



# Exadel Studio

## Getting Started Guide for Creating a Struts Application

We are going to show you how to create a simple Struts application using the Exadel Studio plug-in for Eclipse. The completed application will ask a user to enter a name and click a button. The resulting new page will display the familiar message, “Hello <name>!”

This document will show you how to create such an application from the beginning, along the way demonstrating some of the powerful features of Exadel Studio. You will design the application, generate stub code for the application, fill in the stub coding, compile the application, and run the application all from inside Exadel Studio.

We’ll assume that you have already launched Eclipse with Exadel Studio or Exadel Studio Pro installed and also that the Exadel Studio perspective is the current perspective. (If not, make it active by selecting **Window/Open Perspective/Exadel Studio** from the menu bar.)

### Starting Up

We are first going to create a new project for the application.

1. Go to the menu bar and select **File/New/Project...**
2. Select **Exadel Studio/Struts/Struts Project** in the **New Project** dialog box.
3. Click **Next >**.
4. Enter `StrutsHello` as the project name.
5. Leave everything else as is, and click **Next >**.
6. Click **Next>** again.
7. Make sure that **struts-bean.tld**, **struts-html.tld**, and **struts-logic.tld** are checked in the list of included tag libraries and then click **Finish**.

A **StrutsHello** node should appear in the upper-left Package Explorer view.

8. Click the plus sign next to **StrutsHello** to reveal the child nodes.
9. Click the plus sign next to **WebContent** under **StrutsHello**.
10. Click the plus sign next to **WEB-INF** under **WebContent**.
11. Then, double-click on the **struts-config.xml** node to display a diagram of the Struts application configuration file in the editing area.

## Exadel Studio: Getting Started Guide for Creating a Struts Application

At this point, it's empty except for the background grid lines.

### Creating the Application Components

Now, we will design the application by creating the individual components as placeholders first. (We don't have to complete all of the details inside the components until afterwards.)

#### Creating JSP Page Placeholders

Next, let's create and place two JSP pages. We will not write any code for the files, but only create them as placeholders so that we can create links to them in the diagram. We will write the code a little bit later.

#### Creating the Page Placeholders

12. Bring the Web Projects view to the front of the Package Explorer view by selecting the **Web Projects** tab next to that tab.
13. Right-click the **StrutsHello/WEB-ROOT (WebContent)** folder in the Web Projects view and select **New/Folder...**
14. Enter `pages` for a folder name and click **Finish**.

We will keep our presentation files in this folder.

15. Right-click the **pages** folder and select **New/File/JSP...**
16. For **Name** type in `inputname` (the JSP extension will be automatically added to the file), for **Template** select **StrutsForm**, and then click on the **Finish** button.
17. Right-click the **pages** folder again and select **New/File/JSP...**
18. For **Name** type in `greeting`, for **Template** leave as **Blank**, and then click on the **Finish** button.

Just leave these files as is for now.

#### Placing the Page Placeholders

Let's now place the two pages just created on the diagram.

19. Click on the **struts-config.xml** tab in the Editing area to bring the diagram to the front.
20. Click on the **inputname.jsp** page in the Web Projects view, drag it onto the diagram, and drop it.
21. Click on the **greeting.jsp** page in the Web Projects view, drag it onto the diagram, and drop it to the right of the **/pages/inputname.jsp** icon with some extra space.

You should now have two JSP pages in the diagram.

#### Creating an Action Mappings

Using a context menu on the diagram, we are next going to create an Action mapping.

## Exadel Studio: Getting Started Guide for Creating a Struts Application

22. Right-click between the two icons and select **Add/Action...**
23. Enter the following values:

<b>path</b>	/greeting
<b>name</b>	GetNameForm
<b>scope</b>	request
<b>type</b>	sample.GreetingAction
<b>validate</b>	<leave blank>

(GetNameForm is the name for a form bean that we will create later.)

24. Click **Finish**.

The **/greeting** action should appear in two places, in the diagram and also under the **action-mappings** node under the **struts-config.xml** node in the Outline view. Also, note the asterisk to the right of the name, “struts-config.xml,” in the Outline view showing that the file has been changed, but not saved to disk.

### Creating a Link

Let’s now create a link from the **inputname.jsp** page to the action.

25. On the left-hand side of the diagram in the column of icons, click on this icon:



26. In the connect-the-components mode you are in now, click on the **/pages/input-name.jsp** icon in the diagram and then click on the **/greeting** action.

A link will be created from the page to the action.

### Creating a Forward

Next, we are going to create a forward for the action.

27. On the left-hand side of the diagram in the column of icons, click on this icon, again:



28. Click on the **/greeting** action icon in the diagram and then click on the **/pages/greeting.jsp** icon.

That’s it. A link will be drawn from the action’s new **greeting** forward to the **greeting.jsp** JSP page. Note that the forward’s name will be set based on the name of the target JSP file name. If you don’t like it, you can easily change it.

29. Select the **Tree** tab at the bottom of the editor window (between **Diagram** and **Source**).

## Exadel Studio: Getting Started Guide for Creating a Struts Application

30. Expand the **struts-config.xml/action-mappings//greeting** node and then select the **greeting** forward.
31. In the Properties Editor to the right, change the text to `sayHello` in the **Name** field.
32. Select the **Diagram** tab at the bottom of the editor window and see how the diagram is also updated to reflect the change.

### Creating a Global Forward

One last component that we need to create in the diagram is a global forward.

33. Somewhere in the top-left corner of diagram, right-click and select **Add/Global Forward...**
34. Enter `getName` in the **Name** field.
35. Select the **Change...** button for **Path**.
36. In the **Edit Path** window, switch to the **Pages** tab.
37. Expand the **StrutsHello/WEB-ROOT (WebContent)/pages** node and then select the **inputname.jsp** page.
38. Click **Ok**.
39. Leave the rest of the fields blank and click **Ok**.

A forward object now appears on the diagram and also in the **global-forwards** folder in the Outline view.

40. Tidy up the diagram, by clicking and dragging around each icon, so that the diagram looks something like this:



### Creating a Form Bean

One last thing that we need to do is to create a form bean.

41. Switch to the Tree viewer in the editor for the **struts-config.xml** file, by selecting the **Tree** tab at the bottom of the editor window.
42. Right-click **struts-config.xml/form-beans** and select **Create Form Bean...**
43. Enter `GetNameForm` in the **name** field and `sample.GetNameForm` for **type**.
44. Click **Finish**.

## Exadel Studio: Getting Started Guide for Creating a Struts Application

45. To save your changes to **struts-config.xml**, select **File/Save** from the menu bar.

Note the disappearance of the asterisk next to the name, “struts-config.xml”.

### Generating Stub Coding

We are done with designing the application through the diagram. Now we need to write code for the action component. We also need to write an action class for the **/greeting** mapping along with a `FormBean`. To aid in the coding phase, Exadel Studio can generate Java class stubs for all of the components shown in the diagram.

46. Switch back to the diagram, by selecting the **Diagram** tab at the bottom of the editor window.
47. Right-click a blank space in the diagram and select **Generate Java Code....**
48. Leave everything as is in the dialog box and click **Generate** .

You should see a screen that says:

```
Generated classes: 2
Actions: 1
Form beans: 1
```

49. Click **Finish**.

The Java files will be generated in a **JavaSource/sample** folder that you can see in the Package Explorer view under the **StrutsHello** node. One Action stub and one `FormBean` stub will have been generated.

### Coding the Various Files

We will now code both the Java stub classes just generated, the JSP files left in as placeholders from previous steps, and a new start JSP page we will have to create.

#### Java Stub Classes

50. To finish the two Java classes, switch to the Package Explorer view and expand the **JavaSource/sample** folder.

#### GetNameForm.java

51. Double-click **GetNameForm.java** for editing.

You are looking at a Java stub class that was generated by Exadel Studio. Now we are going to edit the file.

52. Add the following attributes at the beginning of the class:

```
private String name = "";
private String greetName = "";
```

53. Inside the reset method, delete the TO DO and throw lines and add:

## Exadel Studio: Getting Started Guide for Creating a Struts Application

```
this.name = "";
this.greetName = "";
```

54. Inside the validate method, delete the TO DO and throw lines and add:

```
ActionErrors errors = new ActionErrors();
return errors;
```

55. Right-click and select **Source/Generate Getters and Setters...** from the context menu.
56. In the dialog box, check the check boxes for **name** and **greetName**, select **First method** for **Insertion point**, and click on the **OK** button.

The final GetNameForm.java file should look like this:

```
package sample;

import javax.servlet.http.HttpServletRequest;

import org.apache.struts.action.ActionErrors;
import org.apache.struts.action.ActionMapping;

public class GetNameForm extends org.apache.struts.action.ActionForm {

    private String name = "";
    private String greetName = "";

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getGreetName() {
        return greetName;
    }

    public void setGreetName(String greetName) {
        this.greetName = greetName;
    }

    public GetNameForm() {
    }

    public void reset(ActionMapping actionMapping, HttpServletRequest
request) {
        this.name = "";
        this.greetName = "";
    }

    public ActionErrors validate(ActionMapping actionMapping,
```

## Exadel Studio: Getting Started Guide for Creating a Struts Application

```
HttpServletRequest request) {
    ActionErrors errors = new ActionErrors();
    return errors;
}
}
```

57. Save the file.

### GreetingAction.java

58. Open **GreetingAction.java** for editing.

59. Inside the execute method, delete the TO DO and throw lines and add the following:

```
String name = ((GetNameForm) form).getName();
String greeting = "Hello, "+name+"!";
((GetNameForm) form).setGreetName(greeting);
return mapping.findForward(FORWARD_sayHello);
```

The final version of **GreetingAction.java** should look like this:

```
package sample;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import org.apache.struts.action.ActionForm;
import org.apache.struts.action.ActionForward;
import org.apache.struts.action.ActionMapping;

public class GreetingAction extends org.apache.struts.action.Action {

    // Global Forwards
    public static final String GLOBAL_FORWARD_getName = "getName";

    // Local Forwards
    public static final String FORWARD_sayHello = "sayHello";

    public GreetingAction() {
    }

    public ActionForward execute(ActionMapping mapping, ActionForm form,
HttpServletRequest request, HttpServletResponse response) throws
Exception {
        String name = ((GetNameForm) form).getName();
        String greeting = "Hello, "+name+"!";
        ((GetNameForm) form).setGreetName(greeting);
        return mapping.findForward(FORWARD_sayHello);
    }
}
```

60. Save the file.

## Exadel Studio: Getting Started Guide for Creating a Struts Application

61. Close the editors for the two Java files.

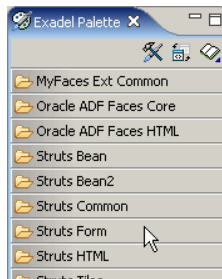
The last thing left to do is to code the JSP files whose editors should still be open from having been created as placeholders.

### JSP Pages

#### inputname.jsp

In this page, the user will enter any name and click the submit button. Then, the `greeting` action will be called through the form.

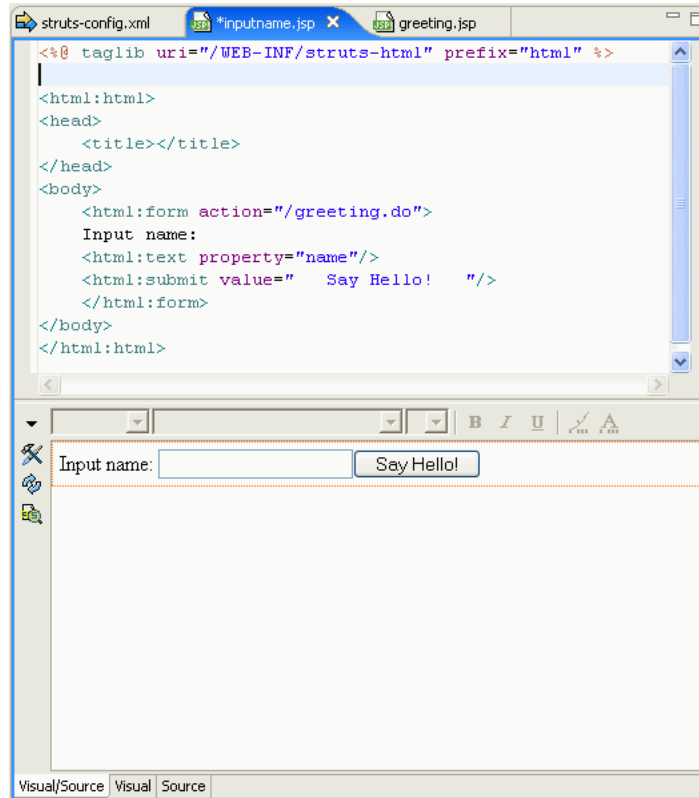
62. Click on the **inputname.jsp** tab in the Editing area to bring its editor forward.
63. In the Web Projects view, expand **StrutsHello/Configuration/default/struts-config.xml/action-mappings** and select **/greeting**.
64. Drag it and drop it between the quotes for the **action** attribute to the **html:form** element in the Source pane of the editor.
65. Then type this text on a new line just below this line:  
`Input name:`
66. Select the Visual pane of the editor.
67. Then, in the Exadel Palette, expand the **Struts Form** library, select **text**, and drag it onto the box.



68. In the Insert Tag dialog box, type in name for **property** and select **Finish**.
69. In the **Struts Form** library in the Exadel Palette, select **submit**, and drag it to right after the the text box in the Visual pane of the editor.
70. Right-click the submit button and select **<html:submit> Attributes** from the context menu.
71. In the Attributes dialog box, select the **value** field and type in `Say Hello!` for its value.

## Exadel Studio: Getting Started Guide for Creating a Struts Application

After tidying the page source, the Editor window for the file should look something like this:



### greeting.jsp

Next, we will fill in the result page.

72. Click on the **greeting.jsp** tab in the Editing area to bring its editor forward.
73. Type in the following code:

```
<html>
  <head>
    <title>Greeting</title>
  </head>
  <body>
    <p>

    </p>
  </body>
</html>
```

To complete editing of this file, we will use macros from the Exadel Palette. This palette is a view that should be available to the right of the editing area.

74. Click on the **Struts Common** folder in the Exadel Palette to open it.
75. Position the cursor at the beginning of the `greeting.jsp` file in the Source pane and then click on **bean taglib** in the Exadel Palette.

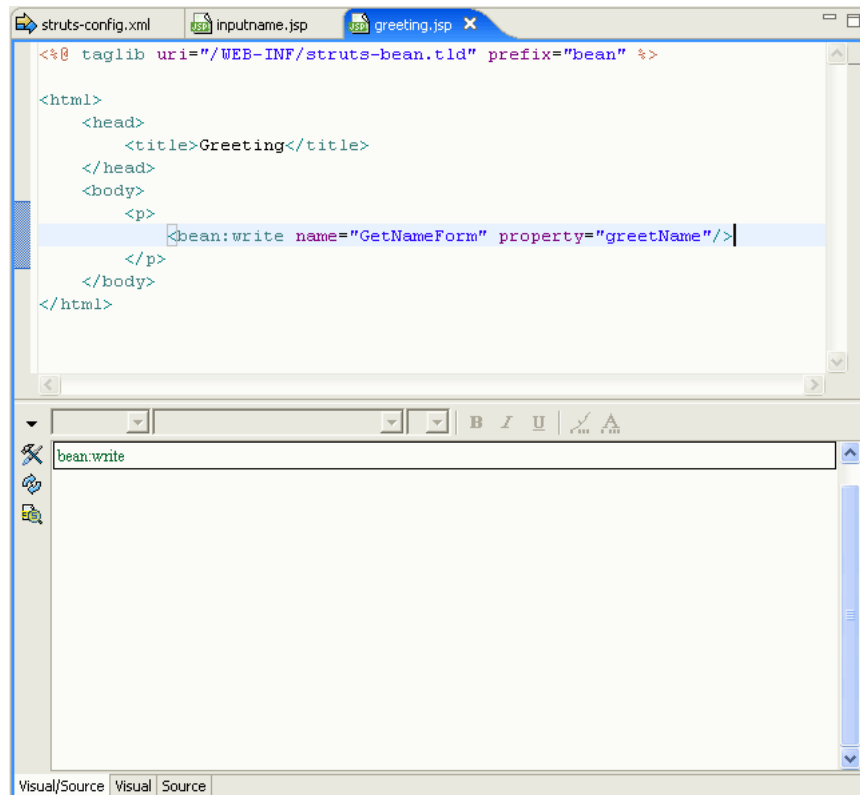
## Exadel Studio: Getting Started Guide for Creating a Struts Application

This will insert the following line at the top of the file:

```
<%@ taglib uri="/WEB-INF/struts-bean.tld" prefix="bean" %>
```

76. Click on the **Struts Bean** folder in the Exadel Palette to open it.
77. Position the cursor inside the `p` element.
78. Click on **write** in the Exadel Palette.
79. Type in `GetNameForm` for the **name** attribute and add a **property** attribute with `greetName` as its value.

The editor should now look like this:



### index.jsp

Finally, we will need to create and edit an **index.jsp** page. This page will use a Struts forward to simply redirect us to the **getName** global forward.

80. In the Web Projects view, right-click on **StrutsHello/WEB-ROOT(WebContent)** node and select **New/File/JSP...**
81. Type `index` for **Name** and click on the **Finish** button.
82. On the Exadel Palette, select the **Struts Common** folder of macros by clicking on it in the palette.
83. Click on the **logic taglib** icon.

## Exadel Studio: Getting Started Guide for Creating a Struts Application

84. Press the Enter key in the editor to go to the next line.
85. Back on the palette, select the **Struts Logic** folder of macros.
86. Click on **redirect**.
87. Delete the ending tag, put a forward slash in front of the closing angle bracket, and type `forward="getName"` in front of the slash.

The finished code for the page is shown below:

```
<%@ taglib uri="/WEB-INF/struts-logic.tld" prefix="logic" %>
<logic:redirect forward="getName"/>
```

88. To save all the edits to files, select **File/Save All** from the menu bar.

## Compiling the Classes

Because this is the Eclipse environment, no explicit compilation step is required. By default, Eclipse compiles as you go.

## Running the Application

Everything is now ready for running our application—without having to leave Exadel Studio—by using the Tomcat engine that comes with the Exadel Studio plug-in. For controlling Tomcat within Exadel Studio, the toolbar contains a panel.



89. Start up Tomcat by clicking on the first icon from left in this panel. (If Tomcat is already running, stop it by clicking on the third icon from the left and then start it again. Remember, the Struts run-time requires restarting the servlet engine when any changes have been made.)
90. After the messages in the **Console** tabbed view stop scrolling, Tomcat is available. At this point, right-click on the **getName** global forward in the **struts-config.xml** diagram view and select **Run on Server**.

The browser should appear with the application started.

