expressive Power

Rendering Technology Agnostic Component Model

- Built on JavaBeans concepts
 - > UIComponent hierarchy
 - > Properties
 - > Methods
 - > Events
 - > Event model
 - > Listener classes
 - > Event objects
- Combine with RenderKit concept

Faces: Expressive Power

POJO Development and Dependency Injection (aka IoC)

- Plain-Ole-Java-Objects – why bother extending base classes or implementing interfaces?
- Expose or define your business logic with faces managed-beans
- Entire object graph can be stated declaratively
- Exposed via the EL or directly through Servlet API calls
- Lazy instantiation and setter injection

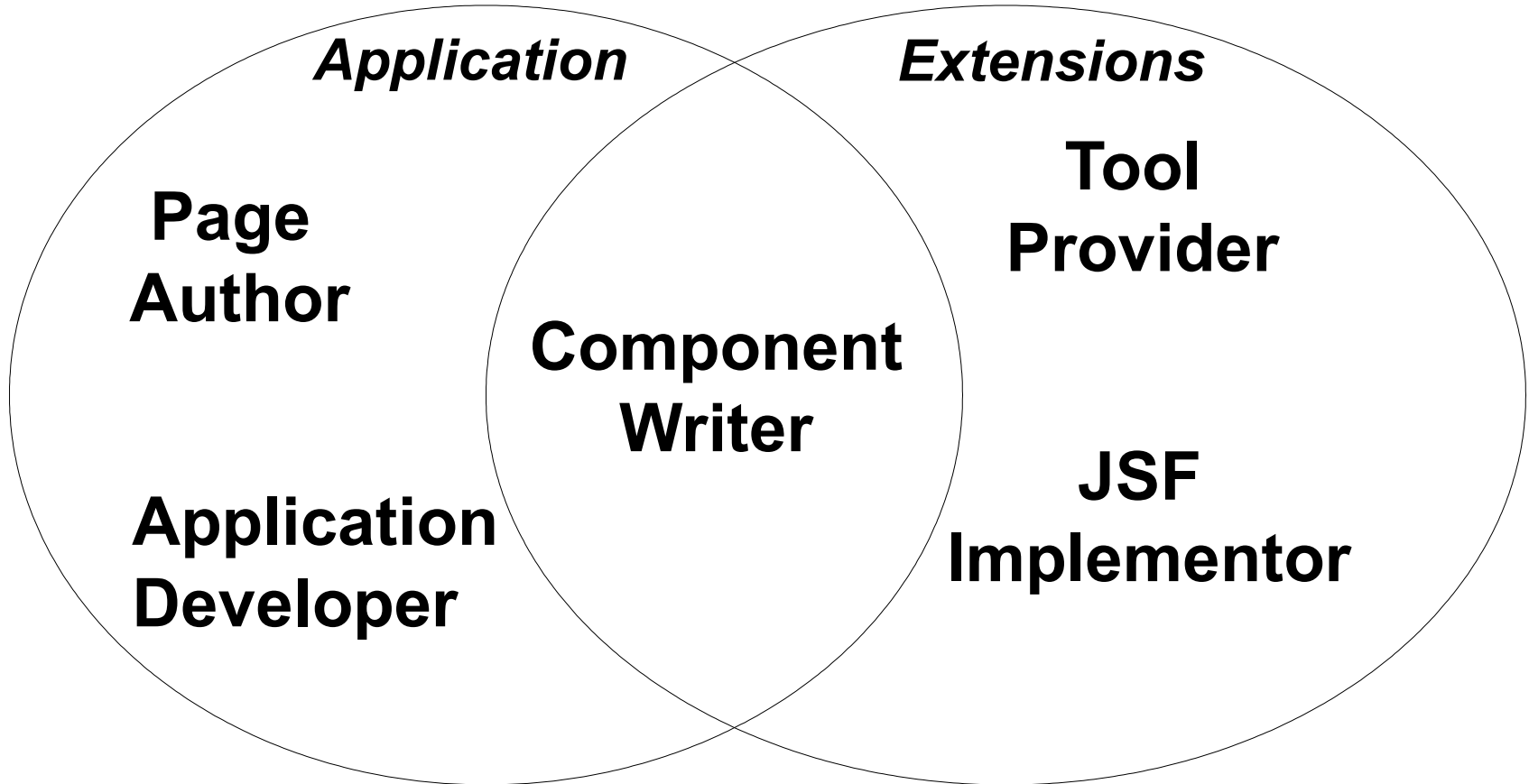
Faces: Expressive Power

Expression Language

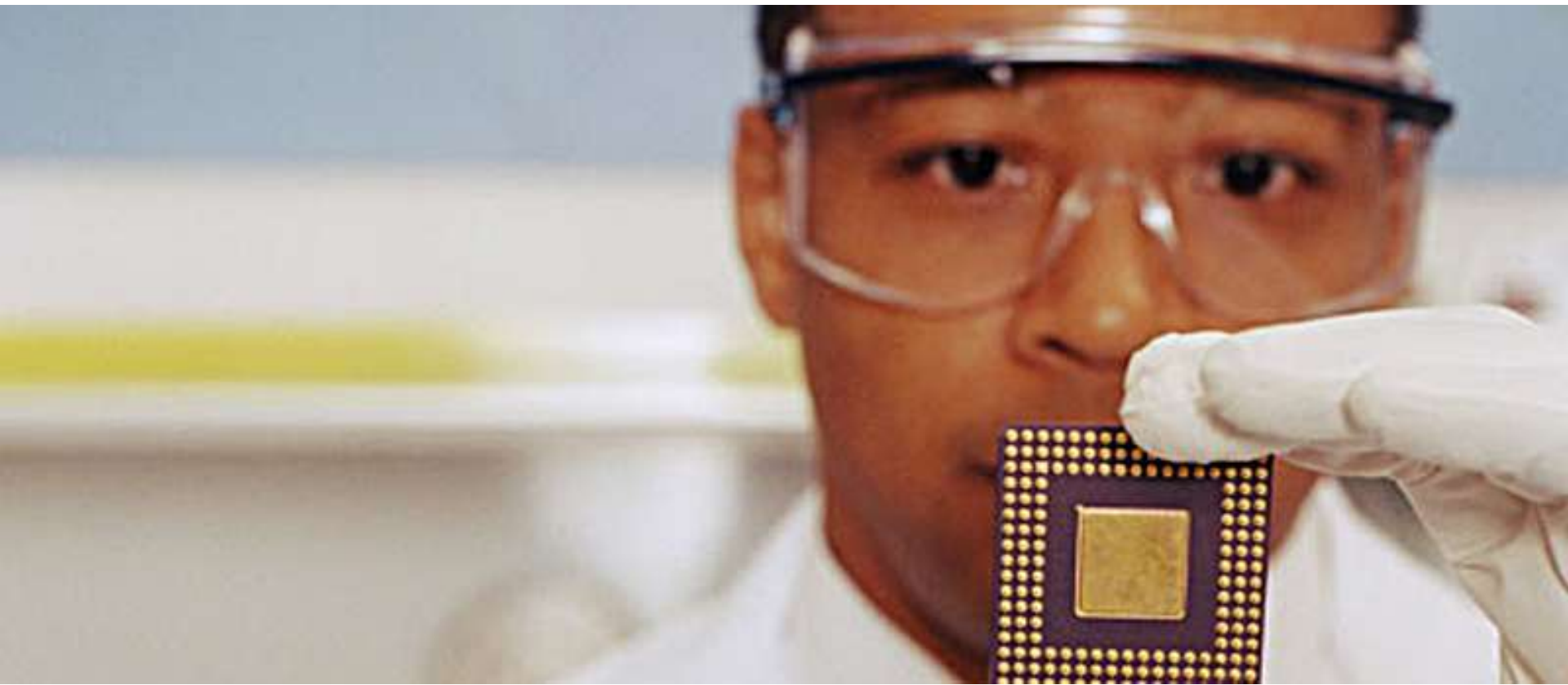
- Navigate your entire business logic object graph
- Set and Get operations: render and postback
- JavaBeans properties, Arrays, Lists, Maps, Collections
- Implicit objects from Servlet environment: cookie, headers, initParams
- Example:
 - > `#{currentUser.prefs.sendSpam}`
 - > `#{products[i].suppliers.name.address}`

Faces: Expressive Power

Separation of Roles by Skillset



Faces: the importance of integration

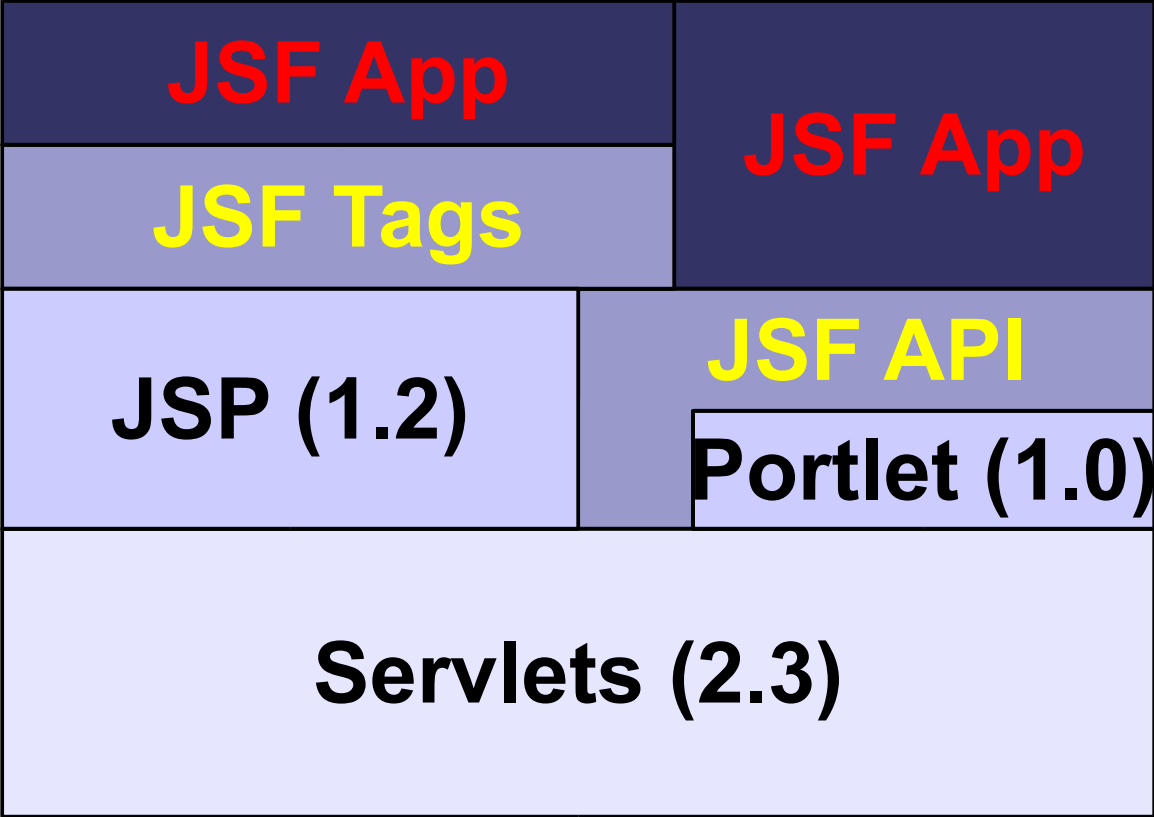


Faces: The Power of Integration

- JSP and JSTL: now works better than ever with Faces
- Easy to integrate back-end logic
 - > `java.sql.ResultSet` and `java.sql.RowSet`
 - > Resources using the `@Resource` annotation (maybe)
 - > Portlet support: designed from the ground up with JSR-168 in mind.
- Massively extensible: nearly everything is overrideable, delegatable, or decoratable

JSF Version History

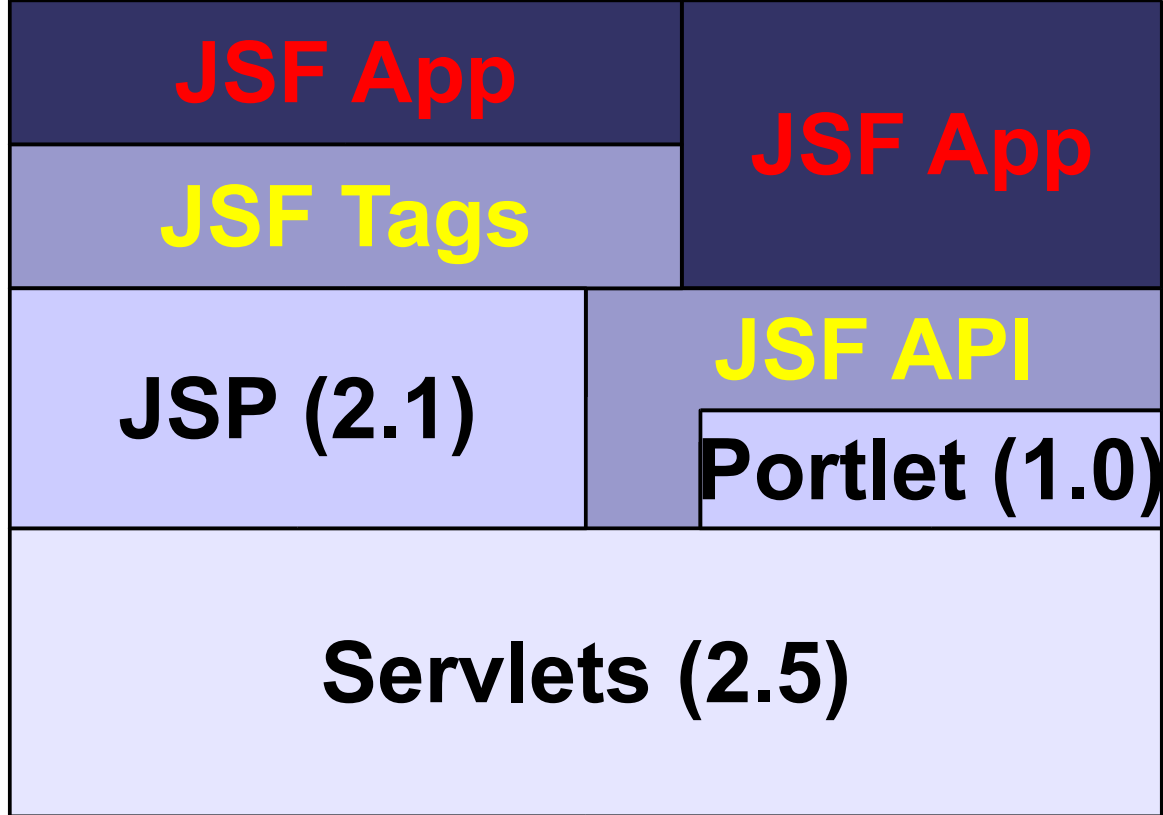
- 1.0 on 11 March 2004
- 1.1 on 27 May 2004
- Based on J2EE 1.3
- Implementation bundled with J2EE 1.4 SDK



JSF Version History

- 1.2 in Proposed Final Draft 25 August 2005

- A core part of Java EE 5
- Implementation bundled with Glassfish, Sun's open-source App Server



Faces: The Power of Making it Easy

- Develop with a tool
- Develop “by hand” aka Emacs or vi.



Faces: Coding “by hand”

- No harder than with any other framework
- Simple login example:

```
<html>
<head>
<%@ taglib uri="http://java.sun.com/jsf/html" prefix="h" %>
<%@ taglib uri="http://java.sun.com/jsf/core" prefix="f" %>
</head>
<body>
<f:view>
  <h:panelGrid columns="2">
    name: <h:inputText value="#{user.username}"/>
    password: <h:inputText value="#{user.password}" />
  </h:panelGrid>
  <h:commandButton action="#{user.loginAction}"/>
</f:view></body></html>
```

Faces: Coding “by hand”

- Managed bean

```
public class User {  
    private String username;  
    private String password;  
    public void setUsername(String n) { this.username = n};  
    public void setPassword(String p) { this.password = p};  
    public String getUsername(void) { return username; }  
    public String getPassword(void) { return password; }  
    public String loginAction() {  
        if (validLogin) { return "success"; }  
        return "failure";  
    }  
}
```

Faces: Coding “by hand”

- Front page (after successful login)

```
<html>
<head>
<%@ taglib uri="http://java.sun.com/jsf/html" prefix="h" %>
<%@ taglib uri="http://java.sun.com/jsf/core" prefix="f" %>
<%@ taglib uri="http://java.sun.com/jstl/core" prefix="c" %>
</head>
<body>
<f:view>
  <f:subview><c:import url="storeFrontPane.jsp"/>
  </f:subview>
</f:view></body></html>
```

Faces: Coding “by hand”

- Config File

```
<managed-bean><managed-bean-name>user</managed-bean-name>  
  <managed-bean-class>User</managed-bean-class>  
  <managed-bean-scope>request</managed-bean-scope>  
</managed-bean>  
<navigation-rule>  
  <from-view-id>/login.jsp</from-view-id>  
  <navigation-case>  
    <from-outcome>success</from-outcome>  
    <to-view-id> /storeFront.jsp </to-view-id>  
  </navigation-case>  
  <navigation-case>  
    <from-outcome>failure</from-outcome>  
    <to-view-id> /login.jsp </to-view-id>  
  </navigation-case>  
</navigation-rule>
```

Faces: Industry Power

- Developer tools already mentioned
- Component libraries
 - > BusinessObjects Crystal Reports
 - > Oracle ADF Faces
 - > ILOG JViews Chart components
- Job postings: three pages of Faces related Jobs on Monster.com
- Five books, from all the major publishers

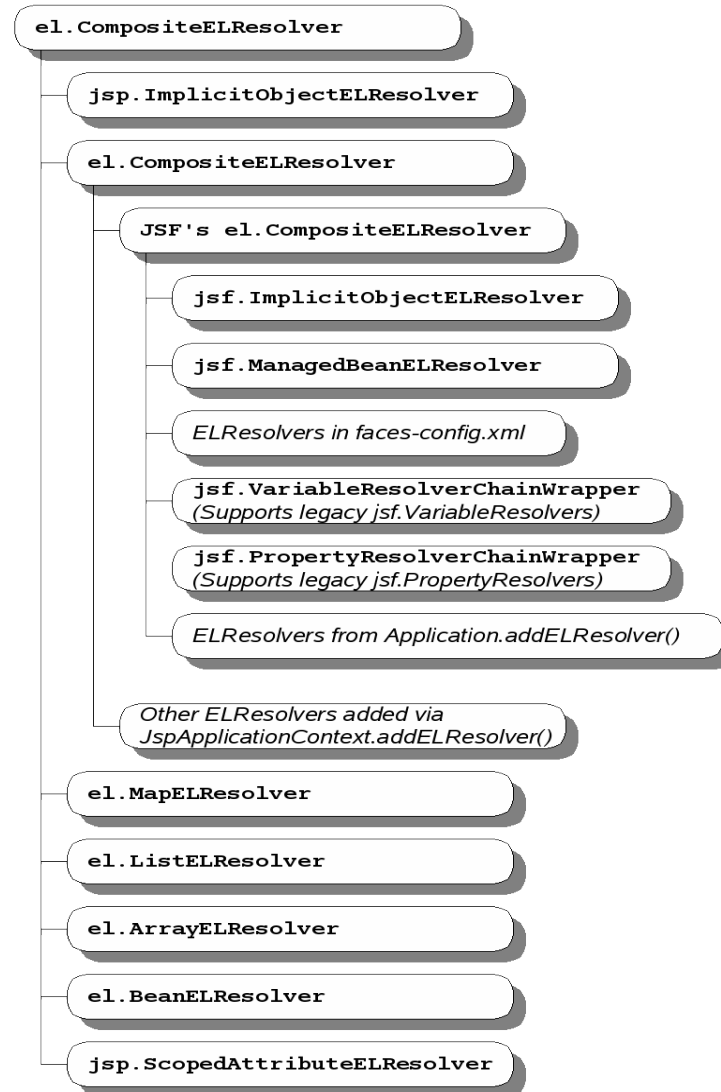
What's next?

- Faces 1.2 and JSP 2.1 (in proposed final draft)
 - > Part of Java EE 5
 - > Dependent on JDK 1.5 / Servlet 2.5
 - > Done via transparent development on java.net
 - > Faces licensed with JDL/JRL
 - > New features – Unified EL
 - > Ultimately in Servlet spec or maybe JDK
 - > Bring new features from Faces 1.0/1.1 EL into JSP
 - > Align the webtier technologies
 - > Break out into separate spec document under JSP 2.1 spec
 - > Defined in new *javax.el* package

What's next?

- > Changes for Unified EL
 - > Page author experience changes:
 - Existing apps still run without change via inspection of web.xml version level
 - Possible to escape “#{” so JSP container doesn't touch it
 - > Java Developer experience changes:
 - ELResolver replaces VariableResolver / PropertyResolver
 - Legacy Variable/Property Resolvers still work
 - > Faces Implementor changes:
 - Encouraged to use new EL APIs
 - Leverage javax.el package

Resolver returned from `JspContext.getELContext().getELResolver()`:



ELResolvers are combined together using CompositeELResolvers, to define rich semantics for evaluating an expression.

Pluggability was the motivation for the ELResolver architecture.

- Spring
- Seam
- Shale

What's next?

- Other Faces 2.1 features (not all mentioned here):
 - > Wrappers for commonly decorated objects
 - > “binding” attribute for converter/validator/listener tags
 - > XML schema instead of DTD
 - > Additional “dir”, “lang” attributes for outputText, outputFormat, messages, message
 - > PhaseListener must guarantee that if “beforePhase()” is called, then “afterPhase()” must also be called
 - > UIViewRoot phaseEvents
 - > “caption” facet on DataTable
 - > Content interweaving
 - > Works perfectly with JSTL 1.2 (new version in Java EE 5)
 - > Tree pre-creation / Content interweaving
 - > Associating label with component for use in messages

What's next?

Resource Injection

- Access to JNDI resources through annotations (`@Resource`, `@EJB...`) standardized across Java EE
- `@PostConstruct`, `@PreDestroy`
- Servlets, Servlet Filters, Servlet Listeners
- Tag Handlers, JSP listeners
- Managed Beans



About Faces: The JavaServer™ Faces Framework

Ed Burns

ed.burns@sun.com

