ChiliProject - Feature # 233: Add a method to return the entire Configuration hash

Status:	Declined	**Priority:**	Normal
Author:	Eric Davis	**Category:**	Refactoring
Created:	2011-02-27	**Assignee:**	
Updated:	2011-12-11	**Due date:**	
Remote issue URL:			
Affected version:			

Description:

Right now you have to access the Configuration hash by it's key directly (@Redmine::Configuration['email_delivery']@). But there is no way to get the entire hash without using @instance_variable_get@ which requires knowing (and not changing) the internal storage attribute.

I propose having a method (or the class itself) return the full hash.

Example

<pre><code class="ruby">
>> Redmine::Configuration['email_delivery']
{
  "delivery_method" => :sendmail
}

>> Redmine::Configuration
Redmine::Configuration

>> Redmine::Configuration.instance_variable_get('@config')
{
          "scm_cvs_command" => nil,
    "scm_mercurial_command" => nil,
    "autologin_cookie_name" => nil,
       "scm_darcs_command" => nil,
  "autologin_cookie_secure" => nil,
      "scm_bazaar_command" => nil,
         "scm_git_command" => nil,
  "attachments_storage_path" => nil,
          "email_delivery" => {
    "delivery_method" => :sendmail
  },
    "scm_subversion_command" => nil,
    "autologin_cookie_path" => nil
}
</code></pre>

History

2011-03-20 07:14 pm - Felix Schäfer

@Redmine::Configuration.load@ returns the whole hash, but reloads it every time, somewhat defeating the use of the module. I'd advocate having @Redmine::Configuration@ return the cached hash, and if possible have @Redmine::Configuration(true)@ return a reloaded hash (should we ever need it).

2011-03-20 07:31 pm - Holger Just

How about letting @Redmine::Configuration@ just inherit from @Hash@? That would give us all the convenience methods for free. And we'd just have to sprinkle a few @super@ calls into the current code.

What could be done is to freeze the hash. That would partly prevent mangling of it if it is passed to certain methods with mess with the passed @options@ hash. However this could also be mitigated by @dup@'ing the result of @[]@ directly on return (which might break stuff on Ruby 1.8.6 as its hash comparison algorithm is rather fubar).

2011-03-21 11:06 pm - Eric Davis

Holger Just wrote:
> How about letting @Redmine::Configuration@ just inherit from @Hash@? That would give us all the convenience methods for free. And we'd just have to sprinkle a few @super@ calls into the current code.

+1

> What could be done is to freeze the hash.

-1 I personally think freezing in Ruby is a bad idea. In a few months someone will need to modify the hash and they can't without removing the freeze. I'd rather keep the hash open and lock it down later if there is problems.

2011-04-06 09:00 pm - Holger Just

- Status changed from Open to Ready for review

See the pull request at https://github.com/chiliproject/chiliproject/pull/35 (towards unstable)

I did several things here.

At first I introduced the top-level @ChiliProject@ module (which is written as @chili_project@ in the file system). Over the time, all of the modules of old @Redmine@ fame should be migrate over there.

Then I introduced @ChiliProject::Configuration@ which is a class derived from @Hash@. It is intended to be used via ChiliProject.config, which is a cached instance of @ChiliProject::Configuration@. The actual configuration is read from @config/configuration.yml@ on initialization of the instance. Additionally, default configuration values are configured on a hash on the class as @ChiliProject::Configuration.defaults@. On access of a configuration value, both hashes are merged.

This means, the actual configuration hash only contains non-default values. The fully merged hash can be retrieved by calling @ChiliProject.config.all@

Following that concept, I got rid of the @default@ section in @configuration.yml@ and instead rely on the data merge feature of YAML to use a default section and included this into the actual environments. For people who don't know YAML, it doesn't make things different, but for people who know it, it's much more clear. Also I got rid of most of the email configuration examples and put them into the Wiki instead.

2011-04-06 09:04 pm - Holger Just

Oh, one thing to consider:

Currently, I set the default value for configuration values near where they are used. This might be confusing if things are used on more than one place. On the upside it gives us locality of concern, that is we don't have a single file where a thousand different things are configured. But I'm open for ideas on which approach is better.

2011-04-06 10:23 pm - Eric Davis

My thoughts:

* Lot of whitespace changes. Any way you can turn off auto-truncate white space in your editor until we do the systemwide change?

* I think adding @lib/chili_project.rb@ is too much for this alone. What if we just wrap @lib/redmine.rb@ for now? (or at least separate this out into another pull request)

<pre><code class="ruby">
chili_project.rb
require 'chili_project/configuration'
require 'redmine'

redmine.rb
remove it's configuration require
</code></pre>

* @Redmine::Configuration@ - I think a standardized "wrapper" for Redmine-to-ChiliProject with a non-production log would be best. Maybe info or debug?

* I prefer having all of the default options in the Configuration class. As it is I can never remember where the SCM code is (model, lib, adapter, etc)

2011-04-07 09:22 pm - Holger Just

See the new pull request at https://github.com/chiliproject/chiliproject/pull/36

Eric Davis wrote:
> Lot of whitespace changes. Any way you can turn off auto-truncate white space in your editor until we do the systemwide change?

I removed all whitespace-only hunks from the commits. The patch is now much cleaner. However, I'd really like to have the system-wide whitespace change soon. There is a massive amount of trailing whitespace in the code (and not just in whitespace-only lines). Also some of the files (like @subversion_adapter.rb@) even use Windows line-endings.

> I think adding @lib/chili_project.rb@ is too much for this alone. What if we just wrap @lib/redmine.rb@ for now? (or at least separate this out into another pull request)

The addition of @lib/chili_project.rb@ is not for this alone, but serves as a starting point. This patch is just the first one to use the new namespace. Gradually, stuff is going to move from the @Redmine::@ namespace out into @ChiliProject::@. But somebody has to make the start. However, I choose your approach. Things should move organically, once they are finished.

> * @Redmine::Configuration@ - I think a standardized "wrapper" for Redmine-to-ChiliProject with a non-production log would be best. Maybe info or debug?

I think we should clearly deprecate old APIs. This would however break compatibility with Redmine plugins. That's why I kept the deprecation warning commented. We could use @ActiveSupport::Deprecation.warn@, but then it would produce a message on each use. And having a @@skip_configuration_deprecation@ is not very sustainable once we start to deprecate more APIs. I don't have a real solution here...

> * I prefer having all of the default options in the Configuration class. As it is I can never remember where the SCM code is (model, lib, adapter, etc)

Okay, moved all of the defaults initialization into @lib/chili_project/configuration.rb@.

2011-04-07 11:17 pm - Eric Davis

Holger Just wrote:

> However, I'd really like to have the system-wide whitespace change soon.

As soon I can can merge in the latest code from Redmine (#288). I'm still looking for some feedback on a few commits though.

> I think we should clearly deprecate old APIs. This would however break compatibility with Redmine plugins. That's why I kept the deprecation warning commented. We could use @ActiveSupport::Deprecation.warn@, but then it would produce a message on each use. And having a @@skip_configuration_deprecation@ is not very sustainable once we start to deprecate more APIs. I don't have a real solution here...

What about having a global toggle for turning internal API deprecation warnings on/off? Keep them off by default but let someone turn them on when they need them (e.g. release testing a plugin). We could take this to the forums if you want...

I'll review the full pull commit soon, probably this weekend. Thanks for re-re-writing it :)

2011-12-11 02:57 pm - Felix Schäfer

Could we get this into 3.0 so as to deprecate Redmine::Configuration and drop it in 4.0? (or in other words: bump :-))

2011-12-11 03:09 pm - Holger Just

I'm not so sure if this is the right way to go anymore.

Instead, I started working on the @Setting@ model to accomplish a couple of features:

* Save settings in correctly typed columns in the database and in the code (as the YAML serializer always sucked, esp. for ints and booleans) (_Status: Prototype_)
* Introduce namespaces (_Status: Prototype_)
** This is especially useful for plugins that don't need to live in a single large hash anymore.
* Merge configuration and settings (_Status: Work in Progress_)
** Forcefully splitting up the configuration into web-based settings and file-based configuration is confusing to users
** *Everything* should be settable from the UI (except for the database.yml)
** The code I wrote for this issue will become part of the @Setting@ model
* Allow the admin to enforce certain setting which can not be changed from the UI (_Status: Work in Progress_)
** This allows admins to enforce things like the URL, path settings, or shellout commands
** These settings can subsequently only be changed by editing a YAML file. By default, there aren't restricted settings
** This helps in environments where only partially trusted people have admin rights to handle custom fields, projects, ...

2011-12-11 03:35 pm - Holger Just

- Status changed from Ready for review to Declined

I'd close this issue in favor of #767. I think we can still deprecate @Redmine::Configuration@ later on, once #767 is finished. And having a single-use compatibility API for that doesn't hurt too much in my eyes.