Spring for JavaServerFaces

A match made in heaven?

Keith Donald
Principal
SpringSource
keith donald at springsource.com

Slides and demos available at
http://blog.springsource.com

Who am I?

- **Web Application Developer**
- **Creator of Spring Web Flow**
- **Technical Lead, SpringSource Web Dev Products**
- **Principal Author, SpringSource Training Curriculum**
- **Director, The Spring Experience Conference Series**
- **Member, JSF 2.0 Expert Group, with Jeremy Grelle**
- **JavaOne, NFJS, and TSSJS Speaker**

Agenda

- **Introduction**
  - **Spring**
  - **JavaServerFaces**
- **Integration Approaches**
  - **JSF-centric**
  - **Spring-centric**
- **Real World Examples**

Introduction

- **What is Spring?**
  - **The leading Java application framework**
- **Spring delivers**
  - **A container that manages your application architecture**
  - **Helpers for integrating popular ORMs like Hibernate**
  - **Declarative transaction management**
  - **Web services and messaging support**
  - **A platform for developing web applications using the MVC paradigm**

Spring and Java EE

- **Spring and Java EE are complementary**
- **Spring provides a lightweight application platform for integrating Java EE Services**
  - **Java Persistence Architecture (JPA)**
  - **Java Database Connectivity (JDBC)**
  - **Java Management Extensions (JMX)**
  - **Java Server Faces (JSF)**

JavaServerFaces (JSF)

- **What is JavaServerFaces?**
  - **A framework for building user interfaces of Java web applications**
- **JSF delivers**
  - **The leading Java-based web UI component model**
    - **Event-driven**
    - **Declarative authoring of pages**
  - **Extension points for binding to managed application services**
  - **Extension points for page navigation handling**

Spring and JavaServerFaces

- **JSF fits into the web layer of a Spring web application**
- **There are two approaches to integration**

Integration Approaches

- **JSF-centric ("basic")**
  - **Looser level of integration**
  - **Spring fits into the JSF world**
    - **Behind a javax.faces.FacesServlet**
- **Spring-centric ("deep")**
  - **Deeper level of integration**
  - **JSF fits into the Spring world**
    - **Behind an org.springframework.web.servlet.DispatcherServlet**

JSF-centric Integration Approach

- **Spring plugs in as a JSF managed bean provider**
  - **Can replace or supplement JSFs built-in facility**
- **Spring can also be used to configure custom JSF artifacts**
  - **Navigation handlers**
  - **Phase Listeners**

Usage Pattern

- **Has evolved from Spring 1.x to Spring 2.x**
- **Spring 1.x**
  - **Spring manages your application-scoped services**
  - **JSF manages your request-scoped and session-scoped beans**
    - **References to services are injected using JSF's facilities**
- **Spring 2.x**
  - **Spring manages your beans across all scopes**
  - **Can replace managed-bean declarations in faces-config.xml**

Benefits of using Spring as a JSF managed bean provider

- **Less verbose configuration**
  - **More concise XML syntax**
  - **Elegant annotation-based syntax (Spring 2.5)**
- **One facility to learn**
  - **If you use Spring elsewhere, why not for configuring JSF too?**
- **Full power of Spring's container is available**
  - **Constructor injection**
  - **Lifecycle callbacks**
  - **Aspects**

Demo

- **JSF-centric integration approach**
  - **Spring Travel Reference Web Application**
- **Code available at the SpringSource Team Blog**
  - **http://blog.springsource.com**

Are there useful things JSF's facility does Spring does not?

- **EL support**
  - **The ability to inject the result of an EL expression evaluation into a managed bean**
  - **Useful for conveniently injecting web state into managed beans**
    - **Request parameters**
    - **Session attributes**
    - **Any implicit web variable**
- **EL support is being added in Spring 3.0**

Agenda

- **Introduction**
  - **Spring**
  - **JavaServerFaces**
- **Integration Approaches**
  - **JSF-centric**
  - **Spring-centric**
- **Real World Examples**

Spring-centric JSF Integration Approach

- **JSF plugs into Spring as a View implementation**
  - **Integrates with Spring MVC and Web Flow**
  - **FacesServlet is not used**
- **Spring is used**
  - **As the managed bean provider**
  - **As the request dispatcher**
  - **As the navigation handler**
  - **As the state manager**
  - **As a lightweight JSF component library**

Usage Pattern

- **Deploy a Spring MVC DispatcherServlet**
- **Implement Controllers for stateless interactions**
  - **These controllers can select JSF views for rendering**
- **Implement Web Flows for stateful conversations**
- **Use the Spring Faces library for**
  - **Client-side validation components**
  - **Ajax in a JSF environment**
  - **Progressive UICommand components**
  - **JSF utilities**

Benefits of a Spring-centric approach

- **Full control over URLs**
- **More concise, powerful expression of UI control logic**
  - **Page navigation rules**
  - **Exception handling rules**
- **Finer-grained state management**
  - **ViewScope, ConversationScope**
- **One model to learn**
  - **If you already familiar with Spring for MVC, why not use it to process standard JSF views too?**

Demo

- **Spring-centric integration approach**
  - **Spring Travel Reference Web Application**
- **Code available at the SpringSource Team Blog**
  - **http://blog.springsource.com**

Demo Comparison (1)

	Plain JSF	Spring-centric JSF
Number of Controller Artifacts	**4**	**2**
Lines of code	**333**	**95**

Demo Comparsion (2)

	Plain JSF	Spring-centric JSF
Ajax	No	Yes
View Scope	No	Yes
Conversation Scope	No	Yes
Modularization	Limited	Yes
View Lifecycle Callbacks	No	Yes
Automatic Redirect After Post	No	Yes
Protected Views	No	Yes

Demo Comparsion (3)

	Plain JSF	Spring-centric JSF
Degradation	No	Yes
Hot Reloadable Changes	No	Yes
Exception Handling	Limited	Yes
URL Control	Limited	Yes
Bookmarkable Pages	Limited	Limited
MVC Architecture	View Driven	Controller Driven

When to use which integration approach?

- **JSF-centric**
  - **Most common for existing JSF applications**
  - **Most natural for a seasoned JSF developer with little Spring experience**
- **Spring-centric**
  - **Best for new JSF web applications**
    - **Deeper level of integration offers simplicity and power for JSF in one integrated offering**
  - **Most natural for Spring developers (“Springers”)**

Other common JSF pain points addressed by the Spring-centric approach

- **Proliferation of verbose XML configuration**
  - **Solved by annotation-based configuration**
  - **Solved by modularizing your web application by domain use case**
- **Poor performance / memory overhead**
  - **Solved by reducing session state and serialization overhead with ViewScope and ConversationScope**
- **POST Only / Lack of control over URLs**
  - **Solved by Spring MVC's robust UrlHandlerMapping machinery**
- **Browser back button usage problems**
  - **Solved by automatic post+redirect+get**

Roadmap for the future

- **Declarative site definition language (DSL)**
  - **Will provide a complete web site modeling language inspired by Jesse James Garret's Visual Vocabulary**
  - **Similar, but broader than Web Flow's existing "flow definition language"**
    - **Provides a complete "site map"**
    - **Allows for a mixing of standalone pages and reusable flows with other "site elements"**
      - **Providing a natural lexicon for UI developers**
- **Use of scripting languages for site flow definition**
  - **Provide a elegant mix of declarative and imperative constructs**

Summary

- Spring and JSF are a great fit
- Spring provides complete JSF support
- Two integration approaches exist for integrating Spring and JSF
  - JSF-centric (“basic”)
  - Spring-centric (“deep”)
- Learn more at www.springframework.org

Questions?