UW Business School

JavaServer Faces (Intro)

MSIS 531 – Spring 2006

Learning Goals

- Old school: JavaServer Pages (JSP)
- Introducing JavaServer Faces (JSF)
- JSF application overview and structure
- JavaBeans and their role in JSF applications
  - JavaBean structure
  - Managed beans
- JSF Navigation
  - Recall HTTP is stateless; "what to do next" must be stated completely and robustly
- Standard JSF tags
  - Core JSF, HTML support

JavaServer Pages (JSP)

- Old-school J2EE view components
  - Combination of Java code and HTML
  - Clear competitor for ASP pages
    - Not sure who came first
    - Tags are very similar
- Implementation issues:
  - Easy for developers to combine presentation, business logic in JSP pages
    - Lacks "separation of concerns"
  - Mixed HTML, Java makes code cleanliness, maintenance an issue
    - Simple examples illustrate this

Examples: echo.jsp, echo2.jsp

Why JSF?

4

- Goal: rapid UI development for server-side Java applications
  - Target market: web projects with/without J2EE
    - J2EE infrastructure is available if needed
  - Competition: ASP.NET
- Key components:
  - Prefabricated UI components
  - Event-driven programming model
  - Easily extended component model, allowing developers to implement new components
    - Significant if JSF use is to expand

UW Business School

JSF and NetBeans

5

- With NetBeans 5.0, JSF support can be included in any project
- Embedded Tomcat allows easy developer deployment, testing of JSF applications
  - Huge benefit over other Java IDEs, which currently require additional setup
- Some need to follow the “NetBeans rules” on JSFs for now; we’ll break them pretty quickly
  - NB naming standards are different from Core JSF book
- Bottom line: strong result from the NetBeans team
  - And NB6 is in the works

UW Business School

Getting Started with JSF

6

- Sample app from CJSF Ch. 1: login screen and navigation to “hello” screen
- Key components:
  - JSP page with JSF tags included
  - JavaBean storing, accessing properties (username, password) needed for the application
    - Imagine this being stored externally (e.g. in a database)
  - Navigation configuration (where to go next)
- These components implement the Model/View/Controller (MVC) pattern
  - JavaBean -> Model; JSF -> View; Config file -> Controller

Example: JSF Login

UW Business School

JavaBeans: Introduction

- JavaBeans are just Java classes that expose properties, events
- Specific format to store, retrieve properties
  - Java does the “translation”, e.g. from variable “name” to getName() and setName()

```
public class UserBean {
    private String name;
    private String password;

    public String getName() { return name; }
    public void setName(String newValue) { name = newValue; }

    public String getPassword() { return password; }
    public void setPassword(String newValue) { password = newValue; }
}
```

UW Business School

JSF Pages

- One JSF page per browser screen
- Externally, we’ll use “.jhtml” for our extension (not .faces)
- Internally, these will map to .jsp
- Within the JSF, the two tag libraries map HTML-like tags to Java code
  - More on this in later chapters
- Input values bound to “user” JavaBean
  - “Value-binding expression”
- Command button invokes action “login”

UW Business School

The Controller – faces-config.xml

- Two key components to note in this important file:
  - Navigation rules
  - Bean definitions
- For our example, an outcome of “login” on login.jsp sends the user to welcome.jsp`
- The bean definition maps the “user” bean to the class shown
  - This separates the “view” reference from the “model” implementation
- We also give the class a scope (request), which we’ll discuss in the next chapter

UW Business School

Message Bundles

- Localization (internationalization) is well supported in JSF
- Messages (groups of string localized for a particular language) are loaded using <f:loadBundle>
- Then specify locale using either
  - <f:view locale="en"> in JSF page
  - <supported-locale> tag in faces-config.xml
- We won't spend much time on message bundles, but we'll install and use them if they're made available in the text (e.g. the one in this chapter)

UW Business School

Example: Number Quiz

- Slightly more complicated example, esp. in terms of the model
  - Beans contain more complex "business logic", not just getters/setters
  - Allows model changes without changes to view
  - Note only QuizBean is exposed in faces-config.xml
- Message bundles are easy to switch out: change the loadBundle line in the JSP page
  - English->German is a view change without affecting model
- Navigation is a little too simple: no way to "end the game" (solution in Ch. 3 example)

Example: Number Quiz

UW Business School

JSF Application Navigation

- Since HTTP is stateless, all navigation must be managed by the developer
  - Must always have reasonable "what to do next" steps
- Static navigation: responses specified in config file
- Dynamic navigation: next navigation step determined programmatically
  - Example in book (p. 69): user validation determines action
- String values are used for result codes
  - Return null to force same page to redisplay

UW Business School

13

Example: Java Quiz

- Another quiz, this time with more complex navigation
  - See transition diagram on p. 75; key design artifact
  - Allows model changes without changes to view
- Some changes from structure in book
  - Name overlap w/previous chapter
  - Solved with renames, subdirectories, separate packages
- Navigation is still a little cheesy, but it gives you a sense of what's possible

Example: Java Quiz

14

Standard JSF Tags

- Those two tag libraries do a lot…
  - Core tags manage components, support HTML
  - HTML tags add input, output, command, selection, layout, data, other components
- We'll treat Ch. 4 as a reference, looking at all the samples projects given
  - Easy way to grasp component use is through examples
- We'll leave data tables (Ch. 5) for our database discussion next week

Examples: Several from Ch. 4