ServingLargeFiles

Serving Large Files

The following tips are useful when serving really large files (let's say 100MB).

If you don't want to read all of this stuff... there's an example at the bottom of this page, ready to be copy/pasted 😉

Disable Caching

You don't want to waste your precious cache space on a 100MB file.

To disable caching add the following attribute to your pipeline:

```
type="noncaching"
```

Details about enabling/disabling caching can be found on in the official documentation.

Disable Response Buffering

Each pipeline can buffer the response, before it is send to the client. The default buffer size is unlimited (-1), which means when all bytes of the response are available on the server, they are send with one command directly to the client. Of course, this slows down the response as the whole response is first buffered inside Cocoon and then send to the client instead of directly sending the parts of the response when they are available.

So, by default, the whole file is loaded in memory before being sent to the client. This is **very bad** and will quickly lead to OutOfMemory errors.

To disable response buffering add the following child element to your pipeline:

```
<map:parameter name="outputBufferSize" value="8192"/>
```

Details about response buffering can be found on in the official documentation.

Support byteranges/resumes

Older versions of Cocoon (2.0.x) had a ByteRangeResourceReader in the scratchpad. This component implemented byterange support which allows clients (browsers or download managers) to resume broken downloads. Since 2.1.x this code was merged into the regular ResourceReader.

Full Example

```
<map:pipeline type="noncaching">
  <map:parameter name="outputBufferSize" value="8192"/>
  <map:match pattern="**.zip">
    <map:read src="{1}.zip" mime-type="application/octet-stream"/>
  </map:match>
</map:pipeline>
```