



Building Web Applications With The Struts Framework

ApacheCon 2003

Session TU23 – 11/18 – 17:00-18:00

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

Slides:

<http://www.apache.org/~craigmc/>



Agenda

- A Brief Description of Struts
- Model-View-Controller (MVC)
- Struts Features Overview
- A First Struts-Based Application
- Struts and JavaServer Faces
- Summary



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

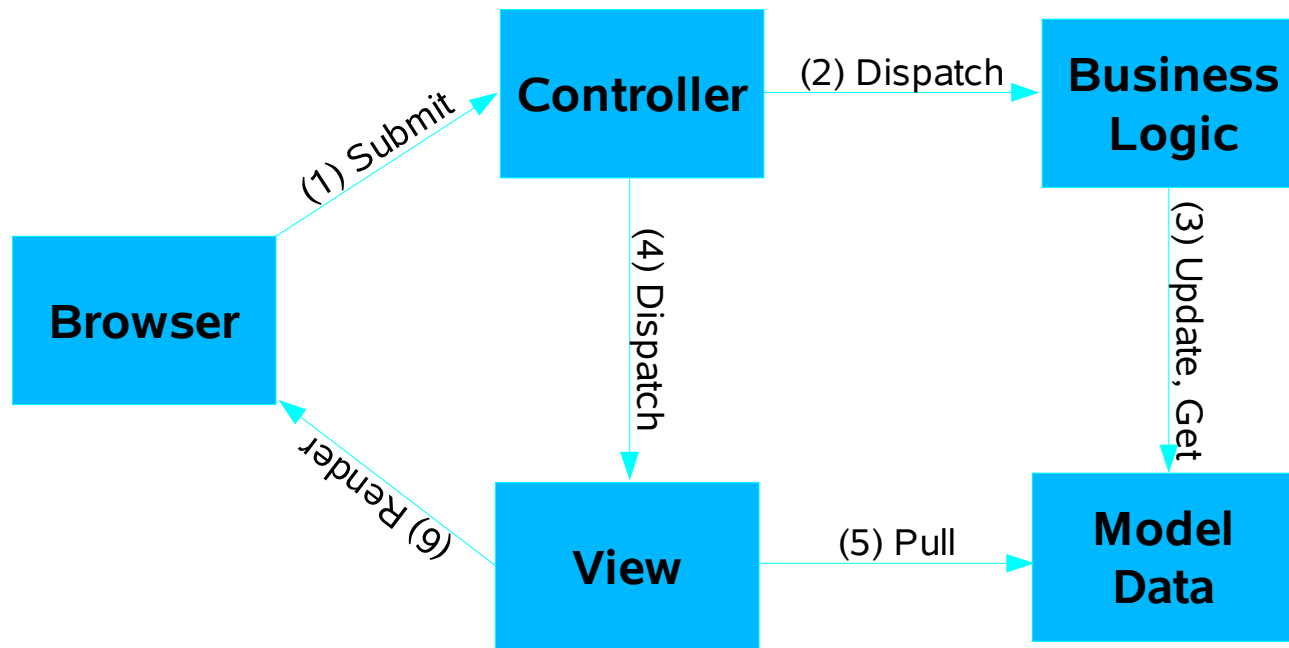


Model-View Controller (MVC)

- *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



Struts Features – Miscellaneous

- Jakarta Commons Libraries:
 - *BeanUtils* – Access bean properties dynamically, supports DynaBeans
 - *Collections* – Extensions to Java2 Collection Classes
 - *Digester* – Parse XML documents and configuration files
 - *FileUpload* – Support file uploading
 - *Lang* – Extensions to core JDK packages



Struts Features – Miscellaneous

- Jakarta Commons Libraries:
 - *Logging* – Abstract layer over JDK 1.4 logging, Log4J, or others
 - *Validator* – Validation framework that can be used in the view and model tiers
- *Jakarta-ORO* – Regular expression parsing



Struts Features – Miscellaneous

- Extensive documentation and JavaDocs
- Wide variety of third party resources
- Example web applications:
 - Blank “starter” app
 - Simple basic example
 - Exercise individual tags
 - Snapshot of all online docs and resources
 - Specific examples for Tiles and Validator



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 – A Simple Struts-Based Application



The Configuration Files

- *web.xml* – Web App Deployment Descriptor
 - “ActionServlet” is the controller
 - Multiple configuration files supported
 - Typically loaded at startup time
 - Mapped to extension path (*.do) or path prefix (/do/*)
 - Identifies the application welcome file
 - (JSP 1.1 only) must declare tag library descriptors



The Configuration Files

- *struts-config.xml* – Struts configuration
 - Form beans (one dynamic, one standard)
 - Global exceptions (none in this app) define handlers for specific types
 - Global forwards provide logical names for physical resources
 - Action mappings map URLs to Actions
 - Can nest exception and forward definitions



The Configuration Files

- *struts-config.xml* – Struts configuration
 - Controller has global configuration settings
 - Message resources elements load sets of localized text for i18n
 - Plugins provide lifecycle (start and stop) support for extensions
- *struts-config-registration.xml* – Illustrates that you can use multiple config files
- *struts-config_1_1.dtd* – Documents content of Struts Config files



Walk Through – Logon Process

- Start on */index.jsp*, second hyperlink:

```
<html:link page="/logon.jsp">  
  <bean:message key="index.logon"/>  
</html:link>
```
- Generated source is localized:

```
<a href="/struts-example/logon.jsp">  
  Log on to the MailReader Demo ...</a>
```
- Automatic URL encoding is performed
- Direct link to JSP is unusual, only useful for “no setup required” transitions



Walk Through – Logon Process

- The */logon.jsp* page is displayed
- Contains a custom form tag:

```
<html:form action="/logon"  
  focus="username" onsubmit="...">
```

- Action attribute must matches configured `<action>` element in `struts-config.xml`
- Focus positions cursor via JavaScript
- Onsubmit invokes client side validation
- Two input fields and two buttons nested



Walk Through – Logon Process

- Submits to */struts-example/logon.do*
- Invokes *ActionServlet* processing
- Selects the correct `<action>` element:

```
<action path="/logon"  
        type="org.apache....LogonAction"  
        name="logonForm"  
        scope="session"  
        input="logon"/>
```



Walk Through – Logon Process

- ActionServlet instantiates *logonForm* bean (if needed), per the form bean definition:

```
<form-bean name="logonForm"  
  type="org.apache....DynaValidatorForm">  
  <form-property name="username"  
    type="java.lang.String" />  
  <form-property name="password"  
    type="java.lang.String" />  
</form-bean>
```



Walk Through – Logon Process

- Server side validation performed according to configured rules:

```
<form name="logonForm">  
  <field property="username"  
    depends="required,minlength,maxlength"> ...  
  <field property="password" ...> ...  
</form>
```

- In this case, we used client-side validation as well, via generated JavaScript



Walk Through – Logon Process

- If validations fail, control goes to the “logon” forward:

```
<action path="/logon"  
        type="org.apache....LogonAction"  
        name="logonForm"  
        scope="session"  
        input="logon" />
```

```
<forward name="logon" page="/logon.jsp" />
```



Walk Through – Logon Process

- If validation succeeds, the *execute()* method of our configured Action class is invoked (the “Command Pattern”)

```
public class LogonAction extends Action {  
    public ActionForward execute  
        (ActionMapping mapping, ActionForm form,  
         HttpServletRequest request,  
         HttpServletResponse response)  
        throws Exception { ... }  
}
```



Walk Through – Logon Process

- The Action checks the username and password against the user database:
 - On unsuccessful match, store an error message and return to the input page
`return (mapping.getInputForward());`
 - On successful match, log user in and indicate “success”
`return (mapping.findForward("success"));`
 - Which transfers to the main menu page
`<forward name="success" page="/mainMenu.jsp"/>`



But What About Prepopulation?

- Often, you need to prepopulate fields to be displayed on a page
- The *Edit Registration* option illustrates a very typical Struts idiom:
 - *Setup Action* populates beans (including the form bean), which forwards to ...
 - *Page* that renders the input form, which submits to ...
 - *Processing Action* that updates the database based on new input



Example Application Summary

- We've seen the basic organization and features that Struts provides
- Struts lives up to its promise to separate the concerns of business logic and presentation logic:
 - Remodel page look and feel – affects pages but not actions
 - Migrate to different database architecture – affects actions but not pages
 - Validation rules integrated separately



Struts and JavaServer Faces

- *JavaServer Faces* is a new API undergoing standardization as JSR-127 in the Java Community Process:
 - Currently in *Public Draft 2* state
 - <http://java.sun.com/j2ee/javaserverfaces>
 - *EA4* release of reference implementation included in Java Web Services Developer Pack (JWS DP), version 1.2 or 1.3
 - <http://java.sun.com/webservices>



Struts and JavaServer Faces

- JavaServer Faces is a server side UI component framework for Java web apps:
 - So, there are overlaps in functionality with the Struts features we have reviewed today:
 - JSP tags for rendering (Struts HTML tag library)
 - Concepts of managing separation of presentation logic and business logic
 - Indeed, Struts has had substantial positive impact on how JavaServer Faces approaches many issues



Struts and JavaServer Faces

- JavaServer Faces has some unique capabilities:
 - Rendering API that is independent of JSP
 - Rendering API that is independent of HTML
- Struts has some unique capabilities:
 - Tiles Framework
 - Validator Framework
 - Sub-application Modules



Struts and JavaServer Faces

- So, is JavaServer Faces going to standardize Struts out of existence?
 - *ABSOLUTELY NOT!*
- OK, so can I use them together?
 - *FOR SURE!*
- How can I do that?
 - *I'm glad you asked ...*



Struts and JavaServer Faces

- If you have existing applications (or expertise) based on Struts, you can treat JavaServer Faces as an alternative view tier library:
 - Thus, maintaining your investment in the model and controller tiers
- An integration library has been developed to make this very easy:
 - EA version available on the Struts website
 - <http://jakarta.apache.org/struts/>



Struts and JavaServer Faces

- Design goals of the integration library:
 - Allow you to take an existing Struts webapp
 - Migrate your JSP pages to use JavaServer Faces tags instead of Struts HTML tags
 - One page at a time ...
 - Without touching your Actions or business logic
- The design goals have been achieved:
 - Proof of concept – converted version of the canonical *struts-example* app



Summary

- Struts is a robust, mature, web application framework suitable as the basis for developing mission critical web applications
- Struts has garnered substantial industry, tool, and developer support
- Struts will continue to incorporate support for new technologies as they become useful



Resources

- Struts Web Site:
 - <http://jakarta.apache.org/struts/>
- Struts Mailing Lists:
 - struts-dev@jakarta.apache.org
 - struts-user@jakarta.apache.org
- **Free** J2EE 1.4 App Server:
 - <http://java.sun.com/j2ee/>



Questions?