Struts and JavaServer Faces

Ken Paulsen
Staff Engineer
Sun Microsystems, Incorporated
(Slides by Craig McClanahan)
Agenda

- Brief description of JavaServer Faces
- Struts or JavaServer Faces?
- Future directions
- Summary
- Creator Demo
JavaServer Faces Is ...

A server side user interface component framework for Java-based web applications
Background

- Web applications a popular entry point for developers new to Java
- Powerful foundational technologies:
  - Servlet API
  - JavaServer Pages (JSP)
  - JSP Standard Tag Library (JSTL)
  - Portlet API
- But no common component standard
Background

• Web applications also represent a key opportunity to attract a new developer market segment to Java
  – Corporate developers

• Attracting this population required something different
  – Ease of use is the #1 criteria
Fundamental Requirements

- Accessible to corporate developers
- Accessible to tools
- Client device neutral
- Usable with or without JSP
- Usable with or without HTML
- Scalable to enterprise applications
Basic Capabilities

- Extensible user interface (UI) component model
- Flexible rendering model
- Event and listener handling model
- Per-component validation framework
- Basic page navigation support
- Internationalization and accessibility
Architecture Overview

• **UIComponent** – Basic component API
  – JavaBean class with default base implementation class (UIComponentBase)
  – Contains render-independent properties

• **Standard generic components:**
  – Examples: UICommand, UIInput, UIOutput

• **Concrete component subclasses with HTML-specific properties and behaviors**
Value Binding Expressions

• Components may have a local value
  – Rendered at output time
  – Updated on subsequent form submit

• Components may be bound to a model tier value
  – #{customer.address.city}
  – Syntax based on JSTL/JSP 2.0 EL
  – Semantics identical when rendering

• Nearly all properties may be bound
Method Binding Expressions

- Specialized version of value binding expression
- Last element of the expression points at a method instead of a property
- Used to bind command components to the corresponding action method that should be called when component is activated
Events and Listeners

• Standard JavaBeans event pattern
• Strongly typed events and listeners
• Two standard events:
  – ActionEvent – broadcast when a UICommand component is activated by the user
  – ValueChangeEvent – broadcast when a UIInput component has been validated, and the new value differs from the old value
Converters and Validators

- **Converters** – Plugins for String-Object conversion
  - Render time – Object to String
  - Update time – String to Object

- **Validators** – Plugins for correctness checking

- **Default implementations** for common use cases
Application Interface

• JavaServer Faces provides a default ActionListener for every UICommand
  – UICommand may contain a method binding to an action method to be executed
  – Each UICommand may have its own method, or they may share
  – Action method invoked “immediately” or after validation
  – Return logical outcome used for navigation
Page Navigation Model

- Pluggable NavigationHandler called to perform navigation duties
- Default implementation uses configured navigation rules based on:
  - What page submitted this form?
  - Which action method was invoked?
  - What logical outcome was returned?
- Result: navigate to new page or redisplay old page
Managed Bean Creation Facility

• In a value binding or method binding expression, the first element is special
  – “Magic” values – provide access to request or application data
  – “Non-magic” values – search request, session, and application scope (like `<jsp:useBean>`)  
  – If not present, automatically instantiate a bean, put it in scope, and populate properties

• Generalized version of Struts form beans
Business Logic In Backing Beans

• Most JavaServer Faces applications will organize event handling code for a particular form into a corresponding JavaBean class (“backing bean”)

• Typical backing bean is also a managed bean, put in request scope

• Similar in concept to ASP.Net “code behind files”
JSF In Action

• The JSF RI ships with several samples
  – samples/jsf-cardemo.war
• Can be dropped into any Servlet 2.3 / JSP 1.2 (i.e. J2EE 1.3 or later) container
• We will see a demo using Creator
Agenda

• Brief description of JavaServer Faces
• **Struts or JavaServer Faces?**
• Future directions
• Summary
• Creator Demo
Struts or JavaServer Faces?

• Long answer on Craig's blog:
  – /20040927#struts_or_jsf_struts_and

• Struts and JSF provide 2 architectures for building Model 2 based webapps

• But is it an either-or choice?
• No – You can use them together
Struts+Faces Integration Library

• Design goals:
  – Take an existing Struts-based application ...
  – Convert one JSP page at a time to use JSF components instead of Struts HTML tags ...
  – Tweak the mapping information as needed in struts-config.xml ...
  – And make no changes in your form beans or actions

• Must work with Validator and Tiles
Struts+Faces Integration Library

• The design goals were achieved
• Struts+Faces Integration Library available at Apache:
  – http://svn.apache.org/builds/struts/nightly/struts-faces
• Converted application will use JSF components, but not JSF lifecycle
• Can convert actions later if desired
Demo – Integration Library

• The integration library ships with two samples (one with Tiles, one without)
• Can be dropped into any Servlet 2.3 / JSP 1.2 (i.e. J2EE 1.3 or later) container
• Let's look at these applications in action
• And browse the source code
Choosing What To Use

• Three choices here:
  – Pure Struts-based architecture
  – Pure JavaServer Faces-based architecture
  – Hybrid Struts+Faces with Integration Library

• More than one right answer
  – Not a one size fits all environment

• Different criteria will have different weights for different users
My Recommendations

• Existing Struts-based application?
  – Consider migration to JSF via integration library, when more sophisticated UI components are needed
  – Migrating form beans and actions is optional

• New application to be developed?
  – Sufficient JSF expertise and functionality? Use JSF (with tools if desired)
  – Else feel free to (continue) adopting Struts
  – Hybrid solution possible, but not recommended
Agenda

• Brief description of JavaServer Faces
• Struts or JavaServer Faces?
• Future directions
• Summary
• Creator Demo
Future Directions

• Struts 1.x is a robust, mature framework:
  – 1.0 released in 2001
  – Subsequent versions backwards compatible
• Struts 1.x was (and is) a defacto standard
• Several other frameworks have emerged over the last four years
• And JavaServer Faces was standardized
Future Directions

• I believe it is time for Struts to harvest what we've learned over the years:
  – Good ideas from other frameworks
  – Embrace JavaServer Faces standard APIs

• If we knew then what we know now, what would Struts look like?

• A new approach to Struts 2.x will let us find out
  – While 1.x development continues
Future Directions

• Craig has proposed a new architecture for Struts 2.x, called “Shale”:

• Fundamentally based on JSF and the plug in architecture it supports

• Decomposes the “monolithic” Struts request processor
Future Directions

• Focused functionality at different levels:
  – ViewController – Backing bean per page with very simple lifecycle callbacks (“View Helper” design pattern)
  – DialogController – Framework for managing user interaction requiring more than one HTTP request to complete
  – ApplicationFilter – Location for centralized functionality (like access control checks)
Future Directions

• Shale has not (yet) been accepted by the Struts developers as the formal choice for the next generation
  – Discussions continue on developer list
  – Likely to become an accepted subproject

• Shale needs to gather a community to become accepted
  – Subscribe to developer list to participate
    • Send an empty message to dev-subscribe@struts.apache.org
Future Directions

• In the mean time, Struts 1.3 is actively being developed
  – Refactor request processor based on Chain of Responsibility design pattern
  – Reorganize artifacts into core library and separate additional modules
  – Focus remains on being fundamentally backwards compatible
Agenda

• Brief description of JavaServer Faces
• Struts or JavaServer Faces?
• Future directions
• Summary
• Creator Demo
Summary

- Struts is a mature, robust, framework for building web applications based on the MVC design pattern
- JavaServer Faces is the standard Java API for building user interface components for web applications
- The two can be used together as needed
Agenda

- Brief description of JavaServer Faces
- Struts or JavaServer Faces?
- Future directions
- Summary
- Creator Demo
Struts and JavaServer Faces

ken.paulsen@sun.com
A Very Common Question

Now that JavaServer Faces is out, does that mean Struts is obsolete?
Agenda

- Introduction
- Brief description of JavaServer Faces
- Comparison of implementation techniques
- Decision criteria for choosing
- Summary
- Brief description of Struts
The Origin of Struts

- The JavaServer Pages (JSP) Specification (version 0.91) described two fundamental approaches:
  - Model 1 – A resource is responsible for both creating a page's markup and processing the subsequent form submit
  - Model 2 – Separate resources are responsible for creating a page's markup and processing the subsequent form submit
The Origin of Struts

• The second approach sounded better
  – Resources for creating markup and accessing databases are separated ...
  – So they can be built by different people ...
  – Perhaps using different tools

• So, I built a “home grown” architecture based on the Model-View-Controller (MVC) design pattern
Model-View-Controller (MVC)

- Model – Persistent data and the business logic that processes it
  - In large applications, often subdivided
- View – The interface with which the user directly interacts
- Controller – Management software to enforce flow control and dispatch logical functions to physical resources
MVC as Implemented in Struts

1. Submit
   - Browser
   - Controller

2. Dispatch
   - Controller
   - Business Logic

3. Update, Get
   - Business Logic
   - Model Data

4. Dispatch
   - Controller
   - View

5. Pull
   - View
   - Model Data

6. Render
   - View
   - Browser
Struts Features – Model Tier

- Struts includes only minimal features
- But you can integrate any desired approach
Struts Features – View Tier

- **Form Beans**
  - Server-side state of input fields on a form
  - Classic (JavaBean) or DynaBean (configured properties, no separate class) style

- **Validation Framework**
  - Abstract validation rules into separate resource
  - Always enforced on the server side
  - Optionally generates JavaScript for client side checking as well
Struts Features – View Tier

- **JSP Custom Tag Libraries**
  - Bean and Logic – General support (superceded by JSTL)
  - Html – Render HTML markup
  - Nested – Navigate bean hierarchies
  - Tiles – Layout management (see next page)

- **Extended Versions (struts-el)**
  - Integrate expression language support
  - Not required in JSP 2.0 or later
Struts Features – View Tier

• Tiles Framework:
  – Templating for common look and feel
  – Definitions created in JSP page or separate XML resource
  – Definitions can inherit from other definitions
  – Advanced techniques for sharing information between tiles
  – Fully integrated into Struts navigation support
Struts Features – Controller Tier

• Standard configuration resource for defining desired behavior
  – Mapping URLs to Action classes
  – Mapping logical Forwards to physical pages
  – Defining form beans and properties
  – Configuring Action behavior
    • Form bean creation, validation, input page
  – Generalized exception handling
  – Localization resources
Struts Features – Controller Tier

• Standard request processing lifecycle
  – Extract action mapping path
  – Select locale (if necessary)
  – Select action mapping to utilize
  – Role-based authorization checks
  – Instantiate and populate form bean
  – Server side validation
  – Invoke application action
  – Forward to requested view tier resource
Struts Features – Controller Tier

• Sub-application modules
  – Logically divide a single web application into several “mini-applications”
  – Session state is shared across modules

• Standard Action implementations
  – Forward to or include other URLs
  – Dispatch to method based on parameter
  – Switch to different module