CSE 510
Web Data Engineering

The Struts Framework

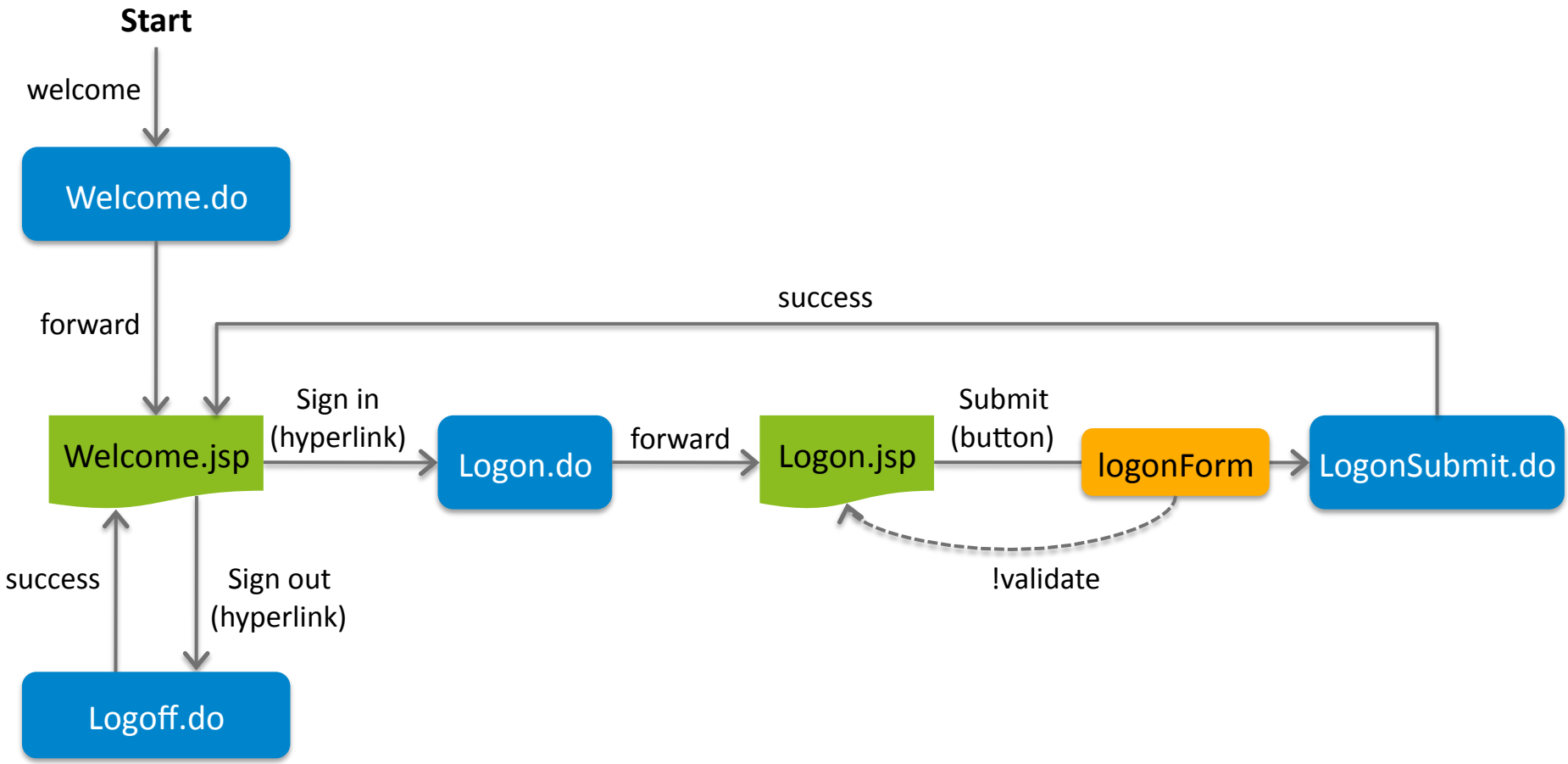
Logon Example

cse@buffalo

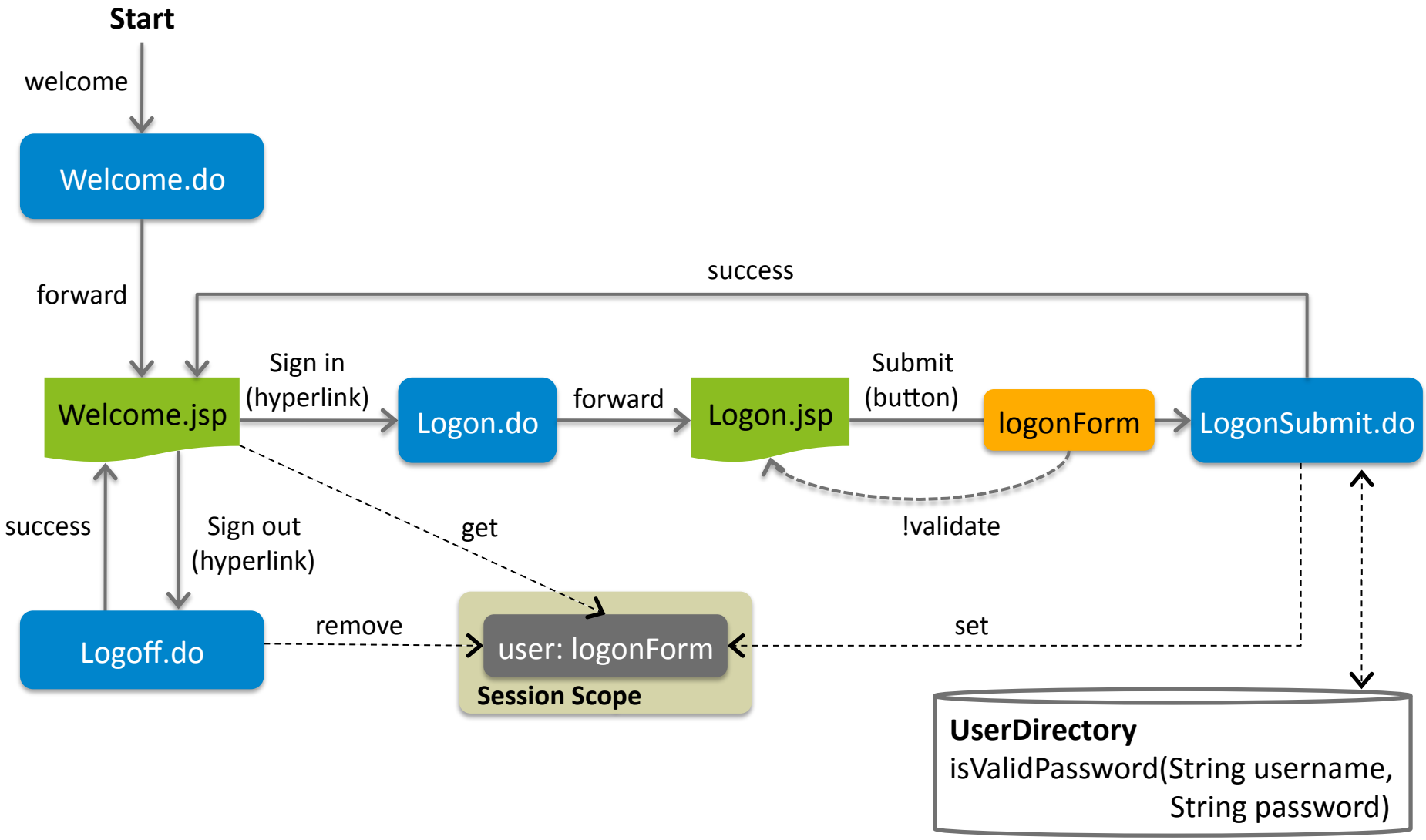
Example

- The example implements a dummy logon functionality
  - Do **not** consider this example to be the best way to implement authorization and access control
  - The example is used only to demonstrate the Struts framework
  - Appropriate authorization and access control will be covered in a separate lecture shortly

WorkFlow


Start
welcome
Welcome.do
forward
success
Welcome.jsp
Sign in
(hyperlink)
Logon.do
forward
Logon.jsp
Submit
(button)
logonForm
LogonSubmit.do
!validate
success
Sign out
(hyperlink)
Logoff.do

WorkFlow


Start
welcome
Welcome.do
forward
success
Welcome.jsp
Sign in
(hyperlink)
Logon.do
forward
Logon.jsp
Submit
(button)
logonForm
LogonSubmit.do
success
Sign out
(hyperlink)
get
!validate
Logoff.do
remove
user: logonForm
Session Scope
set
UserDirectory
isValidPassword(String username,
String password)

/index.jsp

```
<%@ taglib uri="http://struts.apache.org/tags-logic"
        prefix="logic" %>
<logic:redirect forward="welcome"/>

<%--

Redirect default requests to Welcome global ActionForward.

--%>
```

/pages/Welcome.jsp

```
<%@ taglib uri="http://struts.apache.org/tags-bean" prefix="bean"%>
<%@ taglib uri="http://struts.apache.org/tags-html" prefix="html"%>
<%@ taglib uri="http://struts.apache.org/tags-logic" prefix="logic"%>
<html:html>
<head>
   <title><bean:message key="app.title" /></title>
   <html:base />
</head>
<body>

<logic:present name="user">
   <h3>Welcome <bean:write name="user" property="username" />!</h3>
</logic:present>
…
```

/pages/Welcome.jsp (cont'd)

```
...
<logic:notPresent scope="session" name="user">
   <h3><bean:message key="welcome.heading" /></h3>
</logic:notPresent>

<html:errors />

<ul>
   <li><html:link forward="logon">Sign in</html:link></li>
   <logic:present name="user">
      <li><html:link forward="logoff">Sign out</html:link></li>
   </logic:present>
</ul>
</body>
</html:html>
```

Edge Labels In struts-config.xml

```
<!-- ==========Global Forward Definitions →
   <global-forwards>
      <forward
         name="logoff"
         path="/Logoff.do"/>
      <forward
         name="logon"
         path="/Logon.do"/>
      <forward
         name="welcome"
         path="/Welcome.do"/>
   </global-forwards>
```

/pages/Logon.jsp

```
…
<body>
   <html:errors />
   <h3><bean:message key=”logon.heading" /></h3>
   <html:form action="/LogonSubmit" focus="username">
      <table width="100%">
         <tr><th align="right">Username:</th>
               <td><html:text property="username" /></td></tr>
         <tr><th align="right">Password:</th>
              <td><html:password property="password" /></td></tr>
         <tr><td align="right"><html:submit /></td>
              <td><html:reset /></td></tr>
      </table>
   </html:form>
</body>
…
```

Associating the ActionForm Bean with the HTML Form

```
<action
   path="/LogonSubmit”
   type="app.LogonAction"
   name="logonForm"
   scope="request"
   validate="true"
   input="/pages/Logon.jsp">
   <forward
      name="success"
      path="/pages/Welcome.jsp"/>
</action>
```

Form Beans Also Provide Values

/pages/Logon.jsp

```
…
  <h3><bean:message key=”logon.heading" /></h3>
  <html:form action="/LogonSubmit" focus="username">
    <table width="100%">
      <tr><th align="right">Username:</th>
          <td><html:text property="username" /></td></tr>
      <tr><th align="right">Password:</th>
          <td><html:password property="password" /></td></tr>
      <tr><td align="right"><html:submit /></td>
          <td><html:reset /></td></tr>
    </table>
  </html:form>
…
```

Typical Code Of A LogonForm Bean

```
public final class LogonForm extends ActionForm {
   private String password = null;
   private String username = null;

   public String getPassword() { return (this.password); }
   public void setPassword(String password) { this.password = password; }

   public String getUsername() { return (this.username); }
   public void setUsername(String username) { this.username = username; }

   public void reset(ActionMapping mapping, HttpServletRequest request) {
      setPassword(null);
      setUsername(null);
   }
}
```

The Art of Balancing How Many Actions & JSPs to Use

- Consider the "logon" application
- We could have one JSP for each kind of login error
- However, we will see technologies that help consolidate within a few JSPs
  - Form validation features
  - Logic tag library
- Deciding the number of actions and JSPs is an art of design – not a science
  - Examples, practice, then more practice…

Validation

```
public ActionErrors validate(ActionMapping mapping,
                             HttpServletRequest request) {

   ActionErrors errors = new ActionErrors();

   if ((username == null) || (username.length() < 1))
      errors.add("username", new ActionMessage("error.username.required"));

   if ((password == null) || (password.length() < 1))
      errors.add("password", new ActionMessage("error.password.required"));

   return errors;
}
```

Resource File & Internationalization

MessageResources.properties

app.title=Struts Logon Application

welcome.heading=Welcome User!

logon.heading=Sign in, Please!

errors.header=<h3>Validation Error</h3>You must…

errors.prefix=

errors.suffix=

errors.footer=
<hr>

error.username.required=Username is required

error.password.required=Password is required

error.logon.invalid=Username and password provided not found in user…

error.logon.connect=Could not connect to user directory.

…

Action Bean LogonAction.java

```
package app;

import java.io.IOException;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
import org.apache.struts.action.Action;
import org.apache.struts.action.ActionErrors;
import org.apache.struts.action.ActionForm;
import org.apache.struts.action.ActionForward;
import org.apache.struts.action.ActionMapping;
import org.apache.struts.action.ActionMessage;
import org.apache.struts.action.ActionMessages;

public final class LogonAction extends Action {
…
```

Action Bean LogonAction.java

```
…
/**
 * Validate credentials with business tier.
 *
 * @param username The username credential
 * @param password The password credential
 * @returns true if credentials can be validated
 * @exception UserDirectoryException if cannot access directory
 */
public boolean isUserLogon(String username, String password)
            throws UserDirectoryException {

    return (UserDirectory.getInstance().isValidPassword(username, password));
    // return true;
}
…
```

Action Bean LogonAction.java

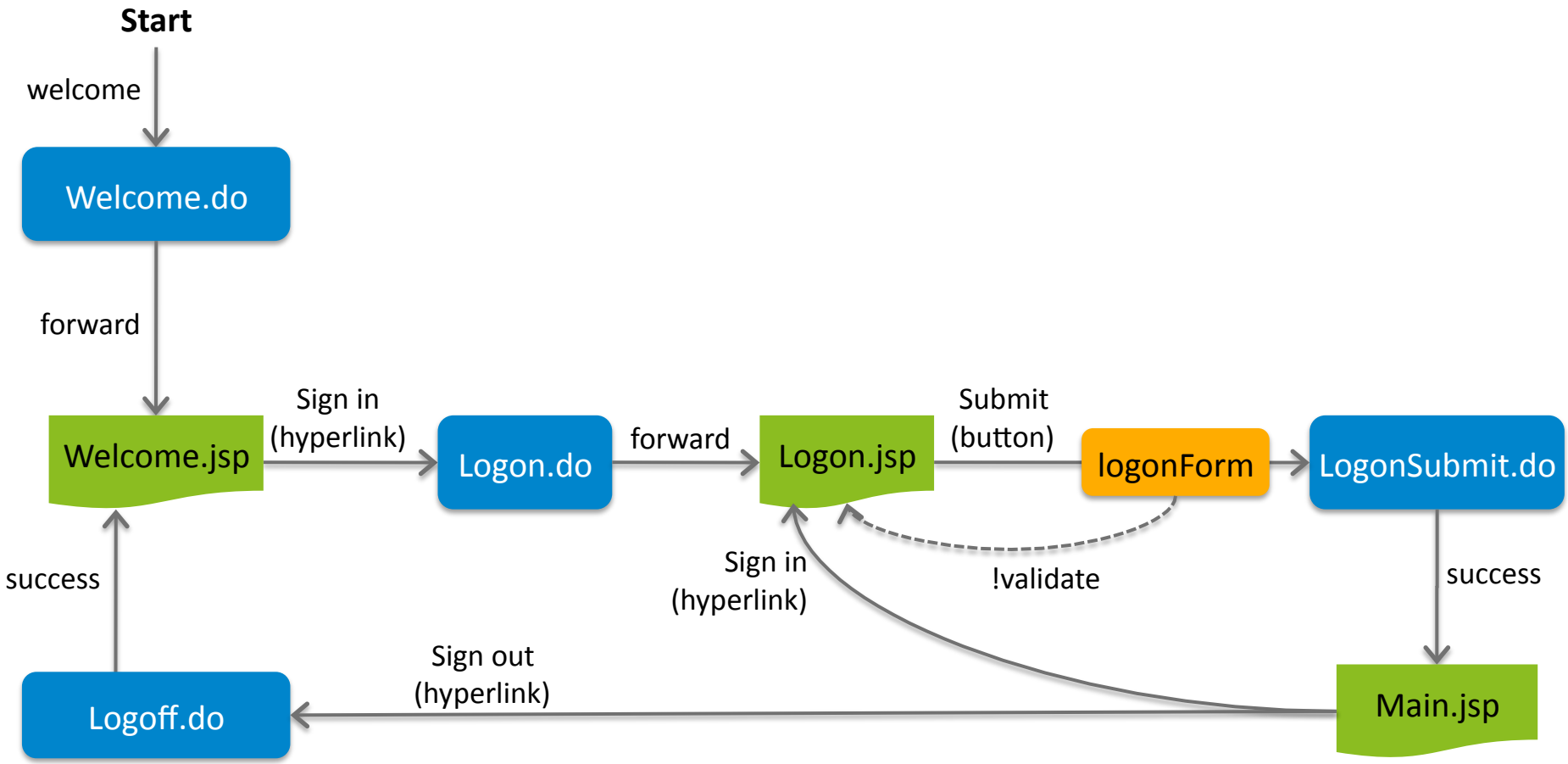
```
…
public ActionForward execute(ActionMapping mapping, ActionForm form,
    HttpServletRequest request, HttpServletResponse response) throws Exception {
    // Obtain username and password from web tier
    String username = ((LogonForm) form).getUsername();
    String password = ((LogonForm) form).getPassword();
    // Validate credentials with business tier
    boolean validated = false;
    try {
            validated = isUserLogon(username, password);
    } catch (UserDirectoryException ude) {
            // couldn't connect to user directory
            ActionErrors errors = new ActionErrors();
            errors.add(…, new ActionMessage("error.logon.connect"));
            saveErrors(request, errors);
            // return to input page
            return (new ActionForward(mapping.getInput()));
    }
    …
```

Action Bean LogonAction.java

```
        …
        if (!validated) {
                // credentials don't match
                ActionErrors errors = new ActionErrors();
                errors.add(…, new ActionMessage("error.logon.invalid"));
                saveErrors(request, errors);
                // return to input page
                return (new ActionForward(mapping.getInput()));
        }
        // Save our logged-in user in the session,
        // because we use it again later.
        HttpSession session = request.getSession();
        session.setAttribute(Constants.USER_KEY, form);

        // Return success
        return mapping.findForward(Constants.SUCCESS);
}
} // End LogonAction
```

WorkFlow Variance



WorkFlow Variance

- The two variances share the same actions
- No modification in Actions source files
- Only need to change “view” (JSPs) and “workflow” (struts-config.xml)