ProposalI18N

Overview

Summary

This wiki summarizes discussions about internationalization(i18n) in Apache Lenya. This summary should lead into a common understanding of the i18n problematic and possible or already implemented solutions in Lenya. Note that I18N was implemented in Lenya as of version 1.2, this page is of historical interest.

The following **topics** concerning i18n are covered here.

- Localization of Lenya system messages i.e. strings displayed on LenyaCMSScreens or notification messages send via email
- Serving of multilingual documents

Discussion Threads

The following links will guide you to the beginning of mailinglist threads related to i18n and Lenya

1. Introducing UserLocaleAction
   http://mail-archives.apache.org/mod_mbox/lenya-dev/200312.mbox/%3c1072122632.2610.278.camel@samson.rkunet.org%3e
2. User language determination
   http://nagoya.apache.org/eyebrowse/ReadMsg?listName=lenya-dev@cocoon.apache.org&msgId=1359385
3. i18n and CMS UI
   http://nagoya.apache.org/eyebrowse/ReadMsg?listName=lenya-dev@cocoon.apache.org&msgId=1359345
4. Documents and language versions
   http://nagoya.apache.org/eyebrowse/ReadMsg?listName=lenya-dev@cocoon.apache.org&msgId=1359117
5. Information hiding for URIs
   http://nagoya.apache.org/eyebrowse/ReadMsg?listName=lenya-dev@cocoon.apache.org&msgId=1359032
6. Localization of Lenya
   http://nagoya.apache.org/eyebrowse/ReadMsg?listName=lenya-dev@cocoon.apache.org&msgId=1359104

Current State Of I18N

Documents can be authored and served in multiple languages. Each language version is stored in a separate file fitting a file name pattern which currently is myfilename_locale.xml i.e. tutorial_de.xml or tutorial_en.xml. The *language negotiation protocol* is simply defined within the request url. Should mean, a Lenya class i.e. the DefaultDocumentBuilder searches for a language identifier like *en or _de in the request uri. Based on that information the appropiate language version of the requested document is served. If you request i.e. tutorial_de.html all links in that page point to _de.html versions assuming that a german version of the destination exists. Request to uris without a language identifier are _always* being routed to a document version in the publication's default language, which is imho not very *user (agent) friendly*.

The current simple protocol of language negotiation has a significant advantage compared to using i.e. mod_negotiation. See the next section for more information on language negotiation.

The Art Of Language Negotiation

There are different approaches for language negotiation, thus answers to the question in which language a document should be served to the user. Two of those approaches are

1. automatic negotiation between a user agent and a web server i.e. using mod_negotiation or other appropiate functionality build into web servers
2. user triggered via hard linked language switches

There are at least three big advantages of the second approach, which are obviously show stoppers for the first approach. Those advantages are

1. Search engines can index all language versions of a document, which is often really important for site owners.
2. Its easy to create multilingual offline versions of the website. Should mean the user can switch the langauge even if pages are not served via a http server with language negotiation capabilities.
3. The user can switch a document's language without changing i.e. browser settings, which is sometimes not possible. Imagine a scenario in an university providing so called internet access points. On those access points changing browser settings is not possible. Of course this is seldom, but real.

Doing the language negotiation to 100% via a web server is not as flexible as we need it.

See the references section for must reads about language negotiation.

Naming Conventions

As we saw in the previous sections it is pragmatic to use language indentifiers in urls. Currently Lenya uses i.e. *en or _de as identifiers. I would prefer a naming scheme like stated in section _Note on hyperlinks and naming conventions* in mod_negotiation.

A good naming scheme is **filename.locale.extension** i.e. tutorial.en.html. Using language indentifiers in uris or better internationalization in general meets another problem discussed in LenyaAndCoolUris. Should mean urls get worse if one encodes to much information in it i.e. a language identifier. What about translating the hole url? Now things get really ridiculous? I think to translate uris or not is worth a separate discussion, but....hey.

I propose to create uri names in the systems default language (or better english names) and thats it. Otherwise the maintenance of a website would get much more complex.

Language Negotiation For System Messages

A system message in my opinion are strings on a LenyaCMSScreen like user adminstration or strings in a workflow notification email send to a reviewer or editor. In the latter case language negotiation via a request url or a accept-language header is not possible since the user is not directly requesting the message via an url. That means a user registered within Lenya cms should be able to specify the preferred language for system messages.

References

- *Techniques for multilingual Web sites* by Jukka Korpela
- *Language Negotiation Notes* by A.J.Flavell
- Apache mod_negotiation
- Apache2 mod_negotiation
- *Content Negotiation* an overview by Norman Walsh

Serving I18N Documents

The current implementation to serve multilingual documents with Lenya is a really pragmatic approach. Nevertheless I want to propose the following enhancements/changes, which lead into a compromise of using automatic language negotiation and user triggered language changes via hyperlinks.

- **The initial document language served should be derived from a user agent's accept-language header if available.** Should mean, the accept-language overrides the default publication language if the requested document is available in the accept-language.This would enhance useability. Of course it costs some effort and the automatic part of negotiation does not work in offline browsing mode. In case of offline browsing (pages are not served via a http server) changing the language shouldn't be a problem since I assume a website provides links to swtich the documents language. See the Debian website for a real straightforward pattern of serving multilingual documents. Changing the browser's language settings while watching the debian website causes every page to be displayed in the newly set language as long as you do not specify a language identifier in the url. At the bottom of the page, the user can access the language of the document being viewed via hyperlinks. This pattern can also be used, if webpages are not served by Lenya but i.e. by Apache Http Server using mod_negotiation. Using this pattern fits imho best for the many cases where the user is able and aware of setting his browser's accept-language. The pattern also suits well enough for the other cases, but the user than has to switch the language on each page he visits, if the user agent doesn't send the user's preferred accept-language. To implement this pattern as shown on the debian website would mean to change the current behavior of Lenya, that if you request i.e. tutorial_de.html all links in the returned page point to _de versions if available. The advantage of this behavior is, that a user has to switch the language only once via a hyperlink, because the user is directed to i.e. tutorial.de.html which contains links to other .de pages if available. Should mean, the user is able to override his user agent's accept-language manually via hyperlinks. Imho, this is a quite cool extension for the debian pattern of serving multilingual documents, but leads into a maintenance problem, since all .de.html documents which reference documents without language identifier need to be changed, if a referenced document gets available in .de. I propose to implement the debian pattern of serving multilingual documents as is.
- **Use filename.locale.extension as basis pattern for multilingual urls.** This is just to go along with naming conventions mentioned in mod_negotiation section *Note on hyperlinks and naming conventions*.

.

Serving I18N Messages

The common way to localize messages using cocoon is to use the I18NTransformer. It translates keys marked up with i18n elements. The transformer looks up keys in so called dictionaries. See docu of the I18NTransformer for details.

The transformer needs to be told for which locale the translation should be done. I propose to introduce a user settings dialog within the admin cms tab called *My Settings*. There each user of Lenya can define the *langauge settings* for system messages. I propose the follwing options for the language settings.

- Use user agents language if possible? (yes/no/default=yes)
- Choose your language. (list of available locales, default=publication's default language or set during user creation)

The first setting is useful, if a user wants to use a user agent i.e. a browser to define his preferred language for web pages. The second setting is to define the language of system messages, which are not directly requested by a user agent i.e. workflow notification emails send to the user by Lenya.

Putting it all together

To implement i18n of system messages and the serving of multilingual documents as proposed in previous sections we need to change some things in Lenya code and develop new components.

- **MySettingsModule** could be the name of an input module to provide the properties set in *My Settings* screen like the preferred system language. This module should take care for the *Use user agents language if possible?* flag. To use the module one could use {my-settings:language} or {my-settings:use-accept-language}.

- Use {my-settings:language} and {my-settings:use-accept-language} to localize system messages i.e. cms screens or generated notification emails. Should mean we use the MySettingsModule to determine the locale, which has to be passed to the I18NTransformer.
- Change the DefaultDocumentBuilder to use the accept-language header as initial document language, assuming the requested document exists in the accept-language. Otherwise the publication's default language should be used.

This wiki was created by RolfKulemann in december 2003. Please post comments to Lenya-Dev.