Rewrite URLs with Apache

Your Rating: ☆☆☆☆☆ Results: ★★★☆☆ 134 rates

Since Magnolia 4.3, Magnolia CMS Enterprise Edition Pro supports native multi-site deployment, which makes it considerably easier to run multiple websites within one installation than the following instructions, which are for the Community Edition. For details see https://documentation.magnolia-cms.com/display/DOCS/Multisite (or for Magnolia 4.5 https://documentation.magnolia-cms.com/display/DOCS45/Multisite)

Requirement

Magnolia Community Edition is not designed to run multiple websites in a single instance, so workarounds are necessary to achieve this.

The obvious solution is to have multiple instances. To make nice URLs as well (i.e. without `MagnoliaPublic`) requires a solution using Tomcat Virtual hosts (this is addressed by Tomcat virtual hosts and Running Magnolia in the root of a host. Those articles describe the setup for one website, but the solution can easily be generalised for multiple sites. However, it quickly becomes unwieldy as the memory requirements build up, and I found that about four sites was the practical limit in my system.

The next solution is to put all the sites in one instance (which may have a nice URL using the technique just linked to). This is commonly done by allocating top-level nodes to each site and setting up the menu to start displaying at a level lower so that each site displays only its own pages. The Tomcat Virtual Server can be made to answer to the URLs of all the sites concerned. The problem is now that all pages in each site have that site's top-level nodename as part of all URLs; neither Magnolia nor Tomcat has a mechanism to change this.

So the solution has to be found outside, and the most general one is to use Apache `httpd` as a front-end reverse proxy. The power that this brings enables virtually any URL rewriting desired, and so can handle the changes required if a default Magnolia installation is used as the basis for multiple hosts.

The use of Apache as a reverse proxy also has the capability of passing requests to other non-Tomcat sites, say to an IIS server, or of serving some sites from Apache itself. Proxied sites can be on different machines, and load balancing is a possibility. There is much (confusing!) Apache documentation dealing with these matters.

Method

The use of Apache for URL rewriting means that no changes need to be made to your current Magnolia configuration. And if you are starting from scratch, you can use the default Magnolia configuration, modified only to the extent necessary to separate the websites into separate trees, maybe with separate template, docroot, and DMS subtrees and so-on; separating the sites like this makes it easier to provide access control so that editors can be set up to see only their own site and its templates. These matters are touched on elsewhere in the wiki, and on the Magnolia mailing list.

The installation of Apache is straightforward if Tomcat is running on port 8080 or 8081 (which are the defaults in different versions) so that port 80 is free (the ports can be changed for testing purposes, of course, but I won't complicate this article with a discussion of that). After installation you edit `httpd.conf` to define the virtual hosts and the document root for any pages you want Apache to serve directly. It may be necessary to uncomment the load commands for the modules used here:

```
* mod_deflate
* mod_rewrite
* mod_proxy
* mod_proxy_html
```

The last of these may need to be compiled in (Linux) or installed separately (Windows), but is crucial to the particular operation we are undertaking.

The guts of this is placed in the `Virtual Hosts` section of `httpd.conf`. Again, Apache documentation of virtual hosts is very confusing, because it either introduces all the different options immediately, or only describes setups a bit different from what you want - so I will simply give a formula which you can use the documentation to understand if you wish. I relied heavily on this page and this page when putting this together.

You must have a `NameVirtualHost` directive, followed by a `VirtualHost` block for each virtual host. Below I show just my first virtual host; the first example assumes that the public instance is running in the root of a Tomcat virtual host; the second example (added later) is using a default Magnolia installation - the comments to it modify some of the earlier remarks.

Example 1 (Magnolia 2.1.5 with the public instance running in the root of the host)

```
NameVirtualHost 82.70.166.73

<VirtualHost 82.70.166.73
    ServerName ambisonic.info
    ServerAlias www.ambisonic.info
    RewriteEngine on
    RewriteRule ^/$       /index.html [P]
    ProxyPass   /docroot/ http://ambisonic.info:8081/docroot/
    ProxyPass   /author/  http://ambisonic.info:8081/author/
    ProxyPass   /ai/      http://ambisonic.info:8081/ai/
    ProxyPass   /         http://ambisonic.info:8081/ai/
    <Location /docroot/>
        ProxyPassReverse /docroot/
    </Location>
    <Location /author/>
        ProxyPassReverse /author/
    </Location>
    <Location />
        ProxyPassReverse /
        SetOutputFilter INFLATE;proxy-html;DEFLATE
        ProxyHTMLURLMap /ai/ /
    </Location>
</VirtualHost>
```

Explanations for Example 1

```
ServerName ambisonic.info
ServerAlias www.ambisonic.info
```

Obvious, really.

```
RewriteEngine on
RewriteRule ^/$ /index.html [P]
```

If I make the base node of the site a page, the menu doesn't open to show the other pages - so I make the home page `index.html` under that node. But when the sites are subnodes, the usual Magnolia mechanism for redirecting `/` to `/index.html` doesn't work (for me), so I do that rewrite here. The [P] tells the rewrite engine to continue to the proxy directives instead of assuming it has done all the work required.

```
ProxyPass /docroot/ http://ambisonic.info:8081/docroot/
ProxyPass /author/  http://ambisonic.info:8081/author/
ProxyPass /ai/      http://ambisonic.info:8081/ai/
ProxyPass /         http://ambisonic.info:8081/ai/
```

These are scanned in order, so must run from the particular to the general. The last of these directives tells Apache to forward requests for `ambisonic.info/...` to `ambisonic.info:8081/ai/...`, and so removes the necessity for `/ai/`, which is the base node of the site, to be used externally. The previous directive accepts the `/ai/` if it is present, to avoid doubling it - this handles existing bookmarks and links, which I wish to honour transparently (the site had already been publicised in the form with the `/ai/` present). The first two handle `/docroot/` and `/author` without translation; if you separate your docroot for the different sites, this would need appropriate modification (see second example below).

```
<Location /docroot/>
    ProxyPassReverse /docroot/
</Location>
<Location /author/>
    ProxyPassReverse /author/
</Location>
```

These directives do the opposite URL rewriting on headers coming out of the server (in this instance, there is explicitly no change for `/docroot/` or `/author/`). They are an expanded version of the directive:

```
ProxyPassReverse /docroot/ http://ambisonic.info:8081/docroot/
```

which I used initially. The expanded form, I understand, deals more effectively with some incorrect header formats (you will see in the discussion of the second example below that I have dropped the expanded form as Magnolia doesn't need it).

```
<Location />
    ProxyPassReverse /
    SetOutputFilter INFLATE;proxy-html;DEFLATE
    ProxyHTMLURLMap /ai/ /
</Location>
```

Here's the meat of it. The `ProxyPassReverse` removes the `/ai/` from the headers that Magnolia has sent back. However, the big issue is interpage links; `ProxyHTMLURLMap` scans the html for links and removes `/ai/` from any that contain it. The `SetOuputFilter` undoes the HTML compression so that the scanning can take place, calls the html scanner, and then recompresses the output. (See also the additional comments following the next example.)

Example 2 (Magnolia 3.5.4 default bundle installation)

```
<NameVirtualHost 163.1.194.162 />

<VirtualHost 163.1.194.162>
    ServerName magnolia.clinpharm.ox.ac.uk
    RewriteEngine on
    RewriteRule      ^/$ /magnoliaAuthor [P]
    ProxyPass        /   http://localhost:8080/
    ProxyPassReverse /   http://localhost:8080/
</VirtualHost>

<VirtualHost 163.1.194.162>
    ServerName  afrox.org
    ServerAlias www.afrox.org
    RewriteEngine on
    RewriteRule      ^/$       /index.html [P]
    ProxyPass        /docroot/ http://localhost:8080/magnoliaPublic/docroot/afrox/
    ProxyPass        /dms/     http://localhost:8080/magnoliaPublic/dms/afrox/
    ProxyPass        /         http://localhost:8080/magnoliaPublic/afrox/
    ProxyPassReverse /docroot/ http://localhost:8080/magnoliaPublic/docroot/afrox/
    ProxyPassReverse /dms/     http://localhost:8080/magnoliaPublic/dms/afrox/
    ProxyPassReverse /         http://localhost:8080/magnoliaPublic/afrox/
    SetOutputFilter INFLATE;proxy-html;DEFLATE
    ProxyHTMLURLMap /magnoliaPublic/docroot/afrox/ /docroot/
    ProxyHTMLURLMap /magnoliaPublic/dms/afrox/     /dms/
    ProxyHTMLURLMap /magnoliaPublic/afrox/         /
    ProxyHTMLDoctype XHTML
</VirtualHost>
```

Additional Explanations for Example 2

In this case I didn't make a separate author URL for each site; instead I have a single authoring site where I (as manager) can see everything, but the editing users only get to see their own site or sites. For this virtual server, then (the first of the two above), there is no attempt to make a nicer URL by hiding bits of it, not even the `/magnoliaAuthor/`.

```
RewriteEngine on
RewriteRule      ^/$ /magnoliaAuthor [P]
ProxyPass        /   http://localhost:8080/
ProxyPassReverse /   http://localhost:8080/
```

All that there is here is a rewrite to get the main URL to point to the Magnolia authoring instance, `ProxyPass` to send the request on to the Tomcat, and `ProxyPassReverse` to rewrites URLs in returned headers. There is no need for `ProxyHTMLURLMap`, which rewrites URLs in the returned HTML, as the URLs are not being changed (the server name is not included in the URLs in the htlp from Magnolia).

The second server is recognisably like the one in the first example, with some developments:

```
ProxyPass        /docroot/ http://localhost:8080/magnoliaPublic/docroot/afrox/
ProxyPass        /dms/     http://localhost:8080/magnoliaPublic/dms/afrox/
ProxyPass        /         http://localhost:8080/magnoliaPublic/afrox/
ProxyPassReverse /docroot/ http://localhost:8080/magnoliaPublic/docroot/afrox/
ProxyPassReverse /dms/     http://localhost:8080/magnoliaPublic/dms/afrox/
ProxyPassReverse /         http://localhost:8080/magnoliaPublic/afrox/
```

Here there are now extra lines for the DMS. Also note that the `docroot` and `dms` trees have the site name appended - this hides the root of the subtree that I have provided in Magnolia for each site, while retaining the words `docroot` and `dms` to identify where the data is. As stated above, remember to order these from the particular to the general, as the rule with just "/" will always match; I got this wrong again when making this new implementation, and got really messed up before I realised!

I decided, after careful reading and some tests that the circumlocution involving the `<Location>` brackets was not necessary, so the `ProxyPassReverse` and `ProxyHTMLURLMap` statements are given simply:

```
ProxyPassReverse /docroot/ http://localhost:8080/magnoliaPublic/docroot/afrox/
ProxyPassReverse /dms/     http://localhost:8080/magnoliaPublic/dms/afrox/
ProxyPassReverse /         http://localhost:8080/magnoliaPublic/afrox/
SetOutputFilter INFLATE;proxy-html;DEFLATE
ProxyHTMLURLMap /magnoliaPublic/docroot/afrox/ /docroot/
ProxyHTMLURLMap /magnoliaPublic/dms/afrox/     /dms/
ProxyHTMLURLMap /magnoliaPublic/afrox/         /
ProxyHTMLDoctype XHTML
```

Note the last line, which I missed last time. Magnolia emits a `DOCTYPE` at the start of the page. However, `mod_proxy_html` does not pass this on; but instead, it provides a command to put the `DOCTYPE` in for itself. I don't pretend to know why...

Lastly, `mod_proxy_html` has some facilities I don't use. It can also scan scripts and CSS files for urls, but does not by default; you may have circumstances when this is required. The command to turn this on is `ProxyHTMLExtended On`, but it should only be used when required because of the extra overheads involved. And finally, mainly for use with scanning of scripts, the search in `ProxyHTMLURLMap` can be a regular expression, if the flag `R` is added to the statement; this is documented on this page.

Coda

All this has been done using Apache v2.2.4 for Windows, with mod_proxy_html v2.5.2. The Magnolia websites are running in v2.1.5 for the first example, and v3.5.4 for the second example.

Paul Hodges (27 July 2007; revised and extended 23 Feb 2008)