ChiliProject - Feature # 604: Liquid Markup on top of Textile

Status:	Closed	**Priority:**	Normal
Author:	Holger Just	**Category:**	Text formatting
Created:	2011-09-02	**Assignee:**	Holger Just
Updated:	2011-11-25	**Due date:**	

Remote issue URL:

Affected version:

Description:

The current state of macros is rather bad, so we decided to remove this in favor of "Liquid Markup":http://www.liquidmarkup.org/ which gives us much richer expressiveness and less self-hacked code. In [[Development Meeting 1]] we decided that we want Liquid inclusion.

Eric started a branch at https://github.com/edavis10/chiliproject/tree/feature/unstable/liquid. I rebased it on the current unstable and started working on it. My branch is at https://github.com/meineerde/chiliproject/tree/feature/unstable/liquid

This is the state of things:

* The full expressiveness of Liquid can be used (by Eric)
* The core macros are ported to Liquid tags (by Eric)
* @include@ is performed by the native include mechanism of Liquid (by me)
* There exists a compatibility layer for macros. (by me)
** This one should be deprecated from 3.0 on and be removed in 4.0
** After the layer has settled and possible quirks are fixed, we should offer a rake task in the 3.1 release which permanently transforms textilizable content towards the new format (esp. the include format). That way, we can drop the compatibility/legacy layers after a while, e.g. in 4.0
* There exists several drops (basically model objects exposed to liquid) for various models (by Eric)

There are still some issues though:

* Tests for @include_page@ break. This was the include mechanism thought of by Eric which I replaced with native functionality.
* I think we should reduce the usage of tags for everything. Most of the tags defined in the core are better suited as Variable implementations.

Feature-wise, I think we should finally add facilities for object querying. That would allow us to e.g. define a query language for issues, get a collection of @IssueDrops@ back and format them using liquid alone. Which would be awesome :)

What do you think?

Associated revisions

2011-11-19 11:31 pm - Eric Davis

[#604] Require the liquid gem

2011-11-19 11:31 pm - Holger Just

[#604] Add base drop

2011-11-19 11:33 pm - Eric Davis

[#604] Add simple Liquid drops for Projects and Principals

2011-11-19 11:34 pm - Eric Davis

[#604] Add basic WikiPageDrop

2011-11-20 02:49 am - Eric Davis
[#604] Add an IssueDrop with custom field support

2011-11-20 02:49 am - Eric Davis
[#604] Disable classic ChiliProject wiki macros

2011-11-20 02:49 am - Eric Davis
[#604] Run the output of the WikiFormatted text into Liquid

The current view's instance variables are sent to liquid dynamically

2011-11-20 02:49 am - Holger Just
[#604] Evaluate Liquid before Textile-to-HTML transformation.

This changes how the liquid integration works. It now integrates the Textile
conversion step. This was necessary because if you first convert the snippets
inside of loops and conditionals from Textile to HTML, you loose some
important context information which is required to e.g. build proper lists in
textile.

We expect the standard case that Liquid tags return Textile markup instead
of HTML. Thus, we can convert the final textile markup to HTML as a very last
step.

To allow existing and new macros (or tags) to return HTML for advanced usage,
we save their respective output into the context and put a placeholder string
into the generated markup. After the transformation to HTML, we insert the
previously generated HTML into the string using search+replace in
lib/chili_project/liquid/template.rb. Tags have to be registered using
:html => true for this special treatment.

2011-11-20 02:49 am - Eric Davis
[#604] Add ability to support legacy macros

2011-11-20 02:49 am - Holger Just
[#604] Add base tag

2011-11-20 02:49 am - Eric Davis
[#604] Port hello_world macro to liquid

2011-11-20 02:49 am - Eric Davis
[#604] Port the macro_list to Liquid: variable_list and tag_list

2011-11-20 02:49 am - Eric Davis
[#604] Port the child_pages to Liquid tag

2011-11-20 02:49 am - Holger Just
[#604] Adapt the upstream include mechanism to work with Wiki pages

2011-11-20 02:49 am - Holger Just
[#604] Introduce compatibility layer for third party macros.

This be removed with complete macro removal.

2011-11-20 02:49 am - Holger Just
[#604] Add some handy filters

2011-11-20 02:49 am - Holger Just
[#604] Add nicer error formatting, similar to the old style

2011-11-20 02:49 am - Holger Just
[#604] Remove the leading newline from Liquid blocks for easier formatting

2011-11-20 02:49 am - Holger Just
[#604] We don't support the old escape style anymore

2011-11-20 02:49 am - Eric Davis
[#604] Add missing test for Liquid

2011-11-20 02:49 am - Holger Just
[#604] Adapt fixtures for new liquid syntax

History

2011-10-13 12:29 am - Eric Davis
Holger:

How is the Liquid stuff going? I have a client running my version of it in there installation and haven't heard of any bugs.

2011-10-17 05:32 pm - Holger Just
Okay, I think I now have most of the stuff ready. All tests pass, except the one checking for "circular inclusion":https://github.com/meineerde/chiliproject/blob/issues/unstable/604-liquid/test/unit/lib/chili_project/liquid_test.rb#L145-152.

As I use the Liquid built-in include mechanism, we have a difference of behavior here. We used to spit out a warning on circle detection, Liquid turnes the offending include into an effective noop instead. As the test currently expects the old behavior, it breaks. What do you think, which behavior is more desirable?

The code, rebased on the current unstable branch, can be found at https://github.com/meineerde/chiliproject/tree/issues%2Funstable%2F604-liquid

2011-10-25 09:54 pm - Eric Davis
Any way we can output a message on circular inclusion? This is a case where someone makes a mistake and it would be best to tell them what is wrong rather then outputting nothing.

If not, then the no-op is probably fine.

2011-10-27 04:49 pm - Holger Just
It turned out, the no-op I observed was actually caused by a bug which caused only the first include level to succeed at all. All other includes were silently dropped. This was fixed in https://github.com/meineerde/chiliproject/commit/e3c9053f6296687a8feb278a77fd3b5714f6c4aa. Subsequently, I added the previously missing check for circular inclusion and updated the test which was previously failing for the wrong reasons.

I think I'm now at 100% featurewise and at 90% on cleanliness. What is still bothering me is the way we use Drops currently. Currently, each model object to be exposed to Liquid requires the explicit implementation of a Drop class and an extension of the model.

Instead, I'd like to use https://github.com/Shopify/liquid/blob/master/lib/liquid/module_ex.rb and define which methods /attributes are to be exposed directly on the model.

2011-10-27 07:41 pm - Eric Davis
Holger Just wrote:

> Currently, each model object to be exposed to Liquid requires the explicit implementation of a Drop class and an extension of the model.
>
> Instead, I'd like to use https://github.com/Shopify/liquid/blob/master/lib/liquid/module_ex.rb and define which methods /attributes are to be exposed directly on the model.

Shoot, there was a technical reason I used drops instead of @liquid_methods@. I started with @liquid_methods@ but refactored/rebased them into drops.

I think it was because:

* @liquid_methods@ created a class based off of LiquidDrop which made it hard to share behavior. My @BaseDrop@ lets us share common methods. (e.g. custom_field should be shared by all classes with them).
* something to do with a whitelist approach: I only wanted to expose specific fields and methods to the Liquid object. (though @liquid_methods@ seems to do this already)
* having a common @BaseDrop@ class and actual classes will let use define object visibility. That way permissions are handled at the lowest model level: "example":https://github.com/edavis10/chiliproject/blob/feature/unstable/liquid/app/drops/issue_drop.rb#L3

Also using Drops will let us expose non-Model classes too. Things like the current theme, version, etc.

2011-10-27 08:11 pm - Holger Just
Eric Davis wrote:
> * @liquid_methods@ created a class based off of LiquidDrop which made it hard to share behavior. My @BaseDrop@ lets us share common methods. (e.g. custom_field should be shared by all classes with them).

We could either extend @::Liquid::Drop@ via open classes or implement our own @liquid_methods@. In the end, we are probably going to implement our own, with a custom base-class. However, I still like the overall design of having the real underlying class decide which methods it would like to expose.

> * something to do with a whitelist approach: I only wanted to expose specific fields and methods to the Liquid object. (though @liquid_methods@ seems to do this already)

Exactly. Only public methods of the Drop class are exposed. Those methods are created by @liquid_methods@. I extended that behavior "in my branch":https://github.com/meineerde/chiliproject/blob/issues/unstable/604-liquid/app/drops/base_drop.rb#L25 so that the default behavior of "all public methods of the drop" can be further restricted. I'm not too confident in the curent implementation though.

(Note that your approach also exposes the original object to liquid, e.g. @{{issue.object.subject}}@

> * having a common @BaseDrop@ class and actual classes will let use define object visibility. That way permissions are handled at the lowest model level: "example":https://github.com/edavis10/chiliproject/blob/feature/unstable/liquid/app/drops/issue_drop.rb#L3

This is good as a last resort. From a design perspective, it shouldn't even be possible to even create a drop for an object that shouldn't be accessible for the current user.

> Also using Drops will let us expose non-Model classes too. Things like the current theme, version, etc.

@liquid_methods@ is defined on @Module@ and is thus available for *all* modules and classes.

The once case where the explicit drops shines is if there is a need to define custum methods on the drop, like I've done for my prototypical "QueryDrop":https://github.com/meineerde/chiliproject/blob/issues/unstable/665-liquid-issue-queries/app/drops/query_drop.rb. However, this could still be accomplished through open classes. As could be the rest. So it probably makes sense to have a hybrid approach. The generated methods could e.g. be defined on a module instead of a class. This module could then be included into a class like the one we have now. This would allow people to extend the class like usually and could overwrite certain behavior.

2011-10-27 09:15 pm - Eric Davis
To be honest, this feels like a bike shed now.

Whether we use drops or the other stuff in Liquid doesn't affect what the end user sees or how they use the code. I'd say lets use the code we have now (Drops) and consider how they are implemented as an "internal API" that we can change and refactor later on. We should focus on exposing all of the current data first (so it can be used by users) and see how to allow 3rd party data to be exposed (plugins). Then we'd be able to make an informed decision on if Drops or @liquid_methods@ is best. In fact the "best" method might even be a combination of the two or even a third unknown...

tl;dr: Review and commit the existing code, refine the internal implementation later.

2011-11-20 02:52 am - Holger Just
Sent the pull request at https://github.com/chiliproject/chiliproject/pull/127.

It would be great if somebody could review it.

2011-11-25 10:19 am - Eric Davis
- Status changed from Open to Closed

Merged into unstable. I didn't have a chance to take a detailed look at how @include@ works but the external API looks good (i.e. tests).

Thanks for finishing this up Holger.