

How to Debug Your Applications

Problem solving and techniques for your debugging adventures



By Jeff Houser

I just spent four hours debugging an error for a client. The client is an application service provider, and they developed an administrator for internal use. The administrator allows for the user to switch between sites, at whim. The

“site switcher” is a drop menu, which loads up a different DNS depending on the selected site.

A certain set of code was working for all sites except one. The application was not throwing errors, but nothing was getting displayed on the page. How do you find the problem? I’m going to step you through the process I took to solve the problem, and along the way, I hope you’ll learn a technique or two that you can use in your own debugging adventures.

Two Types of Errors

There are really two types of errors that you’ll encounter in your code: syntax errors and logic errors. Syntax errors are those ugly ColdFusion errors that you are used to seeing every day as you write code (I know I’m used to seeing them). Generally these are easy to discover and address. These are usually squashed during your development.

The most common reason why I see syntax errors on production sites is due to undefined variables being accessed. Sometimes there is some link somewhere that is missing a URL variable, or perhaps someone changed the Query String and reloaded a page. Sometimes a checkbox form element is accessed, yet the user did not check anything before submitting the form, so it is undefined or a shared scope variable may be accessed before it is set. Other reasons that ColdFusion might throw syntax errors might be invalid tags or attributes, mismatched tags, or database timeout.

Syntax errors are easier to find than logic errors. Logic errors occur when the code that you’ve written doesn’t satisfy the business requirements you were setting out to imple-

ment. Some business requirements might be easy to fix, such as display 20 products per page instead of 15. Others might become murky to catch, such as people who buy at least five products from Product Line A get a 10 percent discount on all products from Product Line A in their order, unless they have three products of Product Line B attached to the same order, whereas they receive the bigger discount of 10 percent off Product Line A products or 5 percent off of all Product Line B products. The best way to avoid business logic errors like these is to have a clear set of business requirements before you start to code the application. Sometimes it’s hard for the client to see the big picture.

Examining the Debugging Output

Going back to my problem of the day, I started out by assuming that the code was fine, since it was working for all of the sites except one. The problem must reside in the data. The page in question was supposed to show a list programs from the database. If no programs existed, then a blank display is not a surprise. I tracked down the query code and executed it manually against the database using Query Analyzer (an SQL Server tool). The query worked fine and the proper data was returned. This query didn’t contain any CF variables, so the cause of this couldn’t have been invalid values.



Figure 1: The ColdFusion Administrator Login

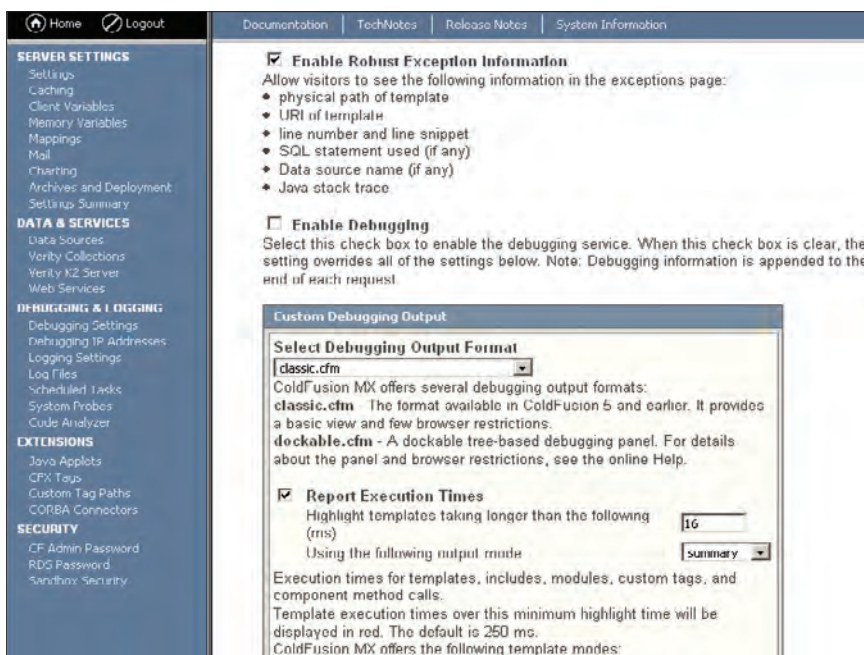


Figure 2: The Debugging and Settings Page

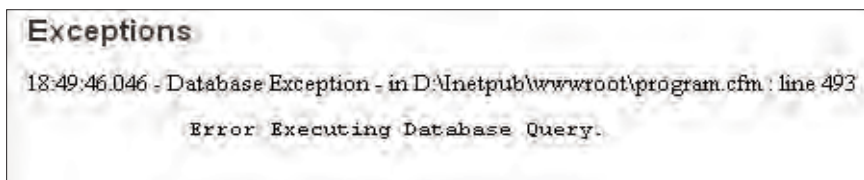


Figure 3: Database error

Just to be sure, I wanted to review the SQL actually being sent to the DB. For this I needed to turn on ColdFusion's debugging output. In normal cases, you wouldn't turn on debugging for a production site, but I decided this case was an exception. The site only has half a dozen users, none of whom could use the site until the error was fixed. You can turn on debugging from your ColdFusion Administrator using these instructions:

1. Go to your administrator. Most likely the URL is `www.yoursite.com/cfide/administrator`. You'll see a screen similar to Figure 1. It might look a little different for those who have already upgraded to CF7. Enter your password to login.
2. Under Debugging and Logging select Debugging Settings. You'll see the different things that can be displayed in debugging output, as shown in Figure 2.
3. Check the Enable Debugging box, if

not already done. I also like to click the Enable Robust Exception Information so that I can view the full SQL statement, and other information about the queries that run.

4. While you're in here, take some time to examine the list of other information that can be displayed in the debugging output. You can select the type of display, either dockable or classic. I prefer classic, the default method. You can also display various variable scopes, tracing information (used in conjunction with the `cfttrace` tag), exception information, database activity, and general debug information. Most commonly I use the variables and database activity.

Back to the problem at hand: I enabled debugging and reran the template. I scrolled down and looked for the debugging output. The query I was looking for was not executed. Why was that?

cfdump and cfabort

The template in question is what I call a "case" template. Based on some mode variable, different code is executed within the template, often with completely different execution and results. I call it a case template because a different `cfcase` statement represents each mode. All of this inside the `cfswitch` tag. For more information, see: <http://livedocs.macromedia.com/coldfusion/7/htmldocs/00000339.htm#1103819>. This type of template is very similar to a fusebox approach to designing Web applications. The mode would be a "fuseaction" and the base template would be the "fusebox" template. I know by looking at the URL that a client-side redirect is not happening, which leaves two options. Either the wrong "mode" is being executed or a server side redirect is happening.

The first thing I wanted to do was to verify that the mode was correct. I turn to `cfdump` tag for that. `cfdump` is a very useful debugging tag that displays the contents of a variable. With simple variables, you could use a `cfoutput` to display the value of that variable, however for complex variables such as an array, structure, or XML document the `cfdump` is invaluable. `cfdump` has a few attributes, but the one we care about for the moment is the `var` attribute. You use the `var` attribute to specify which variable you want to display. Always make sure to enclose the variable name in pound signs, or else the variable name will be displayed instead of the variable value. You can check out the full documentation on `cfdump` at <http://livedocs.macromedia.com/coldfusion/7/htmldocs/00000239.htm#3765824>.

I used `cfdump` to dump the value of the mode variable before entering the `cfswitch` tag. The code looked something like this:

```
<!-- code here -->
<cfdump var="#mode#">

<cfswitch expression="#mode#">
<!-- more code here -->
```

With debugging code in place, I reloaded the template. Nothing was displayed. I switched to a different site and tried again, and the mode was properly

displayed. It was empty, which was the default mode. That led me to believe there must be some sort of redirect that was being called in conditions that existed in the rogue site. In CF the most common method of redirect is to use a cflocation. I started pouring over the code looking for it. There was not an obvious redirect, but it may have been included in any one of a handful of includes or inside some CFC methods. Another tag came to the rescue, cfabort.

cfabort is used to stop the processing of a ColdFusion page. It has a single attribute, ShowError. The ShowError attribute accepts a string value, which is displayed to the screen before the templates execution is stopped. I added this line of code to the template before the switch statement:

```
<cfabort ShowError="It Got This Far">
```

I reran the template, but didn't receive the error message. That told me that the redirect was happening before the cfabort tag. By moving around the cfabort tag and I was able to isolate the line of code that caused the problem. It was a CFC method call that verified that the user was allowed to view this page. If the user didn't have access to view the page, then the system was actually operating properly (although some feedback would have been nice). I went back to the database to verify that my user had proper access to the database, and I did. Before the cfabort I added a dump of the user's access. Yes, the user should have

access. This obviously wasn't the problem.

At this point, I take another in-depth look at debugging output. I noticed something I hadn't noticed before, as shown in Figure 3. There was an error executing the database query. With this new bit of information, I jumped over to the admin and verified the data source. Sure enough ColdFusion could not connect to it. This was probably the cause of the problem. I opened up the data source to discover that the username was incorrect. I corrected the username, verified the data source, and then loaded the trouble page again. Everything started working as expected. Finally!

Catching Errors

So the problem was a failed database query and not a cflocation redirect. You might want to ask why I didn't see a ColdFusion syntax error if the database query failed, right? Good question, it shows that you are thinking. The answer is simple. The error was caught and the user was redirected to the "empty" index page. In this case the site was using the cferror tag to catch the error. cferror is a tag that you would normally put in the application.cfm to launch a specific error page when an error occurs. In CFMX7 you can also use the OnError event that is part of Application.cfc. More information about the cferror tag can be found at <http://livedocs.macromedia.com/coldfusion/7/htmldocs/00000242.htm#2022557>.


<http://livedocs.macromedia.com/coldfusion/7/htmldocs/00000697.htm#1188543>.

Information about the OnError method is found here: <http://livedocs.macromedia.com/coldfusion/7/htmldocs/00000697.htm#1188543>.

cferror usually redirects the user to a page that can process the error, usually displaying a nice error message to the user, logging the error, and/or sending an e-mail to notify a developer about the error. Unfortunately, this particular cferror tag just redirected the user to the site home page, which masked the error from my view. It is an internal site, which unfortunately, often gets fewer resources than the public-facing site, and I'm sure the error page was something that just hasn't been implemented for this site.

Where to Go from Here

What do you want to look at next? If you're a BlueDragon user you can take a look at cfdebugger, which writes detailed information about the code that is executed to a file. More information about cfdebugger is in New Atlanta's enhancement guide, located here: www.newatlanta.com/products/bluedragon/self_help/docs/6_2/BlueDragon_62_CFML_Enhancements_Guide.pdf, and it was written about in the November 2003 article of *CFDJ*. At CFUnited, New Atlanta also announced the BlueDragon Profiler, which will allow you to get line-by-line details of a templates execution.

It is also worth noting that you can create your own debug templates for ColdFusion. The classic and dockable styles are located in the web-INF\debug directory of your ColdFusion installation. You can use them as a start to roll your own. Put your new debug template into the same directory and it will show up in the Administrator under debug settings. 

About the Author

Jeff Houser has been working with computers for over 20 years and in Web development for over 8 years. He owns a consulting company and has authored three separate books on ColdFusion, most recently *ColdFusion MX: The Complete Reference* (McGraw-Hill Osborne Media).

jeff@instantcoldfusion.com

