



Ease of Development in the Java Technology Web Tier

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



Agenda

- Introduction
- JavaServer Faces
- Sun Java Studio Creator
- Apache Struts
- Struts and JavaServer Faces
- Questions and Answers

Agenda

- **Introduction**
- JavaServer Faces
- Sun Java Studio Creator
- Apache Struts
- Struts and JavaServer Faces
- Questions and Answers

Introduction

- **Craig McClanahan is:**
 - Senior Staff Engineer at Sun Microsystems, Inc.
 - Architect for *Sun Java Studio Creator*
- **Previously, Craig has been:**
 - Member of Servlet/JSP Expert Groups
 - Web Tier Architect for J2EE 1.4 Platform
 - Co-Specification Lead, JavaServer Faces 1.0
 - Original creator, Struts Framework

Presentation Themes

- Compare two key ease of use technologies for the Java web tier:
 - JavaServer Faces
 - Apache Struts
- Examine a tool maximizing ease of use development with JavaServer Faces
 - Sun Java Studio Creator
- Future visions

5

Agenda

- Introduction
- **JavaServer Faces**
- Sun Java Studio Creator
- Apache Struts
- Struts and JavaServer Faces
- Questions and Answers

6

Background

- Consider the situation in late 2002:
 - Much innovation in the architecture arena
 - The *de facto* standard web framework (Struts) had no user interface component model
 - Java IDEs focused on supporting the Struts application model
 - Microsoft ASP and (later) ASP.NET focused very strongly on components and associated tooling

7

Background

- Web applications are a popular entry point for developers new to Java
- Powerful foundation technologies available
 - Servlet, JSP, JSTL, Portlet
- Ease of use challenges
- No common standard for user interface components

Background

- Web applications also represent a key opportunity for the Java platform:
 - Attract *corporate developers*
- This goal requires something different:
 - Ease of use is the primary criteria
 - Hide complexity of the platform (until needed)
 - Make it easy to build tools to support this

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 application functionality and performance needs

JavaServer Faces (JSF) is a server side user interface component framework for building Java technology based web applications

11

JSF Basic Capabilities

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

12

JSF Standard Features

- Hierarchical tree of UI components
- Converters (bidirectional)
- Validators (input correctness checks)
- User Interface event handling:
 - Action events
 - Value Change events
 - Custom component events

13

JSF Unique Features

- Value binding expressions:
 - Bind components to model tier data
 - `<h:outputText ... value="#{address.city}"/>`
 - On form submit, used to *push* data back to the model after successful validation
- Method binding expressions:
 - Bind components to “action” methods
 - `<h:commandButton ... action="#{bean.submit_action}"/>`

14

JSF Unique Features

- **Managed Beans:**
 - Instantiate application beans on demand
 - Simple Inversion of Control / Dependency Injection model (setter injection pattern)
- **Outcome Based Navigation:**
 - What view was submitted?
 - What command (button) was executed?
 - What logical outcome was returned
- **Extensibility Points**

15

Request Processing Lifecycle

- JavaServer Faces includes a front controller like most web frameworks
- Customizable via phase listeners:
 - Before Phase
 - After Phase
- Customizable user command processing

16

JSF as an Application Framework

- Pluggable Extension Points:
 - Custom *StateManager* Implementation
 - Determines how component state is saved
 - Custom *ViewHandler* Implementation
 - Support alternate template technologies
- Although focused on the view tier, JSF supplies a basic controller tier as well
 - With extension points for application frameworks to add value on top of JSF

Agenda

- Introduction
- JavaServer Faces
- **Sun Java Studio Creator**
- Apache Struts
- Struts and JavaServer Faces
- Questions and Answers

Creator Background

- Java is an appealing platform to developers from all expertise levels
 - Powerful server side capabilities
 - Object oriented language
 - Secure execution environment in a JVM
- But it has been perceived as “too hard to use” by a significant audience

Creator Background

- Deliberately designed to appeal to corporate developers:
 - May be domain experts, not object oriented software professionals
 - Often familiar with scripting languages
 - Visual Basic, 4GLs, PHP, PERL, ...
- Who create specific types of apps:
 - Consumers of data and services
 - Provide rich UI over complex systems

Creator Background

- With most IDEs, bar is too high:
 - Geared towards technologists / enterprise devs
 - Value = *flexibility* and *richness*
 - Corporate developers have different needs
 - Value = *simplicity* and *understandability*
- Corporate developers also prefer an all-in-one solution
- Enterprise developers can benefit also

23

Creator Quick Orientation

24

Creator Application Model

- Creator leverages the standard JSF request processing lifecycle:
 - Associates “page bean” with each JSP page
 - Contains component instances and event handlers
 - Phase listeners to participate in lifecycle
- Graphical assembly of applications:
 - Individual pages
 - Access to data sources

25

Application Development Demo

Creator and Developers

- Creator clearly appeals to corporate developers
- Enterprise developers will find it valuable as well:
 - Prototyping
 - Rapid application development scenarios
 - Easy access to complex enterprise information system assets

27

Agenda

- Introduction
- JavaServer Faces
- Sun Java Studio Creator
- **Apache Struts**
- Struts and JavaServer Faces
- Questions and Answers

28

The Origin of Struts

- Like many open source projects, Struts started with me “scratching my own itch”
 - Take a US-centric application to Europe ...
 - Supporting multiple languages ...
 - And make it available on the web
- I was familiar with Java and Open Source
- No good architectural model

29

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

30

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

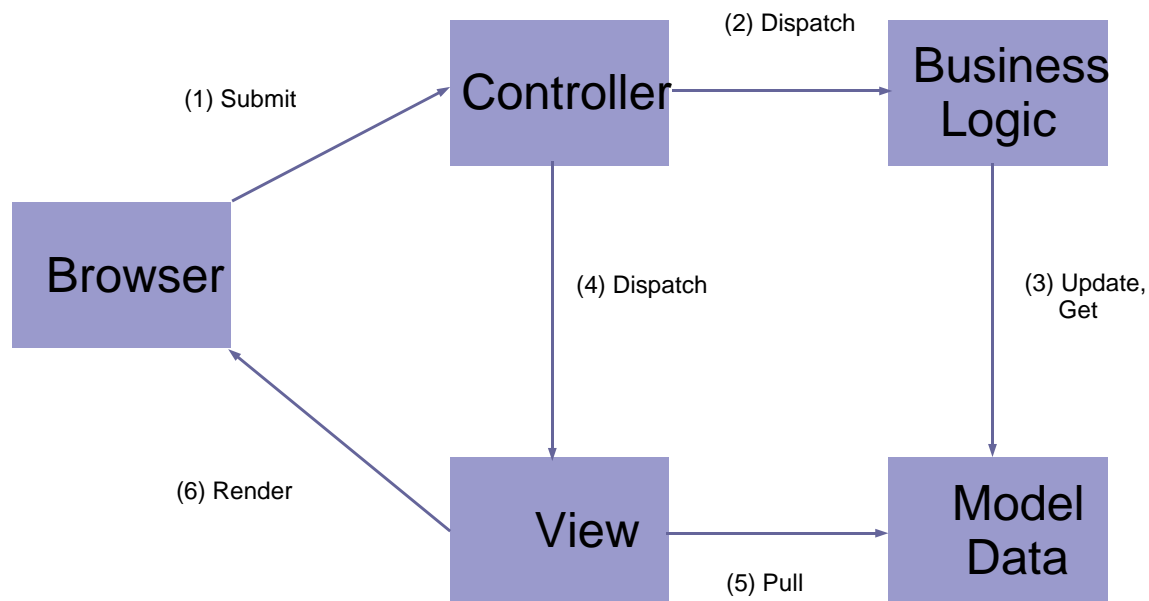
31

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

32

MVC as Implemented in Struts



33

Struts Features – Model Tier

- Struts includes only minimal features
- But you can integrate **any** desired approach

34

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 (superseded 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

Agenda

- Introduction
- JavaServer Faces
- Sun Java Studio Creator
- Apache Struts
- **Struts and JavaServer Faces**
- Questions and Answers

Feature Comparison

- Front controller:
 - *Struts* – ActionServlet processes each request and executes request processing lifecycle
 - *JSF* – FacesServlet processes each request and executes request processing lifecycle
- User interface components:
 - *Struts* – JSP custom tags for simple cases only
 - *JSF* – Full features ⁴² component framework

Feature Comparison

- Field value state saving:
 - *Struts* – Provided by application (ActionForm)
 - *JSF* – Performed by framework automatically
- Data type conversion:
 - *Struts* – Must be performed by application
 - *JSF* – Performed automatically by components
- Input field validation:
 - *Struts* – Client and server side supported

Feature Comparison

- Reusable layout concepts:
 - *Struts* – Tiles framework reuses page fragments
 - *JSF* – Layout components can be created (or Tiles can be plugged in)
- Customization points:
 - *Struts* – Subclass RequestProcessor and other fundamental classes
 - *JSF* – Extensive pluggability points (Template pattern)

Feature Comparison

- Data binding:
 - *Struts* – Limited functionality binding to individual property values
 - *JSF* – Full featured value binding and method binding, with plugin extensibility
- Inversion of Control / Dependency Injection:
 - *Struts* – Create form beans on demand
 - *JSF* – Create any managed bean on demand and configure properties

45

Choosing Between JSF and Struts

- There are many Struts based applications in the world
- Application architects want advice on deciding which technologies to use
- We will look at two decision scenarios:
 - Existing Struts-based applications
 - New development projects

46

JSF and Struts – Existing Apps

- Consider an existing application in which you would like to leverage JSF components in the view tier
- Ideally, one could:
 - Migrate JSP pages (replacing Struts tags)
 - One page at a time
 - Preserving investment in back-end code (Actions, ActionForms, ...)
 - Maintaining support for Struts features (Tiles, client side validation, ...)

JSF and Struts – Existing Apps

- Struts offers a migration library that makes this ideal achievable
- Can migrate view tier and stop
 - Using JSF UI components, Struts controller
- Or, choose to migrate to the JSF application model as well
 - Using JSF UI components, JSF controller
- Struts 1.x is still being maintained

JSF and Struts – New Apps

- For new applications, you should:
 - Evaluate each technology separately
 - If JSF meets your needs, you should use it
 - Near functional equivalence now
 - Or complete equivalence with add-on functionality
 - High quality tools support
 - Basis for future standardization efforts
 - If you need Struts too, use the integration library
- Should not use Struts HTML tags

49

Coming Soon

- Web MVC frameworks *based on* JavaServer Faces:
 - We saw earlier that JSF includes a front controller, like existing web frameworks
 - Next generation frameworks can leverage this to provide *value added* capabilities
 - No need to maintain redundant features
 - Can easily plug in to development tools based on JavaServer Faces components

50

Coming Soon

- One such next generation framework is already under development:
 - “Shale” Proposal (proposed to be Struts 2.0)
 - Accepted as a Struts subproject
 - <http://wiki.apache.org/struts/StrutsShale>
 - Soon nearing functional equivalence with Struts 1.x
 - Radically simpler application model thanks to:

51

Agenda

- Introduction
- JavaServer Faces
- Sun Java Studio Creator
- Apache Struts
- Struts and JavaServer Faces
- **Questions and Answers**



Ease of Development in the Java Technology Web Tier

Craig.McClanahan@sun.com

