Glossary

This page is intended to summarize and define Lenya terminology, to point out usage inconsistencies and synonymous terms, and to collect pointers to more detailed information either in the official Lenya documentation or other pages in this wiki.

It is aimed at new Lenya users looking for a short primer, at proficient users in need of a handy reference and at developers and documentation writers who need a quick refresher in order to maintain consistent usage of technical terms in their code comments and docs.

For a comparison of Lenya's terminology with that of other Content Management Systems see CMSTerminologyComparison.

A more concise top-down overview of those key terms concerning Lenya content structure can be found at GlossaryStructure.

This page is meant to apply to all versions of Lenya, although it currently concentrates on Lenya 2.0. Where Lenya 1.2.X usage differs, please mark it with [Lenya 1.2.X].

---

access control

${renderedContent}

(see also OverviewAuthenticationAndAuthorization)

accreditable

${renderedContent}

- **=>user**
- **=>group**
- **=>IP range**
- **=>world**

archive
${renderedContent}

area
${renderedContent}

- authoring - edit pages
- live - the published pages, visible to site visitors
- archive - archived pages, can be restored
- trash - deleted pages, can be restored until trash is emptied.
- admin - administration area (not really a content area, just a tab for administrative tasks).
- staging - review edited pages (this is not supported in Lenya 2.0)

In Lenya 1.2, the prefix "info-" can be added to the authoring, staging, archive, and trash areas. This is used to display the sitetree and page information (commonly refered to as "info area").

The areas share many properties (notably the presentation of the content), but can have additional properties of their own (an obvious example are the editing menus in the authoring area). Live and authoring can have different content.
A page moves from authoring to live and back according to *=>workflow*s.

In the docs you will sometimes find term **mode** instead of "area" to describe the same concept.

(see also http://lenya.apache.org/1_4/concepts/authoring_live.html)

[The area concept is currently debated and will probably be scrapped for the next major release.]

asset

${renderedContent}

authoring

${renderedContent}

Cocoon

${renderedContent}

(see also: OverviewCocoonKnowledge and http://cocoon.apache.org/)

content

${renderedContent}

content item

${renderedContent}

[This is awkward. "document" seems to be the preferred term on the mailing lists.]

content type

${renderedContent}

context

${renderedContent}

deactivate

${renderedContent}

document

${renderedContent}

delete

${renderedContent}

edit

${renderedContent}

fallback mechanism

${renderedContent}

group

${renderedContent}

identity

${renderedContent}

info

${renderedContent}

internationalization

${renderedContent}

see also: http://solprovider.com/lenya/&cat=Language (Lenya 1.2 specific, but gives a good overview)

see also: http://cocoon.apache.org/2.0/userdocs/transformers/i18n-transformer.html (the cocoon I18nTransformer that governs i18n in Lenya)

i18n

${renderedContent}

live

${renderedContent}

localization

${renderedContent}

l10n

${renderedContent}

metadata

${renderedContent}

see also: http://lenya.apache.org/docs/2_0_x/reference/metadata.html

mode

${renderedContent}

module

${renderedContent}

- a resource type (e.g., docbook module)
- a repository implementation (e.g., jdbc module)

- a collection of XSLTs (e.g., content2svg module)
  (see also http://lenya.apache.org/1_4/reference/modules/index.html)
  navigation::
  policy
  ${renderedContent}

  Example:

- policy 1: user John has the role "edit" on the URL "/news"
- policy 2: IP range 182.12.4.122/255.255.0.0 has the role "review" on the URL "/news"
  If John logs in from a machine in this IP range, he has the roles "edit" and "review" on the URL "/news".
  protocol
  ${renderedContent}
  - from Cocoon:
    - `file://` - read a file from disk, using the operating system's path as the URI
    - `cocoon://` - request a resource from the cocoon servlet (handled by the sitemap pipelines, so this need not exist as an actual file)
    - `context://` - request a file from disk, using the webapp context directory as root
  - Lenya-specific:
    - `lenya-document://` - request a document from the Lenya storage by its **=>UUID** and optionally language and revision (i.e. independent of its current location in the site tree)
    - `site://` - request a document based on its sitetree path
    - `fallback://` - request a file using the **=>fallback mechanism**
    - `fallback-template://` - dito, but skip the current publication and start looking in its template instead
    - `aggregate-fallback://` - concatenate the content of a file in this publication with the contents in all ancestors
      Protocols are implemented using **=>source factories**.

    - `lenya://` - request a file from disk (same as context:/, but the returned source is a [RepositorySource] with additional features) FIXME: is that correct?

      _(see also http://lenya.apache.org/1_4/reference/protocols/index.html)_
      [Needs proofreading and completion. Lenya-Metadata factory is missing.]
      publication

      ${renderedContent}

      (see also http://lenya.apache.org/1_4/concepts/publication.html)
      [Some people feel this term is unfortunate (why not just call it a "site"?), but for now we're stuck with it.]
      publication templating

      ${renderedContent}

      Templating is implemented using the **fallback** mechanism, a lenya-specific uri resolver that can be applied to any uri reference in xml files by using *fallback://* as a protocol specifier. If this is done consistently, publications can share arbitrary **properties** (i. e. xslt files, configuration files, user/group account files, sample pages, resource types etc.) from their template or from the default publication.
      The fallback mechanism operates on a file level. Thus it can only be applied to whole files (not parts thereof), and only if those files are referenced by URIs in configuration files.
      The creation of a new child publication from a template is called **instantiation**. Therefore, you will sometimes find the term "**inst ance** of template X" instead of "child of X".
      Child publications can use features of their template(s) in two ways: by **copying** files from the template during instantiation, or by **referencing** those files.
      Copying severs the link between child and template - later changes to the template will not affect the child. **Referencing** implies that all changes to the template will immediately affect the child as well, since the child uses the template's property.
      (see also http://lenya.apache.org/1_4/reference/publication-templating/index.html)
      publish
      ${renderedContent}

      resource
      ${renderedContent}

      resource type
      ${renderedContent}
  - an XML structure definition (e.g., Relax NG)
  - some presentation pipelines,
  - some presentation XSLT stylesheets,
  - usecases to manipulate documents.
    The default publication contains the resource types *xhtml*, *homepage*, *OpenDocument*, *CForms* and *links*.
    (see also http://lenya.apache.org/1_4/reference/resource-types.html)
    review
    ${renderedContent}

    revision control
    ${renderedContent}

    (see also JcrConfiguration)
    role
    ${renderedContent}

You can define custom roles and workflows.
Roles are frequently assigned via **=>group** membership, but do not confuse roles and groups. OverviewAccessControl has a good explanation of how different they are.

site

${renderedContent}

sitemap

${renderedContent}

(see also OverviewSitemapStructure and http://lenya.apache.org/1_4/reference/lenya-sitemaps.html)
site tree

${renderedContent}

site tree node

${renderedContent}

source factory

${renderedContent}

submit

${renderedContent}

translation

${renderedContent}

trash

${renderedContent}

usecase

${renderedContent}

(see also http://lenya.apache.org/1_4/reference/usecase-framework/index.html)
usecase handler

${renderedContent}

user

${renderedContent}

UUID

${renderedContent}

version

${renderedContent}

workflow

${renderedContent}

To move a page back from live to authoring, a reviewer must **deactivate** it. Afterwards, it can either be re-published or **deleted**.
Workflow transitions are typically invoked when a **=>usecase** is executed. Moreover, in a workflow context, "usecase" is sometimes used as a synonym for "workflow transition".

In Lenya, the workflow of a document is controlled by a finite state machine with arbitrary states, transitions, and events, which is expressed using XML. Each resource type can use its own workflow schema. You can implement custom conditions to be checked before a transition can fire, this requires custom java code.

zones

${renderedContent}

To do

- cross-reference each term to the appropriate docs
- keep each definition short and sweet. perhaps the more verbose explanations can be merged into the docs if it makes them easier to understand?
- maintain alphabetic order

New terms coined for this section

- property (any file within a publication)
- child publication (we have instance, but i think that's too OOP)

Are there better, already established terms for these concepts? If so, let's use them instead and get rid of these new ones.