

# ColdFusion and Java

Andrew Cripps

September 25, 2000 3:44 pm

## Introduction

---

In 1995 Allaire launched the ColdFusion Web Application Server. Since then, java has become an important language for the web, and Allaire has moved right along to combine the power of java with ColdFusion. On June 15th 1999, Allaire purchased JRun, giving them a powerful java server that currently supports java servlets and Enterprise Java Beans. Allaire has a technology road map that will allow your CFML pages to execute directly on the next generation application server technology - a java engine code-named "Pharaoh."

Right now you can develop applications in java and run them with your JRun engine, and you can develop ColdFusion applications in CFML and run them on the ColdFusion Web Application Server. But how do you use ColdFusion and java *together* right now?

Jeremy Allaire, in his column IMHO, August 1999 (Vol.1 Issue 4) said that "ColdFusion and Java are perfect cousins." In this article I want to show you how you can use Java with your ColdFusion applications right now. The good news is that you don't have to wait for Pharaoh or switch to JRun.

**Java on the Server and Java on the client.** When java first came out, many people thought it was going to be primarily useful on the client. Browser manufacturers started including Java Virtual Machines (JVM) in their products, and websites started putting java applets in web pages. The browser downloads the applet, and the applet runs on the client machine. As java has matured, it has come to be seen more as a server-side language than a client-side language. On the server, software systems can be built from a combination of java servlets, java classes and/or applications, and Enterprise Java Beans. You can make use of both client-side java and server-side java with ColdFusion.

There are five methods for calling java (both server-side and client-side) from ColdFusion, and we'll go over them all. The five methods are:

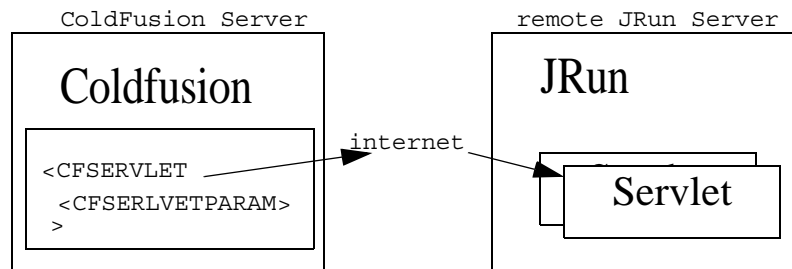
- 1 calling java servlets using `CFSERVLET`
- 2 calling java servlets using `CFHTTP`
- 3 calling and creating java components with `CFOBJECT`
- 4 calling java applets with `CFAPPLET`
- 5 calling java classes with `CFX` tags

## CALLING SERVLETS WITH `CFSERVLET`

---

You can call java servlets from your ColdFusion pages using the `CFSERVLET` tag. The `CFSERVLET` tag executes a servlet on the JRun engine. Note that in order to make use of `CFSERVLET`, the machine on which the servlet exists needs to have JRun installed. You don't have to have JRun on your own server unless you are writing servlets and want to call your own servlets from ColdFusion. You can of course make remote calls to other machines as long as the remote machine is using JRun as its servlet engine.

Let's say that you want to call a servlet-based website that's using JRun. You can call any of the servlets on that machine with `CFSERVLET`. All you have to do is specify the `JRUNPROXY` attribute to be the IP address of the server running JRun.



Depending on your website, you might have JRun and ColdFusion on the same server, or on the same network.

Sometimes you need to specify parameters for the sevlet you are calling. To do this you specify `CFSERVLETPARAM` tags within the `CFSERVLET` call.

For example,

```
<CFSERVLET CODE="ServletName"
            JRUNPROXY="233.125.25.3:8083">

            <CFSERVLETPARAM name="param1" value="35">
</CFSERVLET>
```

Note that your ColdFusion page, when it executes the `CFSERVLET` tag, will wait for a response from the servlet you have called before continuing. And the servlet you have called is executing on the JVM in the JRun engine.

## CALLING SERVLETS WITH CFHTTP

---

Sometimes you may need to call java servlets that are hosted on a web server that is not using JRun as its servlet engine. You can do this using `CFHTTP`. Simply include the `CFHTTP` tag in your CFML page, and make the URL that you call point to the servlet you want to execute. If the servlet you are calling requires form input parameters, you will need to specify the `METHOD="POST"` attribute in `CFHTTP` and include `CHTTPPARAM` tags for the form variables that your servlet requires.

For example:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<html>
<head>
<title>Test CFHTTP call</title>
</head>
<body>

<cfhttp url="http://ippc.nais.luftfartsverket.no/PIBListServlet" method="post">
<cfhttpparam name="RetrievePIBList"
            value="Retrieve List"
            type="FORMFIELD">
</cfhttp>

<Cfoutput>#cfhttp.filecontent#</cfoutput>

</body>
```

</html>

## CALLING JAVA COMPONENTS WITH CFOBJECT

---

CFOBJECT provides a flexible way to call java components. You can create and use java objects and Enterprise Java Beans using CFOBJECT. (You need ColdFusion 4.5 or newer to call java components this way.)

**Set up in ColdFusion Administrator.** To use CFOBJECT (and java CFXs) you need to set up the JAVA section of the ColdFusion Administrator. Example settings are as follows:

- **Load JVM when starting ColdFusion.**  
If you want to load the JVM in the ColdFusion process every time ColdFusion starts, check this box. If you only want to load the JVM when a java request is made, leave it unchecked.
- **Java Virtual Machine Path**  
This is the path to your java virtual machine. For example, on Windows, it might be: "c:\jdk1.2.2\jre\bin\classic\jvm.dll"
- **Class Path**  
Your class path should contain the paths of any java classes you want to use with CFOBJECT. For example:  
"c::c:\jdk1.2.2;c:\jdk1.3;c:\xerces\xerces.jar;  
c:\weblogic\lib\weblogicaux.jar;c:\weblogic\classes"
- **Initial Heap Size**  
Specifies the heap size that is allocated to the JVM when the JVM starts. (Typically 1K)
- **Maximum Heap Size**  
Specifies the maximum size for the heap assigned to the JVM (Typically 16K)
- **System Options**  
You can specify any JVM command-line options here (name-value pairs, separated by semicolon)
- **Implementation Options**  
You can specify any implementation-specific options here (name-value pairs, separated by semicolon)
- **CFX Jar path**

Specify the location of the ColdFusion cfx.jar file which contains interfaces used by Java CFXs.

For example, "C:\CFUSION\JAVA\CLASSES"

Any java class in the class path specified in the ColdFusion administrator can be loaded and used by ColdFusion. Note that classes execute on the JVM embedded in the ColdFusion process.

Your ColdFusion administrator must also enable the `CFOBJECT` tag in the security section before you can use either `CFOBJECT` or the `CreateObject` function.

**Calling Enterprise Java Beans.** You also use `CFOBJECT` to call EJBs. EJBs can be deployed in any EJB server, such as JRun or BEA's weblogic server.

To call EJBs, you first configure the ColdFusion administrator, as described above; second, you create a java bean and deploy it to your EJB server; third, you create a java object that is running on the ColdFusion JVM, and then use that object to call your EJBs running on the EJB Server. The results of the call are returned as a ColdFusion variable that you can then use in your CFML template.

You can create java objects in two ways:

- using the `<CFOBJECT ACTION="CREATE" ...>` tag,
- using the function `CreateObject("JAVA", class)`

Note that `CFOBJECT`, supports `COM` objects, java objects and `CORBA` objects. Solaris installations of ColdFusion supports only `CORBA` objects at the time of writing, while Windows NT installations of ColdFusion support all three types.

For example, suppose you have written an Entity Bean that looks up an employee name given a unique key. In this example, we will assume that the bean has been deployed to BEA's weblogic server.

We now need a ColdFusion page that will call our Entity Bean.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<!--      Andrew Cripps, Sept. 2000      -->
<!--      Create a weblogic environment object that allows us to
      look up the bean "EmployeeHome"
-->
<CFOBJECT ACTION="Create"
      TYPE="Java"
      CLASS="weblogic/jndi/Environment"
      NAME="wlEnv">
<!-- Get the context -->
```

```

<CFSET ctx=w!Env.getInitialContext()>
<!-- Use the lookup method of the context -->
<CFSET ejbHome=ctx.lookup("EmployeeHome")>

<!-- accepts ID as input and calls EJB component to do lookup -->
<cfparam name="form.id" default="1">

<!-- Call the findByEmployeeID method -->
<cfset temp = ejbHome.findByEmployeeID("#form.id#")>
<!-- Call the getEmployeeName method-->
<cfset value = temp.getEmployeeName()>

<!-- Output the results-->
<html>
<head>
<title>form post processor</title>
</head>

<body>
<h1>Calls Employee EJB</h1>
<cfoutput>
Value from EmployeeBean is <b>#value#</b>
</cfoutput>
</body>
</html>

```

## CALLING JAVA APPLETS WITH CFAPPLET

---

Client-side Java    Java applets are supported by ColdFusion using the `CFAPPLET` tag. Applets are stored on your server. They are downloaded by the client whenever a page that contains applets is requested from the server.

**Using Java Applets.** To use a java applet in your CFML page, you need to register the applet with the ColdFusion Administrator **Applets** section. Once it's registered you can embed the applet in your CFML page using `CFAPPLET`. `CFAPPLET` must be used with `CFFORM`.

## CALLING JAVA CLASSES WITH CFX TAGS

---

CFXs - or ColdFusion Extensions - allow you to extend ColdFusion with custom code written in C++ or java. You can write your own CFXs in java or you can just call existing java classes.

To call a java class this way, you need to do two things in the ColdFusion administrator.

First, in the Java section, you need to set up the class path and other information as described in the section above on `CFOBJECT`.

Second, you need to register you custom extensions in the ColdFusion administrator in the CFX Tags section. You give the CFX a name, specify that is a java extension, and then tell ColdFusion which java class the CFX will call.

For example, let's suppose you've registered a class "HelloWorld" In the ColdFusion administrator CFX section that outputs "HelloWorld" when it's called. In the Java section, you've included the path to your HelloWorld.class file in the "ClassPath" option, and now you want to create a sample ColdFusion page.

```
<html>
<head>
<title>Call CFX HelloWorld</title>
</head>

<body>
<h1>Calls CFX_HelloWorld</h1>
<CFX_HelloWorld>
</body>
</html>
```

You should see "HelloWorld" as the output when you call this page. If you get an error like this:

```
java.lang.ClassNotFoundException: HelloColdFusiona. Java
exception occurred in call to method.
```

it is most likely that you have not correctly specified the path to your java class in the "ClassPath" setting of the java section of ColdFusion Administrator. Another possibility is that you have not correctly specified the classname in the CFX "Class" attribute in the CFX section of the ColdFusion Administrator.

Note that if you are using ColdFusion server version 4 or earlier, you need to use `CFX_J`. `CFX_J` has now been deprecated and has been replaced with the mechanism described above. But you can still download and install `CFX_J` if you have an older version of ColdFusion server.

### Summary

In this article, we've covered five ways that you can call both client-side and server-side java from ColdFusion. If you need to call

JRun servlets, use `CFServlet`; if you need to make a simple call to java class, use `CFXs`; and if you need to call EJBs, use `CFObject`.