BlueDragon™ 6.1
Deploying CFML on J2EE Servers
BlueDragon™ 6.1 Deploying CFML on J2EE Application Servers

June 22, 2004
Version 6.1

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 Macromedia, Inc. in the United States and/or other countries, and its use in this document does not imply the sponsorship, affiliation, or endorsement of Macromedia, 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.
# Table of Contents

1. **Introduction** ............................................................................................................................. 4  
   1.1 About CFML ............................................................................................................................ 4  
   1.2 Deploying CFML on J2EE Servers with BlueDragon ............................................................ 4  
   1.3 About BlueDragon ................................................................................................................. 5  
   1.4 System Requirements ............................................................................................................ 6  
   1.5 Technical Support .................................................................................................................. 6  
   1.6 Additional Documentation ..................................................................................................... 7  

2. **J2EE Web Applications** .......................................................................................................... 8  
   2.1 Webapp Directory Structure ................................................................................................. 8  
   2.2 URL Context Path ................................................................................................................. 9  
   2.3 web.xml Deployment Descriptor ......................................................................................... 9  
   2.4 WAR Files ............................................................................................................................ 9  

3. **BlueDragon webapp Template** ............................................................................................. 10  
   3.1 Creating and Deploying a CFML webapp ......................................................................... 10  
   3.2 Creating WAR Files............................................................................................................ 11  
      3.2.1 Creating WAR Files Manually ............................................................................. 11  
      3.2.2 Creating WAR Files with the WAR Deployment Wizard ................................ 12  
      3.2.3 Setting the BlueDragon Working Directory for WAR Files ............................ 13  

4. **Relative URLs within HTML Tags** ...................................................................................... 14  

5. **Protecting CFINCLUDE/CFMODULE Templates** .............................................................. 15  

Appendix A. bluedragon.xml........................................................................................................ 16  
Appendix B. BEA WebLogic 7.0, 7.0.1, and 8.1 ......................................................................... 17  
Appendix C. IBM WebSphere 4.0 and 5.0 ................................................................................. 21  
Appendix D. Sun ONE Application Server 7 ............................................................................. 26  
Appendix E. Oracle 9i Application Server ................................................................................. 29  
Appendix F. JBoss 3.0.4 ................................................................................................................. 30  
Appendix G. New Atlanta ServletExec 4.1 and 4.2 ................................................................. 31  
Appendix H. Macromedia JRun 4 ............................................................................................... 35  
Appendix I. Apache Tomcat 4.0 and 4.1 ................................................................................. 39  
Appendix H. Borland Enterprise Server 5.2.1 ......................................................................... 40
1 Introduction

BlueDragon for J2EE Servers (BlueDragon/J2EE) allows CFML applications to be deployed as standard J2EE web application directories or web application archive (WAR) files. While most web applications on J2EE servers are built with servlets, JSPs, EJBs, and other components of the J2EE specification, BlueDragon makes it possible to deploy CFML applications on J2EE servers as native J2EE components, or to integrate CFML and native J2EE components.

This document offers a brief overview of J2EE web applications, explaining both the benefits of CFML (for J2EE developers) and the benefits of J2EE deployment (for CFML developers). It then describes how to create standard J2EE 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 J2EE application server without requiring the installation of proprietary Allaire/Macromedia ColdFusion Server software.

The appendices of this document describe how to deploy web applications onto J2EE application servers from leading vendors, including BEA WebLogic, IBM WebSphere, Sun ONE Application Server, Oracle 9iAS, Borland Enterprise Server, JBoss, New Atlanta ServletExec, Macromedia JRun, and Apache Tomcat. The examples offered should also assist in deploying BlueDragon applications on other J2EE servers and JSP/servlet engines.

1.1 About CFML

ColdFusion Markup Language (CFML) is a popular server-side markup language for building dynamic database-driven web sites. Unlike scripting-based alternatives such as ASP or PHP, CFML is based primarily on HTML-like markup tags (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.

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 2 Enterprise Edition (J2EE) standard defined by Sun Microsystems, Inc. and its partners. This standardization on J2EE 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 Allaire/Macromedia ColdFusion application servers.

1.2 Deploying CFML on J2EE Servers with BlueDragon

BlueDragon allows existing CFML applications to be redeployed as standard J2EE components (WAR or EAR files) onto standard J2EE application servers, eliminating the need for proprietary Allaire/Macromedia 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 J2EE technologies (servlets, JSP, EJB, etc.), or web developers can continue to enjoy the productivity and ease-of-use of CFML, but in a standard J2EE environment.
A general discussion of the motivation for and benefits of deploying CFML on J2EE is offered in a New Atlanta-written article published in the April 2004 ColdFusion Developers Journal, “Making the Case for CFML on J2EE”:

http://www.sys-con.com/story/?storyid=44481&DE=1

There are many benefits to redeploying CFML applications to J2EE servers, including (but not limited to):

- possibilities of clustering, load-balancing, and fail-over as provided by the J2EE 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 J2EE servers to stop, start, redeploy and otherwise configure and manage web applications;
- reporting mechanisms provided by J2EE servers to track requests, sessions, and more;
- option to use J2EE datasources, providing enhanced pooling, clustering, configuration, management, and more.

1.3 About BlueDragon

The core technology of BlueDragon is a CFML runtime and execution module that is implemented as a standard J2EE servlet. Building web applications that include the BlueDragon CFML runtime allows the deployment of CFML pages onto standard J2EE servers without installing proprietary Allaire/Macromedia ColdFusion server software. BlueDragon is highly compatible with Macromedia’s ColdFusion MX Server, with some limitations and some enhancements. See the BlueDragon CFML Compatibility Guide for details:


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 J2EE 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
1.4 System Requirements
BlueDragon/J2EE allows deployment of CFML applications using standard web applications as defined by the Java Servlet 2.3 specification. Therefore, BlueDragon/J2EE runs on any J2EE server or JSP/servlet engine that supports standard web applications or web application archive (WAR) files; it has been tested to run on the following, which are also described later in this document:

<table>
<thead>
<tr>
<th>J2EE Application Server</th>
</tr>
</thead>
<tbody>
<tr>
<td>BEA WebLogic 7.0 and 8.1</td>
</tr>
<tr>
<td>Borland Enterprise Server 5.2</td>
</tr>
<tr>
<td>IBM WebSphere 4.0, 5.0, and 5.0.1</td>
</tr>
<tr>
<td>JBoss 3.0.4</td>
</tr>
<tr>
<td>Sun ONE Application Server 7</td>
</tr>
<tr>
<td>Macromedia JRun 4.0</td>
</tr>
<tr>
<td>Oracle 9iAS J2EE Application Server</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>JSP/Servlet Engines (non-EJB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>New Atlanta ServletExec 4.2 and 5.0</td>
</tr>
<tr>
<td>Apache Tomcat 4.1 and 5.0</td>
</tr>
</tbody>
</table>

BlueDragon/J2EE should run on any operating system supported by these application servers; New Atlanta has tested and supports the following operating systems:

- Windows NT/2000/XP/2003
- Solaris 2.6, 7, 8, or 9
- Red Hat Linux 6.2 and higher
- AIX 4.3.3 and 4.3.10
- HP-UX 10.20 and 11.0

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


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
1.6 Additional Documentation

The other manuals available in the BlueDragon documentation library are:

- *BlueDragon 6.1 Installation Guide*
- *BlueDragon 6.1 CFML Compatibility Guide*
- *BlueDragon 6.1 User Guide*

The Installation Guide is primarily focused on the BlueDragon Server products, but each of the others offers useful information that may be relevant to developers, installers, and administrators. They are available in PDF format from New Atlanta’s web site:

```
```
2 J2EE Web Applications

This section is an introduction to J2EE web applications for developers who are new to this subject. Experienced J2EE developers may want treat this as a refresher, or skip to the next section, “BlueDragon Webapp Template”.

In J2EE 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 J2EE webapp can be deployed as a single component onto any J2EE application server that implements the Java Servlet API 2.3 (or later) specification.

A J2EE 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 contains an excellent in-depth discussion of J2EE 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

2.1 Webapp Directory Structure

Content that is to be served to the client is placed directly in the top-level directory of a 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 J2EE 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 J2EE server’s classpath or otherwise configured in any way).

In summary, the key features of a J2EE 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 within sub-directories of the top-level directory.
- The WEB-INF sub-directory is located within the webapp top-level directory. The J2EE 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 J2EE server, a URL prefix (referred to as the context path) is associated with the webapp. This deployment process is referred to as registering the webapp and the specific procedures vary based on the J2EE server (see the Appendices of this document for details). After the webapp is registered with the J2EE 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

2.3 web.xml Deployment Descriptor

The web.xml deployment descriptor contains configuration information used by the J2EE 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 J2EE server how to process CFML files (specifically, it instructs the J2EE 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 any utility that can create a ZIP file, such as WinZip on Windows, gzip on UNIX, or the JDK jar utility. The BlueDragon administration console includes a wizard for creating WAR files. See Section 3.2 for more information on creating WAR files.
3 BlueDragon webapp Template

The BlueDragon_webapp_61 directory contains a pre-built standard J2EE 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_61 as a starting point, here is a high-level overview of the steps needed to create a standard J2EE webapp that supports CFML pages:

1. Make a copy of the BlueDragon_webapp_61 directory.
2. Add CFML pages and other content files (HTML, GIF, JPEG, JSP, etc.) to the new webapp directory created in Step 1.
3. Use the BlueDragon administration console or manually edit bluedragon.xml to configure any datasources required by the CFML pages.
4. After testing, deploy the webapp to a production server either as an open directory or packaged within a WAR file (use the WAR Deployment Wizard in the BlueDragon administration console to create the WAR file, or simply use WinZip to compress the directory).

In addition to the BlueDragon CFML runtime servlet, BlueDragon_webapp_61 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 J2EE server classpath. The JDBC-ODBC Bridge included with the standard Java runtime environment can be used to access ODBC datasources.

3.1 Creating and Deploying a CFML webapp

Follow these detailed step-by-step instructions to create a standard J2EE web application that supports CFML pages:

1. If you have not already done so, download the installer file which contains the BlueDragon_webapp_61 directory:
   
   http://www.newatlanta.com/products/bluedragon/download.jsp

2. Make a copy of the BlueDragon_webapp_61 directory (you could simply rename the BlueDragon_webapp_61 directory, but you’ll probably want to use it as a template to create additional webapps in the future). Note that many J2EE 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. Register the webapp with your J2EE server (see the appendices of this document; for development, register the webapp as an “open directory” and not as a WAR file).

4. 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`.

5. 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 just click the “Login” button without entering a password. 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. See the Appendix A of this document for configuring datasources by manually editing the `bluedragon.xml` configuration file.

6. Copy your CFML pages and other content files (HTML, GIF, JPEG, JSP, etc.) to the webapp top-level directory, creating sub-directories as needed.

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 Section 3.2 for more information on creating WAR files. On some servers, you can simply copy the web app 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 app. Refer to the Appendices B-I of this document for instructions for deploying a webapp as either an open directory or as a WAR file on J2EE application servers from several leading vendors.

3.2 Creating WAR Files

As explained previously, you can deploy your J2EE web applications either as an open directory or as a WAR file. Some J2EE administrators may prefer to deploy code as WAR files, and indeed some J2EE servers may offer additional benefits when code is deployed as a WAR file.

You have two choices for creating WAR files. You can use the WAR Deployment Wizard of the BlueDragon administration console, or you can create the WAR file manually using a tool that can create a zip file (such as WinZip on Windows, gzip on UNIX, or the JDK jar utility).

Creating a WAR file manually with a zip tool may sometimes be the fastest approach, yet using the administration console wizard affords greater control in what BlueDragon elements are added to the WAR file. Consider the information in the following subsections about using each approach.

3.2.1 Creating WAR Files Manually

The primary benefit of creating a WAR file manually is that you can do it quickly, without need to launch the Deployment Wizard.
The drawbacks to creating WAR files manually are that:

- You have no automated control of what features (drivers, etc.) can be left out when not needed, to reduce the size of the file, as when using the Deployment Wizard

- You have no automated process of setting the BlueDragon working directory (see “Setting the BlueDragon Working Directory for WAR Files” later in this section for more information)

- The code supporting the administration console is included in the WAR file, unless you remove its related files yourself

For those interested in creating a WAR file manually, the steps depend on whether you’re using a tool such as (such as WinZip on Windows, gzip on UNIX, or the JDK jar utility). 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 sure not to just zip the single directory containing the webapp (which will create a zip file where each file’s path starts with the name of the directory containing the web application). Instead, select all the files and directories in the web app at once and zip that. 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 one-step operation, which makes it appealing over the WAR Deployment Wizard (subject to the drawbacks listed above.)

### 3.2.2 Creating WAR Files with the WAR Deployment Wizard

In contrast to the process of creating a WAR file manually, the benefits of using the WAR Deployment Wizard are that:

- You can control what features (drivers, etc.) are to be included in the WAR file, thus enabling you to reduce the size of the WAR file by leaving out unneeded features

- You are offered an option to override the BlueDragon working directory (see “Setting the BlueDragon Working Directory for WAR Files” later in this section for more information)

- The code supporting the administration console is not included in the WAR file

The WAR Deployment Wizard is available in the BlueDragon administration console, which is provided in the skeletal open web application described in the opening of Section 3. It’s offered as the last link in the left navigational toolbar, labeled “Build WAR File”. The fields in the WAR Deployment Wizard are fairly self-explanatory, but for additional detail, see the available online help.
3.2.3 Setting the BlueDragon Working Directory for WAR Files
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. For webapps deployed as open directories there is no need to modify this
default setting.

For webapps deployed as WAR files it is often not possible or desirable to write runtime files to
the webapp directory (or the WEB-INF sub-directory). Therefore, when deploying a webapp as a
WAR file, configure BlueDragon to use a working directory outside of the webapp directory. We
recommend using the following directories for Windows and UNIX/Linux, respectively:

C:\BlueDragon\<webapp name>
/usr/local/BlueDragon/<webapp name>

Each BlueDragon webapp must be configured to use a unique working directory. The
BlueDragon working directory is configured via the BLUEDRAGON_WORKING_DIRECTORY
parameter in web.xml. The easiest way to configure this parameter is to use the WAR
Deployment Wizard in the BlueDragon administration console to create the WAR file.

Most J2EE servers allow you to edit web.xml via their administration consoles; therefore, you
can edit the working directory parameter after deploying your WAR file.
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 J2EE 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.cfml">
<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 J2EE webapp document root directory. It’s worth noting that these paths can be mapped to the WEB-INF directory, for example:

``` CFML
<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 J2EE server, and therefore can be accessed by users via a URL.
Appendix A. bluedragon.xml

The bluedragon.xml file contains configuration information used by the BlueDragon CFML runtime engine. The most important information in bluedragon.xml is the configuration of datasources used within CFML pages. The bluedragon.xml file can be found within the /WEB-INF/bluedragon directory of the BlueDragon_webapp_61 application template.

You will normally not edit bluedragon.xml manually, but will instead use the BlueDragon administration console to modify configuration settings. However, if you wish to edit bluedragon.xml to manually configure datasources, refer to the datasource-examples.xml file. This file contains commented sample datasources configured for Microsoft SQL Server, Oracle, MySQL, and the JDBC-ODBC Bridge. Cut-and-paste these examples into bluedragon.xml and modify the values as appropriate.
Appendix B. BEA WebLogic 7.0, 7.0.1, and 8.1
This appendix describes the deployment of a webapp onto BEA WebLogic 7.0 or 7.0.1, either as an open directory or as a WAR file; instructions for BEA WebLogic 8.1 are nearly identical. These instructions are provided for convenience only; refer to the BEA WebLogic documentation for detailed instructions for deploying web applications.

1. Launch the BEA WebLogic 7.0 administration console and access the Web Applications page as illustrated below:

![Web Applications Console](image)

2. Click the Configure a new Web Application... link and navigate to the directory that contains your webapp. Click the [select] link next to the directory or WAR file that contains your webapp, as illustrated below:

![Select Directory](image)
3. Select the Target Servers on which to deploy the webapp, then click the **Configure and Deploy** button, as illustrated below:

4. When the webapp is deployed successfully, the administration console page will appear similar to the following, with a **Deployed** state of *true*:
5. If you’ve deployed your webapp as a WAR file and need to examine or modify the BLUEDRAGON_WORKING_DIRECTORY parameter in web.xml, you can do so by clicking the Edit Web Application Deployment Descriptors... link as illustrated below (you must redeploy your webapp after modifying this parameter):

6. The webapp is now successfully deployed; this can be demonstrated by accessing the URL http://localhost:7001/mywebapp/index.cfm, which produces a page similar to the following:
Hello! Welcome to BlueDragon from a CFML custom tag.

Hello! Welcome to BlueDragon from a Java CFX tag.

Hello again from CFX_TestTag!

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Exists</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>DoesNotExist</td>
<td>No</td>
<td></td>
</tr>
</tbody>
</table>

No QUERY parameter found for tag.
Appendix C. IBM WebSphere 4.0 and 5.0

This appendix describes the deployment of a webapp onto IBM WebSphere 4.0; deployment on IBM WebSphere 5.0 is very similar. The example webapp is located in a WAR file at \C:\webapps\mywebapp.war. These instructions are provided for convenience only; refer to the IBM WebSphere 4.0 documentation for detailed instructions for deploying web applications.

1. Launch the IBM WebSphere 4.0 administration console, access the Enterprise Applications page, and click the Install button as illustrated below:

![IBM WebSphere Administration Console](image)

2. Enter the Path to the WAR file, the Application Name, and the Context Root for the webapp, then click the Next button:
3. Specify the *Virtual Host Name* on which to deploy the webapp, and whether or not to *Precompile JSPs*, then click the *Next* button (the decision of whether or not to precompile JSPs has no effect on BlueDragon or processing of CFML pages within the webapp):

4. Confirm the webapp configuration options, then click the *Finish* button:
BlueDragon 6.1: Deploying CFML on J2EE Servers
5. After configuring the webapp, it will be necessary to save the server configuration, and it may be necessary to regenerate the web server plug-in. Click the appropriate links as illustrated below:

![WebSphere Application Server](image)

6. Start the webapp by selecting it via the checkbox and clicking the *Start* button:
7. When the webapp is started, the administration console will appear similar to:

8. The webapp is now successfully deployed; this can be demonstrated by accessing the URL http://localhost/mywebapp/index.cfm, which produces a page similar to the following:
Appendix D. Sun ONE Application Server 7

This appendix describes the deployment of a webapp onto Sun ONE Application Server 7. The example webapp is located in a WAR file at `C:\webapps\mywebapp.war`. These instructions are provided for convenience only; refer to the Sun ONE documentation for detailed instructions for deploying web applications.

1. Launch the Sun ONE Application Server 7 administration console, access the `Applications: Web` page, then click the `Deploy...` button as illustrated below:

2. From the `Upload WAR File` page, enter the full physical path to your webapp WAR file, or click the `Browse` button to locate the WAR file on the file system. Then click the `OK` button:
3. From the Applications:Web:Deploy page, modify the web application deployment options as needed, then click the OK button:

4. When deployment of the webapp is completed successfully, the Status will show as Enabled in the administration console:
5. Access the URL http://localhost/mywebapp/index.cfm to confirm successful deployment of your web application.
Appendix E. Oracle 9i Application Server

This appendix describes how to deploy J2EE web applications on Oracle 9i Application Server. These instructions are provided for convenience only; refer to the Oracle 9iAS documentation for detailed instructions for deploying web applications.

Manual Deployment

To manually deploy a web application on Oracle 9iAS, perform the following steps:

1. Edit the `application.xml` configuration file and add a `<web-module>` element within `<orion-application>` element. For example:

   ```xml
   <!-- The global application config that is the parent of all the other applications in this server. -->
   <orion-application autocreate-tables="true"
                    default-data-source="jdbc/OracleDS">
       
       <web-module id="defaultWebApp" path="../../home/default-web-app" />
       <web-module id="dms0" path="../../home/applications/dms0.war" />
       <web-module id="dms" path="../../home/applications/dms.war" />
       
       <!-- BlueDragon webapp -->
       <web-module id="mywebapp" path="C:\webapps\mywebapp" />
       
    </orion-application>
   
   The `path` attribute of the `<web-module>` element is a full or relative file system path to either a directory or a WAR file.

2. Edit the `http-web-site.xml` configuration file and add a `<web-app>` element within the `<web-site>` element for the web site on which you wish to deploy the webapp. For example:

   ```xml
   <web-site port="80"
             display-name="Oracle9iAS Containers for J2EE HTTP Web Site">
       
       <default-web-app application="default" name="defaultWebApp" />
       <web-app application="default" name="dms0" root="/dms0" />
       <web-app application="default" name="dms" root="/dmsoc4j" />
       
       <!-- BlueDragon webapp -->
       <web-app application="default" name="mywebapp" root="/mywebapp" />
       
    </web-site>
   
   The `root` attribute of the `<web-app>` element specifies the URL context path for the webapp.

You must restart Oracle 9iAS for changes to `application.xml` and `http-web-site.xml` to take effect.
Appendix F. JBoss 3.0.4

This appendix describes how to deploy J2EE web applications on JBoss 3.0.4; these instructions apply to versions of JBoss with either the Jetty or Tomcat web containers. These instructions are provided for convenience only; refer to the JBoss documentation for detailed instructions for deploying web applications.

For instructions on working with Tomcat standalone (as opposed to embedded within JBoss), see the Tomcat appendix below.

Web applications may be deployed on JBoss 3.0.4 by placing them in the “deploy” subdirectory of the desired server configuration directory. To deploy a web application as an open directory, the directory name must end with “.war”.

For example, to deploy a web application for the “default” server configuration, place the WAR file or web application directory in the following location:

```
<Jboss install>\server\default\deploy
```

Restart JBoss to complete the deployment of your web application.

Beware of a potential conflict because of the JBoss requirement to name open web applications as “.war” files. If you use the BlueDragon administration console to build a new WAR file based on the web application, it will default to naming the new WAR file the same name as the web application in the same directory where the web application is installed. Because there will already be a directory of the name <webapp>.war, the attempt to create a new war file of the same name will fail. Either specify a different name while building the WAR file, or indicate that it should be built in a different directory than the current one.
Appendix G. New Atlanta ServletExec 4.1 and 4.2

This appendix describes how to deploy J2EE web applications on New Atlanta ServletExec 4.1 and 4.2, a high-performance servlet/JSP container that can be used to add servlet/JSP support to Microsoft IIS, NES/iWS, and Apache web servers. These instructions are provided for convenience only; refer to the ServletExec documentation for detailed instructions for deploying web applications.

Auto-deployment of WAR Files

ServletExec 4.1 allows you to auto-deploy WAR files by simply copying them to the ServletExec installation directory and restarting ServletExec.

Within the ServletExec installation directory is the `webapps` sub-directory; within the `webapps` directory are sub-directories for each virtual server that ServletExec is configured to support. Copy the WAR file to the sub-directory for the virtual server on which the webapp is to be deployed.

For example, if you have a default installation of ServletExec 4.1 for Microsoft IIS (ServletExec/ISAPI), copy the WAR file to the following directory:

```
C:\Program Files\New Atlanta\ServletExec ISAPI\webapps\default
```

Then restart ServletExec (in the case of ServletExec/ISAPI, restart IIS). After ServletExec is restarted, you can confirm that the webapp is configured using the ServletExec administration console, as illustrated below:

![ServletExec Administration Console](image)

The webapp is now successfully deployed, which can be confirmed by accessing the following URL:

```
http://localhost/mywebapp/index.cfm
```
Manual Deployment

Perform the following steps to manually deploy a webapp on ServletExec 4.1, either as an open directory or in a WAR file.

1. Launch the ServletExec administration console and access the Manage Web Applications page. Click the Add Web Application button, as illustrated below:

2. On the Add Web Application page, enter the Application Name, URL Context Path, and physical Location of the webapp (the Location can specify either a directory or WAR file), then click the Submit button:
3. The webapp is now deployed and is displayed in the ServletExec administration console as follows:

4. The webapp is now successfully deployed, which can be demonstrated by accessing the URL http://localhost/mywebapp/index.cfm, which produces a page similar to the following:
BlueDragon CFML Test Page

Hello! Welcome to BlueDragon from a CFML custom tag.

Hello! Welcome to BlueDragon from a Java CFX tag.

Hello again from CFX TestTag!

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Exists</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>DoesNotExist</td>
<td>No</td>
<td></td>
</tr>
</tbody>
</table>

No QUERY parameter found for tag.
Appendix H. Macromedia JRun 4

This appendix describes how to deploy J2EE web applications on Macromedia JRun 4. These instructions are provided for convenience only; refer to the Macromedia JRun 4 documentation for detailed instructions for deploying web applications.

Auto-deployment of WAR Files

JRun 4.0 allows you to auto-deploy WAR files by simply copying them to the JRun installation directory and restarting the JRun server. To auto-deploy a WAR file, copy it to the `<JRun install>/servers/<server name>` directory. For example, in a default JRun 4 installation, copy the WAR file to:

```
C:\JRun4\servers\default
```

Then restart the JRun default server. After JRun is restarted, you can confirm that the webapp is configured using the JRun administration console, as illustrated below:

The webapp is now successfully deployed, which can be confirmed by accessing the following URL:

```
http://localhost/mywebapp/index.cfm
```
Manual Deployment

Perform the following steps to manually deploy a webapp on JRun 4, either as an open directory or in a WAR file.

1. Launch the JRun 4 administration console and access the J2EE components page. Click the Add button in the Web Applications section, as illustrated below:

2. On the Deploy Web Application page, enter the Source File Path, which is the path to either an open directory or a WAR file, then click the Deploy button:
3. The webapp is now deployed and displayed in the JRun administration console as follows:

4. The webapp is now successfully deployed, which can be demonstrated by accessing the URL http://localhost:8100/mywebapp/index.cfm, which produces a page similar to the following:
Deploying CFML on J2EE Servers
Appendix I. Apache Tomcat 4.0 and 4.1

This appendix describes how to deploy J2EE web applications on Apache Tomcat 4.0 and 4.1. These instructions are provided for convenience only; refer to the Apache Tomcat documentation for detailed instructions for deploying web applications.

Auto-deployment of WAR Files

Tomcat 4.0 allows you to auto-deploy WAR files by simply copying them to the Tomcat installation directory and restarting Tomcat. To auto-deploy a WAR file, copy it to the <Tomcat install>\webapps directory and restart Tomcat. This results in the creation of a subdirectory there of the same name as the WAR file.

Tomcat 4.1.27 and above allow auto-deployment of web applications as well, by copying them to the same \webapps directory.

Manual Deployment

To manually deploy a web application on Tomcat 4.0, edit the server.xml configuration file and add a <Context> element for the webapp within the <Host> element of the virtual server on which the webapp is being deployed. For example:

```xml
<!-- Define the default virtual host -->
<Host name="localhost" debug="0" appBase="webapps" unpackWARs="true">
  ...
  ...
  <Context path="/mywebapp" docBase="C:\webapps\mywebapp"/>
  ...
  ...
</Host>
```

The path attribute of the <Context> element specifies the URL context path for the webapp. The docBase attribute specifies the physical location of either an open webapp directory or a WAR file.

You must restart Tomcat for changes to server.xml to take effect.
Appendix H. Borland Enterprise Server 5.2.1
This appendix describes the deployment of a webapp onto Borland Enterprise Server (BES) 5.2.1. The example webapp is located in a WAR file at C:\webapps\mywebapp.war. These instructions are provided for convenience only; refer to the BES 5.2.1 documentation for detailed instructions for deploying web applications.

1. From the BES Management Console, launch the Deployment Wizard and click the “Add” button. In the “Add J2EE Module” dialog, navigate to the location of mywebapp.war, then click the “OK” button:

![Add J2EE Module dialog](image)
2. The Deployment Wizard Step 1 panel should now appear similar to the following:

![Module and Library Deployment Wizard]

3. Click the “Next” button to go to the Deployment Wizard Step 2. Select the partition or cluster on which to deploy the webapp, then click the “Finish” button:
4. When the webapp is finished deploying, a dialog similar to the following will appear:

5. Expand the hierarchy to display the status of the deployed webapp. If deployed successfully, two green check marks will appear next to the WAR file name:
6. The webapp is now successfully deployed, which can be demonstrated by accessing the URL http://localhost:8080/mywebapp/index.cfm, which produces a page similar to the following:
BlueDragon CFML Test Page

Hello! Welcome to BlueDragon from a CFML custom tag.

Hello! Welcome to BlueDragon from a Java CFX tag.

Hello again from CFX_TeStTag!

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Exists</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>DoesNotExist</td>
<td>No</td>
<td></td>
</tr>
</tbody>
</table>

No QUERY parameter found for tag.

---

```
struct
null_type       [empty string]
content_length  [empty string]
content_type    [empty string]
gateway_interface [empty string]
http_accept    image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/vnd.ms-excel, application/vnd.ms-word, application/octet-stream
http_accept_encoding gzip, deflate
```