ChiliProject - Feature # 207: Long format issue/revision links

Status:	Open	**Priority:**	Normal
Author:	Ian Freeman	**Category:**	Text formatting
Created:	2011-02-19	**Assignee:**	
Updated:	2011-02-19	**Due date:**	
Remote issue URL:			
Affected version:			

Description:

Allow an alternate more descriptive linking syntax for common link types like issues and revisions anywhere wiki format is accepted (especially issue comments).

The base code already has such a feature, for instance the link that is displayed for a related issue is more descriptive than just #207.

One approach for issues:
@#207@ becomes @#207@
@##207@ becomes @Feature 207@
@###207@ becomes @Feature 207: Long format issue/revision links@

You could also do @s@ and @l@ suffixes for short and long links, although @l@ could be easily confused with 1 when editing.

An alternate approach, using extra characters to manually layout the text:
@#207n@ or simply @#207@ becomes @#207@
@#207tn@ becomes @Feature 207@
@#207tns@ becomes @Feature 207: Long format issue/revision links@
@n@ stands for number.
@t@ stands for tracker.
@s@ stands for subject.

You could even allow regex, but that's going overboard:
@#207/n \(t\): s by a/@ becomes @207 (Feature): Long format issue/revision links by Ian Freeman@
@n@, @t@, @s@, @a@ would all need to be special codes (can be escaped with \).

I would like to use this feature to help me build changelog wikis that I generate and link to every new version entry in my active projects.

History

2011-02-19 01:48 am - Eric Davis

- Category changed from Wikis to Text formatting

What about a wiki macro? I have a plugin with one that outputs a bunch of information. See the readme and example image:
"redmine_wiki_issue_details":https://github.com/edavis10/redmine_wiki_issue_details

We could have something like:

* @{issue(100)}@ - "#100"
* @{issue(100, 'tracker id')}@ - "Feature #100"
* @{issue(100, 'tracker id subject')}@ - "Feature #100 The subject"
* @{issue(100, 'Some extra text with id before subject')}@ - "Some extra text with #100 before The subject"

(the string is just searched and replaced based on a list of valid field names)

2011-02-19 11:54 am - Aleksey Zapparov

That's a great idea, but I propose to have a little bit different syntax:

* {issue(100)} - "#100"
* {issue(100, :tracker, :id)} - "Feature #100"
* {issue(100, :tracker, :id, :subject)} - "Feature #100 The subject"
* {issue(100, 'Some extra text with ', :id, ' before subject')} - "Some extra text with #100 before The subject"

2011-02-19 01:16 pm - Holger Just

I fear, the macro syntax is not that rich (though, I'm inclined to change that :), probably for version#5 release).

Here, I propose a simple templating language to distinguish the text from fields. It could be one of the following:

* @!{{issue(100, "#tracker ##id #subject")}}@ -> "Feature #100 The subject"
* @!{{issue(100, "Some extra text with ##id before #subject")}}@ - "Some extra text with #100 before The subject"

* @{{issue(100, "{{tracker}} #{{id}} {{subject}}")}}@ -> "Feature #100 The subject"
* @{{issue(100, "Some extra text with #{{id}} before {{subject}}")}}@ - "Some extra text with #100 before The subject"

I think the second variant is not expressible currently, as the regex to parse the macros does not take quotation, escaping or nesting into account.

But more generally, the macro method should only be used *in addition* to the original proposal of using more hashes for longer formats (which I really like). That way, the format could be changed globally and it is really simple to use. Normal users are not too convinced by that macro syntax I'm afraid...

2011-02-19 01:51 pm - Aleksey Zapparov

Another few cents :)) Probably to make it a little bit more natural:

* #207 -> #207
* #207:"Some alt. description of {{tracker}} {{id}} is just a template"

This will also allow to define default template, let's say "{{tracker}} {{id}}", so @#207@ will become @Feature #207@

2011-02-19 04:06 pm - Ian Freeman

I'm happy with any of the macro solutions.

Holger Just wrote:
> But more generally, the macro method should only be used *in addition* to the original proposal of using more hashes for longer formats (which I really like).

I really like this idea. Whichever more complicated macro syntax we should use, multiple OCTOTHORPEs should be used for users that don't care to use it.

But I don't like the double hash syntax you propose in the first example, I can see users getting confused with the sub-list @##@ directive.

2011-02-19 11:07 pm - Eric Davis

Holger Just wrote:
> I fear, the macro syntax is not that rich (though, I'm inclined to change that :), probably for version#5 release).

<pre><code class="ruby">
output = args[1].gsub('id', issue.id).gsub('subject', issue.subject)

return h(output)
</code></pre>

> Here, I propose a simple templating language to distinguish the text from fields.

If we are going to use a template language then we shouldn't invent our own, there are too many edge cases and it's easy to miss one and have a security hole. Liquid templates are a safer solution and might be useful in other places too (theming). But I think that is overkill for this.

> But more generally, the macro method should only be used *in addition* to the original proposal of using more hashes for longer formats (which I really like).

Personally I don't like the extended hash syntax. Issue number are already different than any other object linking (@object:"text"@ "syntax description":/help/wiki_syntax_detailed.html)