HostSelector

One of the Selector components

Using the Host Selector in a live environment

In an environment where more than one site is hosted on the some Cocoon instance, you'll probably use a different domain name for each site. To make this possible, the host selector is used. The host selector selects a certain pipeline according to the value of the host name. This is the part of the URL between 'http://' and the first slash, such as 'www.apache.org' in 'http://www.apache.org/index.html', consisting of both the subdomain (www) and the domain itself (apache.org).

When a request comes in, the host name of the URL requested is passed on to Cocoon. Within Cocoon it is then possible to conditionally execute pipelines depending on the host name using the host selector.

For example; we have a webserver running Cocoon with the following five host names:

- jakarta.apache.org
- xml.apache.org
- www.apache.org
- apache.org
- www.foobar.org

In this case we would have bought the domains 'apache.org' and 'foobar.org' from some domain name registry service. The DNS routes '*.apache.org' and '*.foobar.org' to the webserver running Cocoon. Tomcat is configured to send all requests to the Cocoon webapp, regardless of the host name. Now it is up to Cocoon to decide what to do when a request comes in.

It is a good use to let the main sitemap of Cocoon receive all requests, and pass the request to a subsitemap depending on the host name. For each host name, there is a sitemap defined that acts as the main sitemap for that site (the hostsitemap). You can let more host names share the same hostsitemap, too.

The host selector definition assigns internal names for use inside the sitemap to different host names. A simple <map:select> can then use this internal name to pass execution the correct pipeline.

The following is defined in the <map:components> section of the main sitemap:

```
<map:selectors>
        <map:selector name="host" logger="sitemap.selector.host"
                                src="org.apache.cocoon.selection.HostSelector">
                <host name="jakarta-apache" value="jakarta.apache.org"/>
                <host name="xml-apache" value="xml.apache.org"/>
                <host name="www-apache" value="www.apache.org"/>
                <host name="www-apache" value="apache.org"/>
                <host name="www-foobar" value="www.foobar.org"/>
        </map:selector>
</map:selectors>
```

As you can see, "www.apache.org" and "apache.org" share the same internal name, "www-apache". In this way, we can use the same hostsitemap for both sites (which will make them identical).

N.B. You wouldn't want to do this in a production environment; it would cause Google to index your site twice. "So what!" you say. Well, Google now penalizes sites with duplicate content... so you just reduced your sites ranking! Better to issue a permanent redirect response from one site to the other. - DavidLegg

You could also add a new host selector to the www-apache hostsitemap, which then again destinguishes between 'www.apache.org' and 'apache.org' using the same internal names.

In the main pipeline of the main sitemap the actual UnderstandingCocoonMounts is done:

```
<map:select type="host">
        <map:when test="jakarta-apache">
                <map:mount uri-prefix=""
                           src="file://home/sites/jakarta-apache/sitemap.xmap" />
        </map:when>
        <map:when test="xml-apache">
                <map:mount uri-prefix=""
                           src="file://home/sites/xml-apache/sitemap.xmap" />
        </map:when>
        <map:when test="www-apache">
                <map:mount uri-prefix=""
                           src="file://home/sites/www-apache/sitemap.xmap" />
        </map:when>
        <map:when test="www-foobar">
                <map:mount uri-prefix=""
                           src="file://home/sites/www-foobar/sitemap.xmap" />
        </map:when>
</map:select>
```

Notice that the uri-prefix is left empty in every <map:mount>. This lets the hostsitemap start with the root of the URI, directly after the host name (i.e. "index.html" in "www.apache.org/index.html").

It's another good use to put all the hostsitemaps in different directories on the server. Every hostname-directory can hold it's own set of stylesheets, resources, sitemaps, etcetera, allowing concurrent development on the different sites.

Using the `cocoon:/` CocoonProtocolExample inside any sitemap will point to a pipeline inside the sitemap itself. However, when using `cocoon://` inside a hostsitemap, you point to the root sitemap, which will automatically forward the request to the hostsitemap corresponding to the host name. This means that you can use `cocoon://` protocol from within any subsitemap, no matter how deep it's nested, to refer to the hostname-sitemap.

The following table shows the host selector process of the example above:

URL	**Host name**	**Internal name**	**Hostsitemap**	URI passed to the Hostsitemap	
http://www.apache.org/index.html	www.apache.org	www-apache	file://home/sites/www-apache/sitemap.xmap	index.html	
http://www.apache.org/foo/bar/index.html	www.apache.org	www-apache	file://home/sites/www-apache/sitemap.xmap	foo/bar/index.html	
http://apache.org/index.html	apache.org	www-apache	file://home/sites/www-apache/sitemap.xmap	index.html	
http://jakarta.apache.org/foo/bar/index.html	jakarta.apache.org	jakarta-apache	file://home/sites/jakarta-apache/sitemap.xmap	foo/bar/index.html	