Jakarta Struts: DispatchAction and Other Advanced Action Classes

Struts 1.2 Version

***Core* Servlets & JSP book: www.coreservlets.com**
***More* Servlets & JSP book: www.moreservlets.com**
Servlet, JSP, Struts, JSF, and Java Training Courses: courses.coreservlets.com

For live Struts training, please see JSP/servlet/Struts/JSF training courses at http://courses.coreservlets.com/.

Taught by the author of *Core Servlets and JSP*, *More Servlets and JSP*, and this tutorial. Available at public venues, or customized versions can be held on-site at your organization.

Agenda

- **Performing different logic based on a radio button, hidden field, or push button value**
  - But not repeatedly checking parameter names in the execute method
  - Using DispatchAction
- **Using Struts form-bean capabilities in non-Struts applications**
  - Without redoing code in Struts
  - Using ForwardAction
- **Other advanced Action subclasses**

DispatchAction

Grouping Related Operations

DispatchAction

- **Scenario**
  - The same form should result in substantially different logic depending on whether certain radio buttons were chosen, or depending on which submit button was pressed
    - But, in HTML, one form has one ACTION
  - Several actions share similar or identical helper methods
- **Problems**
  - Tedious and repetitive checking of parameters just to know what real method to invoke
    - Form bean must have getter/setters for the radio button
- **Goal**
  - Automate the dispatch logic (decision re which method applies), based on struts-config.xml

Example (Radio button named "operation")

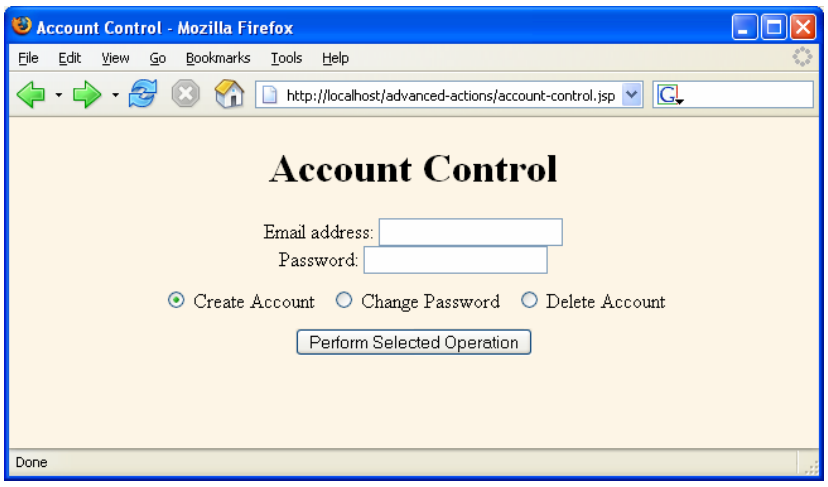
```
public ActionForward execute (ActionMapping mapping,
                              ActionForm form,
                              HttpServletRequest request,
                              HttpServletResponse response)
    throws Exception {
  UserFormBean formBean = (UserFormBean)form;
  if (radioButtonMatches(formBean, "createAccount")) {
    return(createAccount(mapping, formBean, request, response));
  } else if (radioButtonMatches(formBean, "changePassword")) {
    return(changePassword(mapping, formBean, request,
   response));
  } else if (radioButtonMatches(formBean, "deleteAccount")) {
    return(deleteAccount(mapping, formBean, request, response));
  } else {
    return(makeErrorMessage());
  }
}

private boolean radioButtonMatches(UserFormBean formBean,
                                   String value) {
  String operation = formBean.getOperation();
  return((operation != null) && operation.equals(value));
}
```

Using DispatchAction

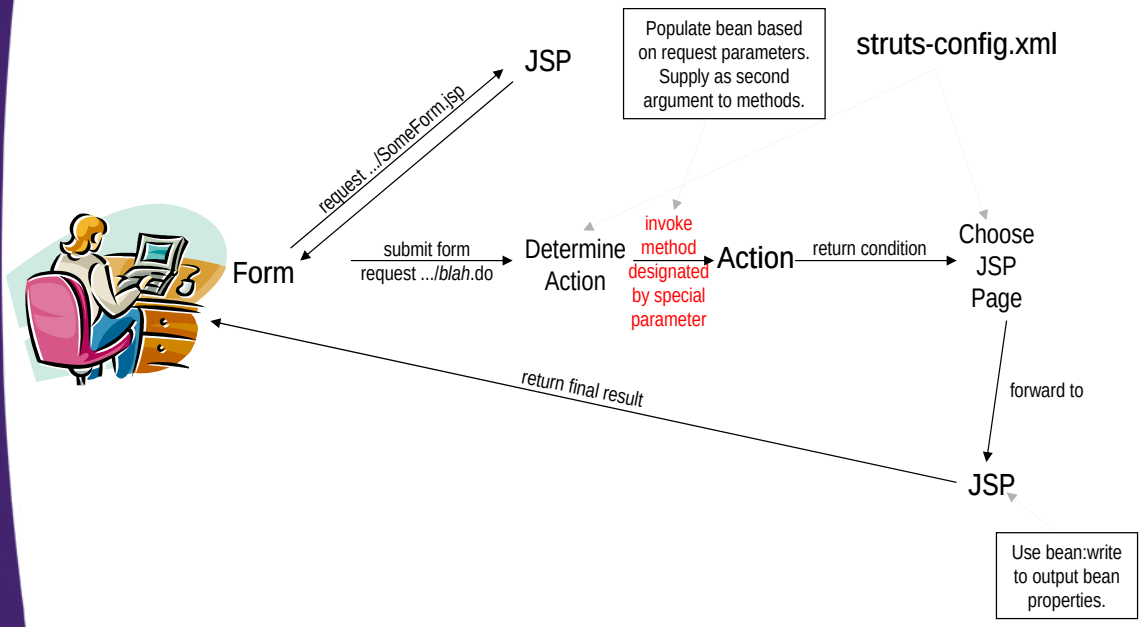
- **Use struts-config.xml to list the parameter used to determine which method will be called**
  - Use the parameter attribute of action
    <action path="..." type="..." parameter="operation">
- **Extend DispatchAction instead of Action**
  - Directly implement desired methods
    - Omit the execute method entirely
    - Custom methods have same signature as execute
  - Note package is ...struts.action**s**, not ...struts.action
- **In form, supply parameter with given name**
  - Value should be the name of the method that applies
    <INPUT TYPE="RADIO" NAME="operation"
    VALUE="createAccount">
- **Form bean needs no setters for button**

Example: Outline

- **Perform different operations depending on which radio button is selected**
- **Use same basic Action class in all cases**



Struts Flow of Control



Struts Flow of Control

- **The user requests a form**
  - The form has radio button, push button, or hidden field with a special name, and whose values correspond to method names
- **The form is submitted to a URL of the form *blah*.do.**
  - That address is mapped by struts-config.xml to an Action class
- **The designated method of the Action object is invoked**
  - The method is selected based on the special form parameter
  - One of the arguments to execute is a form bean that is automatically created and whose properties are automatically populated based on incoming request parameters of the same name
  - The Action object then invokes business logic and data-access logic, placing the results in normal beans stored in request, session, or application scope.
  - The Action uses mapping.findForward to return a condition, and the conditions are mapped by struts-config.xml to various JSP pages.
- **Struts forwards request to the appropriate JSP page**
  - The page can use bean:write or the JSP 2.0 EL to output bean properties
  - The page can use bean:message to output fixed strings

New Capabilities

- **The form has a special parameter**
  - The possible values correspond to method names
- **The parameter is declared in struts-config.xml**
  - With a `parameter` entry in the `action` element
- **The form bean needs no setter method for the special parameter**
  - For html:radio and most other html: tags, getter *is* needed
  - Otherwise, form bean defined and declared normally
- **Extend DispatchAction instead of Action**
  - No execute method
  - One method for each possible value of the special parameter
    - These methods have same signature as execute

The Six Basic Steps in Using Struts: Updates for Bean Use

1. **Modify struts-config.xml. Use WEB-INF/struts-config.xml to:**
   - Map incoming .do addresses to Action classes
   - Map return conditions to JSP pages
   - Declare any form beans that are being used
   - List special param via **`parameter`** entry in **`action`**
   - Restart server after modifying struts-config.xml
2. **Define a form bean.**
   - This bean extends ActionForm and represents the data submitted by the user. It is automatically populated when the input form is submitted.
     - The getter needed for this special parameter determines which radio button is initially selected
     - No setter needed for this special parameter

The Six Basic Steps in Using Struts: Updates for Bean Use

3. **Create results beans.**
   - These are normal beans of the sort used in MVC when implemented directly with RequestDispatcher. That is, they represent the results of the business logic and data access code. These beans are stored in request, session, or application scope.
4. **Define a DispatchAction class to handle requests.**
   - Omit the execute method
   - Define a method named for each possible method of the special parameter
     - Same signature as the execute method
   - The system will call that method automatically

The Six Basic Steps in Using Struts: Updates for Bean Use

5. **Create form that invokes *blah*.do.**
   - Use html:form to associate bean properties with input elements
     - Keeps textfield names in synch with bean property names
     - Allows form prepopulation
     - Allows form redisplay
   - Optionally use bean:message to output standard prompts
6. **Display results in JSP.**
   - The JSP page uses the bean:write tag to output properties of the form and result beans.
   - The page may also use bean:message.

Step 1 (Modify struts-config.xml)

```
<struts-config>
  <action-mappings>
    <form-beans>
      <form-bean name="userFormBean"
                 type="coreservlets.UserFormBean"/>
    </form-beans>
    <action path="/accountMod"
            type="coreservlets.ModifyAccountAction"
            parameter="operation">
      <forward name="create-failed"
               path="/WEB-INF/results/create-failed.jsp"/>
      <forward name="create-success"
               path="/WEB-INF/results/create-confirm.jsp"/>
      <forward name="change-failed"
               path="/WEB-INF/results/change-failed.jsp"/>
      <forward name="change-success"
               path="/WEB-INF/results/change-confirm.jsp"/>
     <forward name="delete-failed"
               path="/WEB-INF/results/delete-failed.jsp"/>
      <forward name="delete-success"
               path="/WEB-INF/results/delete-confirm.jsp"/>
    </action>
  </action-mappings>
</struts-config>
```


Step 2 (Define a Form Bean)

```
package coreservlets;
import org.apache.struts.action.*;

public class UserFormBean extends ActionForm {
  private String email = "";
  private String password = "";
  private String operation="createAccount";

  public String getEmail() { return(email); }

  public void setEmail(String email) {
    this.email = email;
  }
  ...
  public String getOperation() {
    return(operation);
  }
}
```

Step 3 (Define Results Beans)

- **Omitted in this simple example**

Step 4 (Define a DispatchAction Class to Handle Requests)

```
package coreservlets;

import javax.servlet.http.*;
import org.apache.struts.action.*;
import org.apache.struts.actions.*;

public class ModifyAccountAction
             extends DispatchAction {
```

Example: DispatchAction Code (Continued)

```
  public ActionForward createAccount
                          (ActionMapping mapping,
                           ActionForm form,
                           HttpServletRequest request,
                           HttpServletResponse response)
      throws Exception {
    if (isComplexBusinessLogicSuccessful("create")) {
      return(mapping.findForward("create-success"));
    } else {
      return(mapping.findForward("create-failed"));
    }
  }
  public ActionForward changePassword(...) {...}
  public ActionForward deleteAccount(...) {...}

  private boolean isComplexBusinessLogicSuccessful
                                          (String type) {
    return(Math.random() > 0.5);
  }
}
```


Step 5 (Create Form that Invokes *blah*.do)

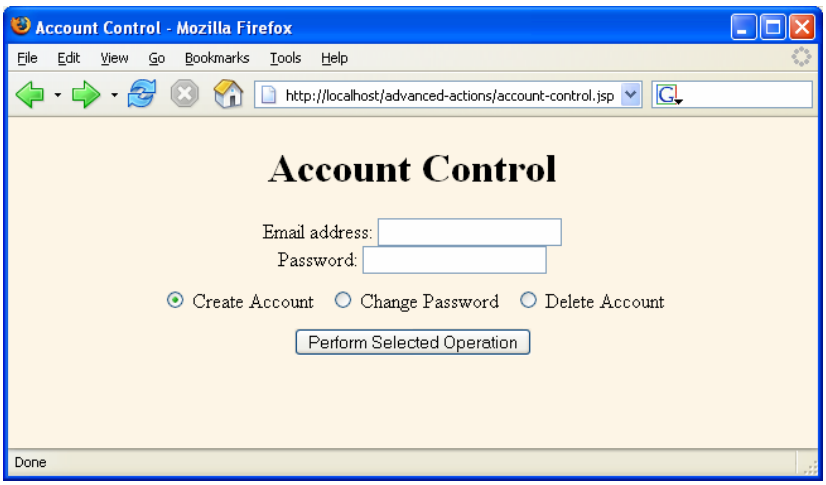
```
<%@ taglib uri="http://struts.apache.org/tags-html"
           prefix="html" %>
<html:form action="/accountMod">
  Email address: <html:text property="email"/><BR>
  Password: <html:password property="password"/><BR>
  <TABLE CELLSPACING="10">
    <TR>
      <TD><html:radio property="operation"
                      value="createAccount"/>
          Create Account</TD>
      <TD><html:radio property="operation"
                      value="changePassword"/>
          Change Password</TD>
      <TD><html:radio property="operation"
                      value="deleteAccount"/>
          Delete Account</TD>
    </TR>
  </TABLE>
  <html:submit value="Perform Selected Operation"/>
</html:form>
```

Step 6 (Display Results in JSP)

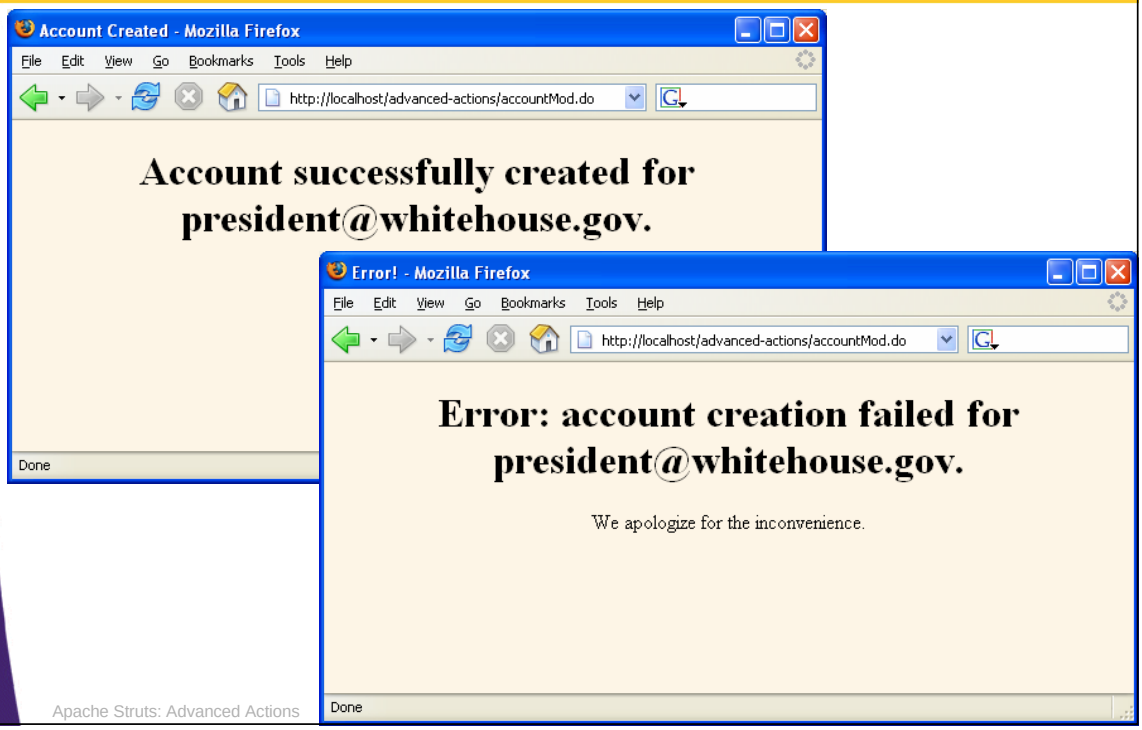
- **WEB-INF/results/create-confirm.jsp**

```
<%@ taglib uri="http://struts.apache.org/tags-bean"
           prefix="bean" %>
<!DOCTYPE ...>
<HTML>
<HEAD><TITLE>Account Created</TITLE></HEAD>
<BODY BGCOLOR="#FDF5E6">
<CENTER>
<H1>Account successfully created for
<bean:write name="userFormBean" property="email"/>.</H1>
Congratulations.
</CENTER>
</BODY></HTML>
```

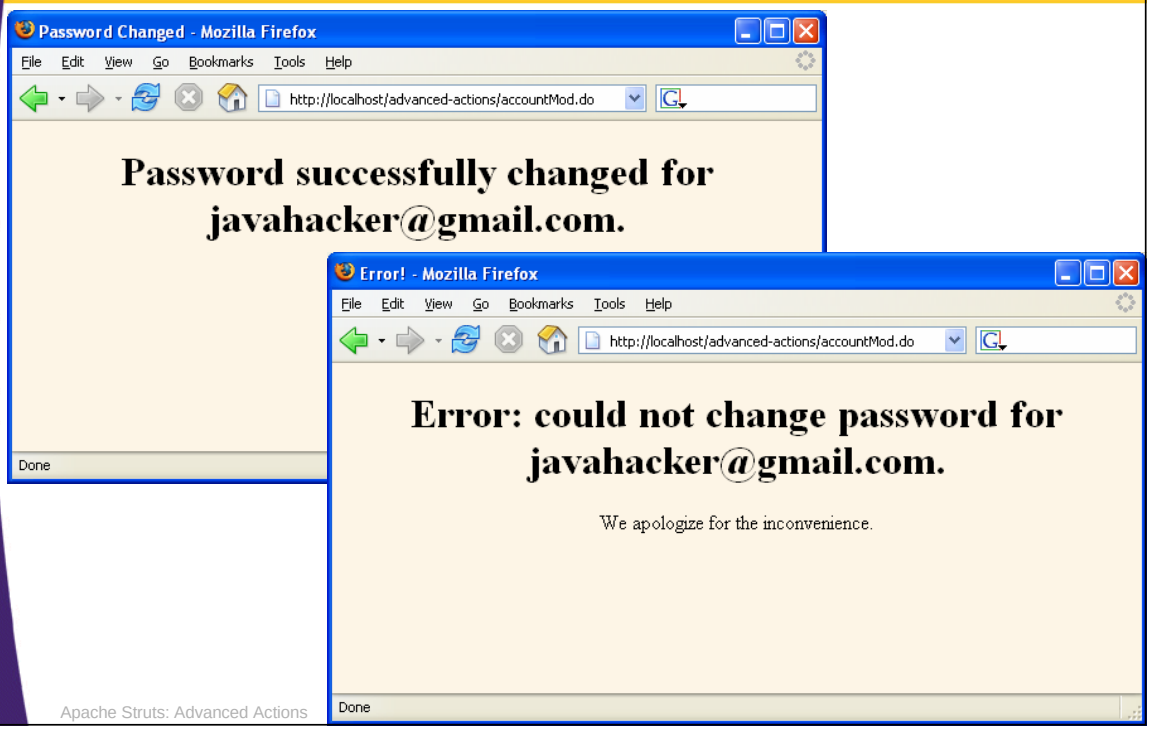

Results (Initial Form)



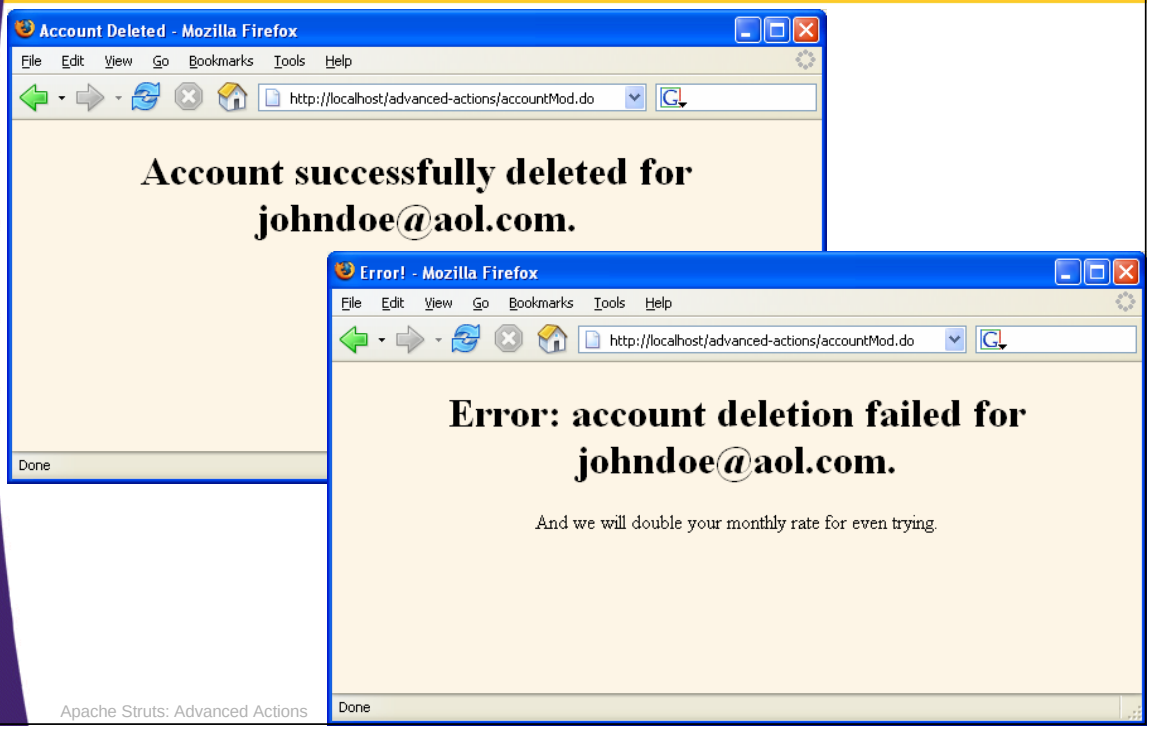
Results (Account Creation)



Results (Password Change)



Results (Account Deletion)



More Details on DispatchAction

- **Using DispatchAction for different forms**
  - Motivation: different forms with similar actions
    - Have the different forms use hidden fields whose name matches the specified parameter and whose value designates the method
- **Using push buttons instead of radio buttons**
  - Conceptually, it seems the same. Problem: push button values are displayed to the user as the button labels
    - You don't want the labels to have to match the method names
  - Solution: override getMethodName.
    - This method maps parameter values to method names
    - But fails in I18N apps where button labels come from properties file, so LookupDispatchAction needed in this complex case
- **Handling unmatched values**
  - E.g., users might submit without selecting a radio button
    - This automatically invokes a method named **`unspecified`**

ForwardAction

Using Form Beans with Non-Struts MVC Apps

ForwardAction

- **Scenario**
  - You already have MVC-based servlet/JSP combo that uses beans
    - Repeated calls to request.getParameter in servlet
  - You already have standalone JSP page that uses beans
    - Scripting elements or jsp:setProperty
- **Problems**
  - Tedious and repetitive reading of request parameters and setting of bean properties
  - Not yet ready to redo code in Struts
- **Goal**
  - Use Struts form beans facility, but keep basic servlet/JSP structure in place

Steps in Using ForwardAction

- **Change bean**
  - Need to extend ActionForm
  - Do not need to implement Serializable
  - Do not need to check for null and empty strings
- **Change servlet**
  - Do not need to create bean or check Request/Session/Application for null
  - Do not need to call request.getParameter
- **Change form**
  - ACTION should refer to *blah*.do

Steps in Using ForwardAction (Continued)

- **Declare bean in struts-config.xml**
  - Same as with a normal form bean
- **Declare action in struts-config.xml**

```
<action path="/blah"
        type="org.apache.struts.actions.ForwardAction"
        name="bean-name"
        scope="request-session-or-application"
        parameter="/path-to-servlet"/>
```

Example: Preferred Colors (Session Scoped)

- **User can specify preferred foreground and background colors**
  - If no color specified, previous choice used
  - If no colors given previously, defaults are used
- **Uses MVC**
  - Servlet uses RequestDispatcher
  - Result page uses JSP 2.0 expression language

Non-Struts Version: Bean

```
package coreservlets;

import java.io.Serializable;

public class ColorBean1 implements Serializable {
  private String foregroundColor = "BLACK";
  private String backgroundColor = "WHITE";

  public String getForegroundColor() {
    return(foregroundColor);
  }

  public void setForegroundColor(String fgColor) {
    if (!isEmpty(fgColor)) {
      foregroundColor = fgColor;
    }
  }

 ...

  private boolean isEmpty(String value) {
    return((value == null) || (value.trim().equals("")));
  }
}
```

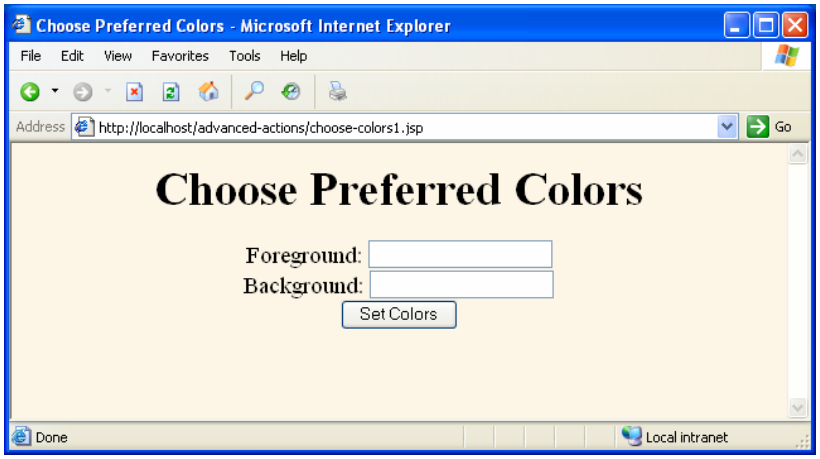
Non-Struts Version: Servlet

```
public class ShowColors1 extends HttpServlet {
  public void doGet(HttpServletRequest request,
                    HttpServletResponse response)
      throws ServletException, IOException {
    HttpSession session = request.getSession();
    ColorBean1 colorBean =
      (ColorBean1)session.getAttribute("colorBean1");
    if (colorBean == null) {
      colorBean = new ColorBean1();
      session.setAttribute("colorBean1", colorBean);
    }
    colorBean.setForegroundColor
      (request.getParameter("foregroundColor"));
    colorBean.setBackgroundColor
      (request.getParameter("backgroundColor"));
    if (colorBean.getForegroundColor().equals
        (colorBean.getBackgroundColor())) {
      colorBean = new ColorBean1();
      session.setAttribute("colorBean1", colorBean);
    }
    String address = "/WEB-INF/results/show-colors1.jsp";
    RequestDispatcher dispatcher =
      request.getRequestDispatcher(address);
    dispatcher.forward(request, response);
  }
```


Non-Struts Version: Form

```
<!DOCTYPE ...>
<HTML>
<HEAD><TITLE>Choose Preferred Colors</TITLE></HEAD>
<BODY BGCOLOR="#FDF5E6">
<CENTER>
<H1>Choose Preferred Colors</H1>
<FORM ACTION="ShowColors1">
  Foreground: <INPUT TYPE="TEXT" NAME="foregroundColor"><BR>
  Background: <INPUT TYPE="TEXT" NAME="backgroundColor"><BR>
  <INPUT TYPE="SUBMIT" VALUE="Set Colors">
</FORM>
</CENTER>
</BODY></HTML>
```

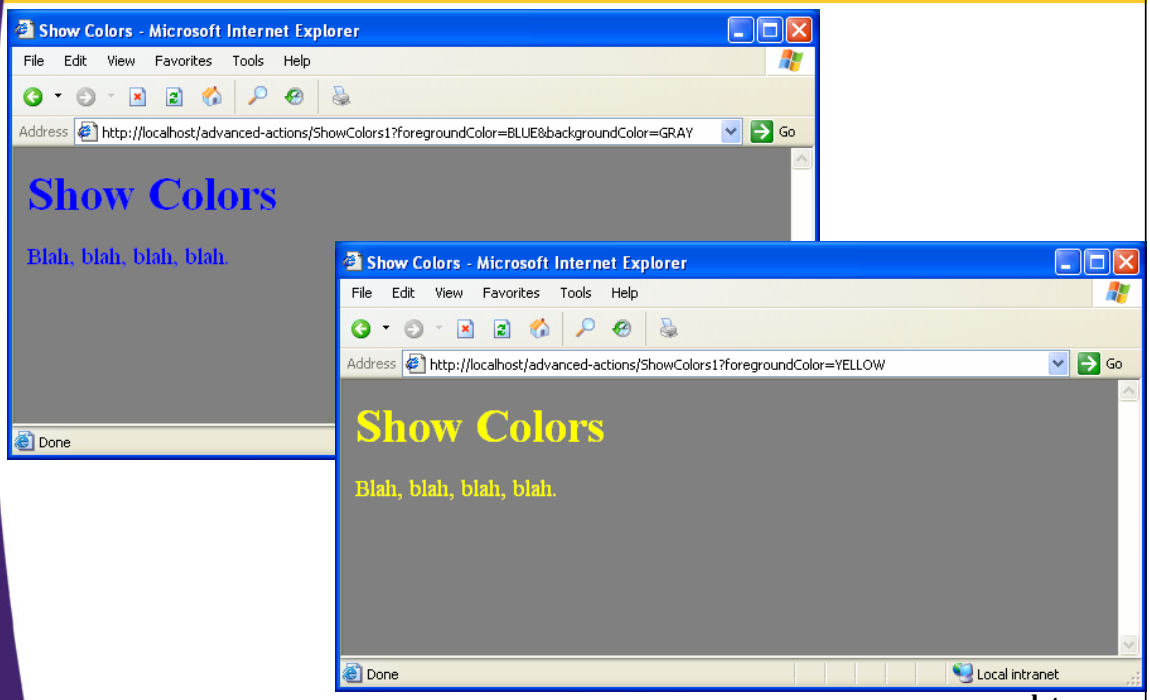
Non-Struts Version: Form



Non-Struts Version: Results Page

```
<!DOCTYPE ...>
<HTML>
<HEAD><TITLE>Show Colors</TITLE></HEAD>
<BODY BGCOLOR="${colorBean1.backgroundColor}"
      TEXT="${colorBean1.foregroundColor}">
<H1>Show Colors</H1>
Blah, blah, blah, blah.
</BODY></HTML>
```

Non-Struts Version: Results



ForwardAction Version: Bean

```
package coreservlets;

import org.apache.struts.action.*;

public class ColorBean2 extends ActionForm {
  private String foregroundColor = "BLACK";
  private String backgroundColor = "WHITE";

  public String getForegroundColor() {
    return(foregroundColor);
  }

  public void setForegroundColor(String fgColor) {
    foregroundColor = fgColor;
  }

  ...

}
```

ForwardAction Version: Servlet

```
public class ShowColors2 extends HttpServlet {
  public void doGet(HttpServletRequest request,
                    HttpServletResponse response)
      throws ServletException, IOException {
    HttpSession session = request.getSession();
    ColorBean2 colorBean =
      (ColorBean2)session.getAttribute("colorBean2");
    if (colorBean.getForegroundColor().equals
        (colorBean.getBackgroundColor())) {
      colorBean = new ColorBean2();
      session.setAttribute("colorBean2", colorBean);
    }
    String address = "/WEB-INF/results/show-colors2.jsp";
    RequestDispatcher dispatcher =
      request.getRequestDispatcher(address);
    dispatcher.forward(request, response);
  }
}
```

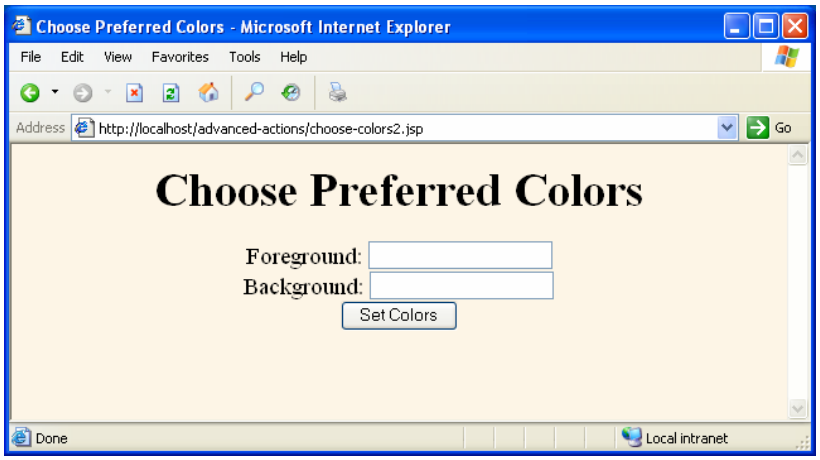

ForwardAction Version: struts-config.xml

```
<struts-config>
  <form-beans>
    ...
    <form-bean name="colorBean2"
               type="coreservlets.ColorBean2"/>
  </form-beans>
  <action-mappings>
    ...
    <action path="/showColors2"
            type="org.apache.struts.actions.ForwardAction"
            name="colorBean2"
            scope="session"
            parameter="/ShowColors2"/>
  </action-mappings>
</struts-config>
```

ForwardAction Version: Form

```
<!DOCTYPE ...>
<HTML>
<HEAD><TITLE>Choose Preferred Colors</TITLE></HEAD>
<BODY BGCOLOR="#FDF5E6">
<CENTER>
<H1>Choose Preferred Colors</H1>
<FORM ACTION="showColors2.do">
  Foreground: <INPUT TYPE="TEXT" NAME="foregroundColor"><BR>
  Background: <INPUT TYPE="TEXT" NAME="backgroundColor"><BR>
  <INPUT TYPE="SUBMIT" VALUE="Set Colors">
</FORM>
</CENTER>
</BODY></HTML>
```

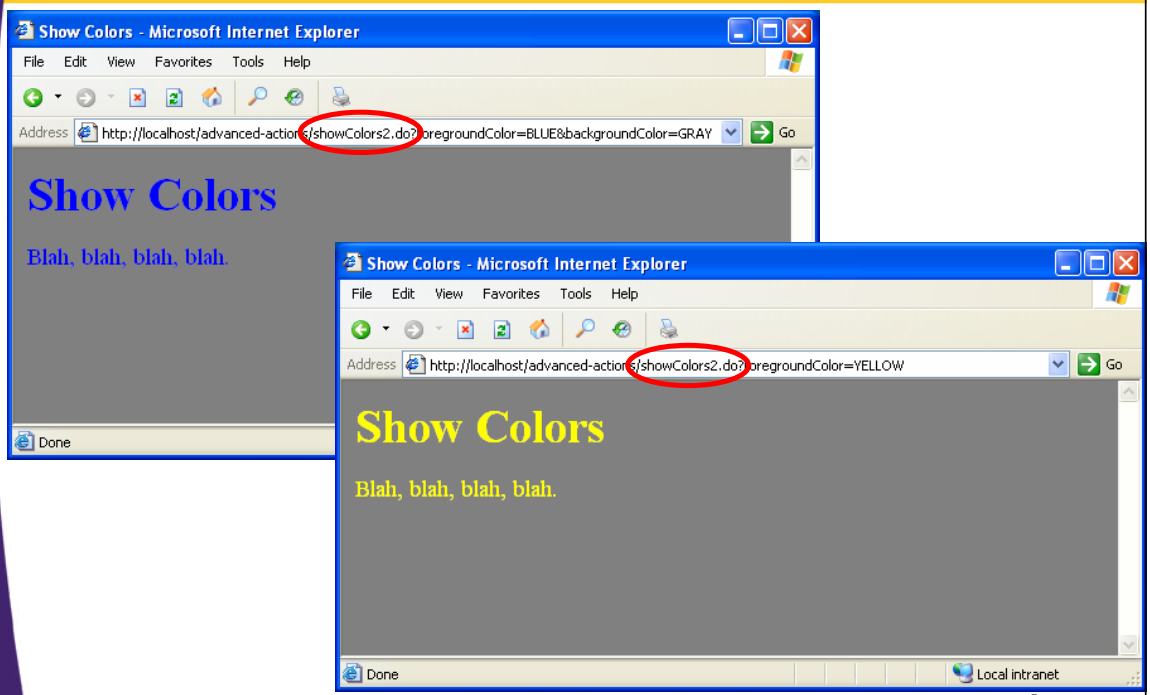

ForwardAction: Form



ForwardAction Version: Results Page

```
<!DOCTYPE ...>
<HTML>
<HEAD><TITLE>Show Colors</TITLE></HEAD>
<BODY BGCOLOR="${colorBean2.backgroundColor}"
      TEXT="${colorBean2.foregroundColor}">
<H1>Show Colors</H1>
Blah, blah, blah, blah.
</BODY></HTML>
```


ForwardAction: Results



Other Specialized Action Subclasses

- **DefinitionDispatchAction**
  - Associates a URL with a Tiles definition name
- **DownloadAction**
  - Used for actions that handle forms with file uploads
- **IncludeAction**
  - Similar to ForwardAction, but includes instead of forwards
    - Populates form beans first
    - *Useful for normal MVC-style servlet that wants to use the Struts form-bean capability*
- **LocaleAction**
  - Uses request params to decide which Locale to use
- **SwitchAction**
  - Invokes resource from a specific Struts Module
- **TilesAction**
  - Accepts one extra parameter: the tiles context

Summary

- **DispatchAction**
  - Use struts-config.xml to list the parameter used to determine which method will be called
    - <action path="..." type="..." parameter="operation">
  - Extend DispatchAction instead of Action
    - Directly implement desired methods
  - In form, supply parameter with given name
  - Form bean needs no accessors for special parameter
- **ForwardAction**
  - Use struts-config.xml to declare the bean and give address of servlet that will use it
    - <action path="/*blah*"
      type="org.apache.struts.actions.ForwardAction"
      name="*bean-name*"
      scope="*request-session-or-application*"
      parameter="/*path-to-servlet*"/>

Questions?

Core **Servlets & JSP book: www.coreservlets.com**
More **Servlets & JSP book: www.moreservlets.com**
Servlet, JSP, Struts, JSF, and Java Training Courses: courses.coreservlets.com