Redmine - Feature #2542

Plugin hooks should have access to the "request" variable

2009-01-19 22:19 - Douglas Manley

Status:	Closed	**Start date:**	2009-01-19
Priority:	Normal	**Due date:**	
Assignee:	Eric Davis	**% Done:**	100%
Category:	Plugin API	**Estimated time:**	0.00 hour
Target version:	0.8.2		
Resolution:	Fixed		

Description

Plugin Hooks do not have access to the "request" variable; this makes linking from them difficult.

In my environment, we have to contend with the different ways that people access the site:

- IP
- redmine
- redmine.myCompany.com

Referencing the Setting class works if the instance of Redmine is accessed only from one URL.

From Eric:

```
request would be best.  Problem is, request is setup in a Rails Controller but plugin hooks are no
t Controllers.
```

Redmine version: 0.8

Related issues:

Related to Redmine - Defect #2754: Setting up the default_url_options :port i...	**Closed**	**2009-02-15**
Is duplicate of Redmine - Feature #2649: Need @controller context in plugin h...	**Closed**	**2009-02-02**

Associated revisions

Revision 2429 - 2009-02-10 04:12 - Eric Davis

Added request and controller objects to the hooks by default.

The request and controller objects are now added to all hook contexts by
default. This will also make url_for work better in hooks by setting up
the default_url_options :host, :port, and :protocol.

Finally a new helper method render_or has been added to ViewListener. This
will let a hook easily render a partial without a full method definition.

Thanks to Thomas Löber for the original patch. #2542

Revision 2430 - 2009-02-10 04:12 - Eric Davis

Renamed variables to be more descriptive. #2542

Revision 2489 - 2009-02-21 01:23 - Eric Davis

Refactored the HookTest methods to use Redmine::Hook::Helper instead
of the Redmine::Hook class directly. Redmine::Hook::Helper is the
public API so it should be one that is tested.

#2542

Revision 2490 - 2009-02-21 01:23 - Eric Davis

Implementing the missing tests now that HookTest has a cleaner setup.

Controller and Views are now tested through the Redmine::Hook::Helper module.

#2542

Revision 2491 - 2009-02-21 01:23 - Eric Davis

Refactored the mess known as Hook default_url_options in favor of the simpler
:only_path as suggested by splatteal on GitHub.

#2542

Revision 2522 - 2009-02-25 08:25 - Eric Davis

Fixing Plugin and Mailer default_url_options.

Both the plugin hooks and Mailer were setting default_url_options incorrectly
and causing ActionContoller::UrlWritter to cache the settings on the module
(mattr_accessor) causing several url generators to fail in either the plugin
hooks or the Mailer.

- Replaced Mailer's use of the default_url_options accessor with the proper class method
- Replaced Hook's use of the default_url_options accessor with the proper class method on the ViewListener class
- Added a test to reproduce the bugs in the Mailer when a hook is registered (thanks Chaoqun Zou)

  #2542

History

#1 - 2009-01-19 22:23 - Douglas Manley

It would be sweet if something like this could be done so that we don't need (as plugin developers) to access the "request" variable.

```
include ActionController::UrlWriter

default_url_options[:host] = request.env[ 'SERVER_NAME' ]
default_url_options[:port] = request.env[ 'SERVER_PORT' ]
```

#2 - 2009-01-20 19:01 - Eric Davis

Douglas Manley wrote:

> It would be sweet if something like this could be done so that we don't need (as plugin developers) to access the "request" variable.

That would solve the host and port issue (would also need protocol) but request has a lot of other useful information. For example, a plugin might want to sniff the User Agent and resent a mobile view.

I'd also love to be able to use render in a hook but I know it's not trivial. Putting "view" code in a class as a string feels so dirty to me.

#3 - 2009-01-23 09:50 - Thomas Löber

This is how I would change hook.rb:

```
module Helper
  def call_hook(hook, context={})
    Redmine::Hook.call_hook(hook, {:project => @project, :request => request}.merge(context))
  end
end
```

And Redmine::Hook.call_hook:

```
# Calls a hook.
# Returns the listeners response.
def call_hook(hook, context={})
  returning "" do |response|
    hls = hook_listeners(hook)
    if hls.any?
      request = context[:request]
      if request
        default_url_options[:host] = request.env["SERVER_NAME"]
        default_url_options[:port] = request.env["SERVER_PORT"]
      end
      hls.each do |listener|
        response << listener.send(hook, context).to_s
```

```
      end
    end
  end
end
```

This works for me at least. I can use "link_to" in a hook listener and I can query context[:response].

What do you think?

#4 - 2009-01-23 13:28 - Thomas Löber

Eric Davis wrote:

> I'd also love to be able to use render in a hook but I know it's not trivial. Putting "view" code in a class as a string feels so dirty to me.

If the call_hook method is extended like this:

```
module Helper
  def call_hook(hook, context={})
    ctx_init = {:project => @project, :controller => controller, :request => request}
    Redmine::Hook.call_hook(hook, ctx_init.merge(context))
  end
end
```

you can use render_to_string in your hook listener method:

```
def view_issues_show_details_bottom(context)
  controller = context[:controller]
  issue = context[:issue]
  controller.send(:render_to_string, :partial => "show_more_data", :locals => {:issue => issue})
end
```

It works, but I wonder if this is the correct way to do it. At least the HTML is in a (partial) view again.

#5 - 2009-01-23 16:47 - Thomas Löber

Actually the above method has to be:

```
module Helper
  def call_hook(hook, context={})
    ctx_init = {:project => @project, :request => request}
    ctx_init[:controller] = self.is_a?(ActionController::Base) ? self : controller
    Redmine::Hook.call_hook(hook, ctx_init.merge(context))
  end
end
```

Then it will work for controller and view hooks.

#6 - 2009-01-24 03:41 - Eric Davis

- Assignee set to Eric Davis

Thomas thanks for the suggestions, I'll see what I can do.

#7 - 2009-02-01 00:07 - Thomas Löber

This is how we can make calling render_to_string more straightforward:

In hook.rb

```
class ViewListener < Listener
  [...]
  def self.render_on(hook, options={})
    define_method hook do |context|
      context[:controller].send(:render_to_string, {:locals => context}.merge(options))
    end
  end
end
```

Then we can define a hook that renders its result using a partial view like this:

```
class MyHook < Redmine::Hook::ViewListener
  render_on :view_issues_show_details_bottom, :partial => "show_more_data"
end
```

The partial view will have access to issue because it is given as a parameter in the call_hook call.

#8 - 2009-02-02 10:00 - Thomas Löber

- File hook.rb.diff added

This is the complete patch.

I split the "call_hook" helper into two methods. One for controllers and one for views. The controller method returns an array of results (which is more appropriate if I want to handle the result inside a controller). The view method uses "join" to convert the array into a string.

#9 - 2009-02-02 11:58 - Thomas Löber

- File hook.rb.revised.diff added

As I just saw it was a bad idea to split the "call_hook" helper. It doesn't work if a controller includes the ApplicationHelper.

So here's a revised patch.

#10 - 2009-02-10 03:09 - Eric Davis

- Status changed from New to 7

- Target version set to 0.8.1

Thomas Löber wrote:

> As I just saw it was a bad idea to split the "call_hook" helper. It doesn't work if a controller includes the ApplicationHelper.
>
> So here's a revised patch.

I'd like to apply this but I'm having a hard time testing it. Could you provide some tests for the different paths of:

- Redmine::Hook.call_hook
- Redmine::Hook::Helper.call_hook
- Redmine::Hook::ViewListener.render_on

#11 - 2009-02-10 04:15 - Eric Davis

- % Done changed from 0 to 80

I found a way to test a few of the methods and committed your patch with some modifications in r2429 and r2430. I'd still like to add some tests for Redmine::Hook::Helper.call_hook and Redmine::Hook::ViewListener.render_on before closing this issue. You can find the tests at the bottom of test/unit/lib/redmine/hook_test.rb.

Commit details

Added request and controller objects to the hooks by default.

The request and controller objects are now added to all hook contexts by default. This will also make url_for work better in hooks by setting up the default_url_options :host, :port, and :protocol.

Finally a new helper method render_or has been added to ViewListener. This will let a hook easily render a partial without a full method definition.

#12 - 2009-02-15 09:40 - Jean-Philippe Lang

- Target version deleted (0.8.1)

See #2754

#13 - 2009-02-21 01:29 - Eric Davis

- Status changed from 7 to Resolved

- % Done changed from 80 to 100

This should be finished now. Instead of using the request and setting host, port, and protocol directly I made the default_url_options use :only_path. This was a much cleaner implementation and should work for the majority of the plugin cases.

Any non standard case (like Douglas Manley) can set the host, port, and protocol in their plugin itself using the request context that's passed in:

```
context[:request].env["SERVER_NAME"]
context[:request].env["SERVER_PORT"]
```

A big thanks goes to Peter Suschlik for the idea.

Implemented in r2489 through r2491

#14 - 2009-02-21 01:30 - Eric Davis

- Target version set to 0.8.2

Adding to 0.8.2, since I couldn't reproduce #2754 and I don't think #2754 would be valid anymore after my refactoring.

#15 - 2009-02-21 08:42 - Peter Suschlik

Great, thank you!

#16 - 2009-02-21 22:48 - Douglas Manley

This should work just fine; thanks!

#17 - 2009-02-22 05:10 - Chaoqun Zou

default_url_options[:only_path] ||= true

This will make the default_url_options[:only_path] to be true permanently. So the link in the notification mail will lost it's protocal, host and port.

#18 - 2009-02-22 05:14 - Eric Davis

Chaoqun Zou wrote:

> default_url_options[:only_path] ||= true >
>
> This will make the default_url_options[:only_path] to be true permanently. So the link in the notification mail will lost it's protocal, host and port.

Can you provide a test case showing the incorrect behavior? The email urls were being generated correctly for me (see #2754).

#19 - 2009-02-22 08:32 - Chaoqun Zou

- File hook_test_diff_for_link_test.patch added

Here is a test (in hook_test unit test) for :only_path :

```
  fixtures :issues

  def test_call_hook_will_break_issue_link_in_mail
    issue = Issue.find(1)

    ActionMailer::Base.deliveries.clear
    Mailer.deliver_issue_add(issue)
    mail = ActionMailer::Base.deliveries.last
    puts mail.body

    @hook_module.add_listener(TestLinkToHook)
    @hook_helper.call_hook(:view_layouts_base_html_head)

    ActionMailer::Base.deliveries.clear
    Mailer.deliver_issue_add(issue)
    mail2 = ActionMailer::Base.deliveries.last
    puts mail2.body

    assert_equal mail.body, mail2.body
  end
```

This test will fail before the line 11-12 be commented out.

A diff patch file is attatched.

#20 - 2009-02-25 08:24 - Eric Davis

I believe I fixed the Plugin and Mailer default_url_options in r2522. It turned out to be a tricky bug that was caused by using default_url_options incorrectly.

Both the plugin hooks and Mailer were setting default_url_options incorrectly and causing ActionContoller::UrlWritter to cache the settings on the module (mattr_accessor) causing several url generators to fail in either the plugin hooks or the Mailer. I found a good description of why this was happening on Stackoverflow.

Thanks for reporting the bug and providing the test case, it helped debugging.

#21 - 2009-03-07 12:41 - Jean-Philippe Lang

- Status changed from Resolved to Closed

- Resolution set to Fixed

Merged in 0.8-stable in r2558.

Files

hook.rb.diff	2.47 KB	2009-02-02	Thomas Löber
hook.rb.revised.diff	3.29 KB	2009-02-02	Thomas Löber
hook_test_diff_for_link_test.patch	1.48 KB	2009-02-22	Chaoqun Zou