

Struts

An Open-source Architecture for Web Applications

David Morris

dmorris@midrangeserver.com

Fall COMMON 2003

450180

47GS

Copyright © 2003 David Morris Consulting

Overview

- Why should you use Struts
- Installing Struts
- Struts Concepts
 - What is MVC
 - Configuring Struts
 - Tag libraries
- Build a practical example
- What do you do when things go wrong?
- Advanced Topics

Important Concepts

Struts is a Servlet that supports your applications

JSP Pages are also Servlets

Struts provides a pluggable framework

Model View Controller separates responsibilities

Tools exist to simplify Struts development

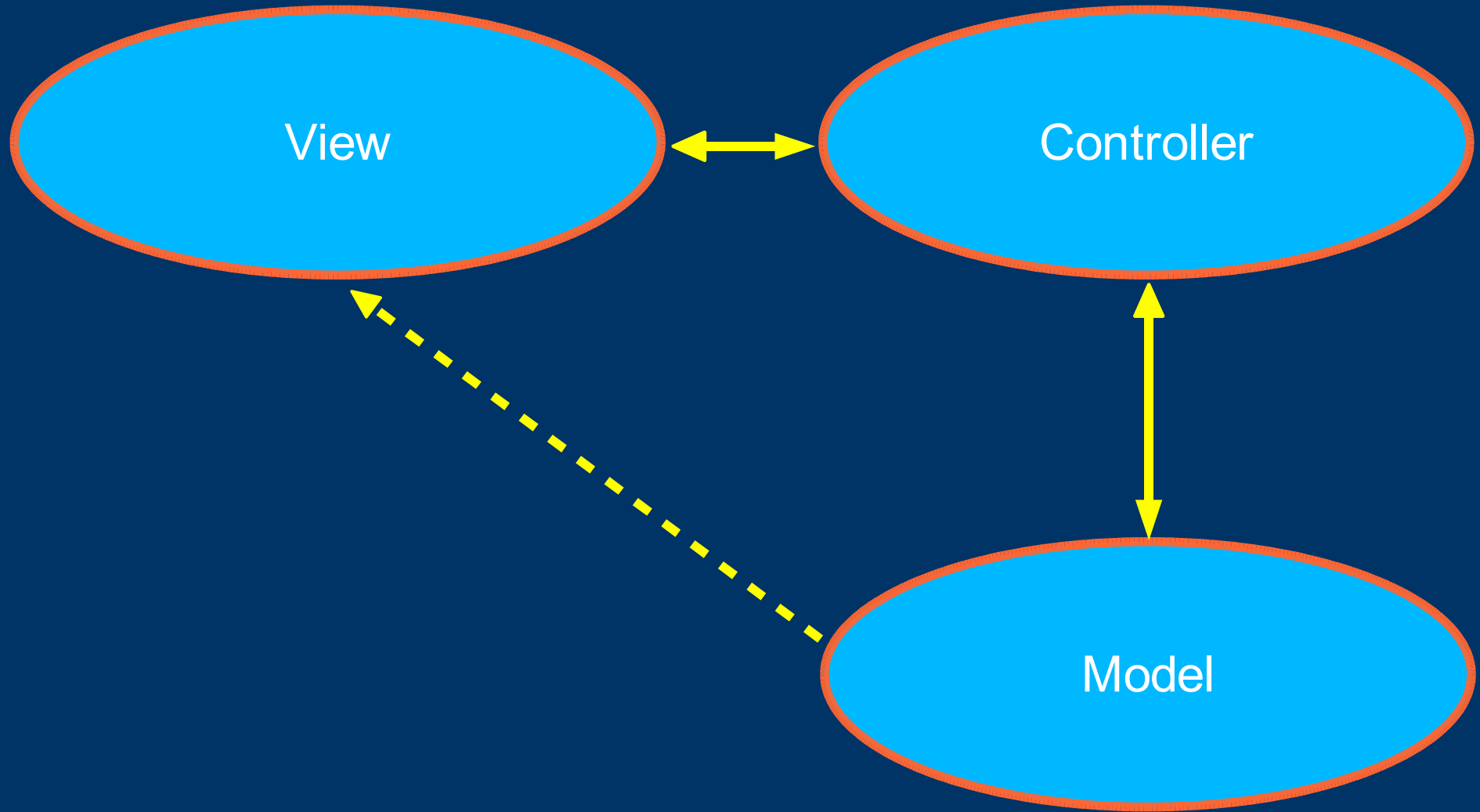
Developer Goals

- Leverage existing skills
 - The pool of Struts developers is large and growing
 - Struts is the standard Servlet/JSP Framework!
- Build on quality components
 - Use widely deployed technologies
 - Define interfaces, preconditions, post-conditions and test to prove it!
- Use known design patterns
 - Model View Controller pattern clearly separates presentation, control, and business logic
 - Avoid vendor lock in!

Fitting the Pieces

- Struts is the "plumbing" – plug in your pieces
- Familiar concepts to iSeries programmers:
 - Taglibs = DDS Keywords
 - Struts Form = Display File
 - Action Class = Main line program
 - Business logic beans = Service programs
- Struts runs on any platform – Standards based
 - Database can be DB2, MySQL, etc.
 - Servlet support can come from Tomcat, WebSphere, WebLogic, etc.
 - There are lots of examples and help available

Model View Controller



Requirements

- A Java Runtime Environment
 - Use latest stable version for best results
 - Add tools.jar for JSP pages
- A Servlet Container
 - Tomcat, WebSphere, WebLogic, etc.
- Memory and CPU
 - More is always better!

Setup

Download and install the following:

- A JDK and Java Runtime Environment
iSeries, Sun, JRocket, Blackdown, etc.
- An application server
Tomcat 4.1.27 will work just fine
- Struts
Currently Struts 1.1

Tomcat Setup

- Download release from <http://jakarta.apache.org>
 - For iSeries get .zip, Linux .tar, Windows .exe
- Unpack zip files or run Windows executable
- Delete folders under webapps
- Eclipse - Import Tomcat as simple project
- Add web application context to conf/server.xml
 - Use no-examples as base
 - Remove Tomcat-Apache service
 - Setup security realm if necessary

Running Tomcat

On a PC:

With .exe version Tomcat starts as a service

With .zip version run ../bin/startup.bat

On the iSeries:

Submit using Qshell (QSH) command

Modify iSeries-toolkit script

sourceforge.net/projects/iseries-toolkit

Within Eclipse:

Use sysdeo or WDSc plugin

Installing Applications

- Struts is a Servlet Application
- Drop a .war file into Tomcat's ../webapps directory
- Create application under ../webapps directory
- Add a context entry into ../conf/server.xml

```
<Context path="" docbase="/webapps/span" debug="2" />
```

Struts Web Deployment

Modify application's web.xml to add Struts Action:

```
<web-app>
  <display-name>myApp</display-name>
  <servlet>
    <servlet-name>action</servlet-name>
    <servlet-class>org.apache.struts.action.ActionServlet</servlet-class>
    <init-param>
      <param-name>application</param-name>
      <param-value>ApplicationResources</param-value>
    </init-param>
    <init-param>
      <param-name>config</param-name>
      <param-value>/WEB-INF/struts-config.xml</param-value>
    </init-param>
    ...
    <load-on-startup>1</load-on-startup>
  </servlet>

  <servlet-mapping>
    <servlet-name>action</servlet-name>
    <url-pattern>*.do</url-pattern>
  </servlet-mapping>
</web-app>
```

Configuring Struts

Once:

- Add action servlet to web.xml

 - Consider using multiple struts-config files

- Modify struts-config.xml

 - Decide whether to use caching

 - Add plugins

Per application:

- Add form definition

- Add action-mappings

Forms

- Struts Display files
- JSP corresponds to a form bean

Basic Flow:

Call reset

Properties set on form bean using reflection

The validate method is called

The form is presented to an Action

The action modifies form values

The JSP page displays modified values from the form (or model)

Taglibs

- Heart of JSP pages
- Encapsulate presentation logic
 - Can encapsulate business logic
- Display file of the Servlet world
- Actually compile into a Servlet
- Easy to understand and separate form from function
- Customizable

Warning

Unfortunately

JSP Pages take a lot of time to develop

Tag libraries are prone to bugs and difficult to keep up to date

ApplicationResources.properties

- Used to store application constants
 - Example: *text.customer.name=Name*
 - Predefined values for errors like: *errors.header*, *errors.footer*, *errors.prefix*, and *errors.suffix*
- Used for error messages
 - Example: *errors.required={0} is required.*

Accessing Bean Properties

- Struts tags use reflection to locate properties
 - Access nested properties with dot notation
 - The name of the bean is optional and will default to the form bean if not specified

```
<bean:write name="myBean" property="customer.name"/>
```

In this example myBean is the name of an attribute set in request or session scope:

```
SomeBean someBean = new SomeBean();  
someBean.getCustomer().setName("David");  
request.getSession().setAttribute("myBean", someBean);
```

Actions

- Extend `org.apache.struts.Action`
 - You must implement `execute` method that receives `mapping`, `form`, `request`, and `response`
- Your `execute` method is called by Struts Action at the appropriate time (remember `web.xml`?)
- Process form data, invoke business rules, decide where to go next, and return status
- `DispatchAction` simplifies Action logic by calling targeted methods

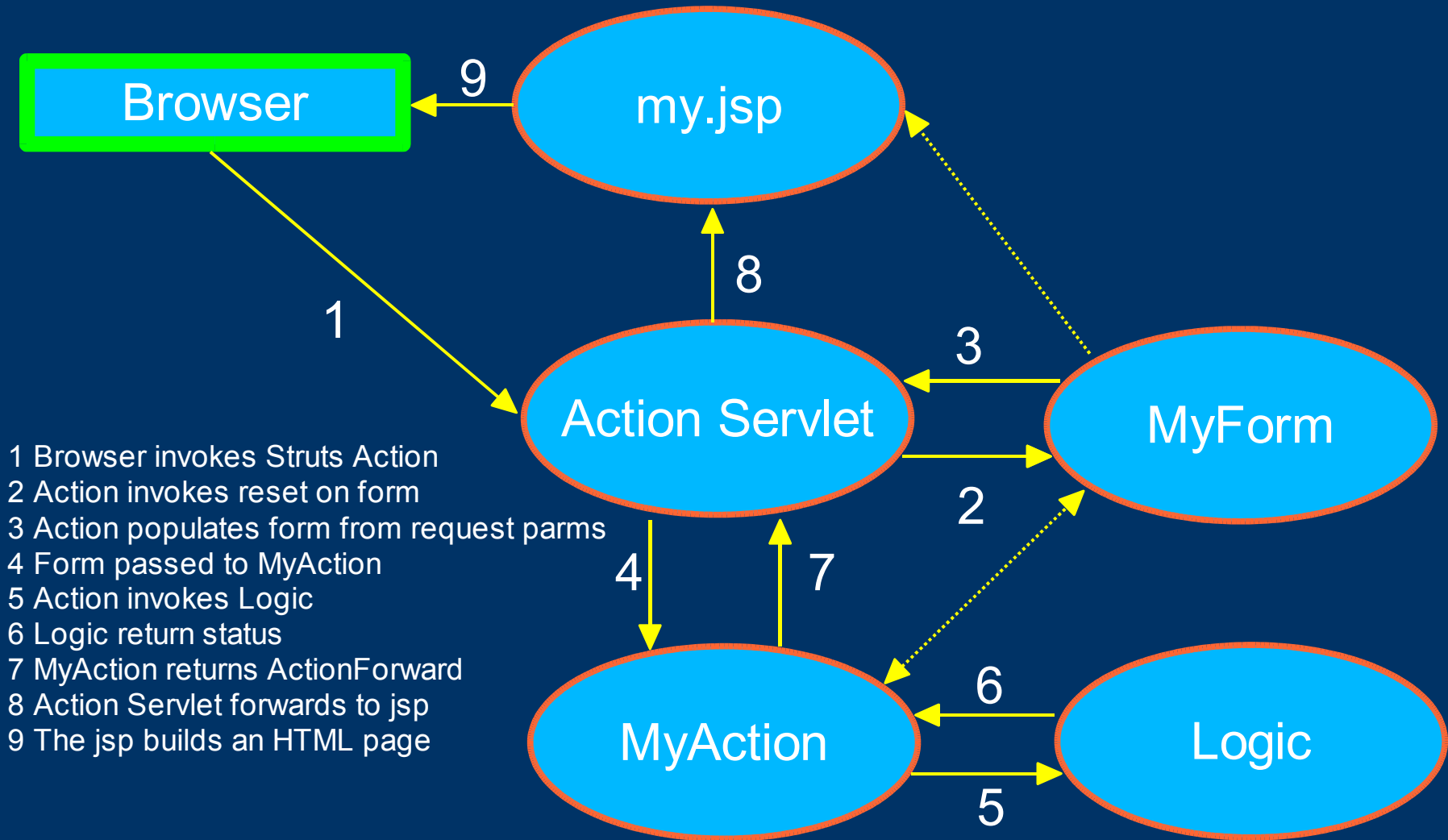
Returning Messages

- Struts contains built in support for messages
 - ActionMessages contains a set of messages
 - ActionMessage is a stores an individual message
- ActionErrors returned from a form's validate method will end processing
- There are two types of messages supported:
 - ActionErrors.GLOBAL_MESSAGE
 - ActionMessages.GLOBAL_MESSAGE
- Call saveMessages(request, messages) in Action to prepare messages for display

Struts Flow

1. User select URL ending in .do ie: /myApp.do
2. Struts Action servlet receives control
3. URL is associated with an Action class
The Action class is associated with a Form bean
4. The Form's reset method is called
5. Request parameters are mapped to the Form
6. The execute method on the Action class is called
An ActionForward is returned ie: /myApp.jsp

Visualizing Struts Flow



Creating a Struts Application

1. Create entity beans
2. Create logic beans to manipulate entity beans
3. Create form bean with necessary properties
 - You skip this step with DynaActionForm
4. Create Action class
 - Extend or Action DispatchAction
5. Update struts-config.xml
 - Define global forward, form, and action mappings
6. Create JSP page to support application

Step 1: Create Entity Beans

- Ideally entity beans are platform independent
- It is usually best if domain model closely resembles underlying database
- Hibernate and ibattis provide good support in this area

```
public class Customer
    implements Serializable {
    /** persistent field */
    private long id;
    private String name;
    private Set orders;
    ...
    public Set getOrders() {
        return this.orders;
    }
    public void setOrders(Set orders) {
        this.orders = orders;
    }
}
```

Step 2: Create Logic Beans

- These will sit between the controller and model
- Service layer that provides APIs over domain model
- Platform independent
 - Don't pass in ServletRequest, etc. use helper classes if

```
public class CustomerService {
    public Customer addCustomer(BaseDao dao, Customer customer)
        throws DatabaseException {
        try {
            dao.add(customer);
        } catch (DatabaseException de) {
            //log error
            throw de;
        }

        return customer;
    }
}
```

Step 3: Create Form Beans

- Form beans are a place holder for html form fields
- Loaded by Action servlet prior to calling associated action's execute method

```
public class CustomerForm extends ValidatorForm {
    private Customer customer;

    public void reset(ActionMapping mapping,
                     HttpServletRequest request) {
        if (customer == null) {
            customer = new Customer();
        }
    }

    public Customer getCustomer() {
        return customer;
    }

    public void setCustomer(Customer customer) {
        this.customer = customer;
    }
}
```

Step 4: Create Action Class

- Controller that directs work
- Called by Action servlet (remember web.xml?)

```
public final class CustomerDispatchAction extends BaseDispatchAction {
    public ActionForward add(
        ActionMapping mapping, ActionForm form, HttpServletRequest request,
        HttpServletResponse response)
        throws Exception {
        // Validate form for errors
        boolean valid = isValid(mapping, form, request);

        if (valid) {
            CustomerForm customerForm = (CustomerForm) form;
            CustomerService service = new CustomerService();
            service.addCustomer(getDao(request), customerForm.getCustomer());

            ActionMessages messages = new ActionMessages();
            ActionMessage message = new ActionMessage(
                "message.customer.add.success", customerForm.getCustomer().getName());
            messages.add(ActionMessages.GLOBAL_MESSAGE, message);
            saveMessages(request, messages);
            customerForm.setAction("update");
            request.setAttribute(mapping.getAttribute(), customerForm);
        }

        return mapping.findForward("refresh");
    }
}
```

Controlled validation

Return messages

What's next

Step 5: Update struts-config.xml

Add form bean, global forward, and action mapping

```
<form-beans>
  <form-bean name="customerForm" type="org.iseriestoolkit.span.CustomerForm" />
</form-beans>

<!-- Global forwards -->
<global-forwards>
  <forward name="customer" path="/customerAction.do" redirect="true" />
</global-forwards>

<!-- Action Mapping Definitions -->
<action-mappings>
  <action path="/customerAction"
    type="org.iseriestoolkit.span.CustomerDispatchAction"
    name="customerForm" scope="request"
    validate="false"
    parameter="action"
    input="/WEB-INF/jsp/customerForm.jsp">
    <exception key="exception.database.error"
      type="org.iseriestoolkit.span.DatabaseException"
      path="/error.jsp" />
    <forward name="refresh" path="/WEB-INF/jsp/customerForm.jsp" />
  </action>
</action-mappings>
```

Step 6: Create JSP Page

- Standard JSP works - favor Struts and JSTL tags
- Buttons are important and a challenge
 - One per page, javascript, or LookupDispatchAction

```
<%@ taglib uri="http://java.sun.com/jstl/core" prefix="c" %>
<html-el:html locale="true" xhtml="true">

<body>
<html-el:errors/>

<html-el:form action="customerAction" focus="customer.name">
  <html-el:hidden name="customerForm" property="action" />
  <html-el:hidden property="customer.id"/>

  <bean-el:message key="text.customer.name"/>
  <html-el:text property="customer.name" size="30" maxlength="50"/>

  <html-el:submit styleClass="button">
    <bean-el:message key="button.add" />
  </html-el:submit>
</html-el:form>
</body>
```

Tips

- Always start at .do – never show your .jsp
- Use some javascript to submit forms onClick() or onChange()
 - See CustomerListForm.jsp
- Don't bother with /do/myAction notation
- When back buttons and refresh won't work
 - Place forms in session scope with nocache="true"
 - Always redirect to a get before presenting page
(use xxx.do?action="refresh")
 - This is an all or nothing solution that could impact performance due to memory utilization

Plug-in Custom Features

- Startup initialization
- Use for any class that takes some time to start
 - Build a menu Map from XML documents, etc.
- Any class that implements the `org.apache.struts.action.PlugIn` interface
 - Must implement an `init` and `destroy` method

Configure in `struts-config`:

```
<plug-in className="org.apache.struts.validator.ValidatorPlugIn">  
  <set-property property="pathnames"  
    value="/WEB-INF/validator-rules.xml, /WEB-INF/validation.xml" />  
</plug-in>
```

Anatomy of an Application - CRUD

Span is a full function application supporting add, update, and delete operations over an iSeries or MySQL database

Goals:

- Minimize coding time

- Support common use cases

- Provide the ability to target any database

- Use reasonable practices

- Demonstrate the capabilities of Struts

Span Overview

Obtain with CVS:

```
cvcs -z3 -d:pserver:anonymous@cvs.sourceforge.net:/cvsroot/iserics-toolkit co span
```

There is no password

- Two operations are supported – List customers and maintain customers
- Hibernate provides database support
- CustomerListAction passes add and update requests to CustomerAction
 - Return point ActionForward is set in forwarding method

Advanced Topics

- Building, Testing, and Debugging Struts applications
 - Using Ant
 - Testing with jUnit
 - What to look for when debugging
 - Chaining actions
- Struts tools
 - Setting up Eclipse and WDS
- Persistence
 - Using a persistence layer

Using Ant

- Ant is a Java based scripting tool
- Scripts defined using XML (build.xml)
- Plugins for many IDEs including Eclipse, Net Beans, and WDSoc
- Use Ant to create, document, distribute, pack, package, test, shake, and stir Java applications

```
<mkdir dir="${build}" />  
<javac debug="${debug}"  
    deprecation="${deprecation}"  
    destdir="${build-target}"  
    failOnError="true"  
    includes="**/*.java"  
    srcdir="${src}">
```

Testing with jUnit

- jUnit is a Java-based unit testing tool
- Available at www.junit.org
- Tests should match package of tested class
- Extend `junit.framework.TestCase`

```
public class BaseDAOTest extends TestCase {
    ...
    public void testAddObject() {
        Customer customer = new Customer();
        customer.setName("Fred Flintstone");
        customer.setEmail("fred.flintsone@toonville.com");

        BaseDao bDAO = new BaseDao();

        try {
            bDAO.add(customer);
        } catch (Throwable t) {
            fail("Unable to add new Customer " + customer);
        }
    }
}
```

Testing Struts Applications

- jUnit provides an easy to use tool for unit testing
- Execute jUnit tests via Ant to regression test applications
- Use jMeter for load testing
- Cactus provides support for testing Servlets, customer tags, etc
jakarta.apache.org/cactus
- Use Struts test case to test actions without a running servlet engine
sourceforge.net/projects/struststestcase

Debugging Tips

- Use an IDE like Eclipse
- Download the source for Struts and bean-utils
- With Eclipse:
 - Add the source to struts.jar and bean-utils.jar
 - Note: I have done this for in the span project jar files
- Set a break point in your form's reset method to catch errors setting bean properties
- Make sure that ActionForward is set on exiting Actions

Common Errors

- Check boxes are not set
 - Use reset method in form to set check box properties to false
- Session timeouts cause null pointer exceptions
 - Use lazy lists in forms with factory
- Properties not set on form
 - Verify type is correct
 - Make sure form values start with lower case and each separator (.) has a getter
- Button does not call proper method
 - Use javascript if necessary to set action for buttons

More Common Errors

- Action class is not being called on submit
 - Verify action path in struts-config matches the form action name
 - Remove validate="true" from struts-config and call form.validate directly from action
- When chaining actions form properties are lost
 - The reset() action is called between actions so add properties to the request string

Chaining Actions

- Used when more than one action will process a single request

Challenges:

- Different actions will have different forms – input may not match up
- Setting return point is difficult

Solutions:

- Avoid chaining actions
- Start with a clean slate each time – use redirect
- Store return point in session

Using Eclipse or WDS*c*

- WDS*c* version 5
 - Wizards that support common tasks
 - Web project, ActionForm, Action class, and Struts JSP
 - Editing support for struts-config
- Run Tomcat within Eclipse
 - www.sysdeo.com/eclipse/tomcatPlugin.html
- Set up Tomcat as a simple project
- Set up Struts applications as Java projects
- Use CVS for source change control

Persistence

The M in MVC

You have to do something!

It can be as simple as data access beans that encapsulate JDBC calls

It can be as complex as EJBs

One platform independent option is Hibernate

Domain to object mapping

A simple but powerful mapping tool is ibattis

Maps SQL statements to objects

Potential Alternatives

- Use a commercial Struts package
 - Espresso
 - Basebeans
 - WebSphere Development Studio
 - Many others
- Start simple
 - Choose a framework after you are comfortable with Struts concepts
 - Hire a mentor to guide you through the learning process

Recommendations

- Download and install Struts
- Extend or modify an example application
- Identify areas that don't meet your needs and explore alternatives
- Join the user list mailing list and ask questions
- Contribute to Struts and other open source projects by offering feedback and help!

Resources

jakarta.apache.org/struts:

The main Struts page containing links to resources, downloads, news, and bug resources

sourceforge.net/projects/struts:

Example applications

sourceforge.net/projects/iserie-toolkit:

iSeries applications and Struts Span example from this presentation

Support and Reference

Mailing lists:

struts-user@jakarta.apache.org

struts-dev@jakarta.apache.org

Books:

See the Struts main page

Tutorials:

www.reumann.net/do/struts/main

Quite a few others are available