Aggregator

Overview

Allows the combination of multiple XML documents. An individual aggregator contains one or more *parts*. Each part is a document fragment, these become children of a new document root, whose element name is defined by an attribute on the aggregator.

E.g.

```
<map:aggregate element="site">
  <map:part src="cocoon:/book-{1}.xml"/>
  <map:part src="cocoon:/body-{1}.xml"/>
</map:aggregate>
```

Will create a site document element, whose children are the contents of the two other documents.

Aggregators can be used as alternatives to Generator.

Aggregate Element

`<map:aggregate>` accepts following attributes:

- `element` – specifies the name of the root element for the aggregated content
- `prefix` – specifies the prefix for the new root element; default is no prefix
- `ns` – specifies namespace URI for the new root element; default is none

Part Element

Each `<map:part>` element inside `<map:aggregate>` specifies an xml document to be aggregated.

`<map:part>` accepts following attributes:

- `src` – specifies the source for the content of the part (see below)
- `strip-root` – if `true`, causes the root node of the part to be stripped and only its content to be aggrated; default is `false`
- `element` – encase the content of the part in the new root element with given name; by default, no new root element is introduced
- `prefix` – specifies the prefix for the new root element *(used in conjunction with `element` attribute)*; default is no prefix
- `ns` – specifies namespace URI for the new root element *(used in conjunction with `element` attribute)*; default is none

The `src` attribute for a part can be any of the following:

- `http://foo/bar` – takes content from a normal URL
- `context://servlet-context-path/foo/bar` – takes content from the servlet context.
- `cocoon:/current-sitemap-pipeline/foo/bar` – takes content from the current sitemap, using the selected pipeline.
- `cocoon://root-sitemap-pipeline/foo/bar` – takes content from the root sitemap, using the selected pipeline.
- `resource://class-path-context/foo/bar` – takes content from the CLASSPATH.
- `jar:http://www.foo.com/bar/jar.jar!/foo/bar` – takes content from a JAR file loaded using http.
- `file://foo/bar` – takes content from the filesystem
- User defined: e.g. `nfs:`, `jndi:`

Question:
Given that the `src` attribute can access pipelines, why oh why not just let it access generators directly so you can trivially merge the contents of two or more generators without having to set up special pipelines for them ??

I second this motion. (See CocoonFeatureRequests.)

See also

- CocoonProtocolExample