TomcatModProxyHTML

Fixing broken links when using a reverse proxy to Tomcat

In this How-To guide, we will show you how to fix broken links when using a reverse proxy between your Apache webserver and your Tomcat server.

Prerequisites

For this you are going to need the following

mod_proxy_html		http://apache.webthing.com/mod_proxy_html/mod_proxy_html.c		mod_proxy_html to fix hyperlinks.
libxml2		http://xmlsoft.org/		Needed by mod_proxy_html. Should be included in most, if not all, Linux distributions.

mod_proxy_html directives can be used in following contexts - server config, virtual host, and in **<Location>**. The examples below will be in the server config context as well as pertain to Apache that has been compiled with the source package from http://httpd.apache.org.

To fix the links in Tomcat pages, we are using mod_proxy_html. This is necessary for proxied pages that use links that are resolved with a URI path or with an absolute URL.

```
<a href="/somefile">
<a href="http://tomcat.server.com/">
```

Links in proxied pages that are set up in the manner above will not work. The first will resolve to http://localhost/somefile which is incorrect compared to the way the reverse proxy to Tomcat was set up. To correctly resolve **somefile**, the link would need to use http://localhost/webapps/somefile. The second link example will yield a "No such server" error for the browser. This also affects images and other resources linked this way on the proxied page. Thus the need to fix these link types with mod_proxy_html.

If you can guarantee that all your links will be of the form:

```
<a href="somefile">
```

then using mod_proxy_html **will not** be necessary. Although, for Tomcat, the management page will contain bad URIs.

Since mod_proxy_html is not included in the Apache package, you need to either download and compile it or get your distro's package, if available. To compile it, we use the **apxs** binary that comes with the Apache package.

```
/path/to/apxs -cia -I/path/to/include/libxml2 mod_proxy_html.c
```

/path/to/ is the path to where apxs and your libxml2 header files are at. You only need to specify the **-I/path/to/include/libxml2** if the libxml2 header files are not in a standard location. Using the apxs command will compile and build the module. It will also put in the correct **LoadModule** in httpd.conf:

```
LoadModule proxy_html_module  modules/mod_proxy_html.so
```

Note: If you get errors starting up Apache regarding mod_proxy_html and libxml2, you can add the **LoadFile** directive below before the **LoadModule** directives.

```
LoadFile /path/to/libxml2.so
```

Make sure **/path/to** is your real system path to the libxml2.so library.

With the module compiled, built and loaded, we can now use mod_proxy_html to fix links. To use the mod_proxy_html directives, you need to make sure that if **<IfModule mime_module>** is being used, then you must put the directives in that **<IfModule>** block. If this block isn't being used, then you can put the directives anywhere in your server configuration file. The Tomcat management page will be used in the following example.

```
# Fix mod_proxy urls.
SetOutputFilter proxy-html
ProxyHTMLURLMap ^/(.*)$ /webapps/$1 R
```

As you can see from the example above, mod_proxy_html can use regular expressions to do its matching which is denoted by the 'R' at the end of the ProxyHTMLURLMap lines. You can check the management page by accessing http://localhost/webapps and clicking the **Tomcat Manager** link (http://localhost/webapps/manager/html). Click on the links on this page and you will see that all the links resolve correctly. If we didn't fix the links, then all the links on Tomcat's management page will yield 404 errors from the browser.

For more complete information on mod_proxy_html, visit Nick Kew's mod_proxy_html page.