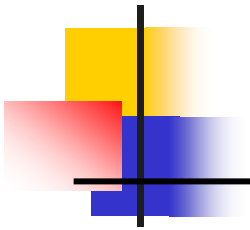
About JavaServer™ Faces *(a.k.a.* Making Faces)

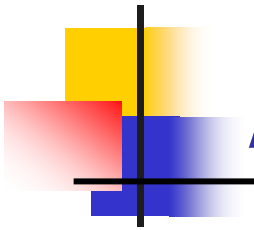
Stephen A. Stelting

Sun Microsystems, Inc.



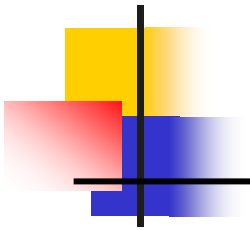
Session Objectives

- Learn about JavaServer Faces (JSF)
- Show how to set up and configure a JSF Web application
- Present standard features of JSF and show how they can benefit Web applications
- Discuss trade-offs for JSF use



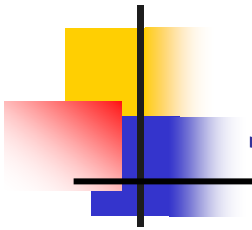
Agenda

- What is JavaServer™ Faces (JSF)?
  - Purpose, Goals, Features
- Installation and setup
- Demo – Adding JSF functionality to a Web App
  - GUI Widgets
  - Form processing
  - Business Components
  - i18n and l10n
  - Navigation



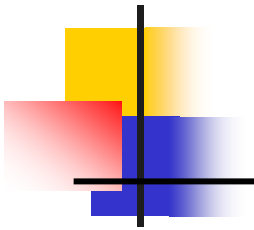
What Is JavaServer Faces?

- Java Web API
- Developed as JSR 127
- Used to build rich Web user interfaces
- Consists of Java classes (for use by Servlets) and custom tag libraries (for JSP use)



JavaServer Faces – Goals

- Make it easier to write and maintain applications that render user interfaces (UIs) to a client
- Provide an extensible framework, allowing custom Luis to be created and used
- Manage UI state across multiple requests
- Provide an event model for server-side code
- Integrate with tool-based technology, for greater ease in development

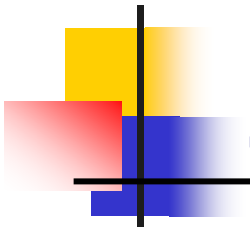


Downloading JavaServer Faces

To get the JSF API and RI, download the Java Web Services Developer Pack 1.2 (JWSDP)

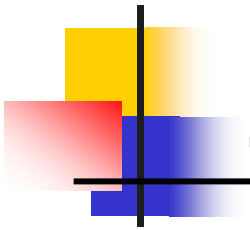
http://java.sun.com/webservices/downloads/webservicespack.html

JavaServer Faces is located in the /jsf subdirectory



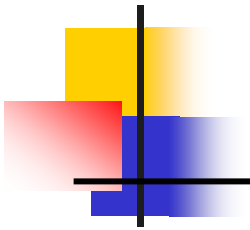
JSF – Base Requirements

- You need the following to run JavaServer Faces:
  - J2SE 1.3.1 or later
  - Servlet 2.3
  - JSP 1.2
- "Standard" Web server environments:
  - JWSDP 1.2 (preconfigured with Tomcat)
  - Any of the Tomcat 4.x builds



JSF – JAR File Requirements

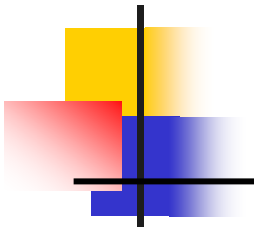
- The JavaServer Faces RI has two JAR files:
  - jsf-api.jar
  - jsf-ri.jar
- Additionally, JSF requires JAR files from a few other APIs in the JWSDP:
  - Java Standard Tag Library (jstl.jar, standard.jar)
  - Shared library resources (commons-beanutils.jar, common-collections.jar, common-digester.jar, common-logging.jar)



Using JSF in a Web Application

To use JSF in a Web application,

1) The WEB-INF/web.xml file must include a standard JavaServer Faces mapping
2) The JSF standard library files must be copied to the WEB-INF/lib directory
3) The Web application must have an application configuration file called WEB-INF/faces-config.xml
4) The Web application must invoke FacesServlet before using JSF features
5) JSPs must be set up to use JSF

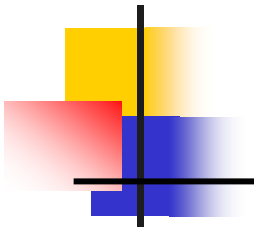


Using JSF – web.xml Config

```
<web-app>
  <!-- FacesServlet defined for the Web app -->
  <servlet>
    <servlet-name>Faces Servlet</servlet-name>
    <servlet-class>
      javax.faces.webapp.FacesServlet
    </servlet-class>
    <load-on-startup>1</load-on-startup>
  </servlet>
  <!-- Map FacesServlet to URLs with /faces/* -->
  <servlet-mapping>
    <servlet-name>Faces Servlet</servlet-name>
    <url-pattern>/faces/*</url-pattern>
  </servlet-mapping>
</web-app>
```

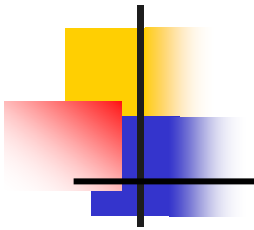
Using JSF – JAR requirements

- From jwsdp-1.2\jsf\lib
  - jsf-api.jar — JSF Core API
  - jsf-ri.jar — JSF Reference Implementation
  - jstl.jar — JSTL support
  - standard.jar — JSTL support
- From jwsdp-1.2\jwsdp-shared\lib
  - commons-beanutils.jar — Java Beans support
  - commons-digester.jar — XML processing
  - commons-collections.jar — Collections extensions
  - commons-logging.jar — Logging API



Using JSF – faces-config.xml

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

<!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>

  <!-- JSF configuration code goes here -->

</faces-config>
```



Using JSF – Invoke FacesServlet

- To use JSF, you must first invoke FacesServlet that is mapped in WEB-INF/web.xml
- Two ways to do this:
  - Add an HTML page that uses the path of FacesServlet to reference the first JSF

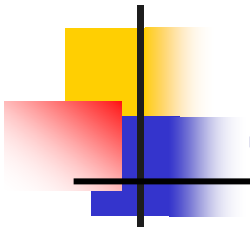
    ```
    <a href="faces/EntryForm.jsp">JSF Page</a>
    ```

  - Directly reference the path in the URL:

    http://javafolks.org/ctx/faces/JSF.jsp

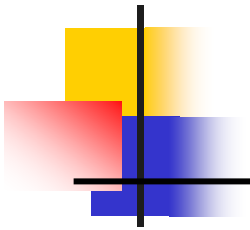
Using JSF in a JSP

```
<!-- JSF Applications use these two tag libraries -->
<%@taglib uri="http://java.sun.com/jsf/html" prefix="h" %>
<%@taglib uri="http://java.sun.com/jsf/core" prefix="f" %>



<!-- Any JSF tags must be embedded in the use_faces tag -->
<f:use_faces>

  <!-- Other JSF tags go here -->

</f:use_faces>
```



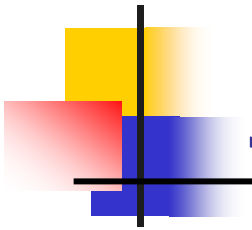
JSF – Key Concepts

- JSF request processing model
  - FacesServlet
  - Lifecycle
- JSF global configuration using faces-config.xml
- Component model
  - UIComponent
  - Tree model used for component hierarchy



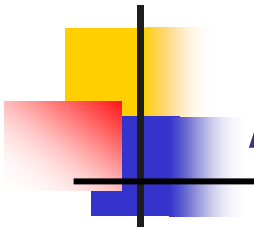
Standard JSF Processing Cycle

- Central to JSF processing is a Servlet – the javax.faces.FacesServlet
- This Servlet acts as a controller for JSF – it marshalls resources and calls crucial methods
- Specifically, for a request, FacesServlet will
  1) Get a javax.faces.FacesContext object
  2) Get a javax.faces.lifecycle.Lifecycle object
  3) Call the Lifecycle method execute(FacesContext)
  4) Call the FacesContext method release



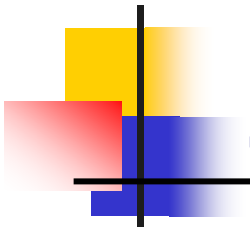
JSF Processing Cycle *(Continued)*

- The javax.faces.lifecycle.Lifecycle class is the processing delegate of FacesServlet
- During a request, Lifecycle does the following:
  1) Builds (or rebuilds) a JSF component tree
  2) Applies request values to components
  3) Processes validations for new request values
  4) Updates model values
  5) Invokes application behavior using event-based model
  6) Renders response to the client



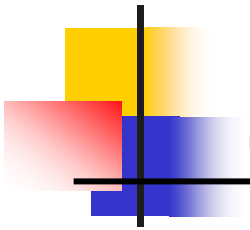
About faces-config.xml

- Every JSF application has an XML config file
- Default is located in the WEB-INF directory
- Lets you declaratively define JSF elements, like:
  - Navigation rules (used to handle page flow)
  - Java Beans components
  - Rendering kits (used to visually depict components)
  - Converters and Validators
  - Custom UI components



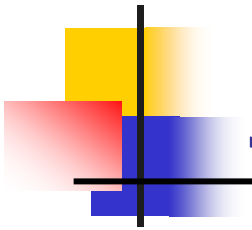
JSF – UI Components

- In JSF, graphical behavior is managed through user interface components
- Basic behavior for components is defined in
  - javax.faces.component.UIComponent (interface)
  - javax.faces.component.UIComponentBase (abstract class)



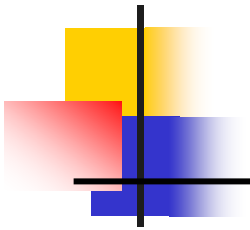
JSF – UI Components *(Continued)*

- Components can use the following APIs for extended functionality:
  - Converters: Converts between a component's value and an appropriate value for the model
  - Listeners/Events: Provides notification of state changes for some types of component
  - Validators: Verifies that the data associated with a component conforms to business rules; validators can send an error to the user if verification fails
  - Renderers: Provides encoding and decoding for a component



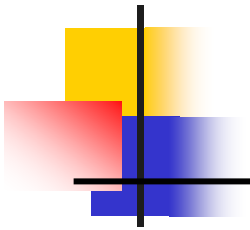
JSF – Features Demonstration

- During the rest of the talk, we'll demonstrate JSF capabilities with a sample Web App
- Progressively add JSF functionality to it
- JavaServer Faces capabilities:
  - Basic form elements
  - Association with model elements
  - Event notification
  - Page navigation
  - Localization and internationalization (l10n and i18n)



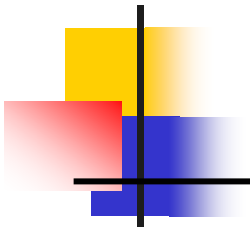
Benefits of JSF

- Benefits of JSF include
  - Rich component functionality
  - Support for event-based notification
  - Support for i18n and l10n
  - XML configuration of things like navigation
  - Built-in integration with beans (Model components)
  - Variable representation of UI components (renderers)
  - Easy integration with builder tools



Drawbacks of JSF

- What's the cost?
  - Non-trivial to set up (with the current version)
  - Complex processing cycle for each JSF request
  - Performance cost to use JSF



Summary

In this session, we

- Discussed what JavaServer Faces is, and what problems it is intended to solve
- Showed how to set up a JSF application
- Talked about general principles of a JSF application
- Looked at a demo of JSF capabilities
- Described benefits and drawbacks of JSF technology

Questions