Overview of the Lenya Sitemaps

Table of contents

Copyright © 2002-2007 The Apache Software Foundation. All rights reserved.

1. Introduction

Lenya is based on Apache Cocoon. To understand how Lenya works, you should have at least some basic Cocoon knowlege. Make sure you know what a Cocoon sitemap is and you understand matchers, generators, transformers and serializers at least.

Lenya uses some more Cocoon components, but if you can spot the matchers, generators, transformers and serizalizers, you will be able to get a good first overview of the Lenya sitemaps.

But Lenya is much more than just a collection of sitemaps and some XSLT stylesheets. Lenya builds on the Cocoon foundation and extends the Cocoon framework with custom

- Matchers
- Actions

Beyond these components, Lenya also defines two proprietary schemes:

- lenya:
- fallback:

These schemes are linked to a custom input module that comes with Lenya, the

- PageEnvelope input module

2. Authoring Area, Live Area, CMS GUI and several Publications

A Lenya installation aggregates a number of different parts into a single Cocoon application. By default, one instance of Lenya can be used to edit and render an arbitrary number of publications which are entirely independent of each other.

There are different concepts of what a publication is, but for now, let's assume each publication represents an independent website.

Inside each publication, there are

- The authoring area
- The live area
- The CMS GUI components (Drop Down Menu, CMS screens, editos, ...)

The URI space is used to organize all this.

3. The Lenya URI space

If you run Lenya in built-in Jetty servlet container, the Lenya webapp is the root application of the container. Therefore http://localhost:8888/ will already hit the Lenya root sitemap.

In case you deployed lenya into a non-root context of any servlet container, the first part of the URI will be handled by the container itself to match the responsible webapp.

If you deployed lenya.war into Tomcat for example, you will most likely have to use http://localhost:8080/lenya/ to get into the Lenya root sitemap.

Copyright © 2002-2007 The Apache Software Foundation. All rights reserved.

For the rest of this document, we pretend Lenya is the root webapp in your container as this is the case with the built-in Jetty. Let's examine , what Lenya does in order to render the document you see when you enter this URL:
`http://localhost:8888/default/authoring/tutorial/new_doctype.html`

3.1. Part 1: The publication ID

The first part is the publication id `default` which selects the *Default Publication*. There is a difference between the publication ID and the name of the publication. The ID should be compatible to both the filesystem implementation as well as the URI encoding because it will become both the name of the publication directory and a part of the URL. Therefore it is good practice to stick to 7-bit ASCII with no spaces or special characters.

In contrast, the display name of the publication (which will show up in the list of publications on the main Lenya entry screen) can be longer and it can contain spaces as well as any Unicode characters.

The publication ID is used to mount the publication specific sitemap.xmap from `$LENYA_HOME/pubs/{publication-id}`. The `map:mount` will strip the publication ID from the URL, so the publication sitemap will just see the `authoring/tutorial/new_doctype.html` portion. Nevertheless a publication has its ID available through the page envelope. More on that later as we're not yet really inside the publication's content.

3.2. Part 2: The area

There are two possible areas.

- Authoring
- Live

You can think of areas as of modes, as in "live mode" and "authoring mode". Live mode is the view of the publication as it is supposed to be displayed on the website to the site visitor. The authoring mode is used by editors and reviewers to edit the publication's content.

Technically speaking, the first major difference between the authoring area and the live area is just that in authoring mode the CMS menus are displayed. Following the WYSIWYG principle of Lenya, the publication content is rendered the same way in authoring mode as it would be in live mode.

Besides displaying the CMS menus or not, there are different copies of the underlying content repository for the authoring and live areas. This allows the editors to edit a working copy without affecting the live site. When a document is published after it was reviewed, it is just beeing copied over to the live repository.

If you're using the default filesystem repository of Lenya, you will find the two different repositories under `$LENYA_HOME/lenya/pubs/content/authoring` and `$LENYA_HOME/lenya/pubs/content/live`.

As well as the publication ID the area is also stored in the page envelope. This will make the actual area available to both the sitemap through the page envelope input module as well as to the components in

Copyright © 2002-2007 The Apache Software Foundation. All rights reserved.

the Java layer of Lenya.

At this point, Lenya parsed the URL to the point that is knows:

- which publication the request belongs to
- which repository is to be used, authoring or live
- wether to display the CMS menus or not

3.2.1. CMS Menus, Usecases and CMS screens

Prior to finally turning over to the actual publication content, some remarks on the CMS menus and CMS screens.

Lenya uses so-called usecases to perform actions. There are usecases such as

- submit (a document)
- publish (a document)
- ...

The CMS menus are nothing but a convenience mechanism for the CMS user to trigger usecases. As an alternative to choosing the "Workflow"->"Submit" menu option, you could as well append `?lenya.usecase=submit` to the document URL. As soon as Lenya finds a lenya.usecase request parameter it will branch into the `lenya/usecases/usecase.xmap` sitemap.

All usecases start in the Lenya core. The core contains most of the common access control, workflow and editing usecases. A publication can introduce publication specific usecases.

Some usecases require CMS screens to be displayed. These are rendered through the usecase sitemap portions as well. That way one could for example introduce a new skin for the CMS screens just by means of the Cocoon sitemap.

For details on usecases, have a look at the Lenya Usecase Framework.

3.3. Part 3: The document URL

Now it's the publication sitemap's task to take the document URL portion of the original URL (`tutorial/new_doctype.html` in our example) and generate and render the content page. This sounds easy, but turns out to be quite complex. This portion is actually the heart and soul of every CMS. This is also where differnt CMS have a lot of differences in terms of features and configurability.

The easiest way to deal with this would be:

- Choose the appropriate content repository (authoring or live)
- Use the document URL to look for a file with that name and generate it.
- If necessary: Apply an XSLT stylesheet and serialize the result.

Lenya can do a lot more here, such as:

- Use a publication specific mapper class to map the document URL to a backend repository URL. This will allow you to hide the actual repository structure from the website visitor.
- Apply some intelligence to decide what language version of the document to use. For example, it no

Copyright © 2002-2007 The Apache Software Foundation. All rights reserved.

specific language version is requested, Lenya will choose the default language. If there is a specific language requested, Lenya will apply a mapping to built a back-end respository path to query for the requested language and generate this if it is available. If the document is not available in the requested language, Lenya will fall back to the default language again.
- Choose among a selection of different rendering pipelines based on the source document's resource type. A resource type can be XHTML, but also any other XML format such as RSS, SVG, SlideML or whatever.

Most of this logic is implemented in the Java layer. The components are Avalon components which are configured on a per-publication basis. In other words: One could implement and plug in custom versions of these components.

Only the mapping between input URL and the path used to access the repository in the backend is implemented as an Avalon component in Java. The actual rendering of the document is done by the means of Cocoon sitemaps, thus allowing anyone with Cocoon knowledge to make changes to the rendering pipelines whithout having to write Java code.

An example of an entirely different mapping between the document URL and the repository is the Lenya Weblog publication that comes as a second sample with the Lenya distribution.

3.4. Part 4: The usecase and workflow parameters (optional)

Right after the document URL there may be additional paramters, especially

- lenya.usecase and mabye
- lenya.step

as well as maybe

- wf.event

For an explaination of the usecase parameters, see the usecase framework.

For an explaination of the workflow engine, see the workflow engine (coming soon).

3.5. The "lenya:" scheme

When reading the sitemaps in Lenya 1.4 you probably came across the lenya-scheme, often also referred to as the lenya protocol. For example, in the doctypes.xmap, you will find this section:

```
    <!-- parametrized doctype matcher -->
    <!-- pattern="{rendertype}/{area}/{doctype}/{document-path}" -->
    <map:match pattern="*/*/*/**.xml">
      <map:generate src="lenya:/{4}.xml"/>
      <map:transform src="fallback://xslt/{3}2xhtml.xsl">
```

The lenya: scheme just needs the document path to locate the source document, i.e. lenya:/concepts/index_en.xml. All other elements of the actual storage location of the document (publication id and area) will be read from the page envelope and interpreted in the LenyaSourceFactory.

Copyright © 2002-2007 The Apache Software Foundation. All rights reserved.

In other words: The lenya: scheme hides the storage implementation details from the sitemap.

3.5.1. What is a scheme?

The lenya scheme is a virtual protocol in Cocoon, similar to the cocoon and context scheme.

Note: In case you are not familiar with the latest official naming of the components of a URI or you still think that a URL is the same as a URI, please read this: http://www.bernzilla.com/item.php?id=100 (http://www.bernzilla.com/item.php?id=100) .

The two sentence summary:

The URI http://lenya.apache.org/ (http://lenya.apache.org/) is a URL. The URL is a special case of a URI. "http" is the scheme, which in a URL referes directly to a TCP/IP protocol. The part left of the colon (http) defines how the part right of the colon (//lenya.apache.org/) is to be interpreted by selecting an appropriate handler.

The JDK provides build-in mechanisms to handle many common schemes such as file, http, ftp and some more. But Cocoon adds an additional layer through its Excalibur SourceResolver system, see: http://excalibur.apache.org/sourceresolve/index.html (http://excalibur.apache.org/sourceresolve/index.html) .

Lenya hooks in there with the LenyaSourceFactory which defines the lenya schema. The actual implementation is in the org.apache.lenya.cms.cocoon.source.LenyaSourceFactory class, see

- LenyaSourceFactory JavaDoc (http://lenya.apache.org/apidocs/1.4/org/apache/lenya/cms/cocoon/source/LenyaSourceFactory.html)
- LenyaSourceFactory source code (http://svn.apache.org/viewcvs.cgi/lenya/trunk/src/java/org/apache/lenya/cms/cocoon/source/LenyaSourceFactor

Copyright © 2002-2007 The Apache Software Foundation. All rights reserved.