Extending Apache Cocoon (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. Introduction

If you want to extend the functionality of Apache Cocoon, it may be unclear how to achieve your goal. This page tries to indicate when to write what, and to give an overview of what already exists (so you don't duplicate other's efforts).

2. When to write a Generator

From the sitemap documentation: "A Generator generates XML content as SAX events and initializes the pipeline processing. "

Thus a Generator is the starting point of a pipeline: it produces the first SAX events on which all other components of the pipeline are triggered.

You may want to write a Generator if you want some other basis for your SAX events (maybe you want a SAX event every time the temperature of your CPU changes?) However, before writing a Generator from scratch, it may be worthwhile to have a look at XSP, which can create a Generator for you.

Existing Generators are:

- DirectoryGenerator - Generates an XML directory listing.
- FileGenerator - Does the job of an XML parser: read an XML file and outputs SAX events.
- HTMLGenerator - Takes an HTML URL, makes an XHTML of it, and outputs the SAX events caused by this XHTML.
- ImageDirectoryGenerator - An extension of DirectoryGenerators that adds extra attributes for image files.
- PhpGenerator - Allows PHP to be used as a generator. Builds upon the PHP servlet functionality. Overrides the output method in order to pipe the results into SAX events.
- RequestGenerator - [FIXME: This looks like just outputing the request headers, the request parameters and the configuration parameters. But I don't see any use of it (besides debugging and demonstration). Are there other situations in which you might want to use this?]
- ServerPagesGenerator - Makes a Generator at compile time, based on the src file you define in the sitemap. This one is responsible for making your XSP pages work.
- StatusGenerator - Generates an XML representation of the current status of Cocoon. This can be considered "for administration use", i.e. your application probably won't deal with this one.

All these classes are in the org.apache.cocoon.generation package. In the same package, you find following helper classes and interfaces:

- Generator - The interface you have to implement if you want to write a Generator.
- AbstractGenerator - Extend this one for easier building of your own Generator.
- AbstractServerPage - [FIXME: This seems to be intended as basis for the ServerPagesGenerator, but it seems to be obsolete now?]
- ServiceableGenerator - Can be used as base class if you want your Generator to be an Avalon Serviceable.
- ServletGenerator - If you want to generate servlets. This is the base class for the ServerPagesGenerator.

3. When to write a Transformer

Let's start again from the sitemap documentation: "A Transformer transforms SAX events in SAX events." In other words, a Transformer outputs SAX events based on SAX events it receives.

You can imagine a Transformer doing many things, from XSLT processing over database querying to sending mail (and much further, of course).

These Transformers are standard available:

- LogTransformer - This is a class that can be plugged into a pipeline to print the SAX events which passes through this Transformer in a readable form to a file. This Transformer's main purpose is debugging.
- SQLTransformer - Can be used for querying a SQL database.
- XalanTransformer - Probably the most intuitive Transformer: it applies an XSL sheet to the SAX events it receives. It uses Xalan in the process.
- XIncludeTransformer - To include other XML documents in your "XML document" (which at transformation time exists in SAX events).
- XTTransformer - The same as XalanTransformer, but this one uses XT.

All these classes can be found in org.apache.cocoon.transformation, along with these helper classes and interfaces:

- Transformer - The interface each Transformer has to implement.
- AbstractTransformer - A helper base class for implementing a Transformer.
- AbstractDOMTransformer - An Abstract DOM Transformer (helper base class), for use when a transformer needs a DOM-based view of the document.

4. When to write a Serializer

No need for re-inventing the wheel, so let's start again with the sitemap documentation: "A Serializer transforms SAX events in binary or char streams for final client consumption." A Serializer is always the last step in a pipeline, and gives the client its final result: an HTML page, a nice PNG picture, a sound stream, or maybe just an XML document.

You should write a Serializer if you want to serve a client with some format that hasn't been provided yet.

Existing Serializers:

- FOPSerializer- Make PDF files.
- HTMLSerializer - Generate an HTML document.
- LinkSerializer- Show the targets of the links in the document.
- SVGSerializer- To construct an SVG.
- TextSerializer - Generate a text document.
- XMLSerializer - Generate an XML document.

Again, these can be found in the package org.apache.cocoon.serialization. And this package also includes following interfaces and helper classes:

- Serializer - The interface every Serializer has to implement.
- AbstractTextSerializer - Use this as base for your Serializer if you want to output a character stream.
- AbstractSerializer - A more general base class.

5. About Action

[FIXME: We have to wait until we can see what is going to happen here. Also, I wonder if this belongs here or should deserve a separate page.]

The Action part will be used for making Cocoon able to react on form input. This will make Cocoon no longer a simple basis for web publishing, but will make it apt for web interaction as well.

See Actions.

6. About XSP

XSP stands for "eXtensible Server Pages". It is the idea to program Generators by means of XML. The basic idea is to put XML tags like <xsp:logic> in your XML file, with in those tags Java code.

This is not the proper way to use XSP's. I just mentioned them here so you wouldn't forget their existence. Look to the XSP page for more information.

1. Comments

add your comments