Configuring Apache for WebObjects

Overview

In a deployment scenario on Mac OS X, Linux, Solaris or even Windows, your applications will most likely use Apache to server static resources. Additionally, if you develop with WODirectConnectEnabled=false (you **should**, see the Direct Connect section for details), you will be running your application locally through Apache as well. Apache is a very extensible web server that provides a huge number of capabilities, some of which we will detail here.

Split Install

WebObjects applications are deployed in a "split install". A split install means that your application code, components, and resources are deployed in one location to be served from your WebObjects application (on OS X, `/Library/WebObjects/Applications/`*`YourApp`*`.woa`), while your `WebServerResources` are installed in another location (on OS X, `/Library/WebServer/Documents/WebObjects/`*`YourApp`*`.woa/Contents/WebServerResources`) to be served directly by Apache. This provides the optimal performance scenario, as Apache is specifically tuned for serving static content, and it does not make sense to send requests for large binary files through WebObjects if it is not necessary.

mod_expires

To get the most performance out of Apache, you should make sure that you have mod_expires enabled. mod_expires controls the caching headers that are applied to static resource requests. Depending on your installation, Apache may default to mod_expires disabled, which would cause your end-users' browser to re-request every resource on your site on every page, even if it's a common header graphic.

An example mod_expires configuration might look like:

```
<IfModule mod_expires.c>
  ExpiresActive On
  ExpiresDefault A60
  ExpiresByType image/bmp A3600
  ExpiresByType image/gif A3600
  ExpiresByType image/ief A3600
  ExpiresByType image/jpeg A3600
  ExpiresByType image/png A3600
</IfModule>
```

You will also need the corresponding type-extension mappings:

```
<IfModule mod_mime.c>
  AddType image/bmp bmp
  AddType image/gif gif
  AddType image/ief ief
  AddType image/jpeg jpeg
  AddType image/jpeg jpg
  AddType image/jpeg jpe
  ...
</IfModule>
```

This tells Apache that when a request is made for a type image/gif, the requesting browser will be told not to request the image again for an hour (A3600 = 3600 seconds).

mod_rewrite

Anyone who has used WebObjects has likely noticed that WebObjects URLs are long `http://yoursite.com/cgi-bin/WebObjects/AppName.woa/wa/something`. It is a common request to make these URLs nicer for end-users who are used to just requesting `http://yoursite.com`. Fortunately Apache provides an amazingly extensive module called "mod_rewrite" that allows you to rewrite the URL requests of your site based on a series of regular expressions and rules.

Aaron Rosenzweig has a very thorough article about using mod_rewrite with Apache.

mod_rewrite with mod_webobjects

I ran into a problem with mod_rewrite when using mod_WebObjects where mod_WebObjects had be loaded first or it just wouldn't work properly (it would work fine with cgi-bin adaptor).

So in http.conf, search for mod_rewrite and change it to:

```
LoadModule WebObjects_module    /System/Library/WebObjects/Adaptors/Apache/mod_WebObjects.so
LoadModule rewrite_module       libexec/httpd/mod_rewrite.so
```

, find again:

```
AddModule mod_WebObjects.c
AddModule mod_rewrite.c
```

There's still a load module in /System/Library/WebObjects/Adaptors/Apache/apache.conf, but you can just ignore it - it produces a warning about being loaded twice.

Mike Schrag

Here's an example mod_rewrite we use on one of our apps:

```
<IfModule mod_rewrite.c>
  RewriteEngine On
  RewriteRule ^/$ /page/HomePage [R]
  RewriteCond %{QUERY_STRING} ^appNum=([-0-9]+)(.*)$
  RewriteRule ^/page/(.*)$ /cgi-bin/WebObjects/AppName.woa/%1/wa/viewPage?pageName=$1%2 [L,PT]
  RewriteRule ^/page/(.*)$ /cgi-bin/WebObjects/AppName.woa/wa/viewPage?pageName=$1 [L,PT,QSA]
</IfModule>
```

The WOA produces URLs in the format `http://site.com/page/HomePage?appNum=2`, which turns into `http://site.com/cgi-bin/WebObjects/AppName.woa/2/viewPage?pageName=HomePage`.

Ray Kiddy

This worked for me (September 2020) and seems easier to parse. I wanted to change all of "http://opencalaccess.org/cgi-bin/WebObjects/app.woa/" to "http://opencalaccess.org/app/" and this worked.

> *RewriteEngine On*
>
> *RewriteRule ^/$ /app [R]*
>
> *RewriteRule ^/index.html$ /app [R]*
>
> *RewriteRule ^/app((.*))?$ /cgi-bin/WebObjects/app.woa$1 [PT,L]*

I inserted this into my file: /etc/apache2/sites-enabled/opencalaccess.org.conf which specifies the values particular to this one domain.

I added this as arguments to the app:

> *-Der.extensions.ERXApplication.replaceApplicationPath.pattern=/cgi-bin/WebObjects/app.woa*
>
> *-Der.extensions.ERXApplication.replaceApplicationPath.replace=/app*

And you can, of course, do this in the Properties file as well. Much thanx to Stefan Gärtner on the mailing list (Subject: Re: Apache rules and SSL, 2020/08/27)

Jeff Schmitz

This one stumped me for a couple days, so thought I'd add it. Was trying to add mod_rewrite functionality as described above, and things went well on my dev machine by adding the rewrite rules just to the /etc/apache2/httpd.conf file. However, on the deployment machine I also had to add them to the /etc/apache2/sites/0000_any_.conf file.

WebObjects Adaptor for Apache 2.2

Travis Cripps

A number of people have expressed interest in using the WebObjects adaptor with Apache 2.2.x. I finally gotten a chance to sit down and work on it today. I'm writing to let you know that it's available in the Project Wonder CVS repository. (also there are precompiled binaries for various OS available)

The necessary changes turned out to be mostly minor updates to change calls to outdated/deprecated functions. The biggest (and non-trivial) change was for SSL support. It's been re-written to use Apache's mod_ssl module.

I've tested with MacOS X 10.4.7, Apache 2.2.2, with and without ssl support. It works in all of my tests.

Configuration of the web server to work with the adaptor turned out to be surprisingly challenging, due to the new, very strict default access rules that ship in Apache 2.2.x httpd.conf file. Once I discovered that, it was trivial to change the setting, but it's worth mentioning here to save some people a lot of frustration.

The new default configuration is:

```
<Directory />
   Options FollowSymLinks
   AllowOverride None
   Order deny,allow
   Deny from all
</Directory>
```

Your options are to comment out the last 2 lines of that block, or to override them in a VirtualHost block. Just setting the usual Location block didn't seem to work for me.

And, of course, either change the name of the WebObjectsAlias setting from /cgi-bin/WebObjects to <foo>/WebObjects or comment out the ScriptAlias definition for the /cgi-bin/ directory.

Note

The default `ScriptAlias` directive in the 10.5 and 10.6 `httpd.conf` files is:

```
ScriptAliasMatch ^/cgi-bin/((?!(?i:webobjects)).*$) "/Library/WebServer/CGI-Executables/$1"
```

This prevents `/cgi-bin/WebObjects` from matching, so no change to `WebObjectsAlias` or `ScriptAlias` is necessary.

Other than these tips, it's pretty much the standard compilation and installation, and configuration.

1. Alter the make.config file in the Adaptors directory of the Wonder repository to reflect your apache installation setup.*
2. Run make to build the Adaptor
3. Curse because of that one setting you forgot. Fix it.
4. make clean; make
5. Install the mod_WebObjects module with apxs
6. Configure your httpd.conf and either link or copy the WebObjects directory from the standard location (if on MacOS X) to your new htdocs directory.
7. apachectl configtest; apachectl graceful
8. Test.
9. Curse again. Change the httpd.conf file as necessary.
10. apachectl graceful. Go to 8 as necessary.
11. Finally! apachectl graceful

Enjoy your shiny new WO adaptor. 🙂

- Note: if you are getting the error

  ```
  libtool: compile: unable to infer tagged configuration
  libtool: compile: specify a tag with `--tag'
  apxs:Error: Command failed with rc=65536
  ```

Add to the end of your make.config the following:
`CC = gcc`

Webobjects Adaptor for Apache 2.4

Access control

Check out http://httpd.apache.org/docs/2.4/upgrading.html#access

In 2.2, access control based on client hostname, IP address, and other characteristics of client requests was done using the directives `Order`, `Allow`, `Deny`, and `Satisfy`.

In 2.4, such access control is done in the same way as other authorization checks, using the new module `mod_authz_host`. The old access control idioms should be replaced by the new authentication mechanisms, although for compatibility with old configurations, the new module `mod_access_compat` is provided.

2.2 configuration:

```
<Location /cgi-bin/WebObjects/>
        Order allow,deny
        Allow from all
</Location>
```

2.4 configuration:

```
<Location /cgi-bin/WebObjects/>
        Require all granted
</Location>
```