JSF: Introduction, Installation, and Setup

Originals of Slides and Source Code for Examples:
http://www.coreservlets.com/JSF-Tutorial/

This somewhat old tutorial covers JSF 1, and is left online for those maintaining existing projects. All **new** projects should use JSF 2, which is both simpler and more powerful. See http://www.coreservlets.com/JSF-Tutorial/jsf2/.

Customized Java EE Training: http://courses.coreservlets.com/
Java, JSF 2, PrimeFaces, Servlets, JSP, Ajax, jQuery, Spring, Hibernate, RESTful Web Services, Hadoop, Android.
Developed and taught by well-known author and developer. At public venues or onsite at *your* location.

For live training on JSF 1 or 2, please see courses at http://courses.coreservlets.com/.

Taught by the author of *Core Servlets and JSP*, *More Servlets and JSP*, and this tutorial. Available at public venues, or customized versions can be held on-site at your organization.

- Courses developed and taught by Marty Hall
  – JSF 2, PrimeFaces, servlets/JSP, Ajax, jQuery, Android development, Java 6 or 7 programming, custom mix of topics
  – Ajax courses can concentrate on 1 library (jQuery, Prototype/Scriptaculous, Ext-JS, Dojo, etc.) or survey several
- Courses developed and taught by coreservlets.com experts (edited by Marty)
  – Spring, Hibernate/JPA, EJB3, GWT, Hadoop, SOAP-based and RESTful Web Services
  Contact hall@coreservlets.com for details

Topics in This Section

- **Understanding JSF**
  - Different views of JSF
  - Comparing JSF to standard servlet/JSP technology
    - Pros *and* cons
  - Comparing JSF to Apache Struts
    - Pros *and* cons
- **Setting Up JSF**
  - Downloading and configuring JSF
    - Apache MyFaces
    - Sun Reference Implementation
  - Testing JSF
  - Setting up JSF applications
  - Accessing JSF documentation

4

Overview

5

What is JSF?

- **A set of Web-based GUI controls and associated handlers?**
  – JSF provides many prebuilt HTML-oriented GUI controls, along with code to handle their events.
- **A device-independent GUI control framework?**
  – JSF can be used to generate graphics in formats other than HTML, using protocols other than HTTP.
- **A better Struts?**
  – Like Apache Struts, JSF can be viewed as an MVC framework for building HTML forms, validating their values, invoking business logic, and displaying results.
- **But which is the <u>proper</u> way to view JSF?**
  – The answer depends on what you are going to use it for, but the 1 & 3 are the most common way of looking at JSF.

Advantages of JSF (vs. MVC Using RequestDispatcher)

- **Custom GUI controls**
  – JSF provides a set of APIs and associated custom tags to create HTML forms that have complex interfaces
- **Event handling**
  – JSF makes it easy to designate Java code that is invoked when forms are submitted. The code can respond to particular buttons, changes in particular values, certain user selections, and so on.
- **Managed beans**
  – In JSP, you can use property="*" with jsp:setProperty to automatically populate a bean based on request parameters. JSF extends this capability and adds in several utilities, all of which serve to greatly simplify request param processing.
- **Expression Language**
  – JSF provides a concise and powerful language for accessing bean properties and collection elements

Advantages of JSF (vs. Standard MVC), Continued

- **Form field conversion and validation**
  - JSF has builtin capabilities for checking that form values are in the required format and for converting from strings to various other data types. If values are missing or in an improper format, the form can be automatically redisplayed with error messages and with the previously entered values maintained.
- **Centralized file-based configuration**
  - Rather then hard-coding information into Java programs, many JSF values are represented in XML or property files. This loose coupling means that many changes can be made without modifying or recompiling Java code, and that wholesale changes can be made by editing a single file. This approach also lets Java and Web developers focus on their specific tasks without needing to know about the overall system layout.
- **Consistent approach**
  - JSF encourages consistent use of MVC throughout your application.

Disadvantages of JSF (vs. MVC with RequestDispatcher)

- **Bigger learning curve**
  - To use MVC with the standard RequestDispatcher, you need to be comfortable with the standard JSP and servlet APIs. To use MVC with JSF, you have to be comfortable with the standard JSP and servlet APIs *and* a large and elaborate framework that is almost equal in size to the core system. This drawback is especially significant with smaller projects, near-term deadlines, and less experienced developers; you could spend as much time learning JSF as building your actual system.
- **Worse documentation**
  - Compared to the standard servlet and JSP APIs, JSF has fewer online resources, and many first-time users find the online JSF documentation confusing and poorly organized. MyFaces is particularly bad.

Disadvantages of JSF (vs. Standard MVC), Continued

- **Less transparent**
  - With JSF applications, there is a lot more going on behind the scenes than with normal Java-based Web applications. As a result, JSF applications are:
    - Harder to understand
    - Harder to benchmark and optimize
- **Undeveloped tool support**
  - There are many IDEs with strong support for standard servlet and JSP technology. Support for JSF is only beginning to emerge, and final level is yet to be determined.
- **Rigid approach**
  - The flip side of the benefit that JSF encourages a consistent approach to MVC is that JSF makes it difficult to use other approaches.

10

Advantages of JSF (vs. Struts)

- **Custom components**
  - JSF makes it relatively easy to combine complex GUIs into a single manageable component; Struts does not
- **Better support for Ajax**
  - Several third-party component libraries have extensive Ajax support (Apache Tomahawk, JBoss Ajax4jsf/RichFaces, Oracle ADF, IceFaces). Struts doesn't have real component libraries (see above).
- **Support for other display technologies**
  - JSF is not limited to HTML and HTTP; Struts is
- **Access to beans by name**
  - JSF lets you assign names to beans, then you refer to them by name in the forms. Struts has a complex process with several levels of indirection.

11

Advantages of JSF (vs. Struts), Continued

- **Expression language**
  - The JSF expression language is more concise and powerful than the Struts bean:write tag.
    - This is less advantageous if using JSP 2.0 anyhow.
- **Simpler controller and bean definitions**
  - JSF does not require your controller and bean classes to extend any particular parent class (e.g., Action) or use any particular method (e.g., execute). Struts does.
- **Simpler config file and overall structure**
  - The faces-config.xml file is much easier to use than is the struts-config.xml file. In general, JSF is simpler.
- **More powerful potential tool support**
  - The orientation around GUI controls and their handlers opens possibility of simple to use, drag-and-drop IDEs

Disadvantages of JSF (vs. Struts)

- **Struts: established base and industry momentum**
  - 4/2009 search at dice.com and monster.com
    - "struts": 896 jobs (dice.com), 525 jobs (monster.com)
    - "jsf":   732 (dice.com),   694 (monster.com)
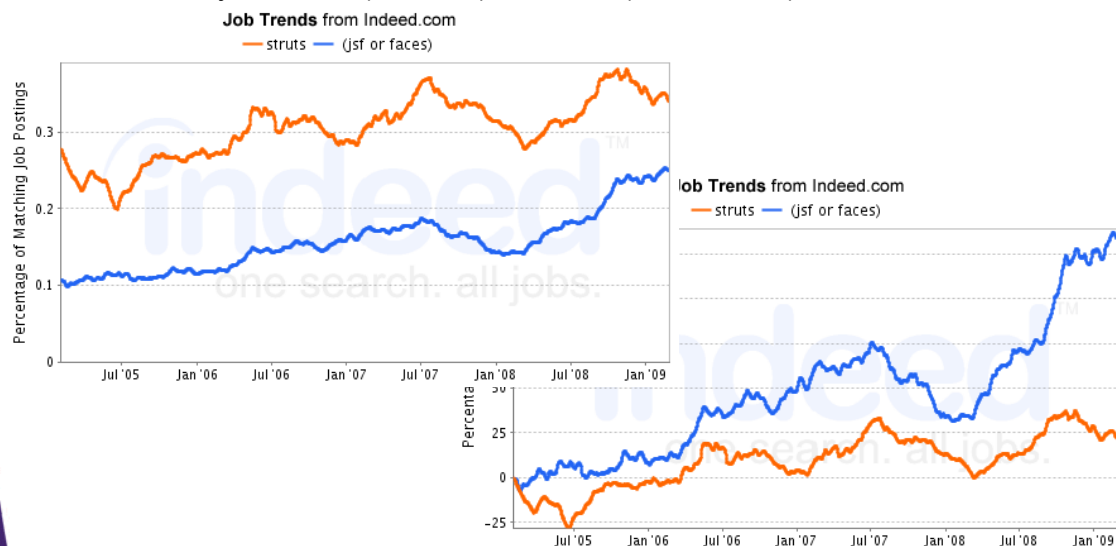
Disadvantages of JSF (vs. Struts), Continued

- **Support for other display technologies**
  - JSF is not limited to HTML and HTTP; Struts is
    - Hey! Didn't I say this was an *advantage* of JSF?
- **Confusion vs. file names**
  - The actual pages used in JSF end in *.jsp*. But the URLs used end in *.faces* or *.jsf*. This causes many problems; in particular, in JSF:
    - You cannot browse directories and click on links
    - It is hard to protect raw JSP pages from access
    - It is hard to refer to non-faces pages in faces-config.xml
- **Self-submit approach**
  - With Struts, the form (blah.*jsp*) and the handler (blah.*do*)have different URLs; with JSF they are the same.

Disadvantages of JSF (vs. Struts), Continued

- **No builtin equivalent to Tiles**
  - Struts comes with a page layout facility; JSF does not
    - Facelets is not yet part of JSF, but is now mainstream
    - Or, you can extract Tiles from Struts and use it with JSF
- **Much weaker automatic validation**
  - Struts comes with validators for email address, credit card numbers, regular expressions, and more. JSF only comes with validators for missing values, length of input, and numbers in a given range.
    - But many powerful 3rd-party validators (including MyFaces)
    - You can use the Struts/Commons validation library with JSF
- **Lack of client-side validation**
  - Struts supports JavaScript-based form-field validation; JSF does not
- **Worse installation**
  - Struts has the struts-blank application to use for a starting point; JSF has nothing similar

JSF and Struts: The Future

- **Possibilities**
  - JSF will fail and developers that want a framework will stick with Struts
    - JSF will die
      - Doubtful at this point, but other MVC frameworks have failed
  - JSF will flourish and replace Struts
    - Struts will die
  - JSF will grow moderately, and developers will be split
    - Both Struts and JSF will be widely used frameworks
- **Prediction is difficult**
  - Technical factors are not usually what decide these things
- **Recommendations**
  - Move ongoing Struts projects to JSF: not yet
  - Start real-world JSF projects: yes, but with some caution

16

Installation

17

Installing and Configuring JSF: Summary

- **To run JSF, you need three things:**
  - Certain JAR files in WEB-INF/lib
    - Specific to the implementation.
  - Certain web.xml entries
    - Must match the specific implementation.
  - A blank faces-config.xml file in WEB-INF
    - Standard. Nothing specific to MyFaces.
- **jsf-blank-myfaces**
  - Blank Eclipse project with pieces taken from Apache
  - Downloadable from http://coreservlets.com/JSF-Tutorial/
  - Import into Eclipse, then copy/paste for new projects
    - Copying projects in Eclipse is painful; see later slide

Installing and Configuring JSF: Details

- **Download the JSF files from one of two places**
  - Get jsf-blank-myfaces from coreservlets.com
    - http://www.coreservlets.com/JSF-Tutorial/
    - Files taken from the MyFaces distribution
    - Configured as an Eclipse project
  - Download from http://myfaces.apache.org/download.html
    - JAR files in WEB-INF/lib (some missing!)
    - Hard to find sample web.xml or faces-config.xml
- **Make compatible Eclipse project**
  - Copy jsf-blank-myfaces
    - Requires you to edit .settings/org.eclipse.wst.common.component
  - Or, create a new Dynamic Web Project in Eclipse
    - Copy JAR files to WebContent/WEB-INF/lib
    - Copy entries to WebContent/WEB-INF/web.xml
    - Copy faces-config.xml to WebContent/WEB-INF
    - Add project facet for JSF so that Eclipse is smart about editing JSF tags in .jsp files and so that it uses the faces-config.xml editor

Bookmark the JSF Documentation

- **API Javadocs**
  - http://java.sun.com/j2ee/javaserverfaces/1.1/docs/api/
    - Same as the MyFaces version
- **Tag library docs**
  - http://java.sun.com/j2ee/javaserverfaces/1.1/docs/tlddocs/
    - Much better than the MyFaces version
- **MyFaces References**
  - http://myfaces.apache.org/
    - User's Guide, extensions documentation, FAQs, etc.
- **faces-config.xml annotated DTD**
  - http://horstmann.com/corejsf/faces-config.html
    - From Cay Horstmann

JSF Books (In Order of My Personal Preference)

- ***JavaServer Faces, the Complete Reference***
  - By Schalk and Burns
- ***JavaServer Faces in Action***
  - By Kito Mann
- ***Core JavaServer Faces***
  - By Geary and Horstmann
    - Second edition available as of May '07
- ***JavaServer Faces***
  - By Hans Bergsten
- ***Mastering JavaServer Faces***
  - By Dudney, et al

Development Process by Copying "Blank" Project

22

The jsf-blank-myfaces Project

- **Ready-to-extend Eclipse project**
  - Contains all required standard JAR files
  - Contains minimal web.xml entries
  - Contains blank faces-config.xml file
  - Added project facet so that Eclipse is JSF-aware
    - Especially so special editor is used for faces-config.xml
  - Download jsf-blank-myfaces from http://www.coreservlets.com/JSF-Tutorial/
    - Web site also has extended version of "blank" project that contains MyFaces extensions

23

Developing in Eclipse by Copying "Blank" Project

- **Why copy?**
  - Creating new JSF apps in Eclipse is cumbersome
    - R-click project → Properties → Project Facets → JavaServer Faces has problems if web.xml exists already
    - You have to configure JAR file locations
- **Issue**
  - Cutting/pasting Web app in Eclipse does not change the context path and internal project deploy name
  - Context path can be changed by R-clicking project, then Properties → Web Project Settings
  - But deployed name is not accessible from within Eclipse!
- **Solution**
  - Go to file system, edit .settings/org.eclipse.wst.common.component
  - Change *two* instances of old project name to new one
  - R-click on project and choose Refresh

Copying jsf-blank-myfaces: Example

- **First, copy project**
  - R-click on jsf-blank-myfaces, choose "Copy"
  - R-click in Project Explorer window, choose "Paste"
    - E.g., name it my-jsf-project
- **Next, edit .component file**
  - Navigate to Eclipse workspace/*projectName*/.settings
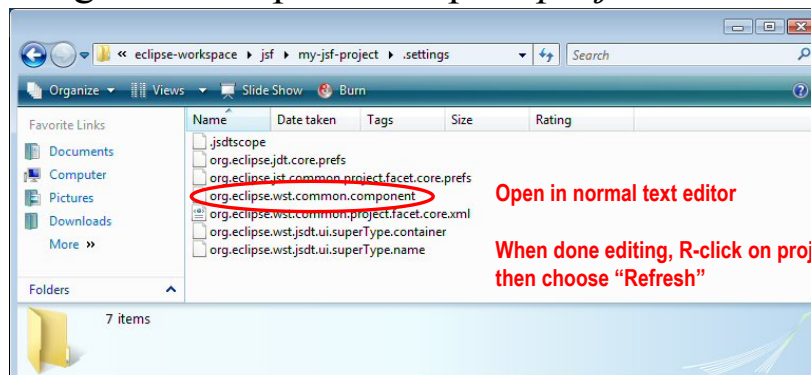
Copying jsf-blank-myfaces: Example (Continued)

- **.component file: before**

```xml
<?xml version="1.0" encoding="UTF-8"?>
<project-modules id="moduleCoreId" project-version="1.5.0">
    <wb-module deploy-name="jsf-blank-myfaces">
        <wb-resource deploy-path="/" source-path="/WebContent"/>
        <wb-resource deploy-path="/WEB-INF/classes" source-path="/src"/>
        <property name="context-root" value="jsf-blank-myfaces"/>
        <property name="java-output-path"/>
    </wb-module>
</project-modules>
```

- **.component file: after**

```xml
<?xml version="1.0" encoding="UTF-8"?>
<project-modules id="moduleCoreId" project-version="1.5.0">
    <wb-module deploy-name="my-jsf-project">
        <wb-resource deploy-path="/" source-path="/WebContent"/>
        <wb-resource deploy-path="/WEB-INF/classes" source-path="/src"/>
        <property name="context-root" value="my-jsf-project"/>
        <property name="java-output-path"/>
    </wb-module>
</project-modules>
```

Using jsf-blank as Starting Point for New JSF Applications

- **Copy and rename jsf-blank-myfaces**
  - R-click, Copy, R-click, paste
  - Edit .component file and change name twice as on previous slides
- **Add JSF content**
  - Edit WebContent/WEB-INF/faces-config.xml
  - Add JSP, HTML, and other Web content to WebContent
    - .jsp files will use JSF-specific tags
  - Add Java files to src
- **Deploy normally**
  - R-click server in Eclipse, "Add and Remove Projects", Select project, R-click, Restart
    - If you do not have a Java-enabled server set up in Eclipse, see http://www.coreservlets.com/Apache-Tomcat-Tutorial/

Summary

- **JSF provides many useful features for developing complex GUIs & handling events**
- **Standard servlet and JSP technology is a viable alternative**
  - Especially when the MVC approach and the JSP 2.0 expression language is used
- **Struts is also a viable alternative**
  - Future of JSF vs. Struts is unknown
- **To get started**
  - Download jsf-blank-myfaces from http://www.coreservlets.com/JSF-Tutorial/
    - Minimal (standard) version or version with MyFaces extensions

28

Questions?