ProfilingPipelinesWithBigSaxEventStreams

Profiling pipelines with big SAX-event streams

The profiliing pipelines, `ProfilingCachingProcessingPipeline` and `ProfilingNonCachingProcessingPipeline` are useful to measure the performance of your Cocoon application. However, when the XML that you process is really big (millions of SAX events), these pipelines crash sometimes with an error like:

```
org.apache.cocoon.ProcessingException: Failed to execute pipeline.: org.xml.sax.SAXException: Index too large
```

This error is caused by the `XMLByteStreamCompiler`, which is used to store a compiled format of the SAX-event stream in a byte-array (I am not sure why that is needed). Especially when you are dealing with large documents, measuring performance can be very important, but if the profiling pipelines give up, some other method must be used.

Warning: In the rest of this page and the accompanying attachments, some ugly coding techniques are used. Do not use the code given here as an example of how to program Cocoon components.

Instead of one of the profiling pipelines, a normal (non-profiling, and probably non-caching) pipeline must be used. The performance measuring is done by a new transformer, the `LOWProfilerTransformer` (see the attachment, available through the "more actions" menu at the top of this page). 'LOWProfiler' means 'Log-file Output Writing Profiler', and also indicates that the standard profiler is preferable. The `LOWProfilerTransformer` must be inserted in the pipeline, just before the serializer, e.g.,

```
      <map:match pattern="test">
        <map:generate src="bigfile.xml"/>
    ...do some transformations etc...
        <map:transform type="lowprofiler"/>
        <map:serialize type="xml"/>
      </map:match>
```

You will need to declare the transformer:

```
    <map:transformers default="xslt">
      <map:transformer name="lowprofiler"
        src="org.apache.cocoon.transformation.LOWProfilerTransformer"
        logger="lowprofiler"/>
    </map:transformers>
```

As you can see, a new log-file is needed where the profiling results are written. This is declared in `logkit.xconf`:

```
    <cocoon id="lowprofiler">
      <filename>${context-root}/WEB-INF/logs/lowprofiler.log</filename>
      <format type="cocoon">
        %7.7{priority} %{time}   [%{category}] (%{uri}) %{thread}/%{class:short}: %{message}\n
      </format>
      <append>true</append>
    </cocoon>
  ...
    <category log-level="INFO" name="lowprofiler">
        <log-target id-ref="lowprofiler"/>
    </category>
```

The `LOWProfilerTransformer` measures the time between calls to its `setup()` and `endDocument()` methods, which is an indication of the time spent in the pipeline. As a bonus, `LOWProfilerTransformer` also attempts to measure the amount of heap memory used by the JVM. This works most reliable if you restart the Cocoon servlet before each call to the pipeline.

The LOWProfiler results can be read with the `LOWProfilerGenerator` (see attachment). This yields the result in a format which is very similar to what is generated by the standard `ProfilerGenerator`, without data for the individual pipeline components. An example output is:

```
<profiler:profilerinfo xmlns:profiler="http://apache.org/cocoon/profiler/1.0"
    date="Fri Feb 25 14:33:12 MET 2005">
  <profiler:pipeline uri="/test" count="4"
  processingTime="956" processingMemory="33376">
    <profiler:average time="239" memory="8344" />
    <profiler:result time="379" memory="8824" index="0" />
    <profiler:result time="193" memory="7903" index="1" />
    <profiler:result time="188" memory="8746" index="2" />
    <profiler:result time="196" memory="7903" index="3" />
  </profiler:pipeline>
</profiler:profilerinfo>
```

The generator is declared as follows:

```
      <map:generator name="lowprofiler"
        src="org.apache.cocoon.generation.LOWProfilerGenerator"
        label="content" logger="lowprofiler"
        pool-grow="4" pool-max="32" pool-min="8"/>
```

When it is used, it gets the location of the log-file as its src-attribute:

```
        <map:generate type="lowprofiler" src="context://WEB-INF/logs/lowprofiler.log"/>
```