Jakarta Struts:
Processing Requests
with Action Objects
Struts 1.*x* Version

For live Struts training, please see 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.

- Courses developed and taught by Marty Hall
  - Java 6, servlets/JSP (intermediate and advanced), Struts, JSF 1.*x*, JSF 2.0, Ajax, GWT 2.0 (with GXT), custom mix of topics
  - Ajax courses can concentrate on 1 library (jQuery, Prototype/Scriptaculous, Ext-JS, Dojo, Google Closure) or survey several
- Courses developed and taught by coreservlets.com experts (edited by Marty)
  - Spring, Hibernate/JPA, EJB3, SOAP-based and RESTful Web Services, Ruby/Rails

Contact hall@coreservlets.com for details

Agenda

- **Struts flow of control**
- **The six basic steps in using Struts**
  - To implement the flow of control
- **Example: one result mapping**
  - Same output page in all cases
- **Example: multiple result mappings**
  - Different output pages depending on the input

Struts Flow of Control
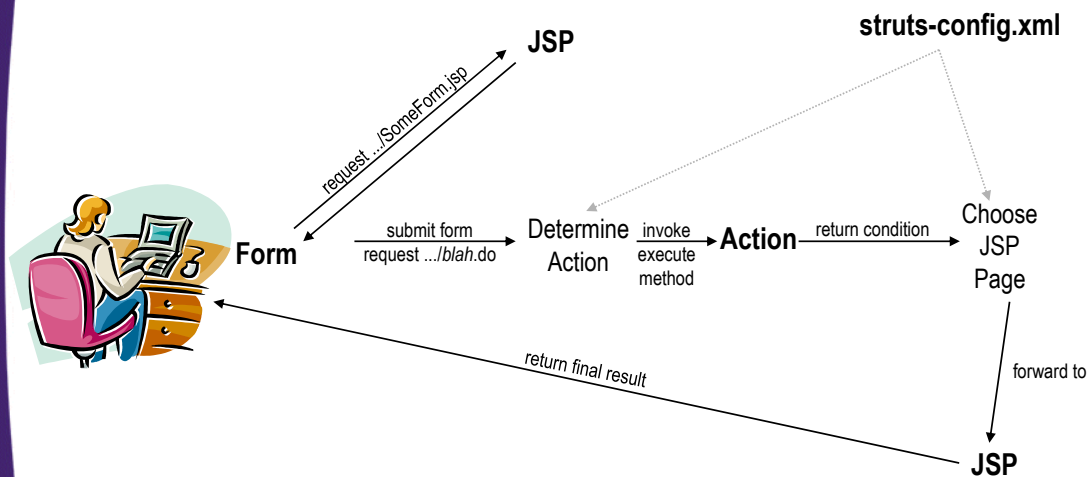
Struts Flow of Control

- **The user requests a form**
  - For now, we use normal HTML to build the form
    - Later we will use the Struts html:form tag
- **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 execute method of the Action object is invoked**
  - One of the arguments to execute is a form bean that is automatically created and whose properties are automatically populated with the incoming form data
  - 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

The Six Basic Steps in Using Struts

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.
   - *Be sure to restart the server after modifying struts-config.xml*; the file is read only when the Web application is first loaded.
2. **Define a form bean.**
   - This bean is a class the extends ActionForm and will represent the data submitted by the user. It is automatically populated when the input form is submitted. Beans are postponed until the next section.

The Six Basic Steps in Using Struts

3. **Create results beans.**
   - In the MVC architecture, the business-logic and data-access code create the results and the JSP pages present them. To transfer the results from one layer to the other, they are stored in beans. These beans differ from form beans in that they need extend no particular class, and they represent the output of the computational process, not the input to the process. Beans will be discussed in the next section.

4. **Define an Action class to handle requests.**
   - The struts-config.xml file designates the Action classes that handle requests for various URLs. The Action objects themselves need to do the real work: invoke the appropriate business- and data-access-logic, store the results in beans, and designate the type of situation (missing data, database error, success category 1, success category 2, etc.) that is appropriate for the results. The struts-config.xml file then decides which JSP page should apply to that situation.

The Six Basic Steps in Using Struts

5. **Create form that invokes *blah*.do.**
   - Create an input form whose ACTION corresponds to one of the .do addresses listed in struts-config.xml.
   - In a later lecture, we will discuss the advantages of using the Struts html:form tag to build this input form.

6. **Display results in JSP.**
   - Since Struts is built around MVC, these JSP pages should avoid JSP scripting elements whenever possible. For basic Struts, these pages usually use the bean:write tag, but in JSP 2.0 the JSP 2.0 expression language is a viable alternative.
   - In most cases, the JSP pages only make sense when the request is funneled through the Action, so the pages go in WEB-INF.
   - If the JSP pages makes sense independently of the Action (e.g., if they display session data), then the JSP pages should be placed in a regular subdirectory of the Web application, and the forward entries in struts-config.xml should say
   <forward ... redirect="true"/>.

Example 1: One Result Mapping

- **URL**
  - http://hostname/struts-actions/register1.do
- **Action Class**
  - RegisterAction1
    - RegisterAction1 extends Action and is in the coreservlets package.
    - The execute method of RegisterAction1 always returns "success"
- **Results page**
  - /WEB-INF/results/confirm.jsp
    - But the URL shown will still be register1.do

Step 1A (Modify struts-config.xml)

- **Map incoming .do URLs to Action classes**
  - In this case, we designate that RegisterAction1 should handle requests for register1.do. To accomplish this, we add an action entry to action-mappings, where action has the following attributes.
    - **path**: the relative path that should be mapped to the Action, minus the .do extension. Thus, path="/register1" refers to http://hostname/webAppName/register1.do.
    - **type**: the fully qualified class name of the Action class that should be invoked when a request for the path is received.

```
<action-mappings>
        <!-- .do implied automatically -->
  <action path="/register1"
        type="coreservlets.RegisterAction1">
    ...
  </action>
</action-mappings>
```

- **Map return conditions to JSP pages**
  - In this case, we use the forward element to say that confirm.jsp applies when the execute method of RegisterAction1 returns "success", as follows:

```
<forward name="success"
         path="/WEB-INF/results/confirm.jsp"/>
```

  - Note: if the same forward is used by multiple actions, you can put the forward declaration in a global-forwards section (before action-mappings) instead of in the action.

```
<global-forwards>
  <forward name="success"
          path="/WEB-INF/results/confirm.jsp"/>
</global-forwards>
```

13

Step 1 (Modify struts-config.xml) – Final struts-config.xml

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE struts-config PUBLIC ... >
<struts-config>
  <action-mappings>
    <action path="/register1"
            type="coreservlets.RegisterAction1">
      <forward name="success"
              path="/WEB-INF/results/confirm.jsp"/>
    </action>
  </action-mappings>
</struts-config>
```

14

Steps 2 and 3

- **Define a form bean.**
  - Beans are postponed until the next section, so this step is omitted for now.
- **Create results beans.**
  - Beans are postponed until the next section, so this step is omitted for now.

Step 4 (Define an Action Class to Handle Requests)

- **Action subclasses should…**
  - Be in a package.
  - In this case, we have
    ```
    package coreservlets;
    ```
    - This means that the source file should go in src/coreservlets
- **Action subclasses should…**
  - Add Struts-specific import statements to whatever imports are otherwise needed.
  - In this case, we have
    ```
    import javax.servlet.http.*;
    import org.apache.struts.action.*;
    ```

Step 4 (Define an Action Class to Handle Requests)

- **Action subclasses should ...**
  – Extend Action
- **Action subclasses should ...**
  – Override execute
  – In this case, we have

```
public class RegisterAction1 extends Action {
  public ActionForward
        execute(ActionMapping mapping,
                ActionForm form,
                HttpServletRequest request,
                HttpServletResponse response)
     throws Exception {
   ...
  }
```

Step 4 (Define an Action Class to Handle Requests)

- **Action subclasses should ...**
  – Return mapping.findForward.
  – The execute method should have one or more return values.
  – These values will then be mapped to specific JSP pages by forward entries in struts-config.xml. In this case, we simply return "success" in all situations.

```
public ActionForward
      execute(ActionMapping mapping,
              ActionForm form,
              HttpServletRequest request,
              HttpServletResponse response)
   throws Exception {
  return(mapping.findForward("success"));
}
```

Step 4 (Define an Action Class to Handle Requests) – Final Code

```
package coreservlets;

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

public class RegisterAction1 extends Action {
  public ActionForward
            execute(ActionMapping mapping,
                    ActionForm form,
                    HttpServletRequest request,
                    HttpServletResponse response)
      throws Exception {
    return(mapping.findForward("success"));
  }
}
```

Step 5 (Create form that invokes *blah*.do)

- **We need an HTML form**
  - That invokes http://hostname/struts-actions/register1.do.

```
<!DOCTYPE ...>
<HTML>
<HEAD><TITLE>New Account Registration</TITLE></HEAD>
<BODY BGCOLOR="#FDF5E6">
<CENTER>
<H1>New Account Registration</H1>
<FORM ACTION="register1.do" METHOD="POST">
  Email address: <INPUT TYPE="TEXT" NAME="email"><BR>
  Password: <INPUT TYPE="PASSWORD" NAME="password"><BR>
  <INPUT TYPE="SUBMIT" VALUE="Sign Me Up!">
</FORM>
</CENTER>
</BODY></HTML>
```

Step 6 (Display results in JSP)

- **In general, there can be several possible JSP pages**
  - Corresponding to the various possible return values of the execute method of the Action.
- **In struts-config.xml, each JSP page is declared in a forward entry within the appropriate action.**
  - In this simple case, the only return value is "success", so /WEB-INF/results/confirm.jsp is used in all cases.
- **This JSP page will just display a simple message (see next slide)**

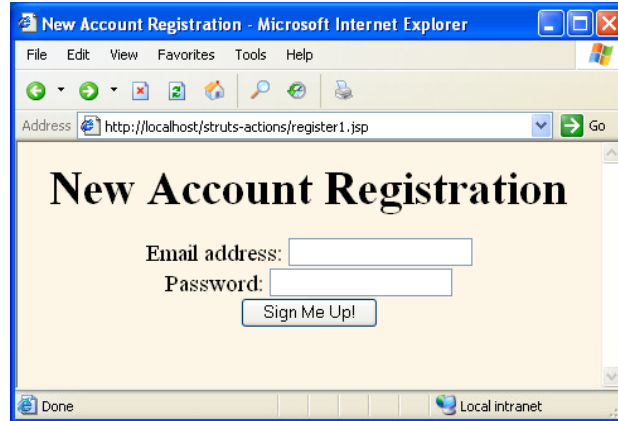# Step 6 (Display results in JSP) – Final Code

```
<!DOCTYPE ...>
<HTML>
<HEAD><TITLE>Success</TITLE></HEAD>
<BODY BGCOLOR="#FDF5E6">
<CENTER>
<H1>You have registered successfully.</H1>
Congratulations
</CENTER>
</BODY></HTML>
```

Example 1: Results

* **First, the HTML form is invoked with the URL http://localhost/struts-actions/register1.jsp**

Example 1: Results

* **This form is then filled in and submitted,**
    – With the form's ACTION resulting in the URL http://localhost/struts-actions/register1.do.
* **This address is mapped by struts-config.xml**
    – To the RegisterAction1 class, whose execute method is invoked.
* **This method returns mapping.findForward**
    – With a value of "success"
* **That value is mapped by struts-config.xml**
    – To /WEB-INF/results/confirm.jsp,
        * Which is the final result displayed to the user.
        * However, since the JSP page is invoked with RequestDispatcher.forward, not response.sendRedirect, the URL displayed to the user is register1.do, not confirm.jsp.
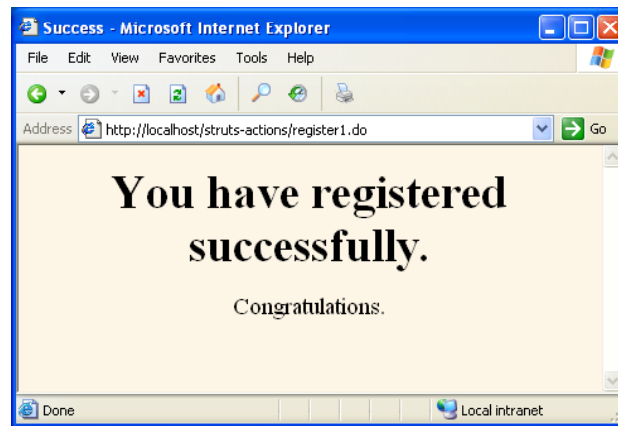
Example 1: Results

Example 2: Multiple Result Mappings

- **URL**
  – http://hostname/struts-actions/register2.do
- **Action Class**
  – RegisterAction2.
    - The execute method of RegisterAction2 returns "success", "bad-address", or "bad-password"
- **Results pages**
  – /WEB-INF/results/confirm.jsp,
  – /WEB-INF/results/bad-address.jsp, and
  – /WEB-INF/results/bad-password.jsp, respectively.
- **Main new feature of this example**
  – The use of multiple forward entries within the action element.

Step 1 (Modify struts-config.xml)

- **Map incoming .do address to Action classes**
  - In this case, we use the action element to designate that RegisterAction2 should handle requests for register2.do (again, note that .do is implied, not listed explicitly).
- **Map return conditions to JSP pages**
  - In this case, we use multiple forward elements, one for each possible return value of the execute method of the RegisterAction2 class.
- **Declare any form beans that are being used.**
  - Beans are postponed until the next section, so this step is omitted for now.

Step 1 (Modify struts-config.xml) – Final Code

```xml
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE struts-config PUBLIC ... >
<struts-config>
  <action-mappings>
    ...
    <action path="/register2"
            type="coreservlets.RegisterAction2">
      <forward name="bad-address"
               path="/WEB-INF/results/bad-address.jsp"/>
      <forward name="bad-password"
               path="/WEB-INF/results/bad-password.jsp"/>
      <forward name="success"
               path="/WEB-INF/results/confirm.jsp"/>
    </action>
  </action-mappings>
</struts-config>
```

Steps 2 and 3

- **Define a form bean.**
  - Beans are postponed until the next section, so this step is omitted for now.
- **Create results beans.**
  - Beans are postponed until the next section, so this step is omitted for now.

Step 4 (Define an Action Class to Handle Requests)

- **Similar to the previous example except for multiple mapping.findForward entries**
  - We return "bad-address" if the email address is missing, is less then three characters long, or does not contain an "@" sign.
  - We return "bad-password" if the password is missing or is less than six characters long.
  - Otherwise we return "success".
- **In this simple example we use request.getParameter explicitly.**
  - In later examples we let Struts automatically populate a bean from the request data.

Step 4 (Define an Action Class to Handle Requests) – Final Code

```
public class RegisterAction2 extends Action {
  public ActionForward
            execute(ActionMapping mapping,
                    ActionForm form,
                    HttpServletRequest request,
                    HttpServletResponse response)
      throws Exception {
    String email = request.getParameter("email");
    String password = request.getParameter("password");
    if ((email == null) ||
        (email.trim().length() < 3) ||
        (email.indexOf("@") == -1)) {
      return(mapping.findForward("bad-address"));
    } else if ((password == null) ||
               (password.trim().length() < 6)) {
      return(mapping.findForward("bad-password"));
    } else {
      return(mapping.findForward("success"));
}}}
```

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

- **We need an HTML form**
  – That invokes http://hostname/struts-actions/register2.do.

```
<!DOCTYPE ...>
<HTML>
<HEAD><TITLE>New Account Registration</TITLE></HEAD>
<BODY BGCOLOR="#FDF5E6">
<CENTER>
<H1>New Account Registration</H1>
<FORM ACTION="register2.do" METHOD="POST">
  Email address: <INPUT TYPE="TEXT"
  NAME="email"><BR>
  Password: <INPUT TYPE="PASSWORD"
  NAME="password"><BR>
  <INPUT TYPE="SUBMIT" VALUE="Sign Me Up!">
</FORM>
</CENTER>
</BODY></HTML>
```

Step 6 (Display results in JSP) First Possible Page

```
<!DOCTYPE ...>
<HTML>
<HEAD><TITLE>Illegal Email
  Address</TITLE></HEAD>
<BODY BGCOLOR="#FDF5E6">
<CENTER>
<H1>Illegal Email Address</H1>
Address must be of the form username@host.
Please <A HREF="register2.jsp">
try again</A>.
</CENTER>
</BODY></HTML>
```

33

Step 6 (Display results in JSP) Second Possible Page

```
<!DOCTYPE ...>
<HTML>
<HEAD><TITLE>Illegal Password</TITLE></HEAD>
<BODY BGCOLOR="#FDF5E6">
<CENTER>
<H1>Illegal Password</H1>
Password must contain at least six characters.
Please <A HREF="register2.jsp">
try again</A>.
</CENTER>
</BODY></HTML>
```

34

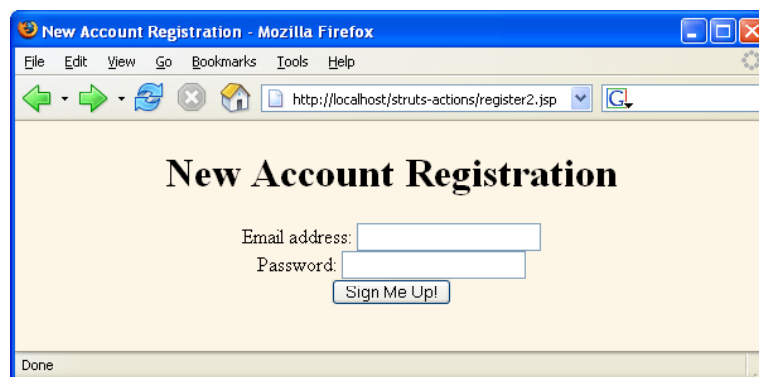# Step 6 (Display results in JSP) Same confirm.jsp Shown Earlier

```
<!DOCTYPE ...>
<HTML>
<HEAD><TITLE>Success</TITLE></HEAD>
<BODY BGCOLOR="#FDF5E6">
<CENTER>
<H1>You have registered successfully.</H1>
Congratulations
</CENTER>
</BODY></HTML>
```
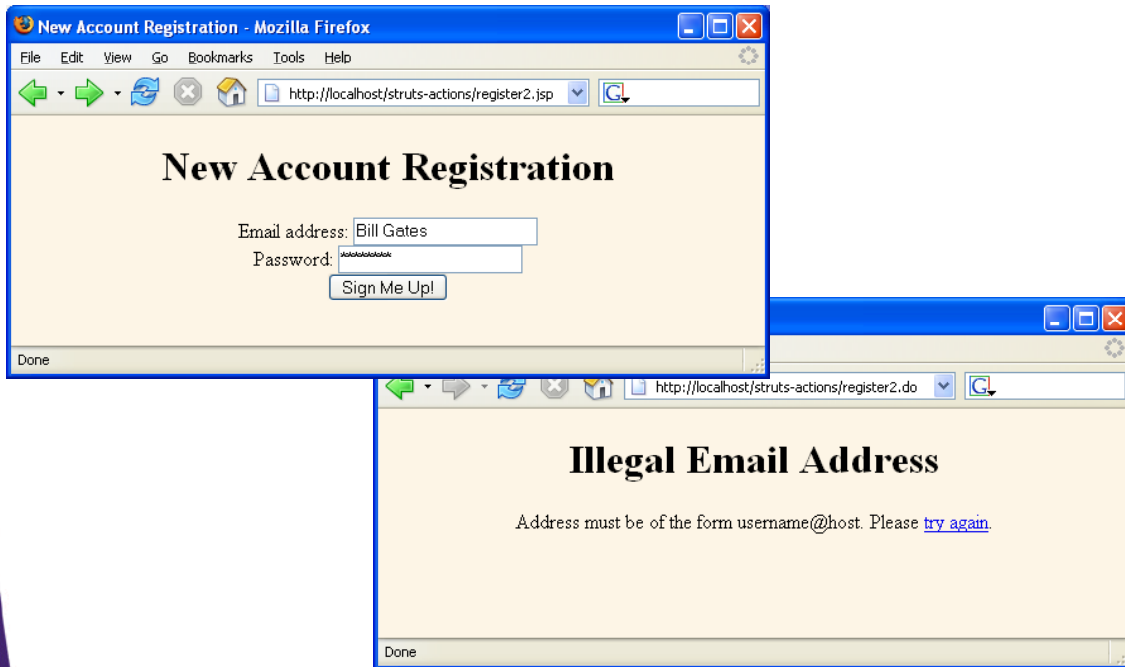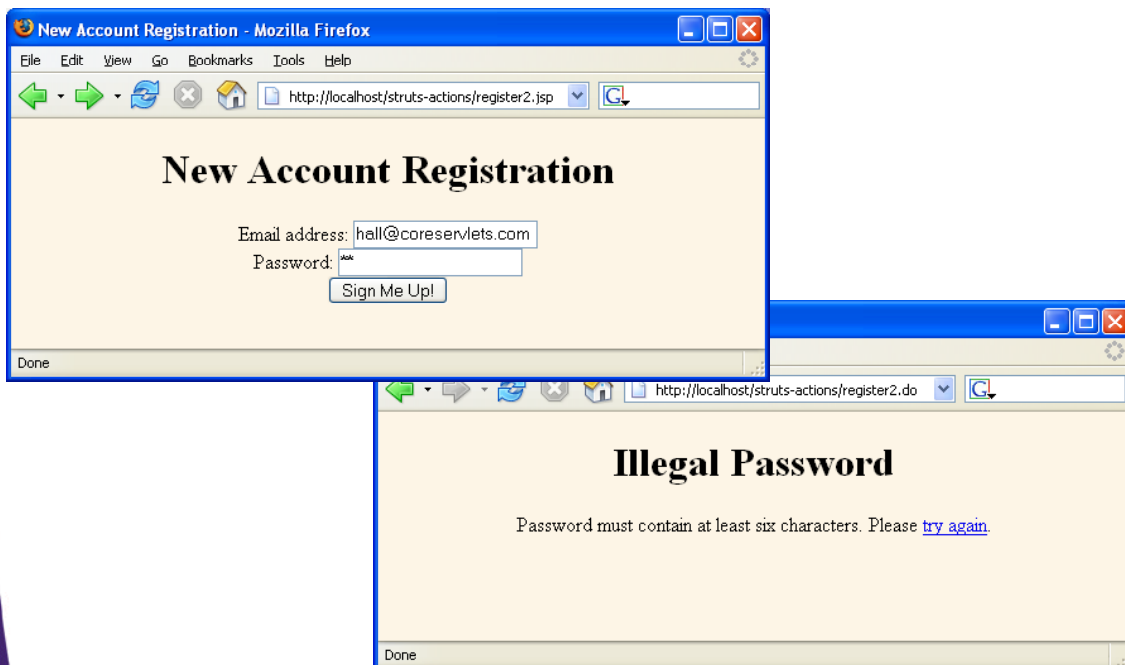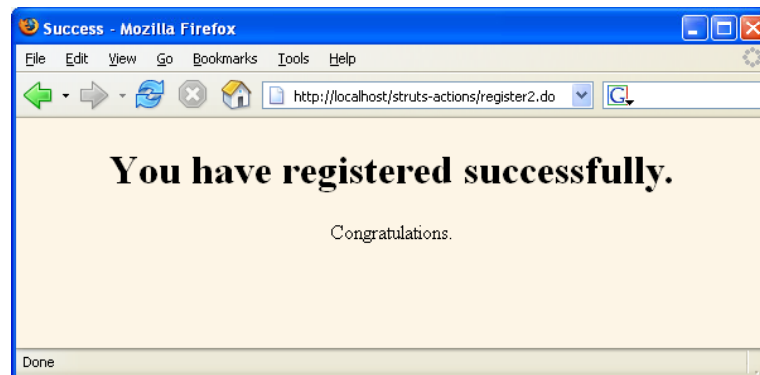
Example 2: Results (Initial Form)

Example 2: Results (Bad Address)

Example 2: Results (Bad Password)

Example 2: Results (Success)

Combining Shared Condition (Forward) Mappings

- **Idea**
  - If the same condition is mapped to the same JSP page in multiple actions, you can move the forward to a global-forwards section to avoid repetition
- **Syntax**
  - The global-forwards section goes before action-mappings, not within it
  - The forward entries within global-forwards have the same syntax and behavior as forward entries within action
- **Example**

```
<global-forwards>
  <forward name="success"
           path="/WEB-INF/results/confirm.jsp"/>
</global-forwards>
```

Combining Shared Condition (Forward) Mappings: Old

```
<action-mappings>
  <action path="/register1"
          type="coreservlets.RegisterAction1">
    <forward name="success"
            path="/WEB-INF/results/confirm.jsp"/>
  </action>
  <action path="/register2"
          type="coreservlets.RegisterAction2">
    <forward name="bad-address"
            path="/WEB-INF/results/bad-address.jsp"/>
    <forward name="bad-password"
            path="/WEB-INF/results/bad-password.jsp"/>
    <forward name="success"
            path="/WEB-INF/results/confirm.jsp"/>
  </action>
  ...
</action-mappings>
```

Combining Shared Condition (Forward) Mappings: New

```
<global-forwards>
  <forward name="success"
          path="/WEB-INF/results/confirm.jsp"/>
</global-forwards>
<action-mappings>
  <action path="/register1"
          type="coreservlets.RegisterAction1">
  </action>
  <action path="/register2"
          type="coreservlets.RegisterAction2">
    <forward name="bad-address"
            path="/WEB-INF/results/bad-address.jsp"/>
    <forward name="bad-password"
            path="/WEB-INF/results/bad-password.jsp"/>
  </action>
  ...
</action-mappings>
```

Summary

- **Modify struts-config.xml**
  - Map *blah*.do addresses to subclasses of Action
  - Map return conditions (from execute) to JSP pages
  - Declare any form beans that are being used.
- **Define a form bean**
- **Create results beans**
- **Define an Action class to handle requests**
  - Extend Action
  - Override execute
  - Return mapping.findForward
- **Create form that invokes *blah*.do**
- **Display results in JSP**

43

Questions?