BEA WebLogic Server™

Using Applets with WebLogic Server

Version 8.1
Revised: June 28, 2002

Copyright

Restricted Rights Legend

Trademarks or Service Marks

Contents

About This Document

Using Applets with WebLogic Server

About This Document

This document describes how to uses applets with WebLogic Server. The document is organized as follows:

- Chapter 1, "Using Applets with WebLogic Server."

What You Need to Know

This document is written forapplication developers who are interested in building Web applications. It is assumed that readers know applets and Java programming.

e-docs Web Site

BEA product documentation is available on the BEA corporate Web site. From the BEA Home page, click on Product Documentation or go directly to the WebLogic Server Product Documentation page at http://e-docs.bea.com/wls/docs60.

How to Print the Document

You can print a copy of this document from a Web browser, one main topic at a time, by using the File—Print option on your Web browser.

A PDF version of this document is available on the WebLogic Server documentation Home page on the e-docs Web site (and also on the documentation CD). You can open the PDF in Adobe Acrobat Reader and print the entire document (or a portion of it) in book format. To access the

PDFs, open the WebLogic Server documentation Home page, click Download Documentation, and select the document you want to print.

Adobe Acrobat Reader is available at no charge from the Adobe Web site at http://www.adobe.com.

Contact Us!

Your feedback on BEA documentation is important to us. Send us e-mail at docsupport@bea.com if you have questions or comments. Your comments will be reviewed directly by the BEA professionals who create and update the documentation.

In your e-mail message, please indicate the software name and version your are using, as well as the title and document date of your documentation. If you have any questions about this version of BEA WebLogic Server, or if you have problems installing and running BEA WebLogic Server, contact BEA Customer Support through BEA WebSupport at http://www.bea.com. You can also contact Customer Support by using the contact information provided on the Customer Support Card, which is included in the product package.

When contacting Customer Support, be prepared to provide the following information:

- Your name, e-mail address, phone number, and fax number
- Your company name and company address
- Your machine type and authorization codes
- The name and version of the product you are using
- A description of the problem and the content of pertinent error messages

Documentation Conventions

The following documentation conventions are used throughout this document.

Convention	Usage
Ctrl+Tab	Keys you press simultaneously.
italics	Emphasis and book titles.

Convention	Usage

`monospace text`	Code samples, commands and their options, Java classes, data types, directories, and file names and their extensions. Monospace text also indicates text that you enter from the keyboard.
	Examples:
	`import java.util.Enumeration;`
	`chmod u+w *`
	`config/examples/applications`
	`.java`
	`config.xml`
	`float`

`monospace italic text`	Variables in code. *Example*: `String CustomerName;`
UPPERCASE TEXT	Device names, environment variables, and logical operators. *Example*s: LPT1 BEA_HOME OR
`{ }`	A set of choices in a syntax line.
`[ ]`	Optional items in a syntax line. *Example*: `java utils.MulticastTest -n name -a address [-p portnumber] [-t timeout] [-s send]`
|	Separates mutually exclusive choices in a syntax line. *Example*: `java weblogic.deploy [list|deploy|undeploy|update] password {application} {source}`

Convention	Usage
. . .	Indicates one of the following in a command line: • An argument can be repeated several times in the command line. • The statement omits additional optional arguments. • You can enter additional parameters, values, or other information
. . .	Indicates the omission of items from a code example or from a syntax line.

CHAPTER 1

Using Applets with WebLogic Server

Introduction

BEA supports the use of applets in limited cases and this document provides other options to help you evaluate your use of applets. For those of you who are working with systems that utilize applets with WebLogic Server outside of our recommended best practices, links are provided to the Sun site for your support.

How Applets Work

This section provides a brief overview of applet functionality. For more information, read about Applets on Sun's Java Web site.

Applets are usually embedded within an HTML page using an `<APPLET>` tag such as:

```
<APPLET CODE="HelloWorld.class"
        CODEBASE="/bea_wls_internal/classes/" WIDTH=150 HEIGHT=25>
</APPLET>
```

When a Web browser requests the HTML page containing the <APPLET> tag, it attempts to find the main applet class referenced by the CODE attribute. The Web browser requests the class from the URL specified by the CODEBASE attribute. Any other classes that the applet uses are requested from the URL specified by CODEBASE.

You should be careful when testing your applet, that the Web browser classpath does not contain any of your applet classes. If the browser cannot retrieve the requested class from the HTTP server, it looks in its local classpath. This may fool you into thinking that you have configured you applet deployment correctly since it works on your local host machine. Yet, when someone attempts to use the applet from a remote client it will fail if you have not deployed all of the required applet classes on your Web server.

When to Use Applets

BEA supports the use of server-side applications with HTTP servlets and Java Server Pages (JSPs) as part of the J2EE platform. We recommend that before you develop new applications, consider using either servlets or JSPs. A well-designed series of interactive web pages using servlets and Java Server Pages (JSPs) generally yield a faster and more reliable website. If you are currently using applets, you may find that most can be converted to Java applications using Java Web Start and you can continue using WebLogic Server. For information, go to Sun's Java Web Start site at http://java.sun.com/products/javawebstart.

You may use applets as part of your distributed application running on WebLogic to provide a more interactive client-side interface in a web browser. In the case of graphics that require that information to be updated over time, applets are a best practice. Applets provide the advantage of running safe client-side code without the need to distribute software.

Applets may be used to extend the functionality of a page, such as in a stateless, click/respond type of application (i.e., a navigation bar or console), or in a polling application (i.e., a stock ticker).

WebLogic Server Web Presentations—Best Practices

The following table provides some recommended BEA best practices for Web presentations when using WebLogic Server.

Table 1-1

If you want to. . .	Advantage/Disadvantage of Applets	Recommend . . .
Display data over time	Can update graphs, tables and charts that display current information Update frames on a web page. For example the left frame containing the hierarchal information in the Administration Console	Applets.
Connections to a database	Restrictions imposed by the Web browser JVMs and maintenance costs for compatibility.	Servlets and JSPs
Manage threads	Thread `ContextClassloader` limitations may create exceptions in the applets.	Servlets and JSPs
Rich GUI	Applets may be the best option for the rich GUI, but be aware that different browers handle applets differently.	Applications with Java WebStart

For information on which browsers and plug-ins have been tested with applets and WebLogic Server, see Browser Support for Applets on the Platform Support page.

Using Applets—Known Limitations

This section discusses the disadvantages of using applets as well as two known limitations with using applets.

Disadvantages of Using Applets

The following is a list of the disadvantages of using applets:

- An applet cannot handle the functionality of a large application.

- Applets generally do not allow for caching which results in classes being downloaded each time an applet is run.
- The caching that is available using the Java Plug-in does not version, so any updates to the applet are ignored.
- Different implementations between browsers and browsers' versions result in poor applet performance.
- Poor performance results when opening two windows that are both running applets and are both on the same page.

Accessing EJBs from Applets

If you use an applet in a Web Application and the applet accesses an EJB deployed with another application, you must include the EJB stubs as part of the Web application. To do this, generate stubs using the `-disableHotCodeGen` option to `ejbc`, and package the stubs as part of the Web Application.

If the stubs are not included as part of the Web Application, the applet will generate a `ClassNotFoundException` when it attempts to access the EJB.

ClassNotFoundException

The ContextClassloaders for the event handling threads are different from the ContextClassloader of the the thread that runs the lifecycle methods. Only the ContextClassloader that runs the lifecycle methods has information about the classes that it loaded from the codebase.

If you try to get the `initialContext` in any thread other than the thread that runs applet lifecycle methods (`init`, `start`, `stop`, `destroy`), a `ClassNotFoundException` may be thrown under the following circumstances:

- when trying to get the `initialContext` in the `actionPerformed()` method of your applet (which implements ActionListener)
- if there is a `getInitialContext` method in an applet and this method is called from the JSP, i.e. `document.`*`AppletName`*`.getInitialContext()`

ClassCastException

When the Weblogic Server client in the applet tries to get some resouce information from the classloader and the cache tags and `codebase=/bea_wls_internal/classes` tag are used together, a `ClassCastException` may be thrown.

To avoid this problem:

- Do not use cache options such as `cache_option` and `cache_archive` while using the classpath servlet as codebase.
- Do not keep the `/classes` (`ClasspathServlet`) as codebase, while using the cache options. To do this, initially package the client side JAR file using the archiver utility.

For more information about this limitation, see http://developer.java.sun.com/developer/bugParade/bugs/4648591.html.

Using the Java Plug-in

BEA recommends that you always use the Java Plug-in for your applets.

Sun provides a browser plug-in that allows applets to run within the standard Java Runtime Environment, instead of the browser's default virtual machine. This ensures consistency in any browser that supports the plug-in, which means compatibility and reliability for your applets. The plug-in also allows you to conveniently determine which JRE is used on the client machine.

The Java Plug-in provides essential compatibility for your applets if they need to communicate with the WebLogic Server. In most cases, the version of the Java virtual machine (JVM) on the client must match the version of the JVM on the server. That is, if the server is running under Java 1.3, then you must use the Java 1.3 Plug-in.

You can find more details at Sun's Java plug-in homepage at http://www.java.sun.com/products/plugin/index.html. The Java Plug-in is a native plug-in for the Internet Explorer or Netscape browsers. When a user first hits a page that requires the plug-in, a message directs them to Sun's Web site to download it. The plug-in need only be downloaded once. The plug-in runs the applet on a stable release of a specific JRE from Sun, yet can run inside the browser like a regular applet.

The plug-in can be tricky to embed into your HTML pages since Internet Explorer and Netscape each use a different syntax. At Sun's Web site are instructions on how to hack both syntax formats into the same HTML file, or you can download the HTML-converter, which will automatically convert your existing `<APPLET>` tags. As you will see, the workaround is quite ingenious, but contrived and difficult to maintain. BEA recommends you evaluate JavaServer pages as a better solution.

In JSP, you use the `<jsp:plugin>` tag to include an applet into a JSP generated web page. The generated servlet detects the type of client Web browser, and sends the appropriate plug-in tags in the response. For more details see Programming WebLogic JSP at http://e-docs.bea.com/wls/docs81/jsp/index.html.

The applet JVM requirements are the same as the stand-alone client JVM requirements. If the stand-alone client must be run on the 1.3 JVM for WebLogic 6.1 server, the applet client should be run on the 1.3 plug-in.

After your applet has been converted to a plug-in-aware applet, it should look like the following example:

```
<HTML>
<HEAD><TITLE>Title of Applet page</TITLE></HEAD>
<BODY>
<OBJECT
CLASSID="clsid:8AD9C840-044E-11D1-B3E9-00805F499D93"
WIDTH = 600
HEIGHT = 350
CODEBASE="http://java.sun.com/products/plugin/1.3/jinstall-13-win
32.cab#Version=1,3,0,0">
<PARAM NAME = CODE VALUE = "Applet1.class">
<PARAM NAME = CODEBASE VALUE =
"/bea_wls_internal/classes/DefaultWebApp@DefaultWebApp/">
<PARAM NAME = ARCHIVE VALUE = "weblogic.jar">
<PARAM NAME="type" VALUE="application/x-java-applet;version=1.3">
<PARAM NAME="scriptable" VALUE="false">
<COMMENT>
<EMBED type="application/x-java-applet;version=1.3"
CODE = "Applet1.class"
CODEBASE = "/bea_wls_internal/classes/DefaultWebApp@DefaultWebApp/"
ARCHIVE = "weblogic.jar"
WIDTH = 600
HEIGHT = 350
scriptable=false
pluginspage="http://java.sun.com/products/plugin/1.3/plugin-insta
ll.html";>
<NOEMBED>
</COMMENT>
alt="Your browser understands the &lt;APPLET&gt; tag but isn't running the
applet, for some reason."
Your browser is completely ignoring the &lt;APPLET&gt; tag!
</NOEMBED>
</EMBED>
```

```
</OBJECT>
</BODY>
</HTML>
```

The CODEBASE Attribute

You use the CODEBASE attribute in an <APPLET> tag to specify a URL where the browser should look for the applet's Java class files. Without the CODEBASE tag, a Web browser will look for the required classes under the same directory as the HTML file containing the <APPLET> tag. Using CODEBASE is convenient, since it allows you to organize your class files under a single directory, separate from the HTML content of your site.

Applets written to work with the WebLogic Server often require WebLogic classes, so it is good practice to use the CODEBASE attribute to allow the browser to load the required classes out of the WebLogic installation. WebLogic automatically provides a special servlet, mapped to /classes, that serves classes from the WebLogic Server's classpath. The servlet is registered by default under the virtual servlet name "classes". When you set CODEBASE to a URL such as:

```
CODEBASE="http://www.weblogic.com/bea_wls_internal/classes/"
```

or

```
CODEBASE="/bea_wls_internal/classes/"
```

the WebLogic Server invokes the servlet, which searches for the requested classes under the WebLogic Server's classpath.

For information on Serving Resources from the CLASSPATH with the ClassPath Servlet see Configuring Web Application Components in *Assembling and Configuring Web Applications*.

If the CODEBASE=/bea_wls_internal/classes/, the classes required by the applet should be available in the system classpath.

If the CODEBASE=/bea_wls_internal/classes/DefaultWebApp@DefaultWebApp, the classes required by the applet should be in the applications/DefaultWebApp/WEB-INF/classes directory or in the system classpath.

The CODE Attribute

Your <APPLET> tag must contain the CODE attribute, which specifies the full package name of the main applet class file. The extension ".class" at the end of the CODE is optional. For example, if you are working with the GraphApplet, your <APPLET> tag looks like this:

```
<APPLET CODE="GraphApplet"
```

```
CODEBASE="/bea_wls_internal/classes/appName@componentName" >
```

where appName is the name of your application and componentName is the name of your WebApp.

For more information on the <APPLET> tag and CODEBASE, see JavaSoft's Overview of Applets at http://java.sun.com/docs/books/tutorial/applet/overview/index.html in the Java Tutorial.

Troubleshooting and Performance

The following topics cover troubleshooting and performance issues.

Troubleshooting Applets

Here are some scenarios you may run into using applets:

Applet Does Not Run in Browser

I'm using WebLogic JDBC in an applet to retrieve data from a DBMS. If I run the class using the Sun Appletviewer on my local machine, I have no problems. But when I try to run the applet in a Netscape browser, it will not connect.

If your applet works in Appletviewer but not in a browser, it is an indication that you are violating a Netscape security restriction; in this case, the violation is that an applet cannot open a socket to a machine other than the one from which it loaded the applet. To solve this problem, you will have to serve your applet code from the same machine that hosts the DBMS.

Note: The IP naming format you use in the applet CODEBASE and the URL you use to make a connection to the WebLogic Server must match exactly. You can't use dot-notation format in one place and domain name format in the other.

ClassFormatError

If you are getting a ClassFormatError, it probably indicates that there is a problem with your HTTP server configuration. It could be that you haven't put the WebLogic or applet classes in the correct directory on the HTTP server, or that you are specifying the CODEBASE or the CODE incorrectly in your APPLET tag. Here are two examples:

You might have packed the applet in a webapp, *MyWar*. If this webapp is part of the application, *MyEar*, the code base should be:

```
CODEBASE=http://host:port/bea_wls_internal/classes/MyEar@MyWar/
```

or

```
CODEBASE=/bea_wls_internal/classes/MyEar@MyWar/
```

This will download all the classes and resource files from the *MyWar* webapp. Keep all the resource files (JPG files , JAR files, etc.) in the `WebApplicationRoot` of the specific webapp which, in this case, is the root directory of *MyWar*.

If you want to test the codebase in the applet having `CODE=com.myapp.MyApplet`, you can frame a URL such as `http://server:host/CODEBASEvalue/com/myapp/MyApplet.class` and try this from the browser window. You should get a download window for this class. Otherwise you need to fix the configuration with the webapp in the server.

For more information, see Programming WebLogic HTTP Servlets at http://e-docs.bea.com/wls/docs81/servlet/index.html.

Testing in Your Local Environment

If you are running the WebLogic Server and Netscape Communicator 4.x on the same host, you will need to remove the `CLASSPATH` from the environment of the shell that is running Communicator. For security reasons, Netscape Communicator will not load classes from your local `CLASSPATH` in order to prevent malicious changes to standard classes. Removing the local `CLASSPATH` when running the browser causes Netscape to load the classes from the WebLogic Server's `CLASSPATH`.

You will still need to set a `CLASSPATH` in the shell in which you start WebLogic. WebLogic recommends that you never set `CLASSPATH` in your environment, but rather set `CLASSPATH` appropriately in the shell from which you run WebLogic.

We recommened testing the prototyped application on an applet before developing the entire application. We recommend testing for problems that cannot be resolved on the WebLogic Server side because they are the result of a dependency on the applet plug-in. This testing is recommened for:

- Applications that need to use any secure protocols inside the applets.
- Applets that have a special design with RMI callback objects.
- Applets that use JMS inside.

Moving From Your Local Development Environment

When you move the applet from your local environment, you will need to make sure that you install the WebLogic classes and the applet class in the proper location on the webserver.

If you are running the applet on the same machine that you installed the WebLogic distribution, this may obscure problems you are having with CODEBASE. The applet will first look for the WebLogic classes in your local CLASSPATH. If you haven't properly installed the classes for serving applets from the HTTP server, the applet will default to your local CLASSPATH and work, obscuring the problem. To test your HTTP configuration properly, you need to temporarily rename the WebLogic classes in your local CLASSPATH or try your applet from another machine.

Speeding Up Applets With Archives

WebLogic has utilities that scan an HTML server log and create a zip file of classes for an applet to speed up file downloading. An even faster alternative is to avoid JDBC in the applet when possible, and obtain DBMS data in HTML form from servlets instead. Servlets can run queries for the applet, and/or retrieve data from Workspaces and supply it as HTML. In concert with WebLogic processes which asynchronously maintain this data, the performance of your applications will improve.

If your applet must download many files before it can run, you can speed this up by using the ARCHIVE parameter inside the APPLET tag on your HTML page. A common problem with multifile applets is that the browser must make a separate HTTP connection for each file used in an applet. Making a connection can take up to several seconds, sometimes longer than downloading the file itself. With the ARCHIVE parameter you can combine these classes into a .jar file (or .cab file, for Microsoft Internet Explorer), which can then be downloaded in a single HTTP connection. Since JAR files can be compressed (CAB files are always compressed), this will also improve download time.

Note: The procedure when using Appletviewer, Netscape Navigator (3.0 and later only), and the HotJava browser differs from that used for Microsoft Internet Explorer (4.0 and later only). For complete compatibility, both methods may be combined.