Matchers (2.1 legacy document)

Table of contents

Table of contents

Warning:
This document was copied as is from the Cocoon 2.1 documentation, but has not yet been fully reviewed or moved to its new home.

1. Goal

This document lists all of the available matchers of Apache Cocoon and describes their purpose. See also the concepts document Using and Implementing Matchers and Selectors.

2. Overview

A matcher is a core sitemap component of Cocoon. Matchers allow Cocoon to associate a pure "virtual" URI space with a given set of "instructions" found in a Cocoon sitemap. Sitemap matchers are used to determine the flow and order of request processing. They typically describe how to generate, transform and present a requested resource(s) to the client. They may also be used to redirect requests to other pipelines or to call other sitemap resources.

Cocoon is driven by the client request. A request typically contains a URI, some parameters, cookies, and much more. Within the Cocoon environment, the request is evaluated to determine what sitemap instructions to use for processing. More specifically, a given request is matched against a pipeline matcher's pattern attribute. When a match is found, processing of the request begins.

As an example, consider the following sitemap snippet:

```
<map:match pattern="body-faq.xml"> <map:generate src="xdocs/faq.xml"/> <map:transform src="stylesheets/faq2document.xsl"/> <map:transform src="stylesheets/document2html.xsl"/> <map:serialize/> </map:match> <map:match pattern="body-**.xml"> <map:generate src="xdocs/{1}.xml"/> <map:transform src="stylesheets/document2html.xsl"/> <map:serialize/> </map:match>
```

Here the two sitemap entries map request URIs to different virtual URIs using the default wildcard matcher (defined earlier in a matcher component configuration). The first entry uses an exact match, "body-faq.xml". Only request URIs composed of this exact string will match this entry. The second sitemap entry uses a wildcard pattern. URI Requests that begin with "body-" and end with ".xml" will meet this matcher's requirement. For example, a URI request for "body-cocoon.xml" would match the second entry.

3. Order

It's important to understand that Cocoon is based on a "first-match" approach. All requests are matched against the different "map:match" entries in the order in which matchers are specified in the sitemap. As soon as a match is successful, the pipeline processing begins. This means that more specific patterns must appear before more generic ones. If the order of the two pipelines in the above example were reversed, a request for "body-faq.xml" not match "body-faq.xml" but "body-**.xml" because it appears first. (This is a familiar concept, especially in router and firewall configurations.)

4. Tokenization

Another important feature of matchers is tokenization. Every "variable" part of a matcher pattern will be kept in memory by Cocoon for additional reuse. It remains available within a pipeline match as a numbered argument. Using the previous example, consider a request URI such as "body-index.xml" matched by the second map:match element. The string "index" which matches the "**" wildcard, is available for reuse by other child elements of map:match. It is identified by the key {1}. This key is

used as a parameter for the generator which will first resolve it to the string "index", and then look for a file named "xdocs/index.html".

5. Wildcard and regular expressions

Most Cocoon matchers are built using two different techniques: regular expressions and wildcards. Regular expressions (or regexps) are a well-known and powerful system for pattern matching. Learning how to master them is beyond the scope of this document. However, you will find a lot of documentation on the web regarding this topic.

Although powerful, regexps can be overkill for most typical Cocoon use cases where simple matching operations are performed. This is why Cocoon offers a simplified pattern matching system based on a small set of basic rules.

- An asterisk ('*') matches zero or more characters, up to the occurrence of a '/' character (which serves as a path separator). A string, such as "/cocoon/docs/index.html", would *not* match successfully against the pattern '/*/*.index.html'. The first asterisk matches up to the first path separator only, resulting in the "cocoon" string. A successful matching pattern would be '/*/*/*.html'.
- A string containing two asterisks ('**') matches zero or more characters. This could include the path separator '/'. In this case, "/cocoon/docs/index.html" would successfully match the '/**/*.html' pattern. The double asterisk, including the path separator, would match the "cocoon/docs" string.
- As with regexps, the backslash character ('\') is used to indicate an escape sequence. The string '*' will match an actual asterisk while a double backslash ('\\') will match the character '\'. A pattern such as "**/a-*-is-born.html" would match strings such as "documents/movies/a-*-is-born.html" or "a/very/long/path/a-*-is-born.html". It would *not* match the string "docs/a-star-is-born.html".

6. Matchers in Cocoon

- **WildCard URI matcher**(The default matcher): matches the URI against a wildcard pattern.
- **Regexp URI matcher:** matches the URI against a full-blown regular expression
- **Request parameter matcher:** matches a request parameters given as a pattern. If the parameter exists, its value is available for later substitution.
- **Wildcard request parameter matcher:** matches a wildcard given as a pattern against the **value** of a configured parameter.
- **Wildcard session parameter matcher**: similar to the Wildcard request parameter matcher, but it matches a session parameter.

1. Comments

add your comments