CocoonAndApacheModRequest

The expires function in cocoon 2.1 looks very exciting, though despite it being currently documented, I couldn't get expire to work in the 2.0 series.

This was too bad, since my webapp comprises some static content, like css files and generated jpg title files, that really don't need to be passed over with each hit. Furthermore, even my cocoon-generated xhtml and svg derives from static xml documents, so a client-side cache of, say, an hour or so, would ensure that the user wouldn't hit my cocoon server again as she's navigating over a previously viewed document. This is good for my server, but it also can appreciably improve the user's experience of the site.

As it happens, I'm serving cocoon through Apache, and *its* Expires module will do the job for cocoon if the setup is well-configured. (I'm using Cocoon 2.0.4 under Tomcat 4.1.15 on RH Linux 7.3 with a Apache 1.3, but I don't think that should matter much for these instructions.)

1. Make sure your Apache http.conf has:

```
LoadModule expires_module modules/mod_expires.so
```

and

```
AddModule mod_expires.c
```

somewhere in it. Mine did out of the box.

2. Make sure your mime-type configuring file has definitions for each mime-type you plan to cache on the client with Expire. On RH Linux, this would be /etc /mime.types, and I had to add a line to define svgs:

```
image/svg+xml svg svgz
```

3. Append the appropriate commands to your Apache httpd.conf, thus:

```
ExpiresActive On
ExpiresByType image/svg+xml A3600
ExpiresByType text/css      A3600
ExpiresByType image/jpeg    A3600
ExpiresByType image/png     A3600
```

The 'Axxx' means cache the content for xxx seconds after access. I'm using a rather conservative single hour. Mod_expires has other functions. Read about them at http://httpd.apache.org/docs/mod/mod_expires.html

4. Restart Apache, and test. I used the 'wget' utility under Linux to ensure that the 'Expire' header was being passed to the client. 'wget -S' includes the header nicely, so

```
wget -S http://localhost/heml-cocoon/site/style/international_names.css
```

gives:

```
--11:12:54--  http://localhost/heml-cocoon/site/style/international_names.css
          => `international_names.css.4'
Resolving localhost... done.
Connecting to localhost[127.0.0.1]:80... connected.
HTTP request sent, awaiting response...
 1 HTTP/1.1 200 OK
 2 Date: Fri, 10 Jan 2003 15:12:54 GMT
 3 Server: Apache/1.3.23 (Unix)  (Red-Hat/Linux) PHP/4.1.2
mod_gzip/1.3.19.1a mod_webapp/1.2.0-dev
 4 Cache-Control: max-age=3600
 5 Expires: Fri, 10 Jan 2003 16:12:54 GMT
 6 Content-Type: text/css
 7 Content-Length: 89
 8 X-Cocoon-Version: 2.0.4
 9 Connection: close

100%[====================================>] 89            86.91K/s    ETA 00:00

11:12:54 (86.91 KB/s) - `international_names.css.4' saved [89/89]
```

Hooray, we're caching this css file on the client for a few hours!

This approach has one advantage over setting the expire within cocoon itself. Client-side caching gets in the way of development, where one wishes immediately to see the results of changes in code. As it stands, my webapp is distibuted without c-s caching, so interested parties can fiddle with it, but once installed in the project's main server the webapp's content is automagically client-side cached.

Further Notes

Many Cocoon urls don't end with extensions. For instance, one might well provide a document entitled 'map' as a pdf, png or svg file. Unfortunately, the system set up above requires the extension to recognize the mime-type of the document, so if you use 'ExpireByType' as listed above, you'll have to ensure that all documents end with proper extensions.

An alternate solution is to use the 'ExpiresDefault Axxxx' command instead of 'ExpireByType' listed in no. 3 above. Using this, everything is given an expires http header. In my little experience so far, this has worked remarkably well. I'd recommend it.

A developer who wants to test recent changes on a live server that is running through Apache and expire can still see un-cached versions by hitting the Tomcat server itself at, usually, :8080