JavaServer Faces and Struts: Competition or Coexistence?

Craig R. McClanahan
Senior Staff Engineer
Sun Microsystems, Inc.

Agenda

- Introduction – A very common question
- Brief description of Struts
- Brief description of JavaServer Faces
- Comparison of implementation techniques:
  - Pure Struts
  - Struts+Faces Integration Library
  - Pure JavaServer Faces
- Decision criteria for choosing

A Very Common Question

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Struts is obsolete?

A Brief Description of Struts

The Origin of Struts

- Like many open source projects, Struts started with me scratching my own itch
  - Take a US-centric application to Europe ...
  - In multiple languages ...
  - And make it available on the web
- I was familiar with Java and open source (Apache JServ, Tomcat)
- But there was no good model for a web application architecture

The Origin of Struts

- The JavaServer Pages (JSP) Specification, version 0.91, described two fundamental approaches:
  - *Model 1* – A resource (such as a JSP page) is responsible for both creating the markup for a form, *and* for processing the subsequent submit
  - *Model 2* – A resource (such as a JSP page) is responsible solely for creating the markup; processing the submit is dispatched to a separate resource

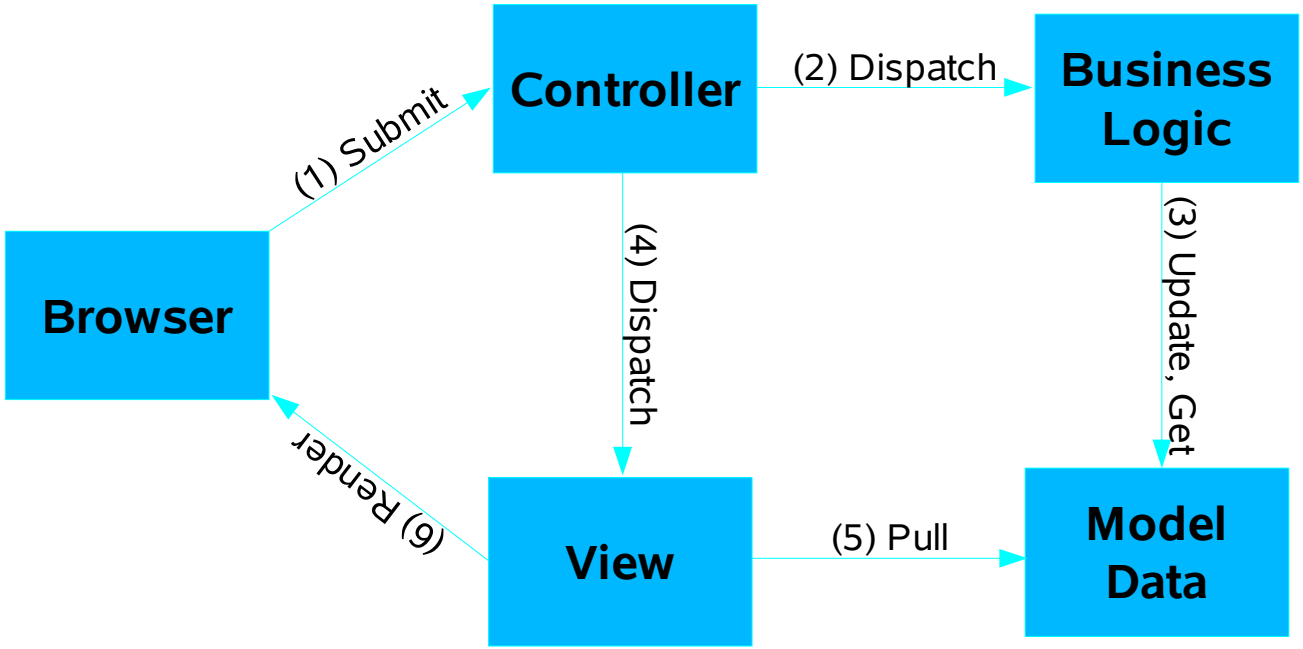
The Origin of Struts

- The second approach sounded better:
  - Resources for creating markup and accessing databases are separated ...
  - So they can be built by different people ...
  - Using potentially different tools
- So, I built a “home grown” architecture based on the Model-View-Controller (MVC) design pattern

Model-View Controller (MVC)

- *Model* – The persistent data (typically stored in a database) and business logic
- *View* – The interface with which the user interacts
- *Controller* – Management software to dispatch form submits to the appropriate business logic functions, and map logical outcomes to the next page

MVC as Implemented in Struts



Struts Features – Model Tier

- Struts includes only minimal features here
- An implementation of *javax.sql.DataSource* (connection pool)
- But you can integrate **any** desired approach

Struts Features – View Tier

- Form Beans
  - Represent the server-side state of input fields on an HTML form
  - Classic (JavaBean style) and DynaBean (configured properties, no separate class)
- Validation Framework
  - Abstracts validation rules into separate document
  - Always enforced on server side
  - Optionally generates JavaScript for client side checking as well
  - Extensible

Struts Features – View Tier

- JSP Custom Tag Libraries:
  - *Bean* – General bean manipulation
  - *Html* – Render HTML markup
  - *Logic* – Conditionals and iteration
  - *Nested* – Versions of standard tags for navigating bean hierarchies
  - *Tiles* – Layout management (next page)
- Extended Version (struts-el):
  - Integrates support for Expression Language (EL) identical to JSTL 1.0
  - Won't be required in JSP 2.0 container (EL expressions work everywhere)

Struts Features – View Tier

- Tiles Framework:
  - Templating for common look and feel
  - Definitions created in JSP page or separate XML document
  - 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 file for defining desired behavior:
  - Mapping Action URLs to Action Classes
  - Mapping Forwards (logical resources) to physical pages
  - Defining form beans (and properties, for DynaBeans)
  - Configuring Action behavior (form bean creation, validation, return-to-input destination, etc.)
  - Generalized exception handling
  - Sources for localized resources

Struts Features – Controller Tier

- Standard request processing lifecycle
  - Extract action mapping path
  - Select locale (if necessary)
  - Select action mapping to utilize
  - Role-based access checks
  - Instantiate and populate form bean
  - Server-side validation (if requested)
  - Invoke application action
  - Forward to view tier resource based on logical outcome

Struts Features – Controller Tier

- Sub-application modules
  - Logically divide a single web application into several “mini-applications”
  - Session state shared across modules
- Standard Action implementations
  - Forward to or include other URLs
  - Dispatch to method based on parameter
  - Switch to different module

A First Struts-Based Application

- Struts ships with a canonical example application (webapps/struts-example.war)
- Can be dropped into any Servlet 2.2 / JSP 1.1 (i.e. J2EE 1.2 or later) container
- Let's take a look at this application in action ...

Demo – Pure Struts Application

A Brief Description of JavaServer Faces

- JavaServer Faces is ... a server side user interface component framework for Java-based web applications

Background

- Web applications are a very common entry point for developers new to the Java platform
- Powerful foundational technologies:
  - Servlet
  - JavaServer Pages (JSP)
  - JSP Standard Tag Library (JSTL)
  - (New!) Portlet

Background

- Web applications represent a key opportunity to attract a completely new developer market segment to Java
  - Traditional Java Developers – 3 million
  - Corporate Developers – 10 million
- Attracting this new developer population requires us (the Java platform folks) to do things a little differently than we've done in the past
  - *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 scale applications

Fundamental Requirements

- Deployable immediately
  - Based on Servlet 2.2 and JSP 1.1
  - Run on any J2EE 1.3 app server
  - (New!) Run on any J2EE 1.4 app server

Java™

COMPATIBLE

ENTERPRISE EDITION

Sun microsystems

J2EE™ 1.4 SDK

Optimized for Java Web Services

- New in 1.4!: Complete Java Web Services & WS-I Basic Profile 1.0 support
- New in 1.4!: Includes a deployable App Server FREE
- New in 1.4!: Optimized for development, prototypes and small deployments
- The most effective way to learn about what's new in J2EE 1.4 featuring documentation, examples, and J2EE Blueprints
- Get the Beta 2 release now!

Download it today for free:

http://java.sun.com/j2ee

Basic Capabilities

- Extensible UI component model
- Flexible rendering model
- Event handling model
- Per-component validation framework
- Basic page navigation support
- Internationalization
- Accessibility

Architecture Overview

- UIComponent / UIComponentBase
  - JavaBean class with standard behavior
  - Render-*independent* properties
- Standard generic component implementations
  - Example: UICommand, UIInput, UIOutput
- (Coming Soon) concrete component subclasses with HTML-specific properties and behaviors

Value Reference Expressions

- Components may have a local value
  - Rendered at output time
  - Updated on subsequent form submit
- Components may have a *value reference expression*
  - "Symbolic link" to a JavaBean property, Map value, array element, ...
  - Example: "customer.address.city"
  - Syntax based on JSTL/JSP expression language syntax for variable references
  - Semantics identical to EL when rendering
  - Used as "lvalue" to update data on submit

Events and Listeners

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

Converters and Validators

- *Converters* – Plugins for conversion
  - Render time – object to string
  - Update time – string to object
- Default implementations automatically selected (like JSP)
- *Validators* – Plugins for correctness checks on input components

Application Interface

- JavaServer Faces implementation provides a default ActionListener on every UICommand
  - UICommand includes an *action reference* to the action to be executed
  - Each UICommand can have its own action reference, or they can share
  - Action can be invoked “immediately” (for user interface changes followed by redisplay) or “normally” after validation and model updates
  - Action returns logical outcome used in navigation

Page Navigation Model

- Pluggable *NavigationHandler* called to perform navigation duties
- Default implementation uses configured navigation rules to select the next page, based on:
  - What page are we currently displaying?
  - What action reference was invoked?
  - What logical outcome was returned?

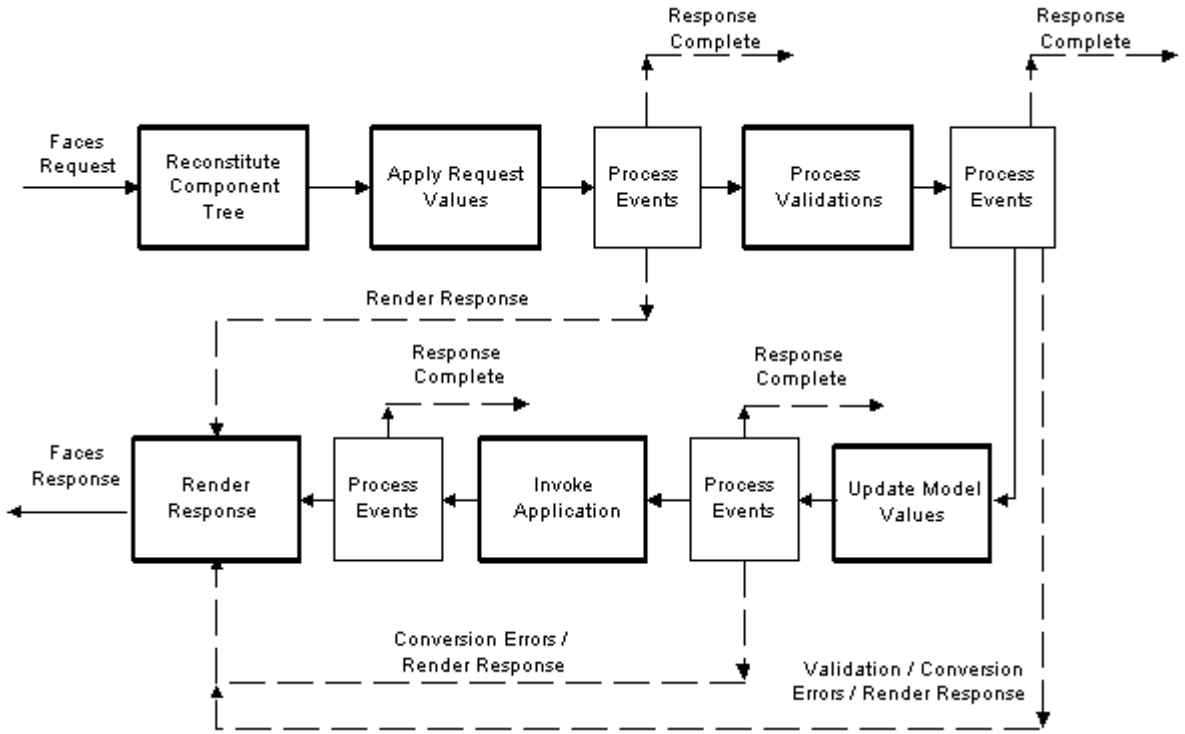
Managed Bean Creation Facility

- In a value reference expression, the first element is treated specially:
  - "magic" values provide access to request or application data (i.e. "initParam" is a map of the context init parameters of this webapp)
  - For non-magic values, searches request, session, and application scope
  - If not present, can automatically instantitate a bean and populate its properties
- Similar to the Struts ability to automatically create form beans, but extended to create *any* bean on demand

Business Logic ("Backing") Beans

- Most JavaServer Faces applications will organize the event handling code for a particular form into a corresponding Java class
- Similar in concept to "code-behind files" in Microsoft ASP.NET
- More flexible, in that you are not tied to a 1:1 relationship between page and code-behind file

Request Processing Lifecycle



Demo – Pure Faces Application

Current Status

- Spec is currently at *Public Draft 2* status
- *Early Access 4* version of RI available in the Java Web Services Developer Pack, version 1.2 or 1.3
  - http://java.sun.com/webservices/
- *Proposed Final Draft* of spec and *Beta* of RI coming soon
- Note – once JavaServer Faces goes final, the RI will be commerically redistributable under a standard Sun binary license

The Struts+Faces Integration Library

- We've seen so far how you can use either Struts or JavaServer Faces as the foundation for a well-architected Model 2 based application
- But is it an either-or choice?
- *NO* – you can use them together

Struts+Faces Design Goals

- Take an existing Struts-based application ...
- Convert *one* JSP page at a time to use JavaServer Faces components instead of Struts HTML tags ...
- Tweak the mapping information in *struts-config.xml* ...
- And make **no** changes in the form beans or actions

Struts+Faces Current Status

- These design goals were achieved
  - Proof of concept – converted struts-example
- Struts+Faces library available at Apache site
- Currently works with EA4 release of JavaS-erver Faces
- Will be updated for subsequent JavaServer Faces releases
- Current limitation – no support for Tiles

Demo – Struts+Faces Library

So How Do You Choose What To Use?

- Recognize that there are really three choices here:
  - Pure Struts-based architecture
  - Pure JavaServer Faces-based architecture
  - Hybrid Struts+Faces with Integration Library
- Recognize that there is more than one right answer:
  - It's not a *one size fits all* environment
- Recognize that the various decision criteria will have different relative weights for different use cases

Project Decision Criteria

- Do you have a project timeline that anticipates deployment soon?
  - Struts 1.1 released June 2003, solid and mature
  - JavaServer Faces planned PFD/Beta in 4QCY2003, final in 1QCY2004
- Do you have substantial expertise and/or reusable code based on Struts?
  - Pure Struts or Struts+Faces likely best

Project Decision Criteria

- Does your development team need rich tools and documentation support?
  - Struts supported by virtually all development tools and IDEs
  - Five books totally focused on Struts, plus chapters in many others
  - JavaServer Faces support coming, but not present yet:
    - Sun and Oracle previewed tools supporting Faces at JavaOne 2003
    - Several books in preparation, including two by well-known authors that are part of the JSR-127 expert group

Project Decision Criteria

- Is your project new?
  - Struts+Faces or Pure Faces likely best
- Does your project require features of Struts that are not present in JavaServer Faces (i.e. Tiles, Validator Framework)?
- Does your project require features of JavaServer Faces that are not present in pure Struts (i.e. Managed beans, rich UI components, event handling)?

A Personal Suggestion ...

- Personally, I would generally decide as follows, based on these use cases:
  - Existing Struts-based app needing minor tweaks – stay with pure Struts
  - Existing Struts-based app needing UI remodel – migrate to Struts+Faces
  - New app – Struts+Faces or Pure Faces depending on other factors
  - Generally shy away from new projects using Struts HTML tags
- But any of the three alternatives might be best for you!

Preview of Coming Attractions

- Struts developers have started defining a roadmap
  - Struts 1.2 – incremental changes, highly back-wards compatible
    - Incorporate commons-resources
    - Incorporate commons-chain (or some other way to support easier refactoring of RequestProcessor)
    - Other features over time (perhaps things like module inheritance) based on user demand
  - Struts 2.0 – rearchitecture based on three years of experience
  - 1.2 and/or 2.0 – support portlet API (JSR-168)

Preview of Coming Attractions

- JavaServer Faces is approaching Proposed Final Draft / Beta
- A couple of hints at nice things coming up:
  - For-real data grid component with general purpose data binding
  - Ability to link component instances to"backing files"
  - Ability to reference backing file methods rather than separate event listener and validator classes
  - Substantial metadata for design-time support in tools

Summary

- So, is Struts obsolete?
  - *Of course not! It is already stable, mature, feature-rich, and widely supported, and will continue to innovate (faster than standards processes can operate)*
  - *But software is like life – evolve or die*

Summary

- Do Struts and JavaServer Faces “compete” with each other?
  - *The two frameworks have overlapping features*
  - *The two frameworks have non-overlapping features*
  - *You can use them together or separately*

Summary

- Will Struts and JavaServer Faces coexist in the future?
  - *Yes – Struts will focus on non-UI-component functionality, while leveraging the ability to use JavaServer Faces as view tier components*
  - *Tools and app servers are not going to gratuitously remove support – that would be a disservice to their customers*
  - *Tools and app servers will likely incorporate support for JavaServer Faces as well.*

Questions?