Basic Struts Web Development

David Lucek
www.lucek.com
dave@lucek.com

Assumptions and Pre-requisites

- You should know how to build a basic Servlet/JSP application and understand web programming before using Struts
- Extract out the sample application. It includes the full source. Can download form www.lucek.com
- Unzip to the c:\ drive or $HOME

What is Struts

- Jakarta Struts is a Model View Controller Framework for J2EE Web Development
- Probably the most popular J2EE Web Framework
- Requires V2.3 Servlet and 1.2 JSP Container
- Web Site: jakarta.apache.org/struts
- Latest Version is 1.1, released in August 2003

Struts Components

- Controller
  - ActionServlet Class – Main dispatcher in a Struts application
    - Configuration Files – struts-config.xml
  - Action Class – decouples the request processing from the business model
- Model – Does really not have a model component
- View
  - Action Forms
  - Tag libraries
  - Resource Bundles
  - Validator

Configuring Struts

- A good starting point is the struts-blank.war application. It's a skeleton Struts setup.
- Add ActionServlet to the web.xml file and add a "*.do" servlet mapping
  - This means that all "struts" enabled requests will must end in "*.do"
- Add tag library descriptors to web.xml, most common are html, bean, and logic

ActionServlet Class

- Performs HTTP Request Processing
- Calls the execute() method on the Action class based on a URL to Action Class mapping. (ActionMapping class)
- This mapping is configured in the struts-config.xml file.
- Forwards request to the URL returned by the execute() method. (ActionForward class)

Action Mapping Configuration

- Allows declarative mappings between URL's and Actions that process the request.
- Config file: struts-config.xml
- Decouples request URL from request processing component and view components.
- Provides declarative exception handling and other features.
- Common Action Mapping is the "Forward Only"

Action Class

- Processes the request and then forwards to the appropriate ActionForward
- Is where is the business model is called
- Receives input from View component via the ActionForm class
- Calls ActionForm.validate() if “validate=true” in config file.
- Must extend the “Action” class and then implement the execute() method
- Common Action Classes
  - Dispatch action – executes class method named in “action” parameter.

View – ActionForms

- Object encapsulating an HTML form, holds submitted form parameters/fields
- If ActionForm is configured in the Action Mapping. The ActionServlet will find an instance or create an instance and populate the parameter fields
- Need to derive off of ActionForm
- Main methods: reset(), validate()

View - ActionForm Continued

- Two Main Types of ActionForms
  - ActionForm – Static, derive from ActionForm
  - Dyna ActionForm – Dynamically created, form definition listed in <form-bean> inside the struts-config.xml file
    - No Java code required
    - Use validator plug-in for validation
    - There are issues with the Dyna Forms
- Need to define using <form-bean> in struts-config file and in each Action Mapping using the “name” attribute
- The ::validate method is used for validation

View – ActionForm Continued

- Works in conjunction with ActionErrors and ActionMessages for user feedback
- Most fields in an ActionForm should be Strings, because an incorrect field or error condition can be easily displayed
  - The Validator Plug-In requires most ActionForm fields to be Strings, otherwise the validator will not work

View – Tag Libraries

- The ActionServlet will normally forward to a JSP after the Action's execute() method is executed.
- Tag libraries cleanly separate the presentation layer
  - A web designer could use the Tag libraries and know nothing about the business model
  - The goal is to have NO Java code in your JSP
- Struts has the following tag libraries
  - HTML, Bean, Logic, Template, Nested, Tiles
  - HTML tag libraries are used for forms processing, works in conjunction with the ActionForm and validation framework
    - Allows convenient error and message feedback mechanism to the user

View – Tag Libraries Continued

- Bean tag library provides Java Bean assessors and mutators
- Logic tag library is used for JSP conditional expressions, looping, and other collection tags
- HTML, Bean, and Logic are the most commonly used Tag libraries
- The HTML tag library is almost required when using Struts ActionForms and its validation Framework.

View – Tag Libraries Continued

- The Bean and Logic are good but are tedious to use. (long tag names and not intuitive)
  - The Bean library might be required when using resources with Struts
- A better choice might be to use the Java Standard Tag Library (JSTL)
  - Included with Tomcat 5.0
  - JSTL 1.1 EA is included in the Java Web Services Developer Pack 1.2
- See example application for usage

View – Resource Bundles

- Provide for Internationalization by using Java resource files
- Must list the resource file name in the <message-resources> tag inside the struts-config.xml file
- Can have multiple resource files
- Most commonly used with <bean:message key="field.name" /> which displays the resource string
- Almost all tags from the HTML and Bean libraries support internationalization.

Form Validation – Standard

- Server Side Based
- Requires two action mappings
  - One with validate=false for first time the page is displayed
  - One with validate=true for form submission
- Requires HTML tag library with the <html:errors/> or other “messages” tag in your JSP HTML form
- Implement the ActionForm::validate() method in your derived class
  - Validate the fields and add errors to the ActionErrors() container object as required

Form Validation – Standard Continued

- If the validate() method returns a non empty ActionErrors object then the controller will re-display the form with the errors shown
  - The <html:errors/> tag displays the errors
  - The original form submission values are preserved
- Can display the error next to the field and other cool stuff
- See example application

Form Validation - With the Validator Plug-In

- Uses the Jakarta Commons Validator Framework to externalize and standardize validation routines
  - Allows validation rules to be defined for each form's fields in a XML file (validation.xml)
- Framework Setup
  - Add Validator Plug-in config in struts-config.xml file
  - Add validator-rules.xml and validation.xml files to project
  - Add required resources to applications message resource file
  - Derive off of:
    - Static Action Forms use ValidatorForm or ValidatorActionForm
    - Dyna Action Forms use DynaValidatorForm or DynaValidatorActionForm

Form Validation - With the Validator Plug-In Continued

- The Validator can produce validation code on both the server and client side.
  - The server side works well
  - The client side is flakey because of JavaScript differences between browsers. A few things will not work.
    - Probably best to stay with server side
    - With client side only you are limited to the small number of validation rules. To extend the client side you have to extend the validator, which is painful for the client side

Form Validation - With the Validator Plug-In Continued

- Basic Validator Rules
  - required
  - mask – regular expression
  - Range – is going away, use intRange
  - maxlength and minlength
  - byte,short,integer, long, float, double
  - date
  - creditCard
  - email
  - requiredif - will be taken out
  - Validwhen - Coming next release, NOT in Struts 1.1 Final Release

Form Validation - With the Validator Plug-In Continued

- Application Setup
  - Derive off of appropriate class
  - For each form in struts-config.xml add a <form> tag with the validation rules for each field in the validation.xml file
  - For client side validation must add special JSP tags
- See example application

Form Validation – Combining Standard Validation with the Validator Framework

- The Validator Framework has limitations
  - Inter-related fields
  - Can not implement special logic without extending the framework
  - Can not compare two fields (The upcoming validwhen rule should fix this)
- Combine Standard and Validator Approach
  - Derive off of the *ValidatorForm and override the validate() method
  - Call super.validate() first and then put custom validation below
  - See example application

Design Guidelines 1

- Have all web application requests go through a Controller component, even if it is just a forward
- Do not use an ActionForm as your business model. Have the Action create DTO's and other service objects
- Use Message resources even if there are no multi-language requirements

Design Guidelines 2

- Struts has Database connection features, use the containers database connection features instead.
- No business logic in the JSPs.
- No Java code in the JSPs
- Keep validations on the server side
- Might want to use JSTL 1.0