WishlistLenya2.0

This is an old wishlist from before Lenya 1.4 (now 2.0) was written. It contains some ideas to reconsider, but it's generally outdated. – JörnNettingsmeier
<<DateTime(2007-07-30T13:40:05Z)>>

"... let's go exploring!" from "Calvin and Hobbes: It's a magical world" by Bill Watterson

Building blocks needed

- repository
  *JSR170
  *WebDAV
- editing
  *Linotype
  *Cocoon Forms
- access control
- workflow
- scheduler
- search
- publishing

Gregor's wishlist

Lenya needs to be compelling for users. Apache technologies often neglect the user side.

User-centric wishlist items

- Intelligent word copy-paste
- diff-like inline versioning (with strikethrough etc)
- workflow overview, to do pane
- browser-based image editing like
  *http://www.ektron.com/news.cfm?doc_id=2669
  *http://www.ulead.com/imediting/features.htm
  *http://www.hotscripts.com/Java/Applets/Image_Effects/Image_Editing/
- user interface localization
- report generation
  *number of documents in draft, rejected, archive state
  *oldest pages
- forms-based workflow configuration
- 508-compliant output
- content import feature (tidy etc)
- make it possible to edit / change stylesheets through the browser
  *maybe a layout selector like http://monorom.to/css_site/css/template.php
  *assemble your pages with xincludes, aggregations in the browser
- improve asset management
  *make images globally available
  *asset search
  *batch upload
  *asset dependencies
- lessons from top 10 intranets 2003
  *emphasize bottom-up over top-down -> more weblog, aggregation technology, xml fragment support, xlink leverage
  *think about email integration
  *put workflow queue in rss
  *leverage trackback to facilitate content discovery / reuse / citation
- support much better navigation
  *extensive link tag support
  *deemphasize layout over content
  *make sure sites follow the readable url rules
- learn from cms grade school lessons
  *alerts for bad html titles
  *enforce good IA as much as possible
- make metadata useful
  *learning objects ideas -> reuse, fragments, bottom-up
- Get rid of the random set of stuff in the documentation and make it useful
- Provide useful backup scripts / backup interface
  *tar.gz of whole site?
- global spell checker
- automatic link creation to a glossary
  *support the forrest link types?
- extend audit trail
  *date accessed, use cases etc
- context-sensitive help
- taxonomy support
- expand related links functionality
  *discover interesting links automatically, suggest

- searching for terms with lucene in the bg
- offer these pages as links
- auto-generation of related articles
- cross-publication content sharing
- template manager
*allow related links lookup from bxe
- keep tabs on workflow, support escalation
- provide "edit this page" link on every live page by default
- embrace rss for notifications, content delivery
- provide full sitemaps

Developer-centric wishlist items

- JSR 170 (implementation started in 2.0)
- WebDAV
- JSR 168 (Portal integration)
- avalonize workflow
- convert lenya core into cocoon blocks
- much better test harness
- much tighter lucene integration
*expose lucene
- RDF search engine
- facilities for XML fragment query and management
- get rid of one-off xml classes (DONE)
- expose and use the container tuning parameters
*caching options
*memory usage
*reloading options
*logging

Andreas Hartmann's Wishlist

Complaints

- It's still to complex to write usecases (too much configuration).
- It's terribly hard to access document properties or meta data, not to mention queries, dynamic aggregation etc.
- The authoring environment is much too slow.

Suggestions

Move to Java 5

I'd really like to use generics and all the other nifty stuff.

Convention over Configuration

I used to prefer configuration for a long time for reasons of flexibility, but I kind-of changed my mind. We have to make flexibility trade-offs to accelerate development. The past showed that many configuration options are apparently not important enough to be worth providing.

Consolidate, Re-Use Concepts

Instead of developing new concepts, we should rather re-use existing ones. A good example was Joern's note to use content items for inbox message storage. An example how not to do it is the rank growth of editor integrations.

The remaining stuff in src/webapp should be moved to modules. Running the build after changing these files is cumbersome. It's so convenient that you don't have to deploy modules after you changed XSLTs and sitemaps.

We should introduce the "copyless" concept for publications too.

Clean and Fast Authoring Environment

Currently, the authoring environment carries out a lot of Cocoon pipeline processing which is necessary to generate the WYSIWYG view, which makes it terribly slow. The same is true for the site environment. I'm convinced that we could implement something faster (and probably easier to maintain) using JSF and AJAX, or by radically simplifying and stripping down our pipeline architecture.

I'd be happy to live without true WYSIWYG when I get in-place editing of page titles and meta data (like Flickr and the like) and can navigate end edit faster.

Stable, Out-of-the-Box Repository

With our homegrown repository implementation we'll never be able to meet any enterprise requirements. We need true transactions and performant querying.

Some options:

- JCR
- Shared-nothing architecture with JPA / Hibernate
- EJB 3.0

Replace all-in-one Sitetree with other concept

Disadvantages of all-in-one sitetree storage:

- doesn't scale
- operations on single nodes require locking of whole sitetree

Some options:

- JCR
- relational DB
- something entirely different, e.g. de-centralized indizes as proposed by solprovider

Queryable, indexable meta data

Examples:

```
/document[@doc:uuid = $uuid]
/document[@xml:lang = $language and wf:versions/wf:version[last()]/@wf:state = 'review']
/document[@doc:resourceType = 'message' and @inbox:recipient = $user]
```

Get rid of the DocumentIdToPathMapper

http://www.nabble.com/-RT--Replace-DocumentToPathMapper-with-language-suffix-in-sitetree-node-IDs-tf4035821.html

Misc

- The publication ID should be part of URLs that enter the publication sitemap. Before we had publication templating, this was not so important, but now we have to use {page-envelope:publication-id} all the time which is cumbersome and verbose.

Torsten Schlabach's Wishlist

User-centric wishlist items

- "Create New, empty publication" button (DONE)
- Browser based editing of CSS, JavaScipt and custom doctypes

Developer-centric wishlist items

- <lenya:lenya-version> should come from the source and not the publication (DONE)
- Servlet-Container based user Management? (Tomcat knows who's there and can be connected to any JNI service such as LDAP. So why does Lenya need it's own user management?)