The Collection Module

Table of contents

1 Introduction

The collection module provides a resource type and a document wrapper class to use a document as a list of references to other documents. Typical examples are category pages, dossiers, etc.

2 Collection Types

Collections can either be static, which means that the reference list is stored in the collection document, or dynamic, which means that the reference list is generated when the collection is accessed. The type is determined by the `type` attribute of the document element (see examples below). At the moment, the following types are supported:

- manual
- children
- link

3 Static Collections (Type "manual")

This is the default type which is used when no `type` attribute is provided. The documents have to be explicitely specified.

The XML format of a static collection document looks like this:

```
<?xml version="1.0" encoding="UTF-8"?>
<collection xmlns="http://apache.org/cocoon/lenya/collection/1.0" type="manual">
  <document uuid="edf..."/>
  <document uuid="a7r..."/>
  ...
</collection>
```

4 Dynamic Collections (Type "children")

Collections of the type `children` assemble the document list from the collection document's children in the site structure.

The XML format of a collection document with the type `children` looks like this:

```
<?xml version="1.0" encoding="UTF-8"?>
<collection xmlns="http://apache.org/cocoon/lenya/collection/1.0" type="children"/>
```

The collection is expanded dynamically (i.e., the child document list is inserted) when the `collection` format is requested.

5 Syndicated Collections (Type "link")

You can use an external resource to provide the contents of the collection. Typically, a separate module provides this content dynamically, for instance by accessing a newsfeed.

The link is specified using the `href` attribute of the `<col:collection/>` element. Here's an example:

```
<collection xmlns="http://apache.org/cocoon/lenya/collection/1.0"
  href="cocoon://modules/feed/atom/http://apache-lenya.blogspot.com/feeds/posts/default"
  includeItems="3" type="link"/>
```

In this example, the feed module accesses the specified URL and returns a valid collection document, wrapping the feed items in `<col:document>` elements.

6 Resource Type Formats

At the moment, only the `collection` format is supported. In the case of static collections, the original XML document is returned. For dynamic collections, the list of documents is inserted. Check out the example below for a usage scenario.

7 Example: CInclude

The collection module provides an XSLT stylesheet (`collection2cinclude.xsl`) to simplify replacing the documents with their content to obtain an aggregation of all documents. Additionally, the meta data of the collection document itself and of all contained documents are included.

In the following example, the `collection` resource type is explicitely specified. This allows to call the collection format from a different resource type (e.g., the `news` resource type extends the `collection` resource type).

```
<map:match pattern="*.xml/*/*/*/*">
  <map:generate src="{resource-type:collection:format-collection}/{2}/{3}/{4}/{5}"/>
  <map:transform src="fallback://lenya/modules/collection/xslt/collection2cinclude.xsl">
    <map:parameter name="uuid" value="{4}"/>
    <map:parameter name="language" value="{5}"/>
  </map:transform>
  <map:transform type="cinclude"/>
  ...
</map:match>
```

8 The CollectionWrapper Class

The `CollectionWrapper` class allows to manipulate collections via the Java API. Here's a simple usage scenario:

```
CollectionWrapper collection = new CollectionWrapper(doc, getLogger());
collection.add(anotherDoc);
collection.save();
```

You can extend the `CollectionWrapper` class to add custom functionality or implement collections with specific behaviour.

For more information, please consult the API documentation.