Including Files and Applets in JSP Documents

Topics in This Chapter

- Including JSP files at the time the main page is translated into a servlet
- Including HTML or plain text files at the time the client requests the page
- Including applets that use the Java Plug-In

Chapter 12

JSP has three main capabilities for including external pieces into a JSP document.

The `include` directive lets you reuse navigation bars, tables, and other elements in multiple pages. The included elements can contain JSP code and thus are inserted into the page before the page is translated into a servlet. This capability is discussed in Section 12.1.

Although including external pieces that use JSP is a powerful capability, other times you would rather sacrifice some power for the convenience of being able to change the included documents without updating the main JSP page. For example, my family's church has a Web page on which it posts snow cancellation announcements. This page is updated by 6:30 AM on Sundays when there is a cancellation. It is hardly reasonable to expect the Web developer to post this update; he probably sleeps in and barely makes the late-late service. Instead, a simple plain text file could be uploaded with the announcement, and the main page could use the `jsp:include` element to insert the announcement into the home page. This capability is discussed in Section 12.2.

Although this book is primarily about server-side Java, client-side Java in the form of Web-embedded applets continues to play a role, especially within fast corporate intranets. The `jsp:plugin` element is used to insert applets that use the Java Plug-In into JSP pages. This capability is discussed in Section 12.3.

12.1 Including Files at Page Translation Time

You use the `include` directive to include a file in the main JSP document at the time the document is translated into a servlet (which is typically the first time it is accessed). The syntax is as follows:

```
<%@ include file="Relative URL" %>
```

There are two ramifications of the fact that the included file is inserted at page translation time, not at request time as with `jsp:include` (Section 12.2).

First, you include the actual file itself, unlike with `jsp:include` , where the server runs the page and inserts its *output*. This approach means that the included file can contain JSP constructs (such as field or method declarations) that affect the main page as a whole.

Second, if the included file changes, all the JSP files that use it need to be updated. Unfortunately, although servers are *allowed* to support a mechanism for detecting when an included file has changed (and then recompiling the servlet), they are not *required* to do so. In practice, few servers support this capability. Furthermore, there is not a simple and portable "retranslate this JSP page now" command. Instead, you have to update the modification date of the JSP page. Some operating systems have commands that update the modification date without your actually editing the file (e.g., the Unix `touch` command), but a simple portable alternative is to include a JSP comment in the top-level page. Update the comment whenever the included file changes. For example, you might put the modification date of the included file in the comment, as below.

```
<%-- Navbar.jsp modified 3/1/00 --%>
<%@ include file="Navbar.jsp" %>
```

Core Warning

If you change an included JSP file, you must update the modification dates of all JSP files that use it.

For example, Listing 12.1 shows a page fragment that gives corporate contact information and some per-page access statistics appropriate to be included at the bottom of multiple pages within a site. Listing 12.2 shows a page that makes use of it, and Figure 12–1 shows the result.

Listing 12.1 `ContactSection.jsp`

```
<%@ page import="java.util.Date" %>

<%-- The following become fields in each servlet that
     results from a JSP page that includes this file. --%>
<%!
private int accessCount = 0;
private Date accessDate = new Date();
private String accessHost = "<I>No previous access</I>";
%>

<P>
<HR>
This page &copy; 2000
<A HREF="http//www.my-company.com/">my-company.com</A>.
This page has been accessed <%= ++accessCount %>
times since server reboot. It was last accessed from
<%= accessHost %> at <%= accessDate %>.

<% accessHost = request.getRemoteHost(); %>
<% accessDate = new Date(); %>
```

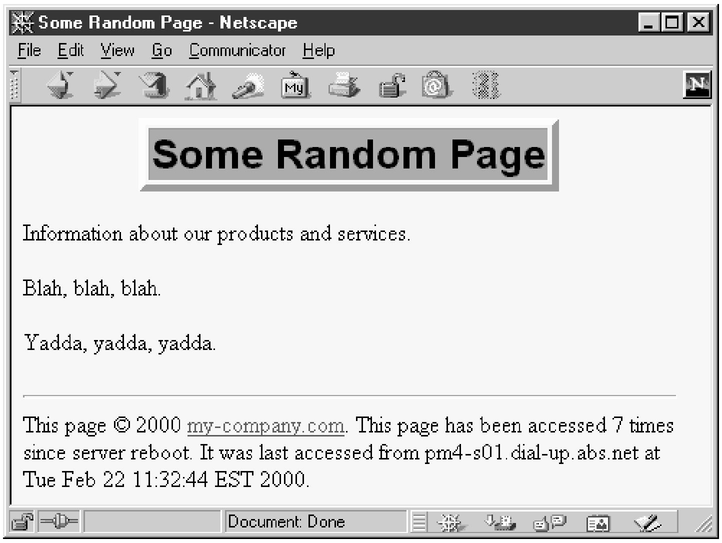
Listing 12.2 `SomeRandomPage.jsp`

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
<HEAD>
<TITLE>Some Random Page</TITLE>
<META NAME="author" CONTENT="J. Random Hacker">
<META NAME="keywords"
      CONTENT="foo,bar,baz,quux">
<META NAME="description"
      CONTENT="Some random Web page.">
<LINK REL=STYLESHEET
      HREF="JSP-Styles.css"
      TYPE="text/css">
</HEAD>

<BODY>
<TABLE BORDER=5 ALIGN="CENTER">
  <TR><TH CLASS="TITLE">
      Some Random Page</TABLE>
<P>
Information about our products and services.
<P>
Blah, blah, blah.
<P>
Yadda, yadda, yadda.

<%@ include file="ContactSection.jsp" %>

</BODY>
</HTML>
```


Some Random Page - Netscape
File Edit View Go Communicator Help
Document: Done

Some Random Page

Information about our products and services.

Blah, blah, blah.

Yadda, yadda, yadda.

This page © 2000 my-company.com. This page has been accessed 7 times since server reboot. It was last accessed from pm4-s01.dial-up.abs.net at Tue Feb 22 11:32:44 EST 2000.

Figure 12–1 Result of `SomeRandomPage.jsp`.

12.2 Including Files at Request Time

The `include` directive (Section 12.1) lets you include documents that contain JSP code into multiple different pages. Including JSP content is a useful capability, but the `include` directive requires you to update the modification date of the page whenever the included file changes, which is a significant inconvenience. The `jsp:include` action includes files at the time of the client request and thus does not require you to update the main file when an included file changes. On the other hand, the page has already been translated into a servlet by request time, so the included files cannot contain JSP.

Core Approach

Use the `include` *directive if included files will use JSP constructs. Otherwise, use* `jsp:include`.

Although the included files cannot *contain* JSP, they can be the result of resources that *use* JSP to create the output. That is, the URL that refers to the included resource is interpreted in the normal manner by the server and thus can be a servlet or JSP page. This is precisely the behavior of the `include` method of the `RequestDispatcher` class, which is what servlets use if they want to do this type of file inclusion. See Section 15.3 (Including Static or Dynamic Content) for details.

The `jsp:include` element has two required attributes, as shown in the sample below: `page` (a relative URL referencing the file to be included) and `flush` (which *must* have the value `true`).

```
<jsp:include page="Relative URL" flush="true" />
```

Although you typically include HTML or plain text documents, there is no requirement that the included files have any particular file extension. However, the Java Web Server 2.0 has a bug that causes it to terminate page processing when it tries to include a file that does not have a `.html` or `.htm` extension (e.g., `somefile.txt`). Tomcat, the JSWDK, and most commercial servers have no such restrictions.

Core Warning

Due to a bug, you must use `.html` *or* `.htm` *extensions for included files used with the Java Web Server.*

As an example, consider the simple news summary page shown in Listing 12.3. Page developers can change the news items in the files `Item1.html` through `Item4.html` (Listings 12.4 through 12.7) without having to update the main news page. Figure 12–2 shows the result.

Listing 12.3 `WhatsNew.jsp`

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
<HEAD>
<TITLE>What's New</TITLE>
<LINK REL=STYLESHEET
      HREF="JSP-Styles.css"
      TYPE="text/css">
</HEAD>

<BODY>
```

Listing 12.3 `WhatsNew.jsp` **(continued)**

```
<CENTER>
<TABLE BORDER=5>
  <TR><TH CLASS="TITLE">
      What's New at JspNews.com</TABLE>
</CENTER>
<P>

Here is a summary of our four most recent news stories:
<OL>
  <LI><jsp:include page="news/Item1.html" flush="true" />
  <LI><jsp:include page="news/Item2.html" flush="true" />
  <LI><jsp:include page="news/Item3.html" flush="true" />
  <LI><jsp:include page="news/Item4.html" flush="true" />
</OL>
</BODY>
</HTML>
```

Listing 12.4 `Item1.html`

```
<B>Bill Gates acts humble.</B> In a startling and unexpected
development, Microsoft big wig Bill Gates put on an open act of
humility yesterday.
<A HREF="http://www.microsoft.com/Never.html">More details...</A>
```

Listing 12.5 `Item2.html`

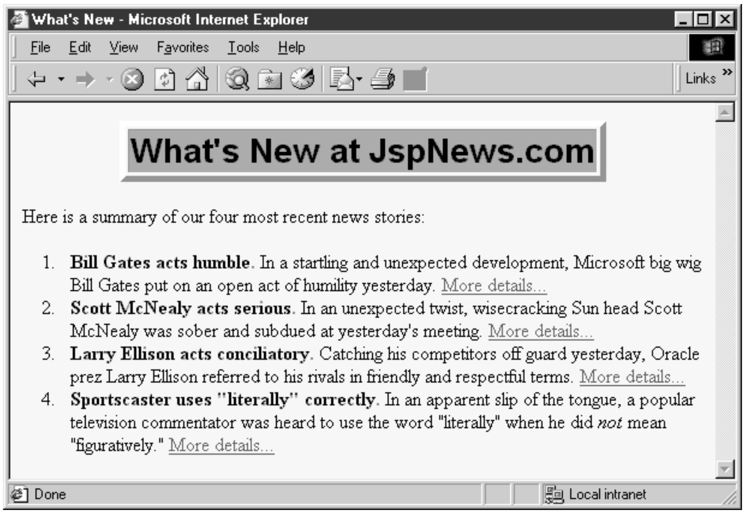
```
<B>Scott McNealy acts serious.</B> In an unexpected twist,
wisecracking Sun head Scott McNealy was sober and subdued at
yesterday's meeting.
<A HREF="http://www.sun.com/Imposter.html">More details...</A>
```

Listing 12.6 `Item3.html`

```
<B>Larry Ellison acts conciliatory.</B> Catching his competitors
off guard yesterday, Oracle prez Larry Ellison referred to his
rivals in friendly and respectful terms.
<A HREF="http://www.oracle.com/Mistake.html">More details...</A>
```

Listing 12.7 `Item4.html`

```
<B>Sportscaster uses "literally" correctly.</B> In an apparent
slip of the tongue, a popular television commentator was
heard to use the word "literally" when he did <I>not</I>
mean "figuratively."
<A HREF="http://www.espn.com/Slip.html">More details...</A>
```


What's New - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Links
Done
Local intranet

What's New at JspNews.com

Here is a summary of our four most recent news stories:

1. **Bill Gates acts humble.** In a startling and unexpected development, Microsoft big wig Bill Gates put on an open act of humility yesterday. More details...
2. **Scott McNealy acts serious.** In an unexpected twist, wisecracking Sun head Scott McNealy was sober and subdued at yesterday's meeting. More details...
3. **Larry Ellison acts conciliatory.** Catching his competitors off guard yesterday, Oracle prez Larry Ellison referred to his rivals in friendly and respectful terms. More details...
4. **Sportscaster uses "literally" correctly.** In an apparent slip of the tongue, a popular television commentator was heard to use the word "literally" when he did *not* mean "figuratively." More details...

Figure 12–2 Result of `WhatsNew.jsp`.

12.3 Including Applets for the Java Plug-In

With JSP, you don't need any special syntax to include ordinary applets: just use the normal HTML `APPLET` tag. However, these applets must use JDK 1.1 or JDK 1.02 since neither Netscape 4.x nor Internet Explorer 5.x support the Java 2 platform (i.e., JDK 1.2). This lack of support imposes several restrictions on applets:

- In order to use Swing, you must send the Swing files over the network. This process is time consuming and fails in Internet Explorer 3 and Netscape 3.x and 4.01-4.05 (which only support JDK 1.02), since Swing depends on JDK 1.1.
- You cannot use Java 2D.
- You cannot use the Java 2 collections package.
- Your code runs more slowly, since most compilers for the Java 2 platform are significantly improved over their 1.1 predecessors.

Furthermore, early browser releases had a number of inconsistencies in the way they supported various AWT components, making testing and delivery of complex user interfaces more burdensome than it ought to have been. To address this problem, Sun developed a browser plug-in for Netscape and Internet Explorer that lets you use the Java 2 platform for applets in a variety of browsers. This plug-in is available at `http://java.sun.com/products/plugin/`, and also comes bundled with JDK 1.2.2 and later. Since the plug-in is quite large (several megabytes), it is not reasonable to expect users on the WWW at large to download and install it just to run your applets. On the other hand, it is a reasonable alternative for fast corporate intranets, especially since applets can automatically prompt browsers that lack the plug-in to download it.

Unfortunately, however, the normal `APPLET` tag will not work with the plug-in, since browsers are specifically designed to use only their built-in virtual machine when they see `APPLET`. Instead, you have to use a long and messy `OBJECT` tag for Internet Explorer and an equally long `EMBED` tag for Netscape. Furthermore, since you typically don't know which browser type will be accessing your page, you have to either include both `OBJECT` and `EMBED` (placing the `EMBED` within the `COMMENT` section of `OBJECT`) or identify the browser type at the time of the request and conditionally build the right tag. This process is straightforward but tedious and time consuming.

The `jsp:plugin` element instructs the server to build a tag appropriate for applets that use the plug-in. Servers are permitted some leeway in exactly how they implement this support, but most simply include both `OBJECT` and `EMBED`.

The jsp:plugin Element

The simplest way to use `jsp:plugin` is to supply four attributes: `type`, `code`, `width`, and `height`. You supply a value of `applet` for the `type` attribute and use the other three attributes in exactly the same way as with the `APPLET` element, with two exceptions: the attribute names are case sensitive, and single or double quotes are always required around the attribute values. So, for example, you could replace

```
<APPLET CODE="MyApplet.class"
        WIDTH=475 HEIGHT=350>
</APPLET>
```

with

```
<jsp:plugin type="applet"
            code="MyApplet.class"
            width="475" height="350">
</jsp:plugin>
```

The `jsp:plugin` element has a number of other optional attributes. Most, but not all, parallel attributes of the `APPLET` element. Here is a full list.

- **`type`**
  For applets, this attribute should have a value of `applet`. However, the Java Plug-In also permits you to embed JavaBeans elements in Web pages. Use a value of `bean` in such a case.
- **`code`**
  This attribute is used identically to the `CODE` attribute of `APPLET`, specifying the top-level applet class file that extends `Applet` or `JApplet`. Just remember that the name `code` must be lower case with `jsp:plugin` (since it follows XML syntax), whereas with `APPLET`, case did not matter (since HTML attribute names are never case sensitive).
- **`width`**
  This attribute is used identically to the `WIDTH` attribute of `APPLET`, specifying the width in pixels to be reserved for the applet. Just remember that you must enclose the value in single or double quotes.

- **`height`**
  This attribute is used identically to the `HEIGHT` attribute of `APPLET`, specifying the height in pixels to be reserved for the applet. Just remember that you must enclose the value in single or double quotes.
- **`codebase`**
  This attribute is used identically to the `CODEBASE` attribute of `APPLET`, specifying the base directory for the applets. The `code` attribute is interpreted relative to this directory. As with the `APPLET` element, if you omit this attribute, the directory of the current page is used as the default. In the case of JSP, this default location is the directory where the original JSP file resided, not the system-specific location of the servlet that results from the JSP file.
- **`align`**
  This attribute is used identically to the `ALIGN` attribute of `APPLET` and `IMG`, specifying the alignment of the applet within the Web page. Legal values are `left`, `right`, `top`, `bottom`, and `middle`. With `jsp:plugin`, don't forget to include these values in single or double quotes, even though quotes are optional for `APPLET` and `IMG`.
- **`hspace`**
  This attribute is used identically to the `HSPACE` attribute of `APPLET`, specifying empty space in pixels reserved on the left and right of the applet. Just remember that you must enclose the value in single or double quotes.
- **`vspace`**
  This attribute is used identically to the `VSPACE` attribute of `APPLET`, specifying empty space in pixels reserved on the top and bottom of the applet. Just remember that you must enclose the value in single or double quotes.
- **`archive`**
  This attribute is used identically to the `ARCHIVE` attribute of `APPLET`, specifying a JAR file from which classes and images should be loaded.
- **`name`**
  This attribute is used identically to the `NAME` attribute of `APPLET`, specifying a name to use for inter-applet communication or for identifying the applet to scripting languages like JavaScript.

- **`title`**
  This attribute is used identically to the very rarely used `TITLE` attribute of `APPLET` (and virtually all other HTML elements in HTML 4.0), specifying a title that could be used for a tool-tip or for indexing.
- **`jreversion`**
  This attribute identifies the version of the Java Runtime Environment (JRE) that is required. The default is 1.1.
- **`iepluginurl`**
  This attribute designates a URL from which the plug-in for Internet Explorer can be downloaded. Users who don't already have the plug-in installed will be prompted to download it from this location. The default value will direct the user to the Sun site, but for intranet use you might want to direct the user to a local copy.
- **`nspluginurl`**
  This attribute designates a URL from which the plug-in for Netscape can be downloaded. The default value will direct the user to the Sun site, but for intranet use you might want to direct the user to a local copy.

The jsp:param and jsp:params Elements

The `jsp:param` element is used with `jsp:plugin` in a manner similar to the way that `PARAM` is used with `APPLET`, specifying a name and value that are accessed from within the applet by `getParameter`. There are two main differences, however. First, since `jsp:param` follows XML syntax, attribute names must be lower case, attribute values must be enclosed in single or double quotes, and the element must end with `/>`, not just `>`. Second, all `jsp:param` entries must be enclosed within a `jsp:params` element.

So, for example, you would replace

```
<APPLET CODE="MyApplet.class"
        WIDTH=475 HEIGHT=350>
  <PARAM NAME="PARAM1" VALUE="VALUE1">
  <PARAM NAME="PARAM2" VALUE="VALUE2">
</APPLET>
```

with

```
<jsp:plugin type="applet"
            code="MyApplet.class"
            width="475" height="350">
  <jsp:params>
    <jsp:param name="PARAM1" value="VALUE1" />
    <jsp:param name="PARAM2" value="VALUE2" />
  </jsp:params>
</jsp:plugin>
```

The jsp:fallback Element

The `jsp:fallback` element provides alternative text to browsers that do not support `OBJECT` or `EMBED`. You use this element in almost the same way as you would use alternative text placed within an `APPLET` element. So, for example, you would replace

```
<APPLET CODE="MyApplet.class"
        WIDTH=475 HEIGHT=350>
  <B>Error: this example requires Java.</B>
</APPLET>
```

with

```
<jsp:plugin type="applet"
            code="MyApplet.class"
            width="475" height="350">
  <jsp:fallback>
    <B>Error: this example requires Java.</B>
  </jsp:fallback>
</jsp:plugin>
```

However, you should note that the Java Web Server 2.0 has a bug that causes it to fail when translating pages that include `jsp:fallback` elements. Tomcat, the JSWDK, and most commercial servers handle `jsp:fallback` properly.

Core Warning

The Java Web Server does not properly handle `jsp:fallback`.

Example: Building Shadowed Text

In Section 7.5 (Using Servlets to Generate GIF Images), Listings 7.9 and 7.11 show a `JFrame` that uses Java 2D to create shadowed text in the size and font of the user's choosing. Listings 12.10 and 12.11 present an applet that uses Swing components to control this frame.

Since the applet uses Swing and Java 2D, it can run only with the Java Plug-In. Listing 12.8 shows a page that uses `jsp:plugin` to load the applet. Listing 12.9 shows the HTML that results from this page (I added some line breaks for readability) and Figures 12–3 through 12–6 show some typical output.

Listing 12.8 `ShadowedTextApplet.jsp`

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
<HEAD>
<TITLE>Using jsp:plugin</TITLE>
<LINK REL=STYLESHEET
      HREF="JSP-Styles.css"
      TYPE="text/css">
</HEAD>

<BODY>

<TABLE BORDER=5 ALIGN="CENTER">
  <TR><TH CLASS="TITLE">

      Using jsp:plugin</TABLE>
<P>
<CENTER>
<jsp:plugin type="applet"
            code="coreservlets.ShadowedTextApplet.class"
            width="475" height="350">
  <jsp:params>
    <jsp:param name="MESSAGE" value="Your Message Here" />
  </jsp:params>
</jsp:plugin>
</CENTER>

</BODY>
</HTML>
```

Listing 12.9 HTML resulting from `ShadowedTextApplet.jsp`.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
<HEAD>
<TITLE>Using jsp:plugin</TITLE>
<LINK REL=STYLESHEET
      HREF="JSP-Styles.css"
      TYPE="text/css">
</HEAD>

<BODY>

<TABLE BORDER=5 ALIGN="CENTER">
  <TR><TH CLASS="TITLE">
      Using jsp:plugin</TABLE>
<P>
<CENTER>
<OBJECT classid="clsid:8AD9C840-044E-11D1-B3E9-00805F499D93"
        width="475"  height="350"
        codebase="http://java.sun.com/products/plugin/1.2.2/jin-
stall-1_2_2-win.cab#Version=1,2,2,0">
<PARAM name="java_code"
       value="coreservlets.ShadowedTextApplet.class">
<PARAM name="type" value="application/x-java-applet;">
<PARAM name="MESSAGE" value="Your Message Here">
<COMMENT>
<EMBED type="application/x-java-applet;"
       width="475"  height="350"
       pluginspage="http://java.sun.com/products/plugin/"
       java_code="coreservlets.ShadowedTextApplet.class"

       MESSAGE="Your Message Here" >
<NOEMBED>
</COMMENT>
</NOEMBED></EMBED>
</OBJECT>

</CENTER>

</BODY>
</HTML>
```

Listing 12.10 ShadowedTextApplet.java

```
package coreservlets;

import java.awt.*;
import javax.swing.*;
import java.awt.event.*;

/** Applet interface to the ShadowedTextFrame
 *  class. Requires Swing and Java 2D.
 */

public class ShadowedTextApplet extends JApplet
                                implements ActionListener {
  private JTextField messageField;
  private JComboBox fontBox;
  private JSlider fontSizeSlider;
  private JButton showFrameButton;

  public void init() {
    WindowUtilities.setNativeLookAndFeel();
    Color bgColor = new Color(0xFD, 0xF5, 0xE6);
    Font font = new Font("Serif", Font.PLAIN, 16);
    Container contentPane = getContentPane();
    contentPane.setLayout(new GridLayout(4, 1));
    contentPane.setBackground(bgColor);

    // Use a JTextField to gather the text for the message.
    // If the MESSAGE parameter is in the HTML,
    // use it as the initial value of this text field.
    messageField = new JTextField(20);
    String message = getParameter("MESSAGE");
    if (message != null) {
      messageField.setText(message);

    }
    JPanel messagePanel =
      new LabelPanel("Message:", "Message to Display",
                     bgColor, font, messageField);
    contentPane.add(messagePanel);

    // Use a JComboBox to let users choose any
    // font available on their system.
    GraphicsEnvironment env =
      GraphicsEnvironment.getLocalGraphicsEnvironment();
    String[] fontNames = env.getAvailableFontFamilyNames();
```

Listing 12.10 ShadowedTextApplet.java (continued)

```
    fontBox = new JComboBox(fontNames);
    fontBox.setEditable(false);
    JPanel fontPanel =
      new LabelPanel("Font:", "Font to Use",
                     bgColor, font, fontBox);
    contentPane.add(fontPanel);

    // Use a JSlider to select the font size.
    fontSizeSlider = new JSlider(0, 150);
    fontSizeSlider.setBackground(bgColor);
    fontSizeSlider.setMajorTickSpacing(50);
    fontSizeSlider.setMinorTickSpacing(25);
    fontSizeSlider.setPaintTicks(true);
    fontSizeSlider.setPaintLabels(true);
    JPanel fontSizePanel =
      new LabelPanel("Font Size:", "Font Size to Use",
                     bgColor, font, fontSizeSlider);
    contentPane.add(fontSizePanel);

    // Pressing the button will open the frame
    // that shows the shadowed text.
    showFrameButton = new JButton("Open Frame");
    showFrameButton.addActionListener(this);
    JPanel buttonPanel =
      new LabelPanel("Show Shadowed Text:",
                     "Open JFrame to Show Shadowed Text",
                     bgColor, font, showFrameButton);
    contentPane.add(buttonPanel);
  }

  public void actionPerformed(ActionEvent event) {
    String message = messageField.getText();
    if (message.length() == 0) {
      message = "No Message";
    }

    String fontName = (String)fontBox.getSelectedItem();
    int fontSize = fontSizeSlider.getValue();
    JFrame frame = new JFrame("Shadowed Text");
    JPanel panel =
      new ShadowedTextFrame(message, fontName, fontSize);
    frame.setContentPane(panel);
    frame.pack();
    frame.setVisible(true);
  }
}
```

Listing 12.11 LabelPanel.java

```
package coreservlets;

import java.awt.*;
import javax.swing.*;

/** A small JPanel that includes a JLabel to the left
 *  of a designated component. Also puts a titled border
 *  around the panel.
 */

public class LabelPanel extends JPanel {
  public LabelPanel(String labelMessage, String title,
                    Color bgColor, Font font,
                    JComponent component) {
    setBackground(bgColor);
    setFont(font);
    setBorder(BorderFactory.createTitledBorder(title));
    JLabel label = new JLabel(labelMessage);
    label.setFont(font);
    add(label);
    component.setFont(font);
    add(component);
  }
}
```

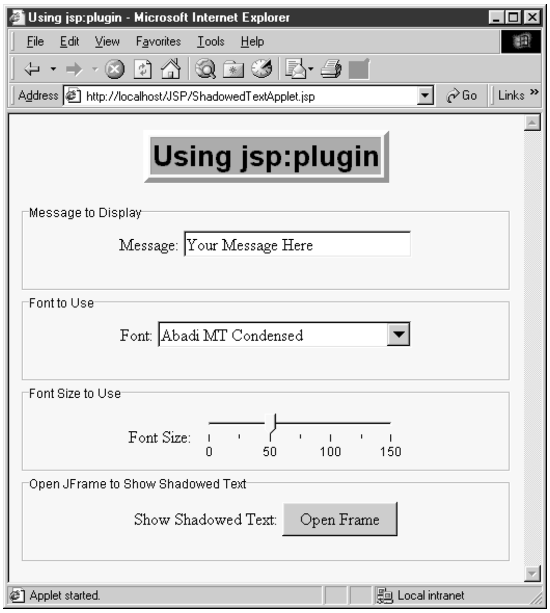

Using jsp:plugin - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://localhost/JSP/ShadowedTextApplet.jsp
Go
Links
Using jsp:plugin
Message to Display
Message: Your Message Here
Font to Use
Font: Abadi MT Condensed
Font Size to Use
Font Size:
0
50
100
150
Open JFrame to Show Shadowed Text
Show Shadowed Text:
Open Frame
Applet started.
Local intranet

Figure 12–3 Initial result of `ShadowedTextApplet.jsp` in a browser that has the JDK 1.2 plug-in installed.

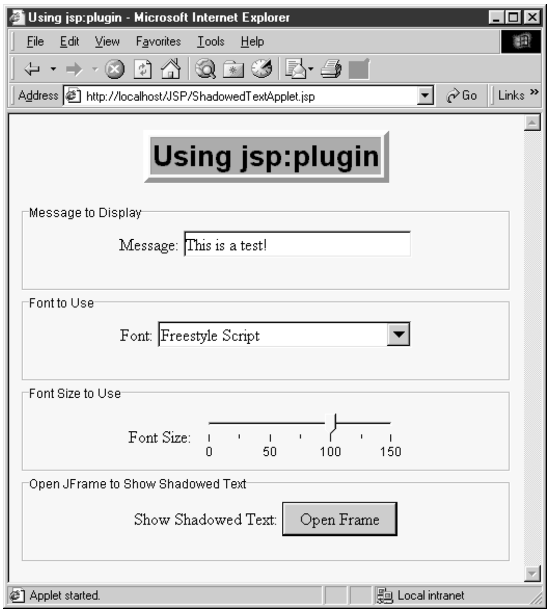


Figure 12–4 `ShadowedTextApplet.jsp` after changing the message, font, and size entries.

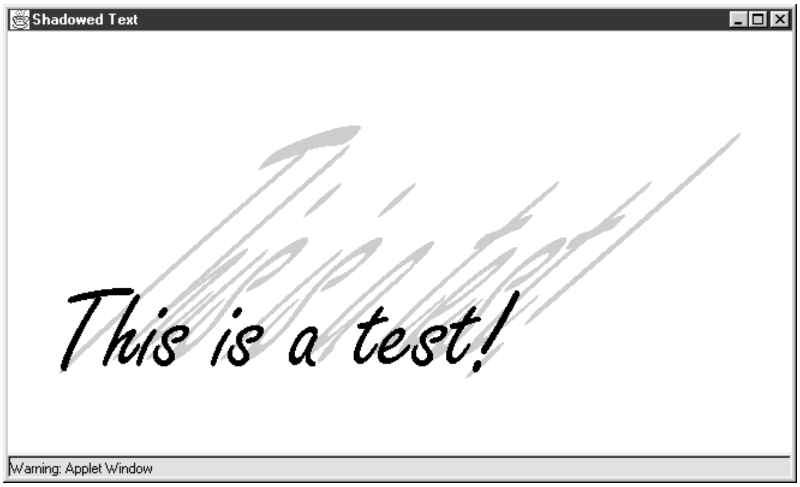


Figure 12–5 Result of pressing the "Open Frame" button in Figure 12–4.

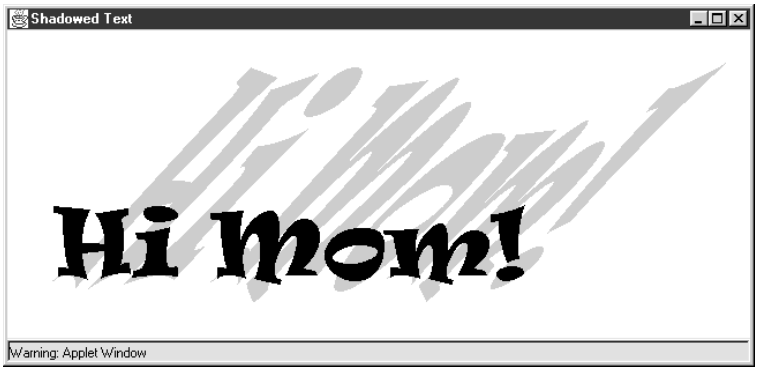


Figure 12–6 Another possible frame built by `ShadowedTextApplet.jsp`.