



Using Struts to create a Web application

Contents

Use Struts to create a Web application 1

Introduction: Use Struts to create a Web application	1
Lesson 1: Create a Struts project	2
Lesson checkpoint	3
Lesson 2: Create Web diagram nodes	3
Lesson checkpoint	5
Lesson 3: Create a form bean	5
Lesson checkpoint	8
Lesson 4: Connect nodes in the Web diagram	8
Lesson checkpoint	10
Lesson 5: Edit a form bean	10
Lesson checkpoint	11

Lesson 6: Edit an action mapping	11
Lesson checkpoint	13
Lesson 7: Add elements to JavaServerPages (JSP) pages	13
Input page.	14
Output page	15
Lesson checkpoint	16
Lesson 8: Display error messages	16
Lesson checkpoint	17
Lesson 9: Test a Struts application	18
Lesson checkpoint	18
Summary: Use Struts to create a Web application.	19

Use Struts to create a Web application

In this tutorial, you will learn how to create a simple Struts application that contains two Web pages, an action mapping, and a form bean. You will use the Web diagram editor to help you visualize the logic flow of the application. After creating your application, you will learn how to set up a runtime test environment and test the application.

Learning objectives

You will learn how to do these tasks:

- Create a Struts project
- Edit your Web diagram
- Create and edit a form bean
- Create input and output JavaServer Pages (JSP) files
- Create an action and an action mapping
- Add ActionError tags to the input JSP page
- Test your Struts application

60 minutes

Related information

[View the PDF version](#)

Introduction: Use Struts to create a Web application

In this tutorial, you will learn how to create a simple Struts application that contains two Web pages, an action mapping, and a form bean. You will use the Web diagram editor to help you visualize the logic flow of the application. The resulting application displays an input Web page where users can type a year, a month, and a date. If the input is valid, the corresponding day of the week is computed and shown on an output Web page. If the input is invalid, an error message is displayed on the input Web page. You will also learn how to set up a runtime test environment and test your application.

Learning objectives

You will learn how to do these tasks:

- Create a Struts project
- Edit your Web diagram
- Create and edit a form bean
- Create input and output JavaServer Pages (JSP) files
- Create an action and an action mapping
- Add ActionError tags to the input JSP page
- Test your Struts application

Time Required

This tutorial should take approximately **60 minutes** to finish. If you explore other concepts related to this tutorial, it could take longer to complete.

Skill level

Advanced

Audience

Web application developers and Web user interface designers

System requirements

To complete this tutorial, you need one of the following servers configured on your local machine:

- A WebSphere® Application Server 5.1 test environment or remote server
- A WebSphere Application Server 6.x runtime server
- An Apache Tomcat 5.x runtime server

Prerequisites

To complete this tutorial, you should be familiar with the following areas:

- Basic Web design concepts, such as sites, pages, browsers, and servers
- Web page elements, such as tables, hyperlinks, forms, and images
- Editing HTML code for Web pages
- Workbench perspectives and views in Eclipse-based products

Lesson 1: Create a Struts project

Create a dynamic Web project that is enabled for Struts.

Struts is an open-source software framework for building Java™ Web applications. Struts uses the *model-view-controller* (MVC) design pattern for application development. The MVC design pattern helps you to create and maintain complex Web applications by separating the application into three parts:

- The *model* maintains the data and the business logic in the application. Since this tutorial does not use a database, and the only data is entered by the user, there is no model in this tutorial.
- The *view* is the interface that displays information to the user. In this tutorial, the view is composed of two Web pages that collect data from the user and display results to the user.
- The *controller* handles events in the application. In this tutorial, the controller is composed of an action mapping and a form bean that work together to process the information that the user enters.

Tip: To learn more about Struts technology, visit <http://struts.apache.org/>.

To create a new Struts-based Web project:

1. In the menu bar, select **File** → **New** → **Project** to open the New Project window.
2. Select **Web** → **Dynamic Web Project** and click **Next** to open the New Dynamic Web Project wizard.
3. In the **Project name** field, type `DayOfWeek`.
4. In the **Target runtime** field, select the runtime that you have configured. Click **Next**.

Note: If you have not configured a runtime environment, click **New** to open the New Server Runtime wizard. For this tutorial, you can use one of the following servers:

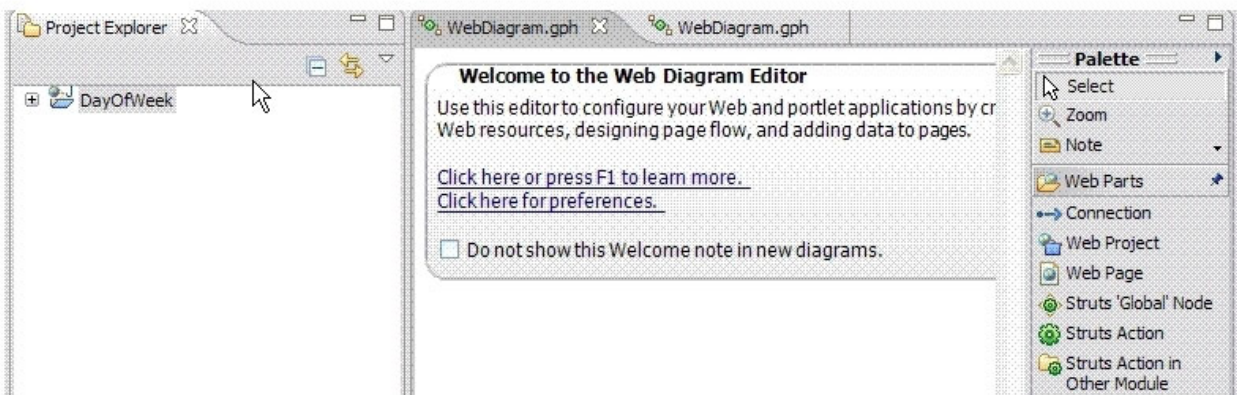
- A WebSphere Application Server 5.1 test environment or remote server
- A WebSphere Application Server 6.x runtime server
- An Apache Tomcat 5.x runtime server

(For more information about configuring an Apache Tomcat server, see <http://tomcat.apache.org/download-55.cgi>.)

5. On the Project Facets page, click **Struts**, then click **Next**.
6. On the Web Module page, click **Next**.
7. On the Struts Settings page, click **Finish**.
8. If you are not in the J2EE perspective, click **Yes** to open this perspective.
9. If this is the first time that you are creating a Web project, click **OK** to enable the required product capabilities.

You can see the new Struts Web project in the Project Explorer. A corresponding Web diagram opens.

Tip: Select the Welcome to the Web Diagram Editor object on the Web diagram, and then press the Delete key to remove the object.



Lesson checkpoint

You learned to create a Struts Web project, select a runtime environment, and open a new Web diagram.

Lesson 2: Create Web diagram nodes

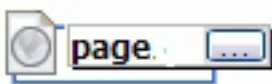
Use this lesson to learn how to add nodes to the Web diagram of your new project.

A Web diagram helps you visualize and change the flow of a Web application. Before using the Web diagram editor, you should understand the following terms:

- A *node* represents a resource, such as a Web page, a Java bean, a Struts action, or a Web application.
- A *connection* represents the logic flow between two of these resources.

To create nodes on your Web diagram to represent two Web pages and a Struts action:

1. Open the Web Parts drawer of the palette. Drag a Web page node into your diagram. The diagram shows a new Web page node with the default name page.jsp.



Tip: When you create a new node in a Web diagram, the name of the node is automatically selected for editing. If you click elsewhere in the diagram before typing a new name, the new Web page name stays the same. To rename a node, click the name to highlight it, and then type a new name.

2. Type `input.jsp` as the new name of the Web page node.

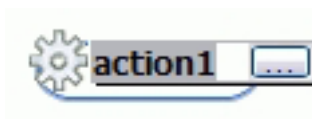


Note: As soon as you name the Web page node, the JavaServer Pages (JSP) file that this node represents is created in your `DayOfWeek\WebContent` Web project folder.

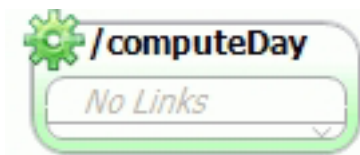
3. Drag another Web page node into the diagram.
4. Type `output.jsp` as the name of the second Web page.



5. Drag a Struts action into the diagram to create a node called `action1`.

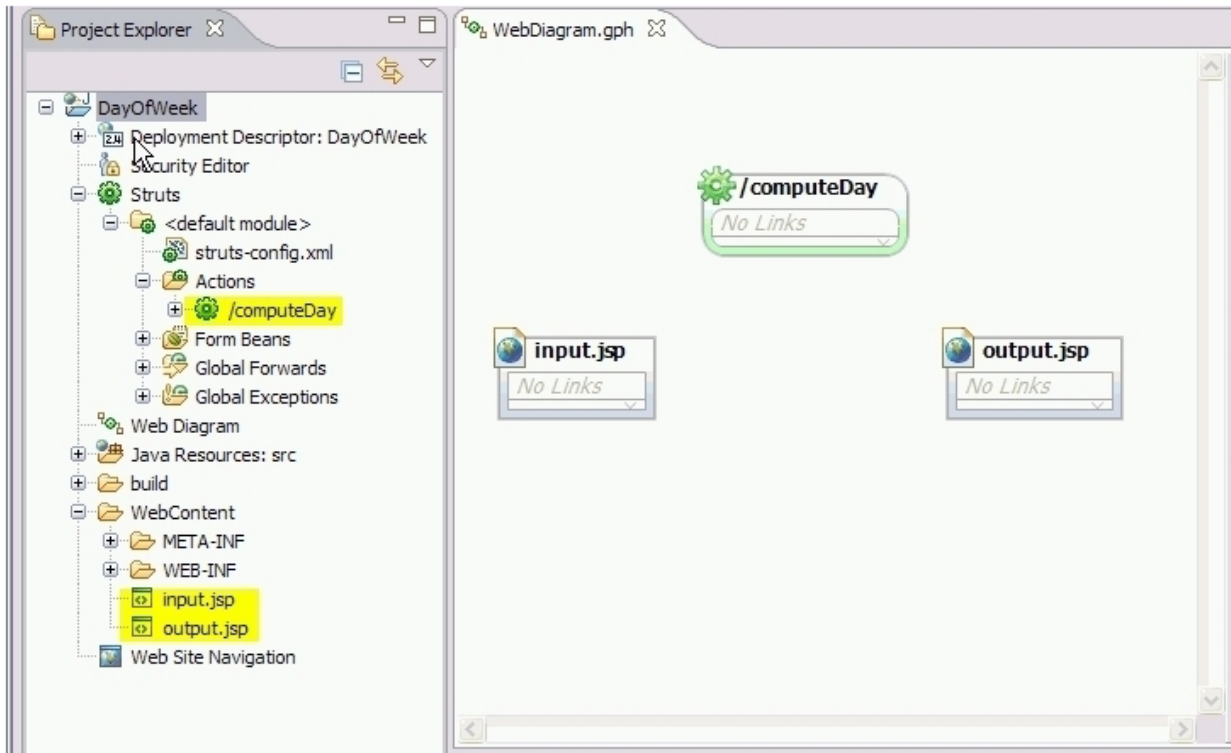


6. Type `computeDay` as the name of the action and press **Enter**. The Web diagram displays an action node named `/computeDay`. To see the `/computeDay` action in the Project Explorer, navigate to your Web project folder, and then navigate to `DayOfWeek\Struts\<default module>\Actions`.



7. Save your Web diagram.

Your Web project should include the following items: `/computeDay`, `input.jsp`, and `output.jsp`.



Lesson checkpoint

In this lesson, you added an input JSP file, an output JSP file, and the `/computeDay` action.

You learned to do these tasks:

- Add JSP files and a Struts action to your Web diagram.
- Locate your JSP files and Struts action in the Project Explorer.

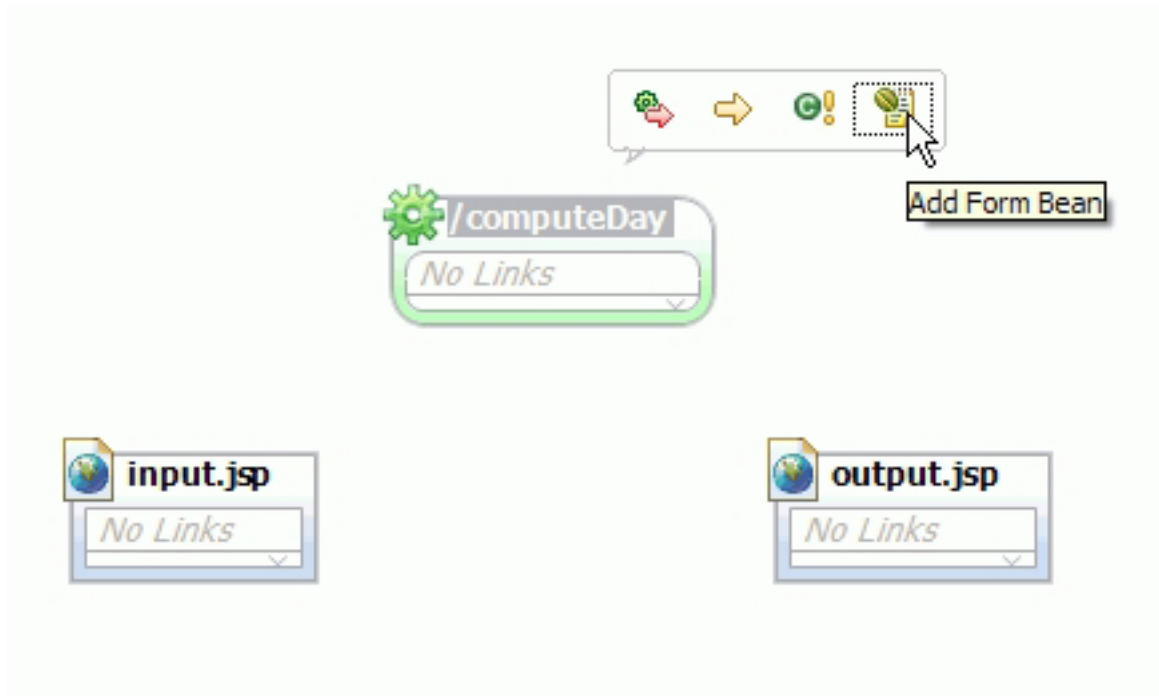
Lesson 3: Create a form bean

Learn how to create a form bean.

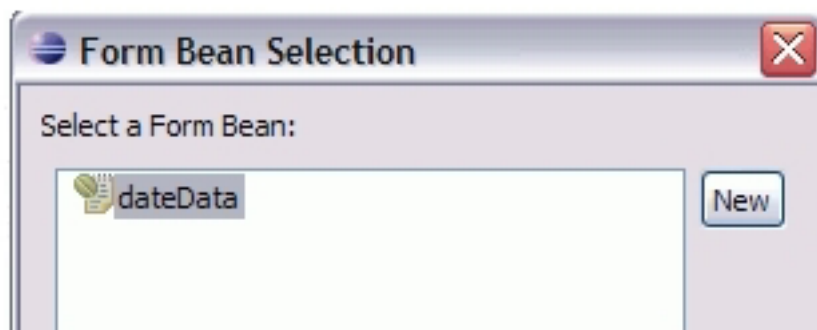
A *form bean* is a type of Java bean. The form bean stores HTML form data from a submitted client request or input data from a Struts action link that a user clicked.

To create a form bean:

1. In the diagram, hold the mouse pointer over the `/computeDay` action and click **Add Form Bean**.



2. In the Form Bean Selection window, click **New** to open the New Form Bean wizard.
3. In the **Form Bean Name** field, type `dateData`. Click **Next**.

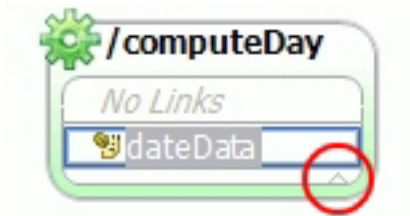


4. Click **Next** to skip the next wizard page. Since your application is new, the pages in your project do not contain any form information yet. If they did, you could pull in fields for this information here.
5. On the page to create new fields for your ActionForm class, add the following fields with the **Add** button, and click **Next**:

Name	Type
<code>year</code>	<code>int</code>
<code>month</code>	<code>int</code>
<code>day</code>	<code>int</code>
<code>dayOfWeek</code>	<code>String</code>

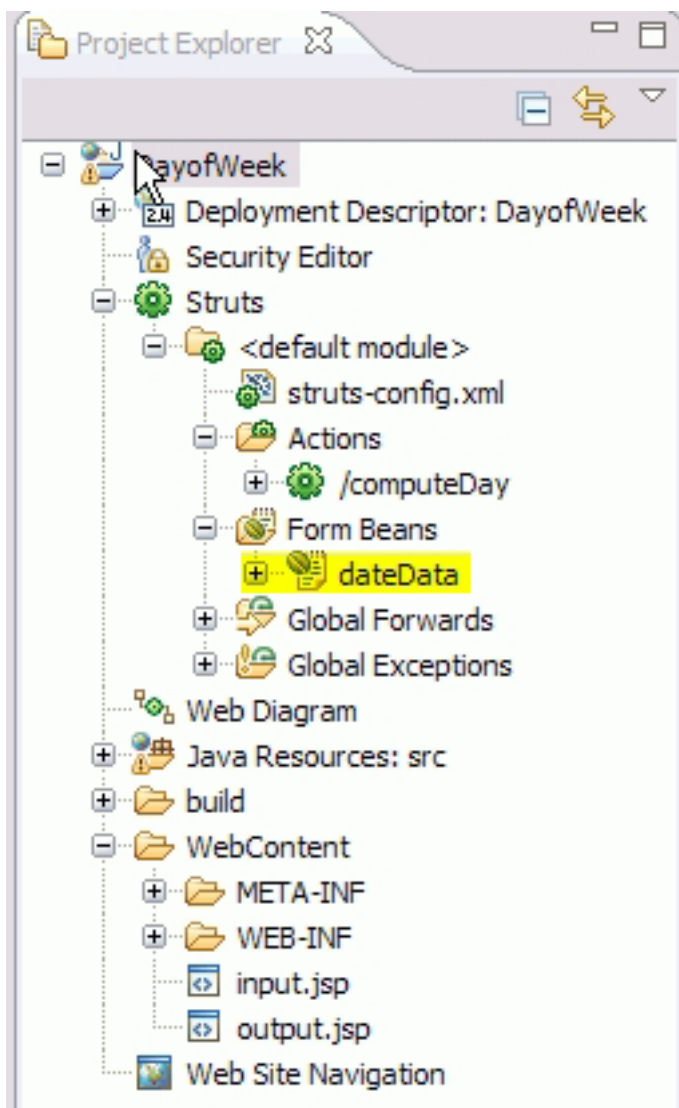
6. On the page to create a mapping for your ActionForm class, in the **Java package** field, type `com.ibm.dayofweek`. Click **Finish**. An updated Form Bean Selection window is displayed.
7. In the Form Bean Selection window, click the `dateData` form bean that you just created. Click **OK**.

8. In the `/computeDay` node, double-click the small triangle near the lower right corner of the node. The Data compartment expands to show the `dateData` form bean that is associated with the node.



The `dateData` form bean is added to the `/computeDay` node, and the `dateData.java` file is created in your Web project folder. In the Project Explorer, navigate to the `DayOfWeek\Struts\<default module>\Form Beans` folder to see the file.

At the end of this lesson, the Project Explorer looks like this:



Lesson checkpoint

In this lesson, you created the dateData form bean and defined the ActionForm class.

You learned to do these tasks:

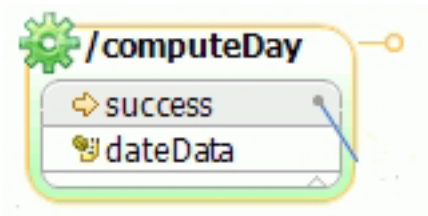
- Bring up the hover menu.
- Define the parameters of the ActionForm class.
- Locate a Java file in the Web project folder in the Project Explorer.

Lesson 4: Connect nodes in the Web diagram

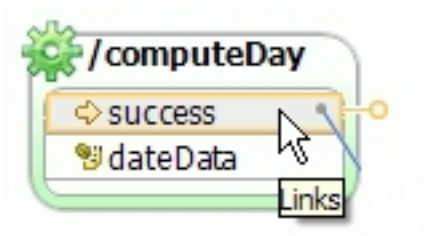
Learn how to use connection handles to connect the nodes in your Web diagram.

Now that you have placed nodes in your Web diagram, the next step is to connect them. You will use connection handles for this purpose. There are two types of connection handles:

- Top-level handles can be used to draw a connection from a node.



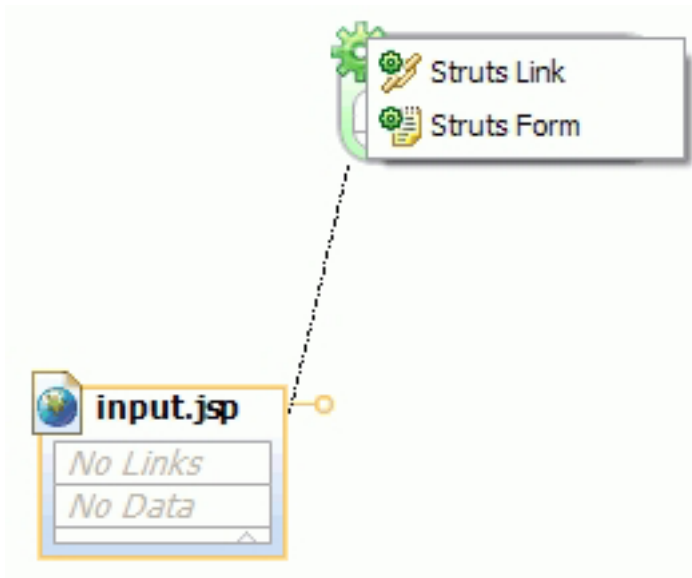
- Nested handles can be used to draw a connection from an item that is in a node compartment.



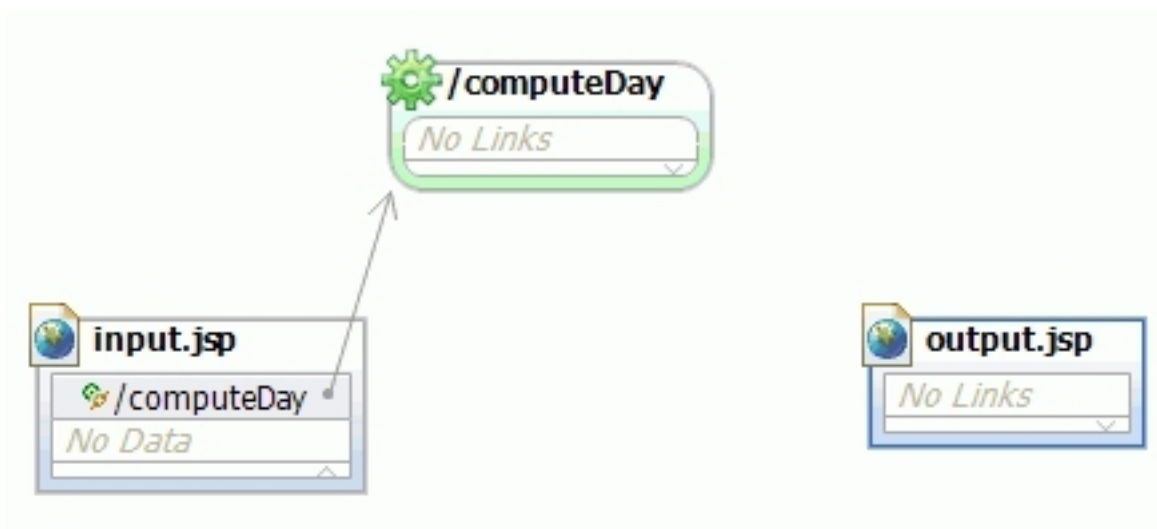
1. In the diagram, hold the mouse pointer over the input.jsp node. The node is highlighted, a connection handle is displayed, and a hover action menu is displayed.



2. Click and drag the handle from the input JSP node to the /computeDay action. A pop-up menu is displayed.



3. Select **Struts form** to submit an input to the action when processing a form. A Struts form is added to the input JSP node with a line connecting the node to the /computeDay action.



Tip: Alternatively, you can use the hover menu to add a Struts link to the input JSP node and then drag the link connection handle to the /computeDay action.

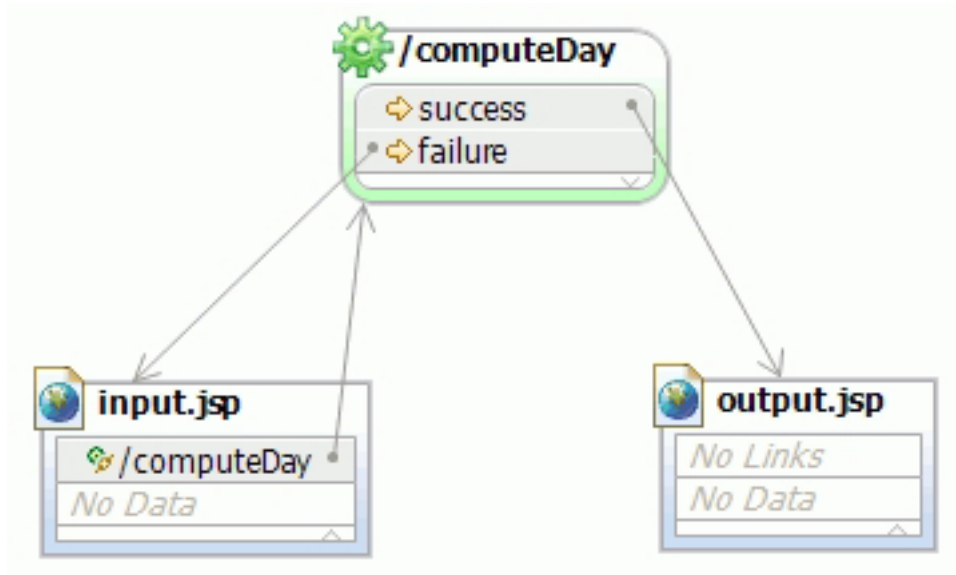
4. Connect the /computeDay action to the output.jsp node using a local forward:
 - a. Hold the mouse pointer over the /computeDay node to display the hover menu.
 - b. Click the **Add Local Forward** icon. A local forward named 'success' is added to the node. (By default, the first local forward that you create in each node is named 'success'.)
 - c. Drag the connection handle from the 'success' link to the output JSP node. A line is drawn from the link to the node to show the connection.

Tip: Each item in the Links compartment of a node also has a connection handle. When you click a node to grab its handle, be sure that you drag the node handle and not a link handle.

5. Connect the /computeDay node to the input JSP node using a local forward:
 - a. Hold the mouse pointer over the /computeDay node and drag its connection handle to the input JSP node.

- b. On the hover menu, select **Local forward**. A local forward named failure is added to the node, and a line connects the local forward to the input JSP node. (By default, the second local forward that you create in each node is named failure.)
6. Save your Web diagram.

The Web diagram editor should show a Struts link and two local forwards.



Lesson checkpoint

Well done! You have connected nodes in a Web diagram.

You learned to do these tasks:

- Choose a top-level handle or a nested handle to draw a connection.
- Add Struts links from the input JSP node to the action node.
- Add a local forward from the action node to the output JSP node.

Lesson 5: Edit a form bean

Edit the form bean source file and the Java resource file so that they work correctly in your Struts application.

1. In the Project Explorer, double-click the dateData form bean in the /computeDay node to open the form bean in the Java editor.
2. In the editor, find the following line of code near the bottom of the file:
`ActionErrors errors = new ActionErrors();`
3. Immediately below this line, insert the following code:

```
if (year < 1582)
{
    errors.add("year", new org.apache.struts.action.ActionError("pre_gregorian"));
}
```
4. Save and close the DateData.java file.
5. In the Project Explorer, navigate to the DayOfWeekJava\Resources: src\com.ibm.dayofweek.resources folder.
6. Double-click the ApplicationResources.properties file.
7. In ApplicationResources.properties, remove the # character from the lines that begin with errors.header and errors.footer.

8. Add the following code to the end of the file:

```
pre-gregorian=<li>Date is before 1582, the year the Gregorian calendar began</li>
```

```
# Optional header and footer for <errors/> tag.
errors.header=<ul>
errors.footer=</ul>
pre-gregorian=<li>Date is before 1582, the year the Gregorian calendar began</li>
```

9. Save and close the file.

Lesson checkpoint

In this lesson, you edited the source file of the form bean.

You learned to do these tasks:

- Locate the source file for the form bean.
- Edit the source code.

Lesson 6: Edit an action mapping

An action mapping represents an action that is run, and when running, it determines the flow of the application.

An action mapping is a configuration file entry that usually contains a reference to an action class. This entry can contain a reference to a form bean that the action can use, and can also define local forwards and exceptions that are visible only to this action. When you write code for a JavaServer Pages (JSP) page, you can refer to the action mapping to trigger an action.

The action class contains an execute method that is called when the form is submitted. In this method, you can supply the logic to determine if the calculated result is success (which is defined by the forward that points to the output JSP page), or failure (which is defined by the forward that points to the input JSP page).

To edit an action mapping:

1. In the Project Explorer, navigate to the DayOfWeek\Java Resources: src\dayofweek.actions folder.
2. Double-click the ComputeDayAction.java file to open it.
3. In the file, select all of the text and press **Delete** to remove it.
4. Copy and paste the following code into the file:

```
package dayofweek.actions;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import org.apache.struts.action.Action;
import org.apache.struts.action.ActionError;
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 com.ibm.dayofweek.DateData;

/**
 * @version 1.0
 * @author
 */

public class ComputeDayAction extends Action {
```

```

/**
 * Constructor
 */
public ComputeDayAction() {

    super();

}

public ActionForward execute(
    ActionMapping mapping,
    ActionForm form,
    HttpServletRequest request,
    HttpServletResponse response)
    throws Exception {

    ActionErrors errors = new ActionErrors();
    ActionForward forward = new ActionForward();
    // return value
    DateData dateData = (DateData) form;
    int year;
    int month;
    int day;
    int dayOfWeek;
    int valcen;
    int valleap;
    int valyear;
    int valmon;
    int valday;
    int[] centuries = new int[4];
    int[] months = new int[13];
    String[] daysOfWeek = new String[7];
    centuries[0] = 2;
    centuries[1] = 0;
    centuries[2] = 5;
    centuries[3] = 3;
    months[1] = 5;
    months[2] = 1;
    months[3] = 0;
    months[4] = 3;
    months[5] = 5;
    months[6] = 1;
    months[7] = 3;
    months[8] = 6;
    months[9] = 2;
    months[10] = 4;
    months[11] = 0;
    months[12] = 2;
    daysOfWeek[0] = "Sunday";
    daysOfWeek[1] = "Monday";
    daysOfWeek[2] = "Tuesday";
    daysOfWeek[3] = "Wednesday";
    daysOfWeek[4] = "Thursday";
    daysOfWeek[5] = "Friday";
    daysOfWeek[6] = "Saturday";

    try {
        day = dateData.getDay();
        month = dateData.getMonth();
        year = dateData.getYear();
        if (month < 3) {
            year--; // Subtract 1 from year
        }
        valcen = centuries[year / 100 % 4];
        valleap = year % 100 / 4;
        valyear = year % 100 % 7;
    }
}

```

```

        valmon = months[month];
        valday = day % 7;
        dayOfWeek = valcen + valleap + valyear + valmon + valday;
        dayOfWeek = dayOfWeek % 7;
        dateData.setDayOfWeek(daysOfWeek[dayOfWeek]);
        request.setAttribute("dateData", dateData);

    } catch (Exception e) {

        // Report the error using the appropriate name and ID.
        errors.add("name", new ActionError("id"));
        e.printStackTrace();
    }

    // If a message is required, save the specified key(s)
    // into the request for use by the tag.

    if (!errors.isEmpty()) {
        saveErrors(request, errors);
        forward = mapping.findForward("failure");
    } else {
        forward = mapping.findForward("success");
    }

    // Finish with
    return (forward);
}
}

```

Tip: The name of the package in your project might differ from `dayofweek.actions`. If so, replace `dayofweek.actions` on the first line of the preceding code with the name of your package.

5. Save and close the file.

Lesson checkpoint

Your action class now contains an `execute` method that is invoked when the Web form is submitted. You have created the logic to determine if the calculated result is success (which is defined by the forward that points to the output JSP page), or failure (which is defined by the forward that points to the input JSP page).

You learned to do these tasks:

- Add an `execute` method to an action class.
- Create logic to determine the success or failure of an action.

Lesson 7: Add elements to JavaServerPages (JSP) pages

Add input fields, push buttons, and output fields to JSP pages to create dynamic user interfaces for input and output.

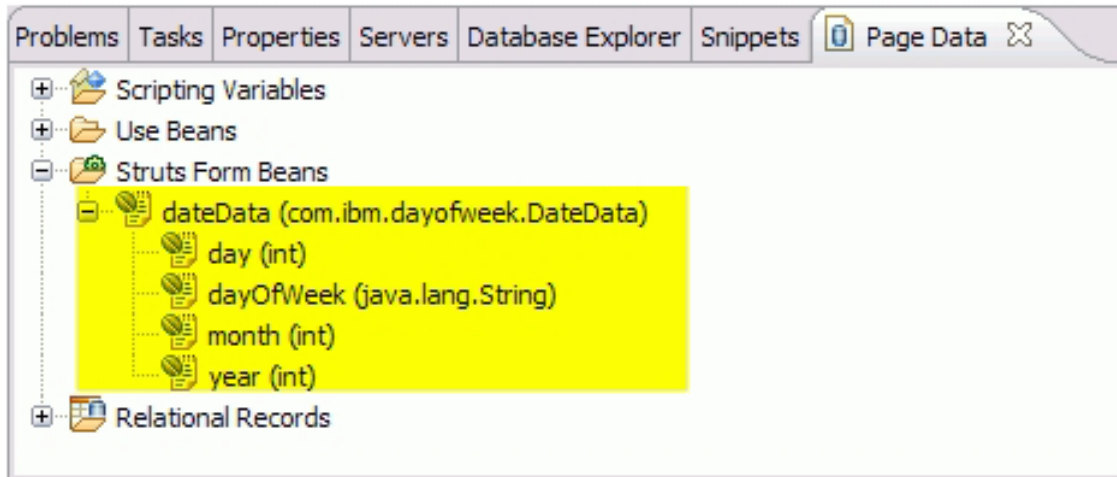
In the input fields, the user of the resulting application will type the day, month, and year of interest. The input page includes Submit and Refresh buttons. In the output field, the corresponding day of the week will be displayed. To create these pages:

1. Add elements to the input JSP page.
2. Add elements to the output JSP page.

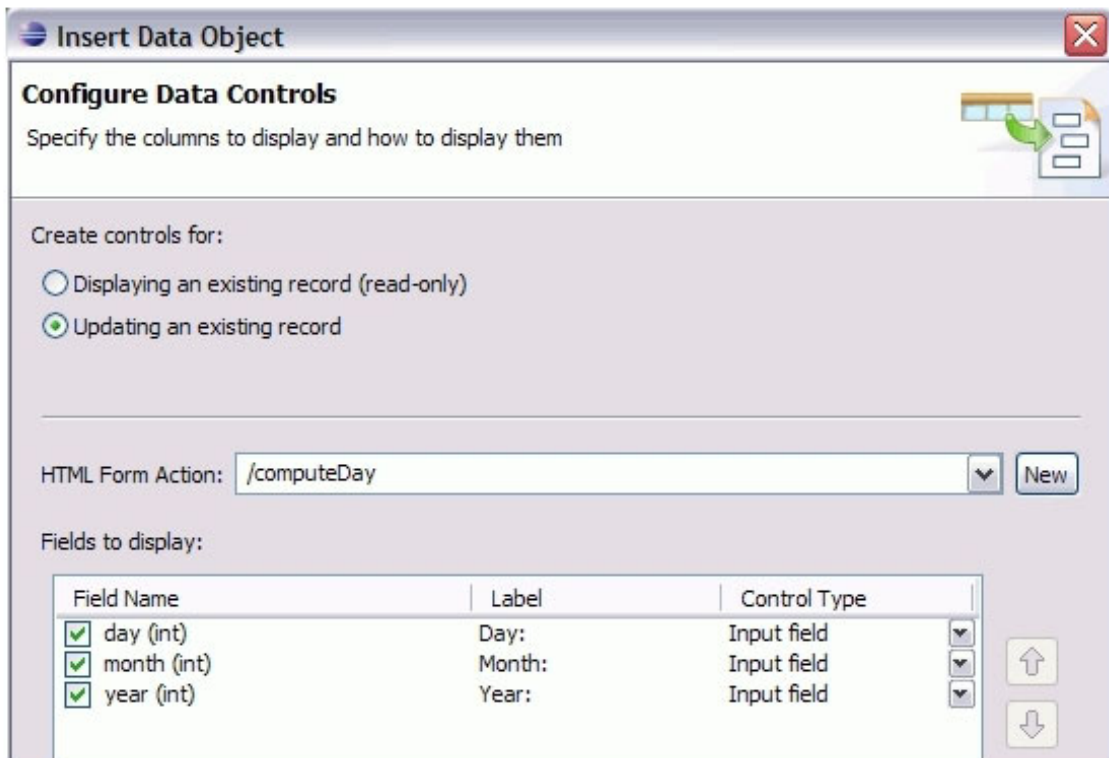
Input page

To add fields and push buttons to the input JSP page:

1. In your Web diagram, double-click the input.jsp page to open the page in Page Designer.
2. In Page Designer, click the Design tab.
3. In the main menu bar, select **Window** → **Show View** → **Other**.
4. In the window, select **Web** → **Page Data** and click **OK** to open the Page Data view.
5. In the Page Data view, expand the Struts Form Beans folder. Find the dateData form bean and the fields that you defined.



6. Drag the day, month, and year fields from the Page Data view to the input JSP page in Page Designer. The Insert Data Object window opens.



7. At the top, select **Updating an existing record**.
8. Click **Options**, and ensure that the **Submit button** and **Delete button** check boxes are selected. Then click **OK**.
9. Click **Finish** to create the fields and buttons in the form.

Day:

Month:

Year:

10. Save and close the input JSP file.

Output page

To add fields to the output JSP page:

1. In your Web diagram, double-click the output.jsp page to open the page in Page Designer.
2. In Page Designer, click the Design tab.

3. In the Page Data view, expand the Struts Form Beans folder. Find the dateData form bean and the fields that you defined.
4. Drag the dayOfWeek field from the Page Data view to the output JSP page in Page Designer. The Insert Data Object window is displayed.
5. At the top, select **Displaying an existing record** and click **Finish** to create the fields in the page.
6. Save and close the output JSP file.

Lesson checkpoint

In this lesson, you added input fields and push buttons to the input JSP page. You also added output fields to the output JSP page.

You learned to do these tasks:

- Open and modify a JSP page.
- Add input fields, output fields, and buttons to a JSP page.

Lesson 8: Display error messages

Display error messages when an action returns them to the input JavaServer Pages (JSP) page.

To use ActionErrors, you must add a Struts HTML tag to display an error message on the input JSP page. You must also add an input field to the action mapping.

To modify the application to return any bean validation errors to the input JSP page:

1. In the validate method of the DateData class, verify that if the input year is less than 1582, the method will return an error with the "pre-gregorian" message.

```
public ActionErrors validate(ActionMapping mapping, HttpServletRequest request) {  
  
    ActionErrors errors = new ActionErrors();  
    if (year < 1582)  
    {  
        errors.add("year", new org.apache.struts.action.ActionError("pre_gregorian"));  
    }  
}
```

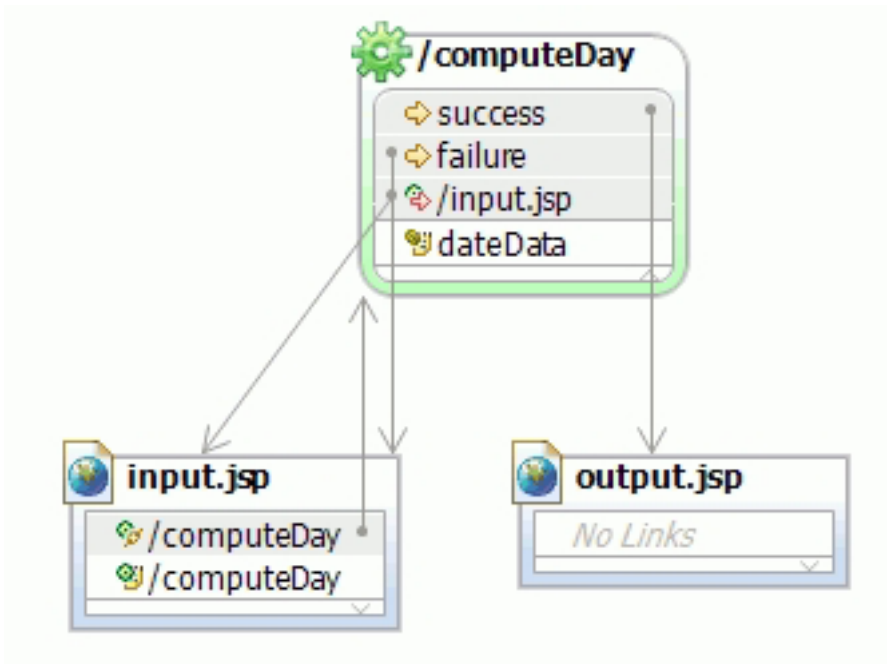
2. Locate the "pre-gregorian" error message in the ApplicationResources.properties file, and verify that the message matches the following definition:

```
pre-gregorian=<li>Date is before 1582, the year the Gregorian calendar began</li>
```

3. To specify the target JSP page for the error message, select an input for the action that uses the form bean:

- a. Open your Web diagram.
- b. In the diagram, hold the mouse pointer over the /computeDay node and drag the node connection handle to the input.jsp node.
- c. In the pop-up menu, select **Action Input**.

Your Web diagram should resemble the following image:



4. Update the input JSP page to display the error message:
 - a. Double-click the input.jsp page to open it in Page Designer.
 - b. From the Struts HTML Tags drawer in the palette, drag an Errors tag to the location in the page where you want the error message to be displayed.

Your input JSP page should resemble the following image:

- * First error message
- * Second error message
- * Third error message

Day:	
Month:	
Year:	
Submit	Reset

5. Save and close your input JSP page and your Web diagram.

Lesson checkpoint

In this lesson, you created the display for an error message in the input JSP page.

You learned to do these tasks:

- Add an HTML tag to display an error message in the input JSP page.
- Modify an action mapping to include an input field.

Lesson 9: Test a Struts application

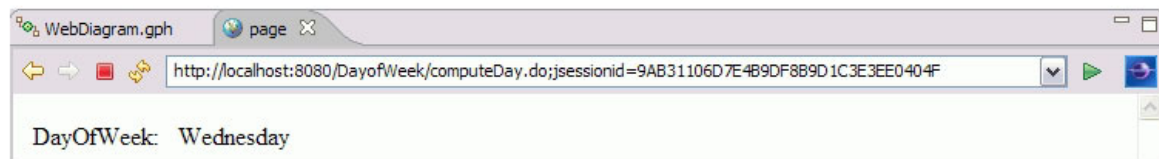
Test your Struts application in an IBM® WebSphere or Apache Tomcat runtime environment. You should be able to see the input and output controls that you previously added to the JSP pages in this environment.

To test your Struts application:

1. In the Project Explorer, right-click the input.jsp file and select **Run As → Run on Server**.
2. In the Server Selection window, select **WebSphere Application Server 6.x** or **Apache Tomcat Server**, and click **Finish**.
3. If you see the Server Operation window, click **Yes** to start the server.
4. Click **Finish**. The input JSP page opens in a Web browser view.



5. In the JSP page, type the following data in the fields:
 - a. In the Day field, type 10.
 - b. In the Month field, type 12.
 - c. In the Year field, type 1962.
6. Click the Submit button. The output JSP page is displayed in the Web browser view. The value in the DayOfWeek field should be Wednesday.



You can enter other dates in the fields of the input JSP to compute the corresponding days of the week.

Lesson checkpoint

Congratulations! You successfully tested your Struts application and viewed the output in a simulated Web browser.

You learned to do these tasks:

- Select a test environment.

- Type data into the fields of an input JSP page.
- Display results in an output JSP page in a string format.

Summary: Use Struts to create a Web application

In this tutorial, you learned how to create a simple Struts application.

Lessons Learned

After completing these lessons, you should be able to perform the following tasks:

- Create a Struts project
- Edit a Web diagram
- Create and edit a form bean
- Create two JavaServer Pages (JSP) files
- Create an action and an action mapping
- Add ActionError tags to the input JSP page
- Test a Struts application

Additional resources

If you want to learn more about the topics that are covered in this tutorial, consider the following sources:

- The Apache Struts Web site (<http://struts.apache.org>) contains information about the Struts framework.
- The online help (**Help** → **Help Contents**) includes documentation for Struts applications and Web diagrams.

Related information

ibm.com

eclipse.org