

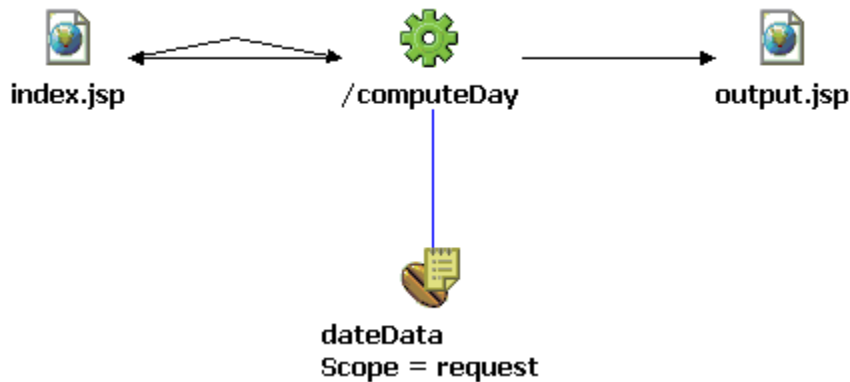
Create Struts Applications

This tutorial teaches you how to create a simple Struts application containing a Web page that accepts a year, month, and date as input and code that manipulates the data to compute the day of the week. If the input is valid, the computed value is displayed on an output Web page; if the input is not valid, an error message is displayed on the input page.

Create Struts Applications introduction

This tutorial shows you how to create a simple Struts application that has two Web pages, an action mapping, and a form bean. The application contains one Web page that accepts a year, month, and date as input and code that manipulates the data to compute the day of the week. If the input is valid, the computed value is displayed on an output Web page; if the input is not valid, an error message is displayed on the input page.

In this tutorial, you will also learn about building Web applications from Web diagrams. A Web diagram is a file that helps you visualize the application flow of a Faces-based or Struts-based Web application. This picture shows the Web diagram you will create in this tutorial:



In this figure, **index.jsp** represents the input JSP file, **output.jsp** represents the output JSP file, **dateData** represents the form bean that stores the input and output data, and **/computeDay** represents a `computeDay` action mapping of the action code that will be run when an input is submitted and its output is directed to **output.jsp**. The action code computes the day of the week for a specified date. The following table shows the input and output fields in this application:

Field Name	File Name	Description
year	index.jsp	Four-digit year
month	index.jsp	Two-digit month
day	index.jsp	Two-digit day
dayOfWeek	output.jsp	String representing the day of the week

Time required

To complete this tutorial, you will need approximately **1 hour and 30 minutes**. If you decide to explore other facets of Struts applications while working on the tutorial, it could take longer to finish.

Prerequisites

In order to complete this tutorial, you should be familiar with the following subjects:

- Basic Web design concepts, such as Web sites, pages, browsers, and servers
- The elements of a Web page, such as tables, hyperlinks, forms, and images

It will also help if you understand how to do these tasks:

- Use workbench perspectives and views
- Edit a Web page's HTML code

Learning objectives

This tutorial is divided into several exercises that must be completed in sequence for the tutorial to work properly. This tutorial teaches you how to create a simple Struts application by completing the following exercises:

- Create a Struts project
- Edit your Web diagram
- Create and edit a form bean

- Create two JSP files
- Create an action and an action mapping
- Test your Struts application

When you are ready, begin Exercise 1.1: Creating a Struts project

Exercise 1.1: Creating a Struts project

Struts is a framework of open-source software that can help you build Web applications quickly and easily. Struts uses the model-view-controller (MVC) design pattern of application development. The MVC design pattern makes it easier to create and maintain complex Web applications by separating the application into these three parts:

- The **View** is the interface that displays information to the user. In this tutorial, the view is the two Web pages which collect data from the user and display results to the user.
- The **Controller** handles the events in the application. In this tutorial, the controller is made up of an action mapping and a form bean which work together to process the information that the user enters.
- 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 the data that the user enters, there is no model in this tutorial.

In this exercise, you will switch to the Web perspective and create a dynamic Web project enabled for Struts.

Switching to the Web perspective

You will use the Web perspective during this tutorial. Follow these steps to switch to the Web perspective:

1. From the menu bar, select **Window > Open Perspective > Other**.
2. In the Select Perspective dialog, click **Web**.

If you don't see **Web**, select the **Show all** check box and it will appear in the list.

3. Click **OK**.
4. You may see a message asking if you want to enable the Web Development (typical) capability. If you see this message, click **OK**.

The workbench switches to the Web perspective.

Creating a Struts project

1. From the menu bar, click **File > New > Dynamic Web Project**.
2. On the Dynamic Web Project page, type this text as the name of the project:

DayOfWeek

3. Click **Next**.
4. On the Features page, make sure that the **Struts** check box is selected.
5. Click **Finish**.

Now you are ready to begin Exercise 1.2: Editing your Web diagram.

Exercise 1.2: Editing your Web diagram

Before you begin, you must complete Exercise 1.1: Creating a Struts project.

A Web diagram is a file that helps you visualize and change the flow of a Web application such as a Faces or Struts-based application. In this exercise, you will open the DayOfWeek Struts project's Web diagram, set the diagram's grid settings, and edit the diagram. The following terms pertain to Web diagrams:

- The **Web diagram editor** is a visual editor for editing Web diagrams.
- A **node** is an icon that represents a resource such as a Web page, Java bean, or Web application.
- A **connection** is a line that connects two nodes. Web diagrams can have four categories of connections: Faces, Java, Struts, and Web.
- A node or connection is termed **realized** if the resource that it represents exists. It is termed **unrealized** if the resource does not exist in the project.

For more information about nodes, connections, and Web diagrams, see Web diagrams and the Web diagram editor and Drawing connections in Web diagrams.

Opening the Web diagram

When you create a Struts-enabled dynamic Web project, a Web diagram is automatically created but not opened. Follow these steps to open the diagram:

1. In the Project Explorer view, expand **Dynamic Web Projects > DayOfWeek**.
2. Double-click **Web Diagram**. The Web diagram editor opens **Web Diagram**, which is an alias for the `diagram.gph` file.

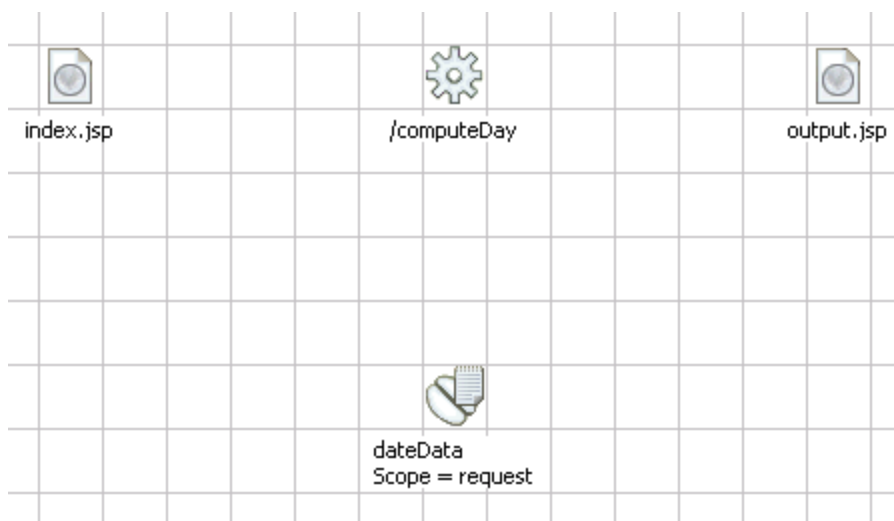
Setting the grid settings

By default, a Web diagram has no alignment grid. You can display a grid and turn on the snap-to-grid option, which aligns the nodes in the diagram to the grid and organizes the Web diagram.

1. Right-click in the Web diagram and click **Snap > Show grid** from the popup menu to display an alignment grid.
2. Right-click in the Web diagram again and click **Snap > Snap to grid on**.

Creating the Web diagram nodes

In this section, you will add nodes to the Web diagram that represent two Web pages, a form bean, and an action mapping. When you are finished, the Web diagram will look like the following image:



1. In the Palette view, click the **Web Parts** drawer to open it.

2. Drag a **Web Page** node from the Palette view onto the left side of your Web diagram.

The new Web page node appears on your Web diagram with the default name `page.jsp` selected.

3. Type this name for the new Web page:

```
index.jsp
```

Tip: When you create a new node in a Web diagram, the name of that node is automatically selected so you can immediately type in a name for that node. If you click anywhere else, the name will no longer be selected. If you want to change the name of a node and it is not selected, right-click on the node and choose **Rename** from the popup menu.

4. Drag another **Web page** node from the Palette onto the right side of the Web diagram.
5. Type this name for the second new Web page:

```
output.jsp
```

6. Click the **Struts Parts** drawer on the Palette view to open it.
7. Drag an **Action Mapping** node from the Palette view onto the Web diagram, directly between the two Web pages.
8. Type this name for the new action mapping:

```
computeDay
```

The Web diagram now shows an action mapping between the two Web pages named `/computeDay`. The Web diagram has added the forward slash mark to the name of the action mapping.

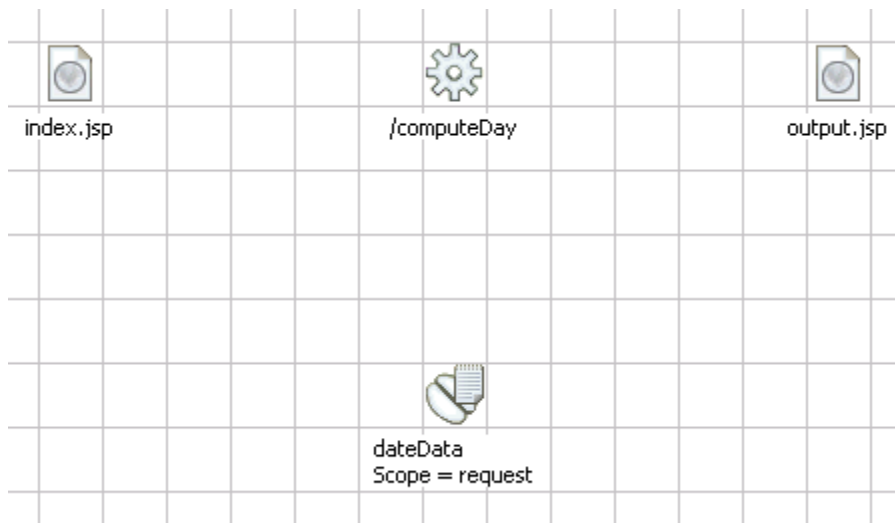
9. Drag a **Form Bean** node from the Palette view to the Web diagram, directly below the action mapping. The Form Bean Attributes window opens.
10. In the Form Bean Attributes window, type this text in the **Form Bean Name** field:

```
dateData
```

Leave the **Form Bean Scope** field set to `request`.

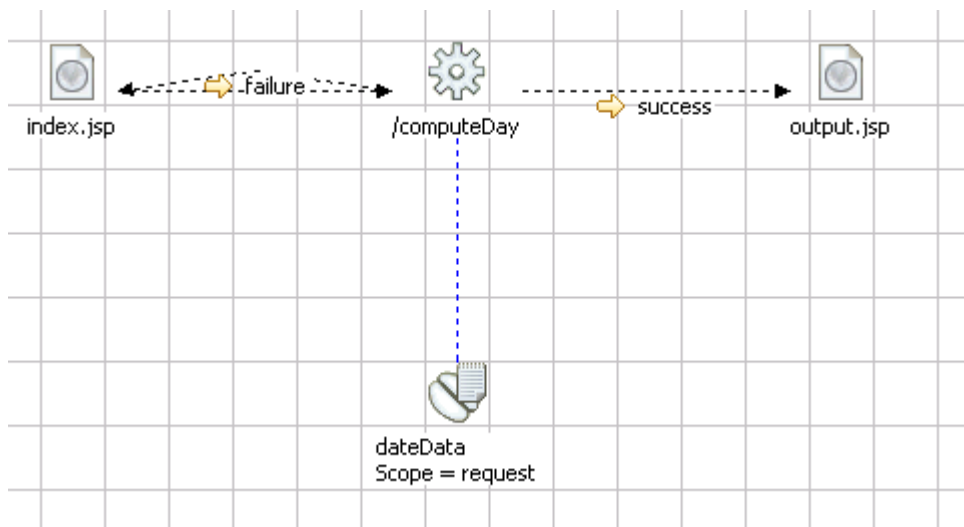
11. Click **OK**.
12. Save the Web diagram.

The Web diagram should now look like the following image:



Connecting the nodes in the Web diagram

Now that the nodes are placed on the Web diagram, the next step is to connect the nodes. When you are finished, the diagram will look like the following image:



1. From the Palette view, click **Connection**.

Now, your mouse cursor is set up to create a connection between the next two nodes you click.

2. Click the **index.jsp** node on the Web diagram.

Now there is a line between your mouse cursor and the `index.jsp` node. This line represents a connection from the `index.jsp` page to another node.

3. Click the **/computeDay** action mapping node.

There is now a dotted line from the `index.jsp` node to the `/computeDay` node, representing a connection between these two nodes.

4. In the same way, connect the **computeDay** action mapping to the **dateData** form bean.

Next, you will connect the computeDay action mapping to the output.jsp node with a local forward.

5. From the Palette view, click **Connection**.
6. Click the **/computeDay** action mapping.
7. Click the **output.jsp** node. The Choose a connection window opens.
8. From the Choose a connection window, expand **Local Forward**.
9. Click **<new>**.
10. Click **OK**.
11. Type this text as the name of the new local forward:

```
success
```

Naming the new local forward connection works the same way as naming the new nodes. When the connection is created, its name is selected and you can type it immediately. If you click somewhere else, the connection is no longer selected and you must right-click on it and choose **Rename** from the popup menu.

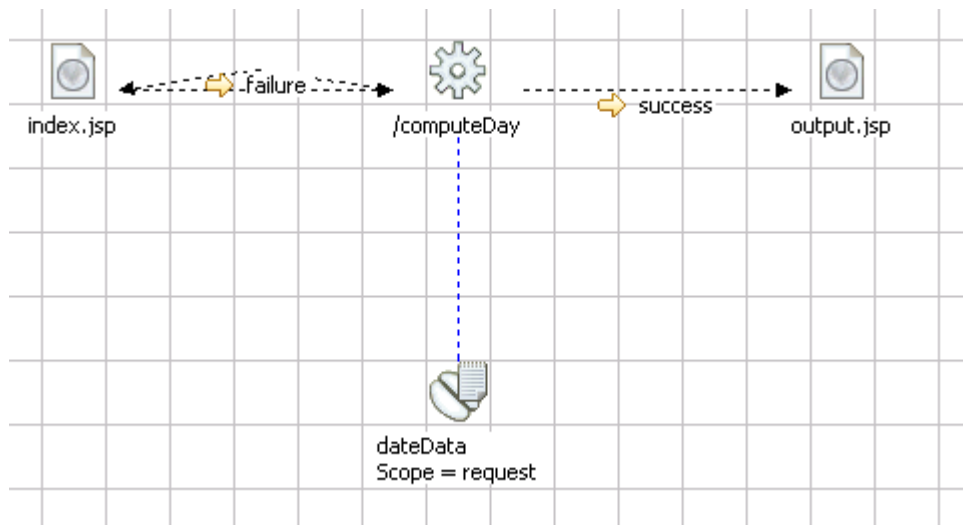
Finally, you will connect the computeDay action mapping to the index.jsp node with a global forward.

12. From the Palette view, click **Connection**.
13. Click the **computeDay** action mapping.
14. Click the **index.jsp** node. The Choose a connection window opens.
15. Expand **Global Forward**.
16. Under **Global Forward**, click **<new>**.
17. Click **OK**.
18. Type this text as the name of the new global forward:

```
failure
```

19. Save the Web diagram.

The Web diagram should look like the following image:



Now you are ready to begin Exercise 1.3: Creating and editing a form bean.

Exercise 1.3: Creating and editing a form bean

Before you begin, you must complete Exercise 1.2: Editing your Web diagram.

A form bean is a type of Java bean. A form bean is an instance of an ActionForm class subclass, which stores HTML form data from a submitted client request or that stores input data from a Struts action link that a user clicked. An HTML form comprises fields in which the user can enter information.

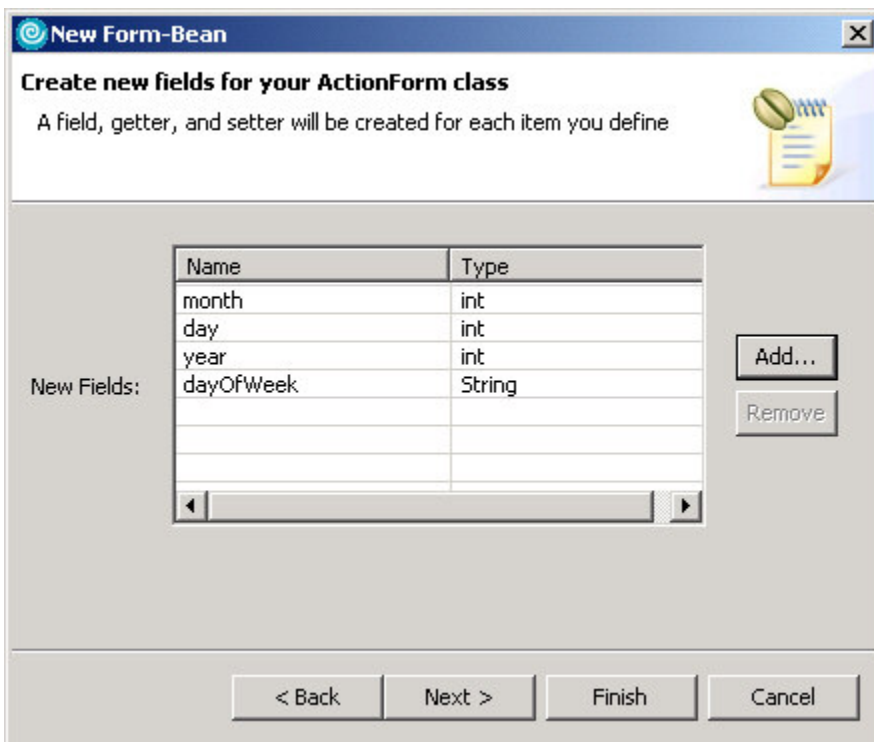
Creating a form bean

Create a Struts form bean from the Web diagram editor:

1. In the Web diagram, double-click the **dateData** form bean icon.
2. On the New Form Bean page, click **Next**.
3. On the Choose new fields for your ActionForm class page, select the **DayOfWeek** check box.
4. Click **Next**.
5. On the Create new fields for your ActionForm class page, click **Add** and specify the following fields:

Name	Type
year	int
month	int
day	int
dayOfWeek	String

The Create new fields for your ActionForm class page should look like the following picture:



6. Click **Next**.
7. On the Create a mapping for your ActionForm class page in the **Java package** field type this text as the name of the Java package:

```
com.ibm.dayofweek
```

8. Click **Finish**. Two things happen:
 - A file named ApplicationResources.properties is created in DayOfWeek\JavaSource\com\ibm\dayofweek\resources (DayOfWeek > JavaSource > com.ibm.dayofweek.resources in Project Explorer).
 - A file named DateData.java is created in the DayOfWeek\JavaSource directory (DayOfWeek > JavaSource > (default package) in Project Explorer). This file opens in the editor. In the next section, you will make changes to this file.

Tip: Creating your form beans before the JSP pages that use them eliminates the need to retype the field names when you create the JSP pages.

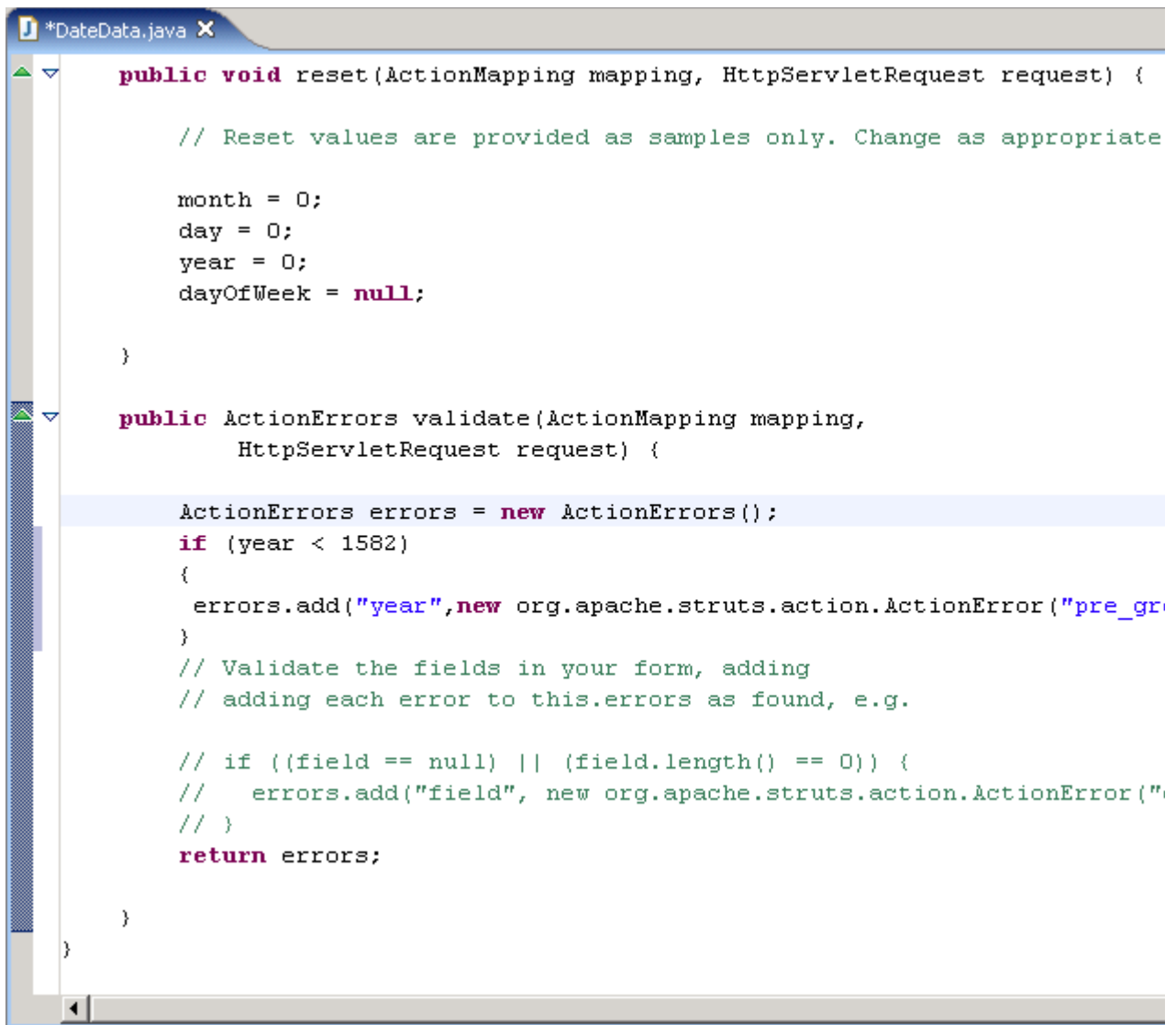
Editing the form bean

Edit the form bean source file and the Java resources file specifically for the application:

1. In the DateData.java file, find the line of code near the bottom of the file that reads:
`ActionErrors errors = new ActionErrors();`
2. Immediately after this code, insert the following code:

```
if (year < 1582)
{
    errors.add("year", new org.apache.struts.action.ActionError("pre_gregorian"));
}
```

The code should look like the following image:



```
public void reset(ActionMapping mapping, HttpServletRequest request) {

    // Reset values are provided as samples only. Change as appropriate

    month = 0;
    day = 0;
    year = 0;
    dayOfWeek = null;

}

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"));
    }
    // Validate the fields in your form, adding
    // adding each error to this.errors as found, e.g.

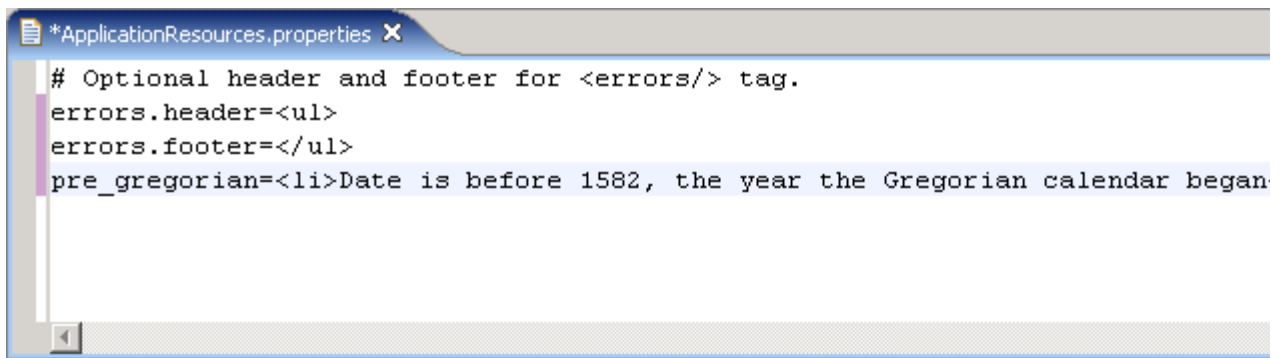
    // if ((field == null) || (field.length() == 0)) {
    //     errors.add("field", new org.apache.struts.action.ActionError(""));
    // }
    return errors;

}
}
```

3. Save and close the file.
4. In the Project Explorer, expand **JavaSource > com.ibm.dayofweek.resources**, and then double-click **ApplicationResources.properties**.
5. In the ApplicationResources.properties file, remove the comment character (#) from the lines that begin with errors.header and errors.footer.
6. Add the following code at the bottom of the file:

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

The ApplicationResources.properties file should look like the following image:



```
# 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
```

7. Save and close the file.

Now you are ready to begin Exercise 1.4: Creating an action and an action mapping.

Exercise 1.4: Creating an action and an action mapping

Before you begin, you must complete Exercise 1.3: Creating and editing a form bean.

In this exercise you will create and edit a Struts action mapping.

Creating an action mapping

Create a Struts action and action mapping, as follows:

1. Open the project's Web diagram.
2. In the Web diagram editor, double-click the **/computeDay** action mapping.
3. On the New Action Mapping page, click **Next**.
4. On the Create an Action class for your mapping page, type this text in the **Java package** field:

```
com.ibm.dayofweek
```

5. Click **Finish**.

An action mapping is added to the Struts configuration file (struts-config.xml), and a file named ComputeDayAction.java is created in DayOfWeek\JavaSource\com\ibm\dayofweek (DayOfWeek > JavaSource > com.ibm.dayofweek in Project Explorer). The ComputeDayAction.java file opens in the Java editor.

Editing the action mapping

1. Select all of the text in the ComputeDayAction.java file and press Delete.
2. Copy and paste the sample code into the ComputeDayAction.java file. This sample code is located at the end of this PDF file, after the summary.
3. Save the file and close it.

The file ComputeDayAction.java is compiled to produce a ComputeDayAction.class file as shown WebContent > WEB-INF > classes > com > ibm > dayofweek in the Web content folder of the Project Explorer.

4. Save the Web diagram.

Now you are ready to begin Exercise 1.5: Creating two JSP files.

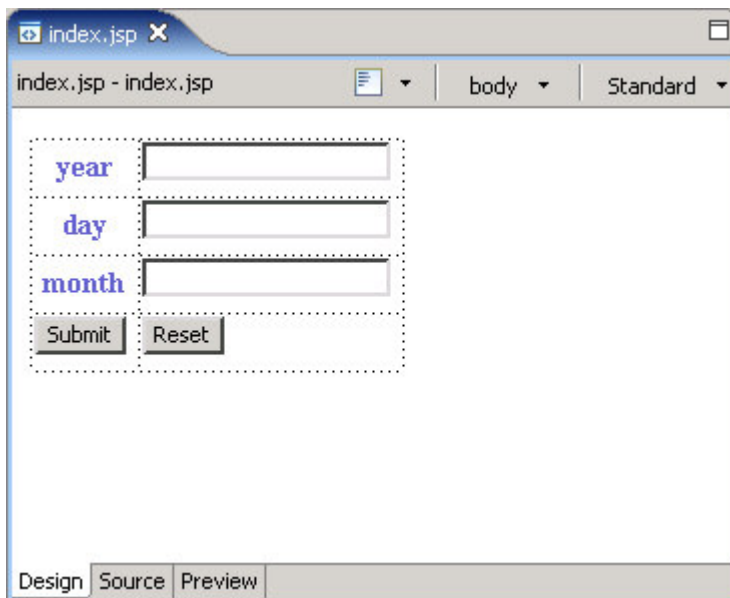
Exercise 1.5: Creating two JSP files

Before you begin, you must complete Exercise 1.4: Creating an action and an action mapping.

In this exercise, you will create two Web page files, one for input and one for output. These files will be named `index.jsp` and `output.jsp` to match the nodes you created in the Web diagram.

Creating the `index.jsp` file

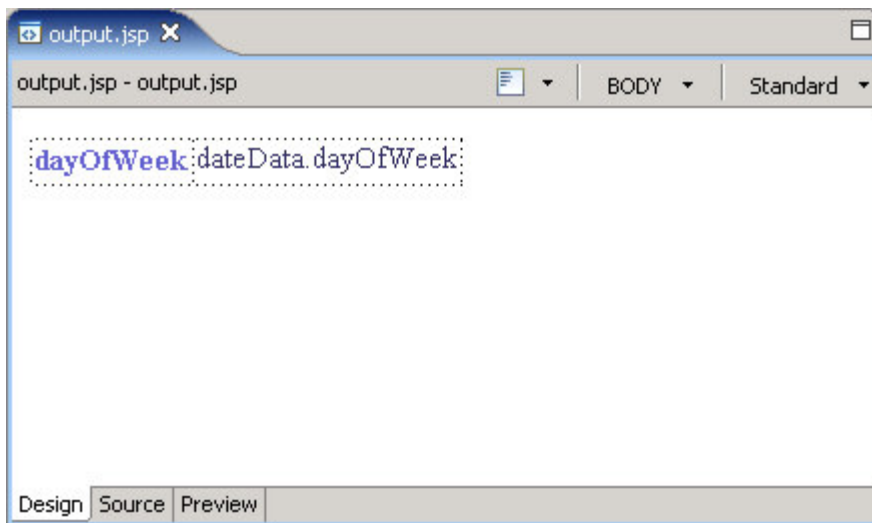
1. In the Web diagram, double-click the **index.jsp** node. The New JSP File wizard opens.
2. In the **Model** field, select **Struts JSP**.
3. Select the **Configure advanced options** check box.
4. Click **Next**.
5. On the following pages, click **Next**:
 - Tag Libraries
 - JSP File
 - JSP File
6. On the Form Field Selection page in the **Form bean entry** menu, click **dateData**.
7. Select the **year**, **month**, and **day** check boxes.
8. Select the **Generate fields in a form** check box.
9. Click **Finish**. The `index.jsp` file opens in the editor. It should look like this picture:



10. Close the `index.jsp` file.

Creating the `output.jsp` file

1. In the Web diagram, double-click the **output.jsp** node. The New JSP File wizard opens again.
2. In the **Model** field, select **Struts JSP**.
3. Select the **Configure advanced options** check box.
4. Click **Next**.
5. On the following pages, click **Next**:
 - Tag Libraries
 - JSP File
 - JSP File
6. On the Form Field Selection page in the **Form bean entry** menu, click **dateData**.
7. Select the **dayOfWeek** check box.
8. Clear the **Generate fields in a form** check box.
9. Click **Finish**. The `output.jsp` file opens in the editor. It should look like this picture:



10. Close the output.jsp file.

Now you are ready to begin Exercise 1.6: Testing your Struts application.

Exercise 1.6: Testing your Struts application

Before you begin, you must complete Exercise 1.5: Creating two JSP files.

In this exercise, you will test your Struts application in the WebSphere test environment.

Starting the test server

Test your application as follows:

1. In the Project Explorer, right-click **index.jsp** and select **Run > Run on Server**. A Server Selection window opens.
2. Make sure that **WebSphere Application Server v6.0** is selected, and click **Finish**.
3. If a Server Operation window opens, asking you whether you want to start the server, click **Yes**.
4. Click **Finish** again. This brings up the **index.jsp** file for input.

Running your Struts application

To test the application, find out what day of the week January 18, 2000 was.

1. Type the number 2000 in the **year** field.
2. Type the number 18 in the **day** field.
3. Type the number 1 in the **month** field.
4. Click **Submit**. The output should look like the following image, which tells you that January 8, 2000 was a Tuesday:

Day of week: Tuesday

You can enter other dates in the fields to determine the day of the week for other dates.

Finish your tutorial by reviewing the materials in the Summary.

Create Struts Applications summary

Congratulations! You have learned how to create a simple Struts application.

Completed learning objectives

If you have completed all of the exercises, you should now be able to do the following tasks:

- Create a Struts project
- Edit your Web diagram
- Create and edit a form bean
- Create two JSP files
- Create an action and an action mapping
- Test your Struts application

Source code for action in simple Struts application

```
package com.ibm.dayofweek;

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 <struts:errors> tag.

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

// Finish with
return (forward);

}
}

```