

# It's Not ColdFusion – It's J2EE!

Setting the record straight

Over the past few months I've had several conversations with developers that have all been pretty much the same. Here's a paraphrase of a recent one: "My management decided to standardize on *<insert your favorite J2EE server here>*, and my company isn't going to use ColdFusion anymore.

"They're telling me I have to convert my existing CFML applications to run on the J2EE server so they can retire the old ColdFusion 4.5 and 5.0 servers.

"The server admins say I have to package my applications into standard J2EE WAR files for deployment. I've looked at BlueDragon – the J2EE edition – and it seems to do everything I need. But when I told my management about it they said, 'Sorry, we've already said that you can't use ColdFusion – you'll have to rewrite it in JSP so that it'll be real J2EE.'

"How am I going to rewrite everything in JSP? I don't know JSP! And it's going to take forever – I have several applications and some of them are made up of hundreds of CFML pages and custom tags! And how am I going to make all these enhancements they want while I'm rewriting everything? What am I going to do?"

My answer to this developer is to first educate his management. There are two important pieces of misinformation in their objection to the use of BlueDragon: (1) it's not ColdFusion (it's CFML); and (2) it's real J2EE. The rest of this column addresses these issues, particularly the second one, and provides details of the BlueDragon architecture to support the claim that it's "real J2EE."



By Vince Bonfanti

Having answered his managers' objections, he should point out to them that the time and costs of rewriting are unnecessary; BlueDragon allows them to migrate their CFML applications to their J2EE server relatively quickly and at a much lower cost than rewriting them in JSP. Finally, having migrated their CFML

applications to a standard J2EE environment, they're free to enhance the applications using either CFML or JSP, or even to slowly rewrite the application in JSP one page at a time if that's what they choose to do.

## CFML is Not ColdFusion

This may seem obvious once you "get it," but it's very hard for some people to separate CFML from ColdFusion because they've been synonymous for so long. CFML is a tag-based markup and server-side scripting language for developing dynamic Web content. ColdFusion is the name of a server product from Macromedia, Inc., that implements CFML. They're not the same thing.

Consider this: there are two completely different implementations of CFML in server products from Macromedia, both named "ColdFusion." One is the original C/C++ implementation embodied in

ColdFusion 5.0, the other is ColdFusion MX, which is not only written in the Java programming language, but is based on a radically different architecture from the original C/C++ version. The fact that the inventors of CFML can create two very different implementations illustrates the separation of the CFML language from the ColdFusion server (even though Macromedia chose to name both of their implementations *ColdFusion*).

Similarly, BlueDragon is a server product that implements CFML using an architecture that differs from both the original C/C++ ColdFusion 5.0 and the Java-based ColdFusion MX. BlueDragon is CFML. BlueDragon is not ColdFusion. CFML is not ColdFusion.

Why is this distinction important? If we go back to the managers' objections, one is, "We've already decided that we're not using ColdFusion anymore." Often, this objection is to the ColdFusion server product and not to the CFML language, but the distinction between ColdFusion and CFML has not been clearly made. Therefore, the rejection of ColdFusion becomes a rejection of CFML. This doesn't have to be the case.

The BlueDragon architecture allows you to integrate CFML into J2EE Web applications as a legitimate, standard, portable, fully compliant presentation-layer alternative to JavaServer Pages (JSP). Therefore, because you can use BlueDragon to deploy CFML in "real J2EE" applications, a rejection of ColdFusion (the server) does not have to be a rejection of CFML (the language).

## BlueDragon is Real J2EE

Java servlets are the core Java 2, Enterprise Edition (J2EE) platform tech-

nology for creating dynamic Web content. JSP pages are translated and compiled into Java servlets at runtime by the J2EE server; furthermore, on many J2EE servers, the translate-compile-execute processing of JSP pages is implemented by a Java servlet. J2EE Web applications, which are supported by all compliant J2EE servers, are defined by the Java servlet specification. Java servlets are as “real J2EE” as it gets.

The BlueDragon CFML runtime is implemented as a standard Java servlet. As such, it can be packaged into a standard J2EE Web application and deployed onto any compliant J2EE server. Web applications containing the BlueDragon servlet have been successfully deployed – without any modifications to the Web application, BlueDragon servlet, or J2EE server – to run CFML pages on BEA WebLogic, IBM WebSphere, Sun ONE Application Server, Oracle 9i Application Server, Borland Enterprise Server, JBoss, Macromedia JRun, New Atlanta ServletExec, and Apache Tomcat. This is what makes BlueDragon “real J2EE.” It is entirely standards-based, is fully compliant with all relevant J2EE specifications, and is completely portable across J2EE servers.

Here are some questions I’d ask someone who thinks CFML via BlueDragon isn’t real J2EE: if you write your own Java servlet and deploy it within a Web application onto a J2EE server, is that real J2EE? If you create a Web application using the servlet-based Struts Framework, is that real J2EE? If you create a Web application using a servlet-based templating system such as WebMacro or Velocity, or a servlet-based XML/XML transformer, is that real J2EE? If the answer to any of those questions is “Yes” (hint: the answer to all of them is “Yes”), then it becomes clear that the BlueDragon servlet-based CFML runtime is also “real J2EE.”

In order to understand this better, let’s take a more detailed look at what a Java servlet is, what a J2EE Web application is, and how to create standard J2EE Web applications that support CFML pages using the BlueDragon servlet.

## Java Servlets

Java Servlets are defined by the `javax.servlet.*` and `javax.servlet.http.*` packages (in Java, packages are collections of classes that perform similar or

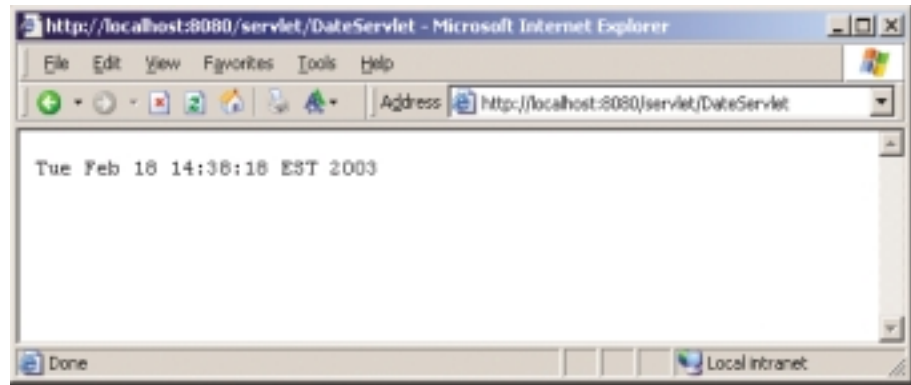


Figure 1

related functions, or that depend on each other to provide a common set of functions). In order to implement a Java servlet you need to do two things:

1. Create a Java class that extends `javax.servlet.http.HttpServlet`.
2. Implement the `service()` method of your class to receive HTTP requests and send HTTP responses.

The `service()` method of the `HttpServlet` class receives two parameters: an `HttpServletRequest` that contains all of the information about a request from the browser (including HTTP headers, URL arguments, and POST data from form submissions), and an `HttpServletResponse` that is used to send responses back to the browser. Here’s a complete Java servlet that simply returns the current date and time back to the browser:

```
import javax.servlet.*;
import javax.servlet.http.*;

public class DateServlet extends HttpServlet
{
    public void service( HttpServletRequest
        request,
        HttpServletResponse response )
        throws ServletException,
        java.io.IOException
    {
        // Set the response content type to
        // text/plain
        response.setContentType( "text/plain"
    );

        // Send the current date and time to
        the client
        response.getWriter().println( new
        java.util.Date().toString() );
    }
}
```

This column is about BlueDragon for J2EE servers, but if you’ve installed BlueDragon Server JX (the standalone server version of BlueDragon that includes servlet and JSP support), then you can see the output of the `DateServlet` by entering the following URL in your browser:

<http://localhost:8080/servlet/DateServlet>

The output produced by the `DateServlet` in the browser is illustrated in Figure 1.

If you don’t have BlueDragon Server JX installed, you’ll need to compile and deploy the `DateServlet` to a servlet container (instructions for which are beyond the scope of this column).

Because the BlueDragon CFML runtime is implemented as a standard Java servlet, it has the same characteristics as the `DateServlet`: it extends `HttpServlet` and implements the `service()` method. Of course, the BlueDragon `service()` method is a little more complicated than the `DateServlet`!

Some servlet containers, such as the one in BlueDragon Server JX, allow you to deploy Java servlets simply by copying the class files (the files generated by compiling Java source code) to a special “servlets” directory. However, the standard J2EE way of doing things is to deploy servlets (and other content) within “Web applications” that conform to a particular set of rules.

## J2EE Web Applications

According to the Java Servlet 2.3 specification, “a Web application is a collection of servlets, HTML pages, classes, and other resources that make up a complete application on a Web server. The

Web application can be bundled and run on multiple containers from multiple vendors.” A J2EE Web application is characterized by a specific directory structure, and a configuration file named web.xml that is also referred to as the Web application deployment descriptor.

Content that is to be served to the client (HTML, JSP, and CFML pages; GIF and JPEG images; etc.) is placed directly in the top-level directory of a Web application. A Web application may contain subdirectories within the top-level directory; for example, it may contain an images sub-directory to hold GIF and JPEG files.

Within the Web application top-level directory is a special subdirectory named WEB-INF. The J2EE server will not serve any content from the WEB-INF subdirectory 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.

The web.xml deployment descriptor contains configuration information used by the J2EE server to support the Web application. For example, the web.xml file provided with BlueDragon 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).

There are two special subdirectories within the WEB-INF directory: the classes subdirectory that is used to hold unbundled Java .class files, and the lib subdirectory that is used to hold Java .jar archives. Any Java classes placed in these subdirectories are automatically available to the Web application (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 Web application are:

- Content that is to be served to the client (HTML, GIF, JPEG, CFM, JSP, etc.) is placed directly within the Web application top-level directory, or within subdirectories of the top-level directory.

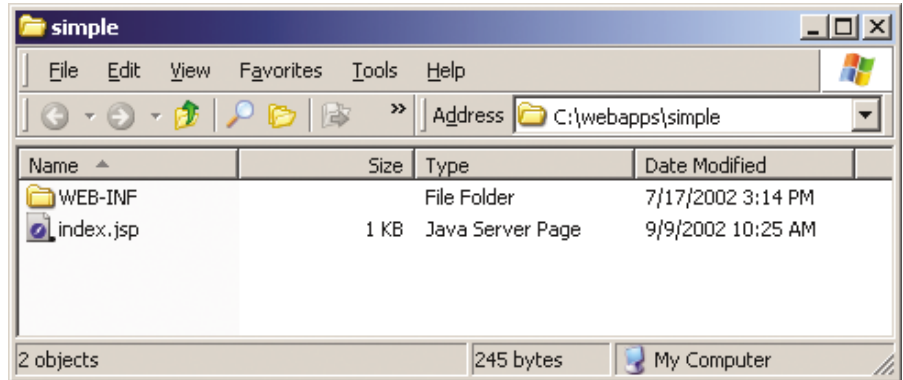


Figure 2

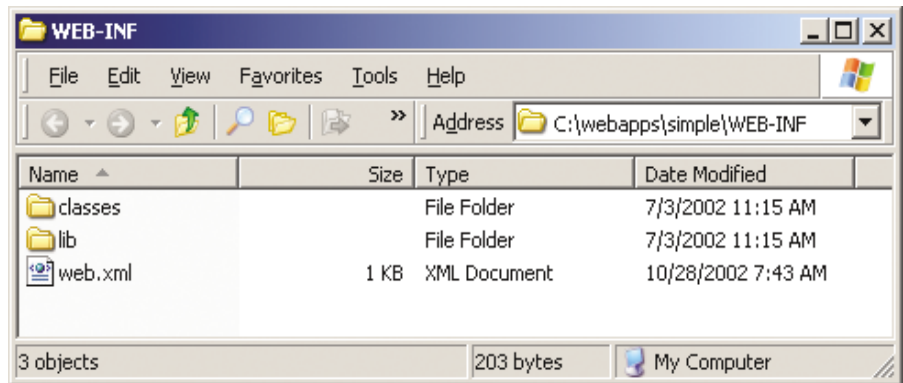


Figure 3

- The WEB-INF subdirectory is located within the Web application 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 subdirectories within WEB-INF are used to store Java .class and .jar files, respectively.

## Deploying CFML in a J2EE Web Application

In order to deploy CFML pages onto a standard J2EE server, start with a basic J2EE Web application, add and configure the BlueDragon CFML servlet, then add your CFML pages and deploy!

The following figures illustrate a simple J2EE Web application. The Web application top-level directory is located at C:\webapps\simple, which contains the WEB-INF subdirectory and a single content file, index.jsp. The WEB-INF subdirectory contains the classes and lib subdirectories (which are

empty), and the web.xml deployment descriptor. If you were to examine the contents of web.xml, you'd find the following:

```
<web-app>
</web-app>
```

In other words, the web.xml deployment descriptor is empty (by definition, J2EE Web applications “know” how to process JSP pages, so no configuration is needed) see Figures 2 and 3.

This is a complete, valid J2EE Web application that is ready to be deployed on any compliant J2EE server. In order to process CFML pages within this Web application, we need to do two things:

- Copy the BlueDragonJ2EE.jar file into the WEB-INF\lib subdirectory. The BlueDragonJ2EE.jar file contains the BlueDragon CFML servlet and supporting classes.
2. Add configuration information to web.xml related to the BlueDragon CFML servlet. The key configuration entries to be added to web.xml are:

```

<servlet>
  <servlet-name>tagServlet</servlet-name>
  <description>BlueDragon CFML
  Engine</description>
  <servlet-
class>com.naryx.tagfusion.cfm.cfServlet</servl
et-class>
</servlet>

<servlet-mapping>
  <servlet-name>tagServlet</servlet-name>
  <url-pattern>*.cfm</url-pattern>
</servlet-mapping>

```

These web.xml entries tell the J2EE server that there's a servlet implemented by the class `com.naryx.tagfusion.cfm.cfServlet` and gives it the name "tagServlet". It then tells the J2EE server to forward all requests that end with ".cfm" to "tagServlet" for processing.

That's it! You can now add CFML pages (with the ".cfm" extension) to the Web application and deploy them onto any standard J2EE server. (Well, that's not really all there is to it, but those are


the main points. There are some additional JAR files needed by BlueDragon, such as JDBC drivers for database access, and some additional directories within WEB-INF, such as the "custom-tags" directory. If you really want to try it, go to New Atlanta's Web site – [www.newatlanta.com](http://www.newatlanta.com) – and download BlueDragon for J2EE servers).

## WAR Files

Finally, a Web application 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 Web application 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. WAR files provide a convenient package for deploying J2EE Web applications as a single component (file).

## Conclusion

The BlueDragon CFML runtime is

implemented as a standard Java servlet that conforms to all of the relevant J2EE specifications. The BlueDragon CFML runtime servlet can be configured in a standard J2EE Web application, which can be used to deploy CFML pages onto any compliant J2EE server. BlueDragon: it's not ColdFusion, it's CFML as "real J2EE"! 

## About the Author

*Vince Bonfanti is president and co-founder of New Atlanta Communications, developers of Java- and CFML-based server products. A charter member of Sun's JavaT Servlet API and JavaServer PagesT Expert Groups, Vince has been a JavaOne speaker and a contributor to Java trade magazines and online publications. He has also been a featured speaker at Toronto's CFNorth and Washington's CFUN conferences as well as at local ColdFusion User Groups around the country.*

[vince@newatlanta.com](mailto:vince@newatlanta.com)

# Ad