Struts
In this presentation we will discuss:

- **Struts Overview**
  - Where to place Struts
  - The Struts controller
  - The Struts Model
  - The Struts View
  - The structure of Struts
  - A Struts example
A PML View

Improve Development Process

Use Frameworks
Struts Description

- Struts is an open source framework
- Struts provides a methodology and framework enabling rapid Java web application development utilizing:
  - Service to Worker (MVC)
  - Front Controller
  - View Helper
  - DTO
Placement of Struts

Client -> Struts Framework -> Custom Code -> Components

Copyright © 2002 by The Middleware Company
The Controller in Struts is embodied by

- The ActionServlet, which serves to:
  - Parse XML configuration file provided by developer
  - Bundle and route HTTP requests to specific Action classes and JSPs as mandated by the configuration file
  - Filter the request and response behavior according to the mapping present and the results of related operations

- Inheritors from the Struts-provided base class: Action
  - Used as Dispatchers and ViewHelpers
  - Should delegate business logic to isolated classes (POJO, SessionBeans) that do not depend upon the Struts APIs
The Controller

Start Here:

URI → Action → Controller → View

Maps to

Returns control to

Forwards to

Reads Configuration file and obtains path mapping... Compares it to
The Model

The Model in Struts is embodied by Java Beans and ActionForm classes

Java Beans can be:
- Application-specific types
- Inheritors from the Struts provided base class: ActionForm
  - Useful for maintaining state and holding form-entry information
  - Used as ViewHelpers
The Controller and Model

ActionServlet

<< dispatch >>

Action

+perform()

<< instantiate >>

ActionForm

+validate()

<< use >>
The view in Struts is embodied by both:

- JSP pages and Servlets
- Non-Java web resources such as XML and HTML pages

- A rich set of taglibs is associated with Struts
  - This can aid in building complex views

- Content (even field labels) can be removed from JSPs

- Struts JSPs provide formatting and error handling
Structure ...continued
Here we will show a simple Struts example including:

- The Struts configuration files
- Action Classes
- ActionForm classes
- JSPs utilizing struts-specific taglibs

We will not show:

- The Helper class that accesses the EJB layer
- The EJBs
- The DTOs
- The error.jsp
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE struts-config PUBLIC "-//Apache Software Foundation//DTD Struts Configuration 1.0//EN" "http://jakarta.apache.org/struts/dtds/struts-config_1_0.dtd">
<struts-config>
  <form-beans>
    <form-bean name="inventoryForm" type="com.test.InventoryForm"/>
  </form-beans>
  <action-mappings>
    <action path="/inventory"
      type="com.test.InventoryAction"
      name="inventoryForm"
      scope="request"
      input="/selectCategory.jsp">
      <forward name="error" path="/error.jsp"/>
      <forward name="success" path="/viewItems.jsp"/>
    </action>
  </action-mappings>
</struts-config>
inventory.page.title=First Struts Inventory System
viewItems.page.done=That is all of the items in that category. 
<p />&nbsp;Thankyou for visiting!
select-category.page.heading=Greetings!
select-category.page.message=Please enter the name of the item category in the textfield below. Then press submit to view all items belonging to that category: (example: FRUIT)
error.inventory.itemCategory.wrong=Please enter one of these choices into the item category field: FRUIT, MEAT, VEGETABLE
error.inventory.itemCategory.required=Please enter a value into the text field before clicking submit.
package com.test;

import javax.servlet.*;
import javax.servlet.http.*;
import org.apache.struts.action.*;
import java.util.*;

public final class InventoryForm extends ActionForm {
    private String itemCategory;
    private Collection items;

    public InventoryForm(){}
    public void setItems(Collection c){
        this.items = c;
    }
    public Collection getItems(){
        return items;
    }
    ...
}
public ActionErrors validate(ActionMapping mapping, HttpServletRequest request) {
    ActionErrors errors = new ActionErrors();
    if (itemCategory == null || itemCategory.length() < 1) {
        errors.add("itemCategory", new ActionError("error.inventory.itemCategory.required"));
    }
    else if (!
             ("FRUIT".equalsIgnoreCase(itemCategory)) ||
             ("MEAT".equalsIgnoreCase(itemCategory)) ||
             ("VEGETABLE".equalsIgnoreCase(itemCategory))) {
        errors.add("itemCategory", new ActionError("error.inventory.itemCategory.wrong"));
    }
    return errors;
}

public String getItemCategory() {
    return itemCategory;
}

public void setItemCategory(String itemCategory) {
    this.itemCategory = itemCategory;
}
package com.test;

import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
import org.apache.struts.action.*;
import java.util.*;

public final class InventoryAction extends Action {
    
    com.test.HelperBean bean = null;
    public InventoryAction() {
    }

    public ActionForward perform(ActionMapping mapping, ActionForm form, HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        String fwd = "success";
        ActionErrors errors = new ActionErrors();
        InventoryForm iform = (InventoryForm) form;
        Collection c = null;

        ...
try{
    bean = new HelperBean();
    c = bean.getItemsInCategory(iform.getItemCategory());
    iform.setItems(c);
}
catch(DataAccessException dae){
    errors.add("org.apache.struts.action.GLOBAL_ERROR", new
    ActionError("error.inventory.page.dataAccess"));
}
finally{
    if(!errors.empty()){
        fwd = "error";
        return mapping.findForward(fwd);
    }
}
return mapping.findForward(fwd);
<jsp:forward page="SelectCategory.jsp" />
SelectCategory.jsp

```jsp
<%@ page language="java" %>
<%@ taglib uri="/WEB-INF/struts-bean.tld" prefix="bean" %>
<%@ taglib uri="/WEB-INF/struts-html.tld" prefix="html" %>
<html>
<title><bean:message key="inventory.page.title"/> : </title>
<body>
<h2><bean:message key="select-category.page.heading"/></h2>
<p/>
<html:errors/>
<html:form action="inventory.do" focus="itemCategory">
<br/>
<html:text property="itemCategory" />
<br/>
<html:submit value="getItems"/>
<html:reset/>
</html:form>
</body>
</html>
```
Greetings!

Please enter the name of the item category in the textfield below. Then press submit to view all items belonging to that category. (example: FRUIT)
<%@ page language="java" %>
<%@ taglib uri="/WEB-INF/struts-bean.tld" prefix="bean" %>
<%@ taglib uri="/WEB-INF/struts-logic.tld" prefix="logic" %>
<html>
<title>
  <bean:message key="inventory.page.title"/>
</title>
<body>
<table border="1">
  <logic:iterate id="item" name="inventoryForm" property="items" >
    <tr>
      <td><bean:write name="item" property="itemName" /></td>
      <td><bean:write name="item" property="category" /></td>
    </tr>
  </logic:iterate>
</table>
<bean:message key="viewItems.page.done" />
</body>
</html>
That is all of the items in that category.

Thank you for visiting!
Consequences

- Using any framework
  - speeds development efforts
  - helps to improve maintenance efforts on large projects
  - Requires an investment of time and training

- Open source frameworks are particularly beneficial in that
  - They do not cost anything to use
  - They are usually maintained and improved upon by a knowledgeable and motivated collection of programmers

- Struts is
  - Well established
  - Based upon existing standards
  - Helpful in separating the Model from View from Controller
Related Frameworks

- Velocity
- Sitemesh
- OSCache
- Cocoon
- Pushlets
- Tapestry
- WebWork
In this presentation we discussed:

- Struts Overview
  - Where to place Struts
  - The Struts controller
  - The Struts Model
  - The Struts View
  - The structure of Struts
  - A Struts example