Generator

The starting point for a Pipeline. It generates the input to the pipeline as a series of SAX events.

Therefore the contract between a Generator and the downstream elements in the pipeline are SAX events: if it can generate SAX events (or can be made to do so), then it can be a generator.

The simplest generator is one that reads an XML document from the file system. i.e. an SAX Parser.

Declaration

Generators are declared in the Sitemap:

```
<map:generators default="file">
<map:generator name="file"
               src="org.apache.cocoon.generation.FileGenerator"
               label="content"/>
...
</map:generators>
```

The `default` attribute on the generators element indicates that if a pipeline does not declare a generator, then this default will be automatically used.

Usage

Example generator entry:

```
<map:match pattern="hello.html">
<map:generate src="docs/samples/hello-page.xml"/>
<map:transform src="stylesheets/page/simple-page2html.xsl"/>
<map:serialize type="html"/>
</map:match>
```

Provided Implementations

There are several generators provided with Cocoon:

- File Generator – reads document from the file system or a URL
- HTMLGenerator – reads HTML document from the file system or a URL, uses JTidy to produce XHTML for subsequent processing
- DirectoryGenerator – generates an XML description of a filesystem
- Image Directory Generator – extension of the above, adds height and width information for images
- Fragment Extractor – transformer/generator pair that allows portions of a SAX event stream to be handled by separate pipelines, e.g. serving SVG images up separately
- JSP Generator – launches an embedded JSP engine which is used to evaluate the request, and produce well-formed XML input for Cocooon. Doesn't use Tidy. Provided as migration step. Could also use HTML Generator I suppose, with network overhead.
- Script Generator – captures the output of a script and parses it as XML. Script language can be any supported by BSF: JavaScript, Jython, ...
- XSP Generator – compiles, caches, and runs XSP pages
- Velocity Generator – produces SAX events from a Velocity template
- Request Generator – uses request data to produce an XML document, includes headers, URL parameters, and parameters from sitemap
- Status Generator – generates XML report on current status of engine
- StreamGenerator – reads XML document from POST request (either as a named form parameter, or the request body), document is parsed and presented to pipeline
- Error Generator – special generator activated by Sitemap Manager in response to errors, used to drive error handlers defined within pipelines.
- PHPGenerator – uses PHP pages as source of content for pipeline.
- WebServiceProxyGenerator – Integrate content from web services into a pipeline.

Some generators require additional libraries (PHP servlet, Velocity engine, JSP engine, HTMLTidy, etc).