Cocoon Profiler (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. Profiler Overview

The Cocoon Profiler has a double goal:

- it shows how much time is spent in each step of the SAX pipeline. Because of the nature of SAX, this information is difficult or impossible to obtain using general-purpose Java profilers. Additionally, it shows the time spent in the setup() method of each component of the pipeline (sometimes setup takes more time than the actual processing!).
- it allows to take a look at the XML produced in each step of a pipeline. You can use this to get insight in how pipelines work or to see what a particular transformer actually does. It is much easier and comfortable than fuzzing around with the LogTransformer.

The profiler does not show the time spent in components during pipeline setup. These include selectors and matchers, but most importantly, actions. Since Actions are used to execute all kind of logic such as communication with databases or EJB's, they can take up quite some time. Use a general purpose Java profiling tool to analyze them.

The Cocoon samples include a demonstration of the profiler. It can be found below "Block samples".

2. Usage

To use the profiler, two things need to be done:

- Change the pipeline implementation
- Add pipelines to generate the profiler information

2.1. Change pipeline implementation

First, check that the profiling pipeline implementations (caching and/or noncaching) are declared in the map:components section of the sitemap. Here is an example:

```
<map:pipes default="caching"> [...] <!-- The following two can be used for profiling:-->
<map:pipe name="profile-caching"
src="org.apache.cocoon.components.profiler.ProfilingCachingProcessingPipeline"/>
<map:pipe name="profile-noncaching"
src="org.apache.cocoon.components.profiler.ProfilingNonCachingProcessingPipeline"/>
</map:pipes>
```

You can now turn on the profiling in two ways:

- Change the default pipeline implementation by changing the value of the default attribute on the map:pipes element.
- Change the pipeline implementation of a specific map:pipeline by adding a type attribute to it with as value "profile-caching" or "profiling-noncaching".

2.2. Add pipelines to generate the profiler information

Instead of following the instructions below, you could also reuse the profiler demonstration from the Cocoon samples as-is, and mount it into your own application.

The information gathered by the profiler can be retrieved using the ProfilerGenerator. Make sure the

profiler generator is declared inside the map:generators element in the sitemap, as follows:

```
<map:generator label="content,data" logger="sitemap.generator.profiler" name="profiler"
src="org.apache.cocoon.generation.ProfilerGenerator"/>
```

Now add the following two matchers in an appropriate place in your sitemap:

```
<map:match pattern="profile.html"> <map:generate type="profiler"/> <map:transform
src="profile2html.xsl"> <map:parameter name="use-request-parameters" value="true"/>
</map:transform> <map:serialize type="html"/> </map:match> <map:match
pattern="profile.xml"> <map:generate type="profiler"/> <map:serialize type="xml"/>
</map:match>
```

Make sure the profile2html.xsl stylesheet is available, you can find it in te Cocoon distribution.

You also need the pretty-content view (which is included in the default Cocoon sitemap). It can be added in the map:views section of the sitemap as follows:

```
<map:views> <map:view name="pretty-content" from-label="data"> <map:transform
src="simple-xml2html.xslt"/> <map:serialize type="html"/> </map:view> </map:views>
```

Again, make sure the simple-xml2html.xsl stylesheet is available, it can also be found in the Cocoon distribution.

Now you are ready to use the profiler. First make a series of requests on the pages you want to profile, then go look at the profiler results by requesting the profile.html page.

3. Notes

- the profiler is contained in a seperate Cocoon block. Unless you deactivated it, it is included in the standard build.
- if you are streaming very large documents through the pipeline, the profiler will use a lot of memory since it buffers the data in between pipeline components.
- profiling has a very negative impact on performance and memory usage, since it buffers all data between pipeline components. Only activate it when required, and never activate it on production systems.
- when using the profile-caching pipeline implementation the XML generated by components will only be available on non-cached executions.

1. Comments

add your comments