How to use the Cocoon Flow Debugger (2.1 legacy document)

Table of contents

Copyright © 1999-2005 The Apache Software Foundation. All rights reserved.

Table of contents

Copyright © 1999-2005 The Apache Software Foundation. All rights reserved.

Warning:
This document was copied as is from the Cocoon 2.1 documentation, but has not yet been fully reviewed or moved to its new home.

1. Overview

This how-to describes how to enable and start the flow debugger in Cocoon Web Applications that use the JavaScript based flow engine. The flow debugger allows you to visually debug flow scripts using breakpoints, tracing, variable watches, etc, similar to other common visual debugging environments currently available.

For a full description of how to use the debugger itself, please refer to the Mozilla Rhino debugger page.

Since: 2.1 2002-12-07

2. Purpose

The process of writing and debugging flow script can be tedious and repetitive. JavaScript is a language that features typed objects but untyped references, which means at times it can be difficult to locate programming errors until runtime.

The flow debugger intends to ease the development, debugging and maintenance of flow scripts by allowing you to visually inspect and influence your flow script, while it's running, without having to resort to many superfluous print or log statements.

3. Intended Audience

This document is intended for Cocoon web application developers who are using the JavaScript flow engine inside of their Cocoon application.

4. Prerequisites

Until Cocoon 2.1 is released, to use the flow debugger you will need a CVS version or development snapshot of Cocoon, at least as recent as 7th December 2002.

You will also need a JavaScript flow based web application, for example the flow webapp samples that are currently shipped with Cocoon, and you will need to be running Cocoon on a server that has a display attached (either local or remote).

No special files need to be installed to use the debugger, support for the debugger is included in the Rhino jar file which is part of Cocoon.

5. Steps

The process of enabling and starting the debugger is quite simple. Essentially, all you have to is enable support for the debugger in the JavaScript interpreter's configuration section of the cocoon.xconf file, and start your application. The debugger will be instantiated by the next request that invokes map:call function or continuation.

Once the debugger is up and running, it should look similar to:

Flow debugger image

So, let's get started.

Copyright © 1999-2005 The Apache Software Foundation. All rights reserved.

5.1. Configuration

By default, the flow debugger is disabled. To enable it, you will need to modify the cocoon.xconf file to include one element named **debugger**, which contains the value *enabled*, as follows:

<?xml version="1.0"?> <flow-interpreters default="JavaScript" logger="flow">
<component-instance name="JavaScript"
class="org.apache.cocoon.components.flow.javascript.JavaScriptInterpreter">
<load-on-startup>resource://org/apache/cocoon/components/flow/javascript/system.js</load-on-startu
<reload-scripts>true</reload-scripts> <check-time>4000</check-time>
<debugger>enabled</debugger> <!-- JavaScript Debugger support -->
</component-instance> </flow-interpreters> If you are using the debugger with the Cocoon flow samples, you will need to modify the *cocoon.xconf* file in build/cocoon/webapp/WEB-INF, or the *flow.xconf* file in src/java/org/apache/cocoon/components/flow/, which is merged into the cocoon.xconf file at build time

5.2. Compile and start your environment

Compile and start your application as normal.

5.3. Start the flow engine

Once your application is running and awaiting requests, attempt to access a page that calls a flow function. The invocation will take longer than normal, but on your server's display you should see the visual debugger being created and populated with your flow script.

Portions of Cocoon's flow script management code will also be viewable in the debugger.

A debugger instance is created per JavaScript interpreter. This means if you make another request to a different sitemap for example, a new debugger instance will be created alongside the previous one. This allows you to debug flow scripts locally to the sitemap they are defined in.

5.4. Use the debugger

At this stage, the debugger is in control. By default, all entry points into the debugger result in an automatic breakpoint being set at that entry point (ie. at the beginning of each call function and handle continuation invocation). To continue processing, press any of the *step* buttons to trace through your code.

Pressing *Exit* from the *File* menu will not actually exit the debugger or JVM, but will make it invisible. It will be automatically made visible again upon entering a new call-function/handle-continuation frame.

Using the flow debugger, you will be able to trace through your code at any level of detail your require. Continuations are also supported, when a sendPageAndWait() is invoked, the debugger will become inactive until the interpreter is called upon to continue that suspended flow.

6. Improvements

6.1. Remote debugging

Currently to use the flow debugger you really need to be working on the same machine where your Cocoon application is running, either via a remote login or locally on the system. The ability to

Copyright © 1999-2005 The Apache Software Foundation. All rights reserved.

remotely attach to a running system and debug flow would be nice.

6.2. Enable/disable without restart

Currently to enable/disable the debugger one has to modify the cocoon.xconf/flow.xconf files, perhaps it would be good if this was possible dynamically, ie. without a restart of the servlet environment or Cocoon.

7. Comments

Do you have an idea about how to improve the Cocoon flow debugger? Please post it to the cocoon-dev mailing list. Care to comment on this How-To? Help keep this document relevant by passing along any constructive feedback to the cocoon-docs mailing list.

8. Revisions

11-12-2002: Content originally submitted by Marcus Crafter.

1. Comments

add your comments

Copyright © 1999-2005 The Apache Software Foundation. All rights reserved.