Apache Struts 2.0
Open-source framework for creating Java web applications

CSCI-5448 Object Oriented Analysis and Design

Present by Ming Lian
Topics Discussed

- History of Struts
- Basic features of struts 2.0
- Struts2 vs struts1.1
- Architecture of struts2.0
- MVC 2 Model Architecture and Overview
- Basic flow of struts2.0
- Core Components
- Pros and Cons
History of Struts

- The Apache Struts Project was launched in May 2000 by Craig R. McClanahan to provide a standard MVC framework to the Java community.
- In July 2001, version 1.0 was released.
- Taken over by Apache Software Foundation in 2002.
Revolution To Next Generation

Struts

Struts 2.0
History of Struts (Cont)

- Struts 2 was originally known as WebWork2
- WebWork and Struts were combined in 2008 to create Struts 2
- Struts 1 is not obsolete and will be supported for many years
Features

- Model 2 - MVC Implementation
- Internationalization (I18N) Support
- Rich JSP Tag Libraries
- Annotation and XML configuration options
- POJO-based actions that are easy to test
Based on JSP, Servlet, XML, and Java
Less xml configuration
Easy to test and debug with new features
Supports Java’s Write Once, Run Anywhere Philosophy
Supports different presentation implementations (JSP, XML/XSLT, etc)
## Comparison

<table>
<thead>
<tr>
<th>Struts 1</th>
<th>Struts 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Action</td>
<td>Action</td>
</tr>
<tr>
<td>ActionForm</td>
<td>Action or POJO’s</td>
</tr>
<tr>
<td>ActionForward</td>
<td>Result</td>
</tr>
<tr>
<td>struts-config.xml</td>
<td>struts.xml</td>
</tr>
<tr>
<td>ActionServlet</td>
<td>FilterDispatcher</td>
</tr>
<tr>
<td>RequestProcessor</td>
<td>Interceptors</td>
</tr>
</tbody>
</table>
Architecture

Struts

HttpServletRequest

ActionContextCleanUp

Other filters (SiteMesh, etc)

FilterDispatcher

ActionProxy

Configuration Manager

struts.xml

Action Invocation

Interceptor 1

Interceptor 2

Interceptor 3

Action

Result

Interceptor 3

Interceptor 2

Interceptor 1

HttpServletRequestResponse

Tag Subsystem

HTML, Dojo, forms, etc

Template

JSP, FreeMarker, Velocity, etc

Key:
- Servlet Filters
- Struts Core
- Interceptors
- User created
Concepts

- Implemented as a server-side Model-View-Controller
- Combination of JSP’s, JSP tags, and Java servlets → **MVC Model 2 Pattern**
- Does not provide an specialized model components
MVC Model 2 architecture

Servlet

Java function

Controller

Model
Database, WS, etc.

View
Templates, layout

Client-Browser

JSP File

request
HTTP, CLI, etc.

response
HTML, RSS, XML, JSON, etc.

demand
data
Model Overview

- **Client browser**
  An HTTP request from the client browser creates an event. The Web container will respond with an HTTP response

- **Controller**
  The Controller receives the request from the browser, and makes the decision where to send the request. With Struts, the Controller is a command design pattern implemented as a servlet. The struts-config.xml file configures the Controller.
Model Overview (Cont)

- **Model state**
  The model represents the state of the application. ActionForm bean represents the Model state at a session or request level, and not at a persistent level. The JSP file reads information from the ActionForm bean using JSP tags.

- **View**
  The view is simply a JSP file. There is no flow logic, no business logic, and no model information -- just tags. Tags are one of the things that make Struts unique compared to other frameworks like Velocity.
Flow of Struts 2.0

- User Sends request
- FilterDispatcher determines the appropriate action
- Interceptors are applied
- Execution of Action
- Output rendering
- Display the result to user
Struts 2 Core components

- Action handler
  Interacts with other layers
- Result Handler
  Dispatches the request to view
- Interceptors
  configured to apply the common functionalities like workflow, validation etc.. to the request
- Custom Tags
  Render the dynamic content
<struts>
  <include file="struts-default.xml"/>
  <constant name="struts.custom.i18n.resources" value="MessageResources"/>
  <package name="default" extends="struts-default">
    <action name="list" class="web.DefectsList">
      <result>/pages/defects.jsp</result>
    </action>
    <action name="action_*" method="{1}" class="web.DefectsAction">
      <result name="input">/pages/editDefect.jsp</result>
      <result type="redirect">list.action</result>
    </action>
  </package>
</struts>
**<include>**

- Used to modularize application
- Always a child of `<struts>` tag
- Only attribute "file" implies the config file
  
  `<include file="module1-config.xml">`

- Order of including files are important
  
  explicitly include: "struts-default.xml" and the "struts-plugin.xml" files
<package>

- name – unique
- extends - “struts-default”
- namespace- admin, test
- Abstract-if “true” actions configured will not be accessible via the package name
Interceptor

- They provide a way to supply pre-processing and post-processing around the action
- Examples include exception handling, file uploading, lifecycle callbacks and validation
Interceptor (Cont*)

```xml
<interceptors>
  <interceptor name="autowiring"
      class="interceptor.ActionAutowiringInterceptor"/>
</interceptors>

<action name="my" class="com.fdar.infoq.MyAction" >
  <result>view.jsp</result>
  <interceptor-ref name="autowiring"/>
</action>
```
com.opensymphony.xwork2

Interface Action

Field Summary
- static String ERROR -- The action execution was a failure.
- static String INPUT -- The action execution require more input in order to succeed.
- static String LOGIN -- The action could not execute, since the user most was not logged in.
- static String NONE -- The action execution was successful but do not show a view.
- static String SUCCESS -- The action execution was successful.

Method Summary
- String execute() -- Where the logic of the action is executed
Pros

- **Use of JSP tag mechanism**
  The tag feature promotes reusable code and abstracts Java code from the JSP file. This feature allows nice integration into JSP-based development tools that allow authoring with tags. FilterDispatcher determines the appropriate action.

- **Tag library**
  Why re-invent the wheel, or a tag library? If you cannot find something you need in the library, contribute. In addition, Struts provides a starting point if you are learning JSP tag technology.
Pros (cont)

- **Open source**
  You have all the advantages of open source, such as being able to see the code and having everyone else using the library reviewing the code. Many eyes make for great code review.

- **Sample MVC implementation**
  Struts offers some insight if you want to create your own MVC implementation

- **Manage the problem space**
  Divide and conquer is a nice way of solving the problem and making the problem manageable. Of course, the sword cuts both ways. The problem is more complex and needs more management.
Cons

- Youth
- Changes in Struts
- Correct level of abstraction
- Limited scope
- J2EE application support
- Complexity
Demo
References

- Webwork official site: http://www.opensymphony.com/webwork/
- Apache Software Foundation: http://struts.apache.org/2.2.1/index.html
- Struts2 Tutorial: http://struts.apache.org/2.0.9/docs/tutorials.html
- Starting Struts2 Ian Roughly: http://static.raibledesigns.com/repository/presentations/MigratingFromStruts1ToStruts2.pdf
Thanks!