DirectoryGenerator

Generates an XML directory listing.

The root node of the generated document will normally be a directory node, and a directory node can contain zero or more file or directory nodes. A file node has no children. Each node will contain the following attributes:

- name : the name of the file or directory
- lastModified : the time the file was last modified, measured as the number of milliseconds since the epoch (as in java.io.File.lastModified)
- size : the file size, in bytes (as in java.io.File.length)
- date (optional) : the time the file was last modified in human-readable form

Configuration options:

- depth (optional) : Sets how deep DirectoryGenerator should delve into the directory structure. If set to 1 (the default), only the starting directory's immediate contents will be returned.
- sort (optional) : Sort order in which the nodes are returned. Possible values are name, size, time, directory. directory is the same as name, except that the directory entries are listed first. System order is default.
- reverse (optional) : Reverse the order of the sort
- dateFormat (optional) : Sets the format for the date attribute of each node, as described in java.text.SimpleDateFormat. If unset, the default format for the current locale will be used.
- refreshDelay (optional) : Sets the delay (in seconds) between checks on the filesystem for changed content. Defaults to 1 second.

Definition

This is how you have to define a DirectoryGenerator (as defined in sitemap.xmap of the cocoon webapp directory):

```
<map:generator label="content"
               logger="sitemap.generator.directory"
               name="directory"
               pool-grow="2" pool-max="16" pool-min="2"
               src="org.apache.cocoon.generation.DirectoryGenerator"/>
```

Usage

Example of a simple usage of the DirectoryGenerator:

```
<map:match pattern="directory-listing">
    <map:generate type="directory" src="../directory/to/list/">
        <map:parameter name="depth" value="2"/>
    </map:generate>
    <map:serialize type="xml"/>
</map:match>
```

Result

This is an example of what you can expect as result of the DirectoryGenerator.

```
<dir:directory name="dir1" lastModified="1067545661156" date="30/10/03 21:27"
               size="0" sort="name" reverse="false" requested="true">
    <dir:directory name="dir11" lastModified="1067295952187"
                   date="28/10/03 0:05" size="0"/>
    <dir:directory name="dir12" lastModified="1067295971203"
                   date="28/10/03 0:06" size="0"/>
    <dir:directory name="dir13" lastModified="1067543010781"
                   date="30/10/03 20:43" size="0"/>
    <dir:directory name="dir14" lastModified="1067545661156"
                   date="30/10/03 21:27" size="0"/>
    <dir:file name="file.xml" lastModified="1067541801328"
                   date="30/10/03 20:23" size="79"/>
</dir:directory>
```

The Namespace

As mentioned in the user documentation, DirectoryGenerator will generate all elements with the namespace *http://apache.org/cocoon/directory/2.0*, so that the afore mentioned example will read like:

```
    <dir:directory name="dir1" lastModified="1067545661156" date="30/10/03 21:27"
                   xmlns:dir="http://apache.org/cocoon/directory/2.0
                   size="0" sort="name" reverse="false" requested="true">
        <dir:directory name="dir11" lastModified="1067295952187"
                       date="28/10/03 0:05" size="0"/>
        <dir:directory name="dir12" lastModified="1067295971203"
                       date="28/10/03 0:06" size="0"/>
        <dir:directory name="dir13" lastModified="1067543010781"
                       date="30/10/03 20:43" size="0"/>
        <dir:directory name="dir14" lastModified="1067545661156"
                       date="30/10/03 21:27" size="0"/>
        <dir:file name="file.xml" lastModified="1067541801328"
                       date="30/10/03 20:23" size="79"/>
    </dir:directory>
```

Beginners, when converting this kind of output to HTML, sometimes are facing problems on dealing with this namespace. Matching the elements is one of them and removing that namespace from the resulting HTML documents is the other. I've even seen some Cocoon Live Sites still bearing this namespace within HTML pages, even though (non-X)HTML standards do not allow any namespace.

The solution is quite easy. The XSLT Standard says:
NOTE: When a stylesheet uses a namespace declaration only for the purposes of addressing the source tree, specifying the prefix in the exclude-result-prefixes attribute will avoid superfluous namespace declarations in the result tree.

As an example, the following stylesheet will transform DirectoryGenerator's output into an HTML unordered list with **no** such superfluous namespace:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
                xmlns:dir="http://apache.org/cocoon/directory/2.0" exclude-result-prefixes="dir"
                xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match="dir:directory|dir:file">
        <li>
                <xsl:value-of select="@name"/>
                <xsl:if test="name(.)='dir:directory'">
                        <!-- creating subdirectories -->
                        <ul><xsl:apply-templates/></ul>
                </xsl:if>
        </li>
</xsl:template>

<xsl:template match="/dir:directory">
        <ul>
                <xsl:value-of select="@name"/>
                <!-- creating subdirectories -->
                <xsl:apply-templates/>
        </ul>
</xsl:template>

</xsl:stylesheet>
```

All you need to get rid of the namespace is in line 3.