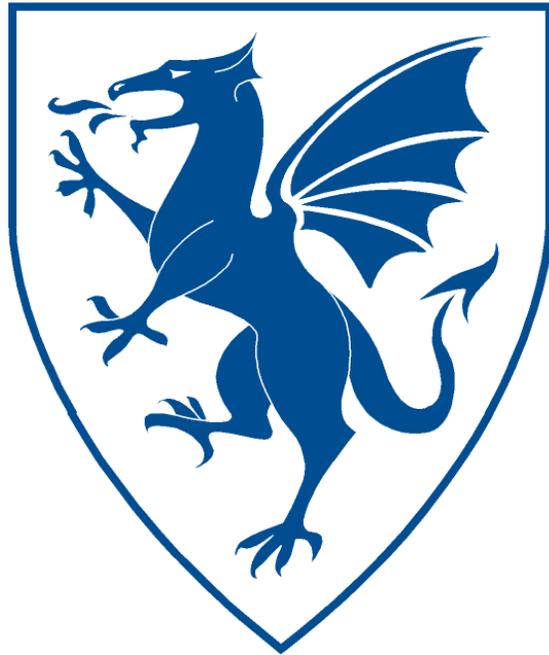


BlueDragon



BlueDragon™ 7.1

Deploying CFML on Java EE

Application Servers

NEW ATLANTA COMMUNICATIONS, LLC

BlueDragon™ 7.1 Deploying CFML on Java EE Application Servers

May 5, 2009
Version 7.1



Copyright © 1997-2009 New Atlanta Communications, LLC. All rights reserved.
100 Prospect Place • Alpharetta, Georgia 30005-5445
Phone 678.256.3011 • Fax 678.256.3012
<http://www.newatlanta.com>

BlueDragon is a trademark of New Atlanta Communications, LLC. ServletExec and JTurbo are registered trademarks of New Atlanta Communications, LLC in the United States. Java and Java-based marks are trademarks of Sun Microsystems, Inc. in the United States and other countries. ColdFusion is a registered trademark of Adobe System, Inc. in the United States and/or other countries, and its use in this document does not imply the sponsorship, affiliation, or endorsement of Adobe Systems, Inc. All other trademarks and registered trademarks herein are the property of their respective owners.

This product includes software developed by the Apache Software Foundation (<http://www.apache.org>).

No part of this document may be reproduced, transmitted, transcribed, stored in a retrieval system, or translated into any language or computer language, in any form or by any means, electronic, mechanical, magnetic, optical, chemical, manual, or otherwise without the prior written consent of New Atlanta Communications, LLC.

New Atlanta Communications, LLC makes no representations or warranties with respect to the contents of this document and specifically disclaims any implied warranties of merchantability or fitness for any particular purpose. Further, New Atlanta Communications, LLC reserves the right to revise this document and to make changes from time to time in its content without being obligated to notify any person of such revisions or changes.

The Software described in this document is furnished under a Software License Agreement (“SLA”). The Software may be used or copied only in accordance with the terms of the SLA. It is against the law to copy the Software on tape, disk, or any other medium for any purpose other than that described in the SLA.

1	INTRODUCTION	1
1.1	About CFML.....	1
1.2	Deploying CFML on Java EE Servers with BlueDragon.....	1
1.3	BlueDragon for Java EE Licensing.....	2
1.4	About BlueDragon.....	2
1.5	System Requirements.....	3
1.6	Technical Support	3
1.7	Additional Documentation.....	3
2	JAVA EE WEB APPLICATIONS	4
2.1	Java EE webapp Directory Structure.....	4
2.2	URL Context Path	5
2.2.1	Choosing an Alternative Context Path	5
2.3	web.xml Deployment Descriptor	6
2.4	WAR Files	6
3	DEPLOYING CFML AS A JAVA EE WEB APPLICATION	7
3.1	Creating and Deploying a CFML webapp	7
3.2	Creating WAR Files	8
3.3	Options to Consider Before Production Deployment	9
3.3.1	Setting the BlueDragon Working Directory	9
3.3.2	Setting the BlueDragon Administration Console Password	10
3.3.3	Removing the BlueDragon Administration Console	10
3.3.4	Preventing the Auto-load of ODBC DataSources on Windows Servers	10
3.3.5	Configuring Datasources in the Java EE Server.....	11
3.3.6	Deploying CFML Without Source Code.....	11
4	RELATIVE URLS WITHIN HTML TAGS.....	12
5	PROTECTING CFINCLUDE/CFMODULE TEMPLATES.....	13

1 Introduction

BlueDragon for Java Enterprise Edition (Java EE) allows web applications developed using ColdFusion® Markup Language (CFML) to be deployed as standard Java EE web applications in either open directories or as web application archive (WAR) files. While most web applications on Java EE servers are built using servlets, JSPs, EJBs, and other components of the Java EE specifications, BlueDragon makes it possible to deploy CFML applications on Java EE servers as native Java EE components, and to integrate CFML and native Java EE components.

This document offers a brief overview of Java EE web applications, explaining both the benefits of CFML (for Java EE developers) and the benefits of Java EE deployment (for CFML developers). It then describes how to create standard Java EE web application components that include the BlueDragon CFML runtime. CFML pages are then added to these web applications, which can be deployed onto any standard Java EE application server *without requiring the installation of proprietary Adobe ColdFusion Server software*.

1.1 About CFML

CFML is a popular server-side scripting language for building dynamic database-driven web sites. Unlike alternatives such as ASP or PHP, CFML is based primarily on HTML-like markup tags (though CFML also contains a scripting language component). CFML is characterized by its low learning curve and ease-of-use, particularly for web developers who do not have a technical background in programming languages such as C/C++ or Java. CFML was originally developed by Allaire Corporation in the late 1990's; Allaire was acquired by Macromedia, Inc. in early 2001, which in turn was acquired by Adobe Systems Inc. in late 2005.

Over the past several years, many organizations have begun adopting standards-based application servers for their Internet and intranet web site deployments. In particular, there has been a significant migration to application servers based on the Java Enterprise Edition (Java EE) standards defined by Sun Microsystems, Inc. and its partners. This standardization on Java EE servers creates a problem for organizations that have legacy applications implemented in CFML: prior to the introduction of BlueDragon these applications could only be deployed on proprietary Adobe ColdFusion application servers.

1.2 Deploying CFML on Java EE Servers with BlueDragon

BlueDragon allows existing CFML applications to be redeployed as standard Java EE components (WAR or EAR files) onto standard Java EE application servers, eliminating the need for proprietary Adobe ColdFusion servers, and without requiring a lengthy and expensive rewrite of the CFML into JSP. The redeployed legacy CFML applications can then be enhanced using standard Java EE technologies (servlets, JSP, EJB, etc.), or web developers can continue to enjoy the productivity and ease-of-use of CFML, but in a standard Java EE environment.

A general discussion of the motivation for and benefits of deploying CFML on Java EE is offered in a New Atlanta-authored article published in the April 2004 ColdFusion Developers Journal, "Making the Case for CFML on J2EE":

<http://coldfusion.sys-con.com/node/44481>

There are many benefits to redeploying CFML applications to Java EE servers, including (but not limited to):

- possibilities of clustering, load-balancing, and fail-over as provided by the Java EE server for all web applications, including options such as session replication across multiple servers;
- options of persisting sessions across server restarts;
- deploying multiple independent instances (where each independent deployed web application is isolated from others—even on the same machine—with its own administrative settings, JVM configuration, and more);
- management controls provided by Java EE servers to stop, start, redeploy and otherwise configure and manage web applications;
- reporting mechanisms provided by Java EE servers to track requests, sessions, and more;
- option to use Java EE datasources, providing enhanced pooling, clustering, configuration, management, and more.

Finally, it's possible to integrate CFML with JSPs, Java objects, EJBs, and other Java EE components, including sharing session and application variables, as well as including back and forth between CFML pages and JSPs/servlets. Because this functionality is supported also in our BlueDragon Server JX edition, the topic of integrating CFML with Java is covered in the *BlueDragon 7.1 User Guide*, and is not discussed further in this document

1.3 BlueDragon for Java EE Licensing

The BlueDragon Java EE web application (discussed in section 3, below) by default has no BlueDragon license key embedded within it, so that it runs as a single-user (localhost-only) developer mode. You may request a 30-day license key to extend it to be accessible from any client, or you may purchase a commercial license key. Contact sales@newatlanta.com for further information. The license key can be entered in the BlueDragon Administration console or in the `bluedragon.xml` configuration file found in the web application's `WEB-INF/bluedragon` directory.

1.4 About BlueDragon

The core technology of BlueDragon is a CFML runtime and execution environment that is implemented as a standard Java EE servlet. Building web applications that include the BlueDragon CFML runtime allows the deployment of CFML pages onto standard Java EE servers without installing proprietary Adobe ColdFusion server software. BlueDragon is highly compatible with Adobe's ColdFusion MX 7.0.2, with some limitations and some enhancements. See the BlueDragon CFML Compatibility Guide for details:

http://www.newatlanta.com/products/bluedragon/self_help/docs/index.cfm

BlueDragon is a highly optimized, high-performance CFML runtime engine. CFML pages are compiled into an internal representation that is cached in memory and executed by the BlueDragon runtime when CFML pages are requested by client browsers.

In addition to allowing the deployment of CFML pages onto standard Java EE servers, BlueDragon is packaged in a standalone server configuration built on New Atlanta's award-winning ServletExec web application server. Additional information about BlueDragon and ServletExec can be found on New Atlanta's web site:

<http://www.newatlanta.com/products/index.jsp>

1.5 System Requirements

BlueDragon for Java EE allows deployment of CFML applications using standard web applications as defined by the Java Servlet 2.3 specification. Therefore, BlueDragon runs on any Java EE server or JSP/servlet engine that supports standard web applications or web application archive (WAR) files. BlueDragon 7.1 has been tested and is supported on the following Java EE application servers:

Java EE Application Server
BEA WebLogic 9.2 and 10.0
IBM WebSphere 6.1
JBoss 4.0.x, 4.2.x, and 5.0

BlueDragon for Java EE will run on any operating system supported by these application servers, including 64-bit operating systems. BlueDragon for Java EE 7.1 requires JDK 1.5 (Java 5) and is also supported on JDK 1.6 (Java 6).

1.6 Technical Support

If you're having difficulty installing or using BlueDragon, visit the self-help section of the New Atlanta web site for assistance:

http://www.newatlanta.com/products/bluedragon/self_help/index.cfm

Details regarding paid support options, including online-, telephone-, and pager-based support are available from the New Atlanta web site:

<http://www.newatlanta.com/biz/support/index.jsp>

1.7 Additional Documentation

The other relevant manuals available in the BlueDragon documentation library are:

- *BlueDragon 7.1 CFML Compatibility Guide*
- *BlueDragon 7.1 CFML Enhancements Guide*
- *BlueDragon 7.1 User Guide*

Each of these documents offers useful information that may be relevant to developers, installers, and administrators. They are available in PDF format from New Atlanta's web site:

http://www.newatlanta.com/products/bluedragon/self_help/docs/index.cfm

2 Java EE Web Applications

This section is an introduction to Java EE web applications for developers who are new to this subject. Experienced Java EE developers may want to treat this as a refresher, or skip to the next section.

In Java EE terminology, a web application (or *webapp*) is a collection of Java servlets, JavaServer Pages (JSP), JSP tag libraries, Java classes, HTML documents, GIF/JPEG images, style sheets, and other resources. A Java EE webapp can be deployed as a single component onto any Java EE application server that implements the Java Servlet API 2.3 (or later) specification.

A Java EE webapp is characterized by a specific directory structure, and a configuration file named `web.xml` that is also referred to as the webapp *deployment descriptor*. The following book—freely available as an online PDF document—contains an excellent in-depth discussion of Java EE web applications:

More Servlets and JavaServer Pages

by Marty Hall

Sun Microsystems Press / Prentice-Hall PTR, 2002

ISBN 0-13-067614-4

<http://www.moreservlets.com>

Detailed Java EE reference information is also available on Sun's web site:

<http://java.sun.com/javaee/technologies/webapps/>

2.1 Java EE webapp Directory Structure

Content that is to be served to the client is placed directly in the top-level directory of a Java EE webapp. A webapp may contain sub-directories within the top-level directory; for example, it may contain an `images` sub-directory to hold GIF and JPEG files.

Within the webapp top-level directory is a special sub-directory named `WEB-INF`. The Java EE server will not serve any content from the `WEB-INF` sub-directory to the client; therefore, this is the place to put configuration files, Java class files, or other resources that need to be protected. The `web.xml` deployment descriptor is placed directly within the `WEB-INF` sub-directory (see further discussion of `web.xml`, below).

There are two special sub-directories within the `WEB-INF` directory: the `classes` sub-directory that is used to hold unbundled Java `.class` files, and the `lib` sub-directory that is used to hold Java `.jar` archives. Any Java classes placed in these sub-directories are automatically available to the webapp (that is, they don't need to be added to the Java EE server's classpath or otherwise configured in any way). This is the location to put files to be called via `CFOBJECT`, `createObject`, and Java CFX tags.

In summary, the key features of a Java EE webapp are:

-
- Content that is to be served to the client (HTML, GIF, JPEG, CFM, JSP, etc.) is placed directly within the webapp top-level directory or its sub-directories.
 - The `WEB-INF` sub-directory is located within the webapp top-level directory. The Java EE server will not serve any files from `WEB-INF` to the client.
 - The `web.xml` deployment descriptor is located directly within the `WEB-INF` directory.
 - The `classes` and `lib` sub-directories within `WEB-INF` are used to store Java `.class` and `.jar` files, respectively.

2.2 URL Context Path

When deploying a webapp onto a Java EE server, a URL prefix (referred to as the *context path* or context root) is associated with the webapp. This deployment process is referred to as *registering* the webapp and the specific procedures vary based on the Java EE server. After the webapp is registered with the Java EE server, all URLs that match the context path are mapped to the webapp for processing.

The top-level directory serves as the *document root* for the webapp. When an incoming URL matches (starts with) the context path of a webapp, the portion of the URL after the context path is interpreted as being relative to the webapp top-level directory. In this way, the context path acts as a sort of virtual directory that maps to the webapp physical directory.

For example, if the file `index.html` resides within the webapp top-level directory and the webapp has been configured with a context path of `/mywebapp`, the following URL will serve `index.html` from the webapp:

```
http://www.myserver.com/mywebapp/index.html
```

The following URL would serve `logo.gif` from within the webapp `images` sub-directory:

```
http://www.myserver.com/mywebapp/images/logo.gif
```

2.2.1 Choosing an Alternative Context Path

Most Java EE servers will use the web application directory name or WAR file name as the name of the context path. If you need to deploy the application to a different context path, particularly if you want to deploy it as the root (`/`) context path, you must configure the context path during deployment.

Most Java EE servers permit designation of the context path during deployment, whether using the Java EE server's administration console or command-line tool. Additionally, most also permit indication of the context path using an XML file entry (typically in a file name and using XML entries specific to each Java EE server).

In BEA WebLogic, you must create (or edit) a `weblogic.xml` file in the web application's `WEB-INF` directory, placing in it the following lines:

```
<weblogic-web-app>
  <context-root>/bd71</context-root>
</weblogic-web-app>
```

For information on setting the context path in other Java EE servers, see the appropriate documentation for that server.

2.3 *web.xml* Deployment Descriptor

The `web.xml` deployment descriptor contains configuration information used by the Java EE server to support the webapp. For example, the `web.xml` file provided with the BlueDragon web application template (see below) contains configuration information that tells the Java EE server how to process CFML files (specifically, it instructs the Java EE server to forward all URLs that end with the `.cfm` extension to the BlueDragon CFML runtime servlet).

A detailed discussion of `web.xml` is beyond the scope of this document. However, the `web.xml` file provided with the BlueDragon web application template (see below) contains all of the configuration information needed to deploy CFML pages; this `web.xml` file will normally not be modified in any way.

2.4 *WAR* Files

A webapp can be deployed unbundled in an open (or *exploded*) directory structure, or bundled into a Web ARchive (WAR) file. A WAR file is simply a webapp directory structure bundled into a ZIP file and given the “.war” extension. WAR files can be created using the JDK `jar` utility or any utility that can create a ZIP file, such as WinZip on Windows or `gzip` on UNIX. See section 3.2 for more information on creating WAR files.

3 Deploying CFML as a Java EE Web Application

When you download BlueDragon for Java EE Servers, you'll find it creates a directory of files including a single war file named `BlueDragon71.war`, and a directory of files named `BlueDragon_webapp_71`. You can use either of these to deploy BlueDragon and your CFML onto any standard Java EE server or servlet engine. This section explains how to do that.

The `BlueDragon_webapp_71` directory contains a pre-built standard Java EE web application that includes the BlueDragon CFML runtime servlet, and contains all of the Java archives (`.jar` files) and configuration files needed to support CFML pages. Using `BlueDragon_webapp_71` as a starting point, here is a high-level overview of the steps needed to create a standard Java EE webapp that supports CFML pages:

1. Make a copy of the `BlueDragon_webapp_71` directory.
2. Add CFML pages and other content files (HTML, GIF, JPEG, JSP, etc.) to the new webapp directory created in Step 1.
3. Deploy the webapp to your Java EE server.
4. Use the BlueDragon administration console to configure any datasources required by the CFML pages.
5. After testing, deploy the webapp to a production server either as an open directory or packaged within a WAR file (See section 3.2 for more information on creating WAR files.)

In addition to the BlueDragon CFML runtime servlet, `BlueDragon_webapp_71` contains JDBC drivers for Microsoft SQL Server, Oracle, and PostgreSQL databases; JDBC drivers for additional databases can be added by placing their JAR files into the `/WEB-INF/lib` directory or adding them to the Java EE server classpath. The JDBC-ODBC Bridge included with the standard Java runtime environment can be used to access ODBC datasources.

The `BlueDragon71.war` file is offered as a compressed version of the `BlueDragon_webapp_71` directory. While you can deploy it instead, the following section focuses on using the open directory, and a later section discusses creating your own WAR file from your deployed CFML/Java EE web application, for subsequent deployment on other servers.

3.1 Creating and Deploying a CFML webapp

Follow these detailed step-by-step instructions to create a standard Java EE web application that supports CFML pages:

1. If you have not already done so, download the installer file which contains the `BlueDragon_webapp_71` directory:

<http://www.newatlanta.com/products/bluedragon/download.jsp>

2. Make a copy of the `BlueDragon_webapp_71` directory (you could simply rename the `BlueDragon_webapp_71` directory, but you'll probably want to use it as a template to

create additional webapps in the future). Note that many Java EE servers automatically use the name of the webapp top-level directory as the URL context path (see above for a discussion of URL context paths).

3. Copy your CFML and other content files (HTML, GIF, JPEG, JSP, etc.) into the webapp top-level directory, creating sub-directories as needed, either before deploying the web application or afterward, depending on your environment and requirements.
4. Deploy the webapp to your Java EE server. On some servers, you can simply copy the webapp or WAR file to a particular directory (hot deployment), but on most, you use the application server's administration console or a command line utility to deploy the web application. See section 3.2 for more information on creating WAR files.
5. Test the webapp by serving the files `index.jsp` and `index.cfm`. For example, if you configured a context path of `"/mywebapp"` use the following URLs (replace `"www.myserver.com"` with the host name or IP address of your computer):

<http://www.myserver.com/mywebapp/index.jsp>

<http://www.myserver.com/mywebapp/index.cfm>

After verifying that your webapp is registered properly, you can delete the files `index.jsp` and `index.cfm`.

6. Access the BlueDragon administration console via the following URL (assuming a context path of `"/mywebapp"`; replace `"www.myserver.com"` with the host name or IP address of your computer):

<http://www.myserver.com/mywebapp/bluedragon/admin.cfm>

The password to the BlueDragon administration console is blank by default, so you are not required to enter a password to login. You can set the password after logging in.

The main task that needs to be performed via the BlueDragon administration console is to configure any datasources required by your CFML pages. See the BlueDragon User Guide or built-in help files for further information about using the BlueDragon administration console.

7. Test your webapp on the development machine.
8. Deploy your webapp onto the production server either as an open web app directory or by packaging the webapp into a WAR file, as desired or as may be required by the production server. See step 4 above, as well as the remaining sections in this chapter, for more information on web application deployment.

3.2 Creating WAR Files

As explained previously, you can deploy your Java EE web applications either as an open directory or as a WAR file. Some Java EE administrators may prefer to deploy code as WAR files,

and indeed some Java EE servers may offer additional benefits when code is deployed as a WAR file.

You can create the WAR file manually using the JDK `jar` utility (which is installed by most application servers) or by using a tool that can create a zip file (such as WinZip on Windows or `gzip` on UNIX).

The details of using each is beyond the scope of this manual, but the bottom line is that in either approach you simply want to create a compressed file comprising all the files and directories of the open web application you've created.

With a tool like WinZip, be careful about how you perform the zip operation. Select all the files and directories in the web app directory at once and zip that (rather than selecting the directory itself and zipping that). Also, do not choose the option to "save full path info". The resulting zip file contents should appear to contain the files (and subdirectory names) just as it did in the open web application, with the path to `web.xml` (for instance) being simply `WEB-INF` rather than `yourappname\WEB-INF`. Finally, once the zip file is created, simply rename it to a WAR file (or pay attention to name it a WAR file during the zip processing).

Once you're familiar with the process, it can become a two-step operation.

3.3 Options to Consider Before Production Deployment

Before deploying your web application into production, consider the following options which may be required or desirable.

3.3.1 Setting the BlueDragon Working Directory

BlueDragon needs to read and write files used by the CFML runtime to a working directory. By default, BlueDragon is configured to use `/WEB-INF/bluedragon/work` within the webapp as its working directory. Usually, there is no need to modify this default setting.

If you wish to modify the BlueDragon working directory location, the BlueDragon working directory is configured via the `BLUEDRAGON_WORKING_DIRECTORY` `init-param` in `web.xml`. The location to store the `bluedragon.xml` configuration file (holding admin console configuration changes) is also specified by an `init-param`, `BLUEDRAGON_XML`, which can also be changed.

If you prefer to control the working directory without modifying each web application's `web.xml` file, note that the following also applies:

- If the `BLUEDRAGON_WORKING_DIRECTORY` `init-param` is not defined within `web.xml`, then BlueDragon uses the `javax.servlet.context.tempdir` attribute provided by the Java EE server to determine the location of its working directory; that is, it creates a sub-directory named "working" within this directory.
- If the `<tempdirectory>` element is not defined within `bluedragon.xml`, then BD creates the "temp" directory as a sub-directory of the working directory.

-
- It's also possible to define the working directory via the `-Dbuedragon.workdir` JVM property

The Java Servlet 2.4 specification (paragraph 3.7.1) explains the temporary directory behavior in more detail. So to let the Java EE server control where the working directory is placed (by its designation of the temporary directory for the web application), simply delete the `BLUEDRAGON_WORKING_DIRECTORY` init-param from `web.xml` and the `<tempdirectory>` element from `bluedragon.xml`.

You can determine the value for `javax.servlet.context.tempdir` by running the following within a JSP page:

```
<%=application.getAttribute( "javax.servlet.context.tempdir" )%>
```

3.3.2 Setting the BlueDragon Administration Console Password

By default, there is no password set for the BlueDragon administration console in the web application made available in the original download. Before deploying the web application to production, you will want to set a password, if not completely remove the administration console as discussed in the next section. You can set the password within the Administration console itself, under `General->License & Security`, or you can set it in the `bluedragon.xml` file.

3.3.3 Removing the BlueDragon Administration Console

Before deploying your web application into production, you can remove the code supporting the BlueDragon administration console, if you prefer for security reasons to prevent access in production.

To remove it, simply delete the following from the web application:

- the `[webapp]\bluedragon\admin\` directory
- the `[webapp]\WEB-INF\bluedragon\admin.bda` file

If you delete these files after a web application is deployed, you must either redeploy or restart the web application for the change to take effect.

3.3.4 Preventing the Auto-load of ODBC DataSources on Windows Servers

On Windows Servers, BlueDragon is configured by default to automatically load any existing ODBC datasources as defined on the server in the Windows ODBC Datasources panel (available in Control Panel or Administrative tools). See the *BlueDragon 7.1 User Guide* for more information on configuring and using ODBC Datasources.

While this feature of auto-loading ODBC datasources is generally considered a productivity enhancement, it may be desirable in some situations to prevent the process. To do so, simply delete the `[webapp]\WEB-INF\bin\ODBCNativeLib.dll` file.

If you delete this file after a web application is deployed, you must either redeploy or restart the web application for the change to take effect.

3.3.5 Configuring Datasources in the Java EE Server

When using BlueDragon for Java EE, you can either use the BlueDragon administration console's datasource configuration capability, or you can instead use the datasource configuration capability provided by your Java EE application server. In that case, you do not need to configure the datasource in the BlueDragon administration console.

Once configured, you can use the datasource name (technically the JNDI name, in Java EE terms) in tags like `CFQUERY`, etc. BlueDragon will search first to see if there is a datasource of that name in its configuration, and if none is found, it will then look for the JNDI name in the Java EE environment.

Defining datasources in the Java EE environment may give you additional benefits, such as deployment, clustering, replication, connection pooling and other beneficial features that are provided by the Java EE server. Consult your Java EE server documentation.

3.3.6 Deploying CFML Without Source Code

As discussed in the *BlueDragon 7.1 User Guide*, BlueDragon offers an option for deploying your CFML applications in such a way that the source code is not readable (and optionally not executable beyond a given date, or without a decryption key). Called “precompiled templates”, they can be created using the BlueDragon administration console option `Deploy->Precompile`. For more information, see the section *Precompiled, Encrypted CFML Templates* in the *User Guide*.

4 Relative URLs within HTML Tags

The use of URL context paths by webapps may result in the need for special handling of relative URLs within HTML tags (this is not an issue that is specific to BlueDragon, but is common to JSP and static HTML pages within a Java EE webapp). Relative URLs within HTML tags that don't start with the slash character (/) work properly when deployed within a webapp and do not require any modification. For example, the following HTML tags work the same in both webapp and non-webapp deployments:

```
<A HREF="file2.html">
<FORM ACTION="store/processOrder.cfm">
<IMG SRC="../images/logo.gif">
```

However, relative URLs that start with the slash character (/), do not work properly within webapps. This is because the browser interprets these URLs as being relative to the web server document root (that is, relative to the URL "/"), and the browser strips the URL context path when converting these relative URLs to an absolute URL.

BlueDragon provides two CFML enhancements to allow you to specify URLs that are relative to the webapp root directory (the webapp top-level directory) without using URLs that start with "/". The first method is to use the `CGI.Context_Path` variable when constructing relative URLs. For example:

```
<CFOUTPUT>
<IMG SRC="#CGI.Context_Path#/images/logo.gif">
<CFOUTPUT>
```

The above example is equivalent to the following tag in a non-webapp deployment:

```
<IMG SRC="/images/logo.gif">
```

If you have a large number of relative URLs within a CFML page, then you can use the `CFBASE` tag to set the "base" URL for all relative URLs within the page. For example:

```
<HTML>
<HEAD>
<CFBASE TARGET="optional">
<TITLE>Relative URL Test Page</TITLE>
</HEAD>
<BODY>
<IMG SRC="images/logo.gif">
</BODY>
</HTML>
```

In the above example, "images/logo.gif" is interpreted by the browser as being relative to the webapp top-level directory (that is, "images" is a sub-directory within the webapp top-level directory). When using the `CFBASE` tag, all relative URLs within the page are interpreted as being relative to the webapp top-level directory, and not relative to the current page as they would be without `CFBASE`. The `CFBASE` tag is expanded by BlueDragon into an HTML `BASE` tag with the `HREF` attribute set to the webapp top-level directory. The optional `TARGET` attribute to `CFBASE` has the same function as the `BASE` tag optional `TARGET` attribute.

5 Protecting CFINCLUDE/CFMODULE Templates

The issue of template paths also applies to `CFINCLUDE/CFMODULE` template path mappings. Template paths that start with `"/` are mapped by BlueDragon to the Java EE webapp document root directory. It's worth noting that these paths can be mapped to the `WEB-INF` directory, for example:

```
<cfinclude template="/WEB-INF/includes/header.cfm">
<cfmodule template="/WEB-INF/modules/navbar.cfm">
```

The advantage of using `WEB-INF` is that files within it are never served directly by the Java EE server, and therefore cannot be accessed by users via a URL.