

RETURN TO MAIN MENU

# **Recital Web Developer**

## **Recital Web Developer**

Recital Corporation,  
100 Cummings Center, Suite 318J  
Beverly, MA 01915

Recital may have patents and/or patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents.

COPYRIGHT ©1988-2006 Recital Corporation. All rights reserved. All Recital products are trademarks or registered trademarks of Recital Corporation, Inc. Other brand and product names are trademarks or registered trademarks of their respective holders.

Last Updated June, 2006

## Developing Web Database Applications with Firecat

---

This section provides details on how to generate and display dynamic HTML pages using Recital Firecat.

### Overview

Recital Firecat is a data centric HTTP Server that incorporates a Database and 4GL. Using Recital Firecat you can generate dynamic content web pages with the Recital/4GL. Recital Firecat can coexist with the existing Web Server on your system, so there is no disruption to your existing website infrastructure. Using Recital Firecat, you can generate rich reports that can be integrated in with Recital Mirage .NET applications or Recital Terminal applications.

### Firecat/RSP Basics

The Recital Firecat HTTP server handles the generation of dynamic content HTML and sends this back to the browser that requested the page. These pages have a “.rsp” extension (Recital Server Page). Recital Firecat can serve static html pages, images, and Recital Server Pages (.rsp pages) similar to the way PHP dynamic pages are generated. With Recital Firecat .rsp pages, you can embed Recital/4GL scripting commands inside HTML documents. Recital Server Pages are really just HTML files with Recital/4GL scripting embedded within them. Any scripting commands that are embedded within the file are executed then removed from the results. All scripting code is hidden from the user viewing your web pages.

### The Tags of Firecat/RSP

To distinguish the RSP code from the regular HTML inside the .rsp file, RSP code is placed between <% and %> tags. The tag combination notifies the Firecat HTTP Server that the code within the <% and %> should be executed by the server and removed from the results. Any output generated by the Recital/4GL code (e.g. using the ? command) will be written to the HTML output that is sent back to the web browser that requested the page.

The following example generates an HTML table that lists all the records in the Suppliers table.

```
<table>
<%
    set sql to vfp
    open database southwind
    use suppliers
    scan
        ? "<tr>"
        for f=1 to fldcount()
            ? "<td>"
            ? &(field(f))
            ? "</td>"
        next
        ? "</tr>"
    endscan
    use
    close databases
%>
</table>
```

## RETURN TO MAIN MENU

You can also echo the result of inline expressions directly using the `<%=` and `%>` tags.

```
<input type="text" name="customer_name" value="<%= field(1) %>">
```

Firecat/RSP also recognizes several directives. These are enclosed within `<% @` and `%>`. The following directives are handled.

```
<%@ include="xxx" %>
```

This directive allows you to include either other RSP files, JavaScript files or HTML files within your .rsp page. This provides the ability to build library pages that can be re-used by many .rsp pages.

```
<%@ codebehind=".wsp filename" %>
```

This directive allows you to write a .wsp file (Recital program file with a .wsp extension) which can be used to dynamically generate and output the HTML. The advantage of using this approach is that you can edit the .prg file in the Recital Enterprise Studio with full color syntax highlighting and IntelliHelp. If you use this approach, it is advisable to keep all of the .wsp files in the scripts subdirectory of webroot. e.g.

```
<%@ codebehind="scripts/myproc.wsp" %>
```

When you use Recital Enterprise Studio to develop and test your .rsp pages, it will automatically synchronize JavaScript and .wsp files into the wwwscripts directory and images (.gif, .jpg, and .png) into the wwwimages directory.

## Quick Start Guide

---

1. Create or add an existing RSP page to your project. To add existing RSP pages to the project, select Project|Add Files To Project... from the Menu Bar .
2. Test the RSP page by right clicking on it in the “Project Explorer” then choose “Run”.

### Tip

If you use the attribute `<%@ codebehind="scripts/filename.wsp" %>` in the RSP file, you can edit the Recital 4GL code with full syntax color highlighting and IntelliHelp.

3. Use the `showdocument()` function in your server based Mirage application to send the request for an RSP page and for the content to be generated dynamically and then rendered in the browser on the desktop PC.

```
showDocument("http://" + getLocalHost() + ":8001/myrspfile.rsp",  
"_blank")
```

### Note

By default the Firecat web server listens on port 8001 so this must be post fixed to the hostname that it is running on as is specified in the sample code above. You can however configure it to run on port 80 too, or alternatively use the Recital ISAPI filter on Windows or the Apache loadable module (`mod_recital`) on Linux/UNIX.

## Setting a default page to display

---

The `DB_WWWDEFAULT` environment variable can be set to point to a default page that will be referenced if a URL is given which does not have a page specified. If this is not found, then Firecat will use the page `default.rsp`. e.g.

`http://localhost:8001`

Would reference:

`http://localhost:8001/default.rsp`

## Firecat/RSP Array Variables

Server configuration and request information, including form parameters are accessible from several public arrays that are created automatically by the Firecat web server when it is servicing a request for a page. The elements stored in these public arrays can be accessed by name e.g.

```
// get the hostname for the remote machine requesting the page
host = _server["REMOTE_HOST"]
```

### The `_SERVER[ ]` array

Argument	Description	Example
SERVER_SOFTWARE	A string that identifies the server	Apache/1.3.22 (Unix)
SERVER_NAME	The hostname, DNS alias, or IP address for self referencing URLs	www.yoursite.com
SERVER_PROTOCOL	The name and revision of the requested protocol	HTTP/1.1
SERVER_PORT	The server port number to which the request was sent	8001
REQUEST_METHOD	The method the client used to fetch the document	GET
PATH_INFO	Extra path elements given by the client	/list/records
PATH_TRANSLATED	The value of the PATH_INFO translated by the server into a full path	/usr/recital/webroot/list/records
SCRIPT_NAME	The URL path of the current page	/list/records/customers.rsp
QUERY_STRING	Everything after the ? in the URL	count=10&data=true
REMOTE_HOST	The hostname of the machine that requested this page	mypc.mydomain.com
REMOTE_ADDR	A string containing the IP address of the machine that requested the page	192.168.0.100
CONTENT_TYPE	The content type of the information attached to queries such as PUT and POST	x-url-encoded
CONTENT_LENGTH	The length of the information attached to queries such as PUT and POST	3866
AUTHORIZATION	The authorization realm	BASIC (This is the only one supported)
USERNAME	The username specified after response.authenticate()	
PASSWORD	The password specified after response.authenticate()	

### The `_GET[ ]` and `_POST[ ]` arrays

It is easy to process forms in the Firecat/RSP as the forms parameters are available in the `_GET[ ]` and `_POST[ ]` arrays. Use these arrays to access form parameters from your Recital/4GL code in the Firecat/RSP page. The keys to the array elements are the parameter names and the values are the values of those parameters.

### The `_COOKIE[ ]` array

This array contains all of the cookies that are sent from the client browser when a page is requested. You reference these by cookies name. You can send cookies to the client browser (session state can be saved and restored) using the `response.addcookie()` method call. (See the response object below for further details).

### The `_ARGS[ ]` array

You can reference the value of any arguments passed to the Firecat/RSP page by referencing them by name in the `_ARGS[ ]` array.

### The Firecat/RSP Request Object

When a web page is requested, along with the HTTP request, information such as the URL of the web page request and the format of the data requested is also passed. It can also contain the name and contents of form variables on the user's form that requested the page. The response object allows you to control the way the server interacts with the browser

The methods that can be specified are as follows:

Argument	Description	Example
WRITE	Writes a string of text	<code>response.write("&lt;table&gt;")</code>
REDIRECT	Redirects to another URL	<code>response.redirect("http://www.recital.com")</code>
WRITEFILE	Writes out the contents of a file	<code>response.writefile("/usr/recital/temp/myreport.html")</code>
APPENDTOLOG	Writes a line of text to the Recital system log	<code>response.appendtolog("opening database southwind")</code>
FLUSH	Flushes out the buffers and sends to the browser	<code>response.flush()</code>
CLEAR	Clears (resets) the output buffer	<code>response.clear()</code>
ADDHEADER	Sends response headers	<code>response.addheader("Content-Type: text/plain")</code>
ADDCOOKIE	Sends a cookie to the browser	<code>response.addcookie("my_cookie_name", "my_cookie_value")</code>
AUTHENTICATE	Causes the browser to prompt for a username/password	<code>response.authenticate()</code>
IMPERSONATEUSER	Impersonates the specified user. This is used to handle file permissions for accessing pages.	<code>response.impersonateUser("username","password")</code>

Special note: When sending response headers or cookies, these must be sent before any text that is output.

## Setting up the Apache loadable module

---

This section provides details on how to install the Recital Firecat Apache loadable module for Apache 1.3 on Linux and UNIX.

### Enable DSO (Dynamic Shared Object) support in Apache

The DSO support for loading individual Apache modules is based on a module named `mod_so.c` which has to be statically compiled into the Apache core. It is the only module besides `http_core.c` which cannot be put into a DSO itself (bootstrapping!)

Your Apache HTTP Server must have this compiled into it in order to use the Recital Firecat Apache Module. If it is not then you must re-configure the Apache HTTP Server as follows in the directory that contains the Apache source code:

```
./configure --enable-module=so
make
make install
```

### Load the Recital loadable module

#### Version 1.3

1. Stop Apache

```
/usr/local/apache/bin/apachectl stop
```

2. Edit the apache `httpd.conf` file

```
vi /usr/local/apache/conf/httpd.conf
```

In the `LoadModule` section add the following line:

```
LoadModule recital_module libexec/mod_recital.so
```

In the `AddHandler` section add the following lines:

```
AddHandler recital-handler .rsp
AddHandler recital-handler .wsp
```

In the `Location` section add the following lines:

```
<Location /recital>
  SetHandler recital-handler
</Location>
```

3. Install the Recital module with the Apache Extension Module tool

```
/usr/local/apache/bin/apxs -i -n recital -a /usr/recital/apache/mod_recital.so
```

4. Restart Apache

```
/usr/local/apache/bin/apachectl start
```

### Version 2.0

1. Stop Apache

```
/usr/local/apache/bin/apachectl -k stop
```

2. Edit the apache httpd.conf file

```
vi /usr/local/apache/conf/httpd.conf
```

In the AddHandler section add the following lines:

```
AddHandler recital-handler .rsp  
AddHandler recital-handler .wsp
```

3. Install the Recital module with the Apache Extension Module tool

```
/usr/local/apache/bin/apxs -i -n recital -a /usr/recital/apache/mod_recital2.so
```

4. Restart Apache

```
/usr/local/apache/bin/apachectl -k start
```

### Version 2.2

1. Stop Apache

```
/usr/local/apache/bin/apachectl -k stop
```

2. Edit the apache httpd.conf file

```
vi /usr/local/apache/conf/httpd.conf
```

In the AddHandler section add the following lines:

```
AddHandler recital-handler .rsp  
AddHandler recital-handler .wsp
```

3. Install the Recital module with the Apache Extension Module tool

```
/usr/local/apache/bin/apxs -i -n recital -a /usr/recital/apache/mod_recital2.2.so
```

4. Restart Apache

```
/usr/local/apache/bin/apachectl -k start
```