Redmine - Feature #11104

Proposal for views substitution mechanism for plugins

2012-06-06 18:51 - Vitaly Klimov

Status:	New	**Start date:**	
Priority:	Normal	**Due date:**	
Assignee:		**% Done:**	0%
Category:	Plugin API	**Estimated time:**	0.00 hour
Target version:			
Resolution:			

Description

I would like to propose to include to the Redmine as a rake task mechanism which i implemented in my Plugin views revisions plugin.

In short, logic behind it eliminates necessity to create full plugins distribution packages if some of the views which plugin changes, changed in the Redmine itself.

If this method would be de-facto standard for plugin development, it would remove last obstacle in the flexibility - right now it is possible to patch controllers, models and helpers but for the views we have to provide whole file, and every change to this file in the Redmine core causes plugin malfunction.

My proposal goes below:

Rake task

Rake task called **redmine:plugins:process_version_change** and has following attributes:

plugins	List of plugins folders to process, could be skipped. In this case all plugins are processed
log	Name of the log file. If not present, *stdout* is used

How this works

For this task to process plugin, plugin should have folder named **rev** in its root folder. Folders in this folder mirror folder structure of the plugin folder tree.

Each file in folder has following pattern:

revision(=|!)-version_high.version_low.version_tiny(=|!)-original_filename

- **revision**
  Lowest revision number for which this file should be used. Could be omitted if version presents.
- **version_high.version_low.version_tiny**
  Lowest version number in major.minor.tiny format for which this file should be used. Could be omitted if revision presents.
- **(=|!)**
  Optional. One of the following symbols:
  = - means that this file should be used only for this **exact** version/revision
  ! - means that if Redmine version/revision is higher than this, this file should be removed from plugin folder

Example

Let's say we have plugin that has modifications in files **app/views/issues/new.html.erb** and **app/views/messages/_form.html.erb**

Redmine has different version of first file starting from revision **7723**, also this file changed in version **1.4.1**. Second file is changed in version **1.4.1**. Also for version **1.3.1** exists another special version of this file. And starting from version **2.0.0** this file is not used at all (renamed, let's say).

In this case we should have following folder and files structure in plugin root folder:

```
rev/
```

```
app/
   views/
      issues/
         0000-new.html.erb
         7723-new.html.erb
         1.4.1-new.html.erb
      messages/
         1.3.0-_form.html.erb
         1.3.1=-_form.html.erb
         1.4.1-_form.html.erb
         1.9.9!-_form.html.erb
```

Now lets consider following cases:

1. Redmine version 1.3.2, revision is 7800
   Files selected: **7723-new.html.erb** and **1.4.1-_form.html.erb**
2. Redmine version 1.3.1, revision is unknown but is less than 7723
   Files selected: **0000-new.html.erb** and **1.3.1!-_form.html.erb**
3. Redmine version 2.0.0, revision is unknown
   File selected **1.4.1-new.html.erb**, file ***form.html.erb*** is removed from _app/views/messages folder

For 'real-life' example please check my Board watchers plugin

File selection rules

Files validity check

1. Filename has revision only
   If revision of Redmine is unknown then revision is taken from **redmine_revisions.yml** file. File is valid if Redmine revision is higher or equal. If equal sign presents file is valid **only** if the revision matches redmine revision
2. Filename has only version
   File is valid if Redmine version is higher or equal. If equal sign presents file is valid **only** if the version matches Redmine version
3. Filename has both version and revision
   1. If revision of redmine is known then rules for revision apply
   2. If revision of redmine is unknown then rules for version apply

Selection rules

If there are file that has equal sign it is chosen, otherwise file with highest revision/version is chosen.

Special files

Plugin checks for two special files in Rails root folder:

1. **.version**
   If exists used for current version override. Should contain string in format XX.YY.ZZ (major,minor,tiny).
2. **.revision**
   If exists used to determine current revision of Redmine. Should contain numeric value which is considered as revision number or word **.ignore** - in this case revision would be excluded from process.
3. **<plugin_root>/config/redmine_revisions.yml**
   This file contains mapping between revisions and versions of the Redmine.

History

#1 - 2012-06-06 22:51 - Terence Mill

+1

#2 - 2012-06-09 11:55 - Jean-Philippe Lang

Interesting but I think that the revisions mapping is a bit too complex and not so usefull as most users run stable versions of Redmine (plus Redmine revision is not necessarily known at runtime). I'd like to propose a different scenario that would not require users to run a rake task for substitution.

We could have this plugin structure:

```
app
  views
    issues
      new.html.erb
```

```
versioned_views
  1
    issues
      new.html.erb
  1.4
    issues
      new.html.erb
  1.4.2
    issues
      new.html.erb
  2
    issues
      new.html.erb
```

Redmine could load the appropriate folders at runtime based on the actual version that is running.

Depending of the Redmine version, the following folders will be loaded at runtime in this order:

- Redmine 1.4.0 or 1.4.1: versioned_views/1.4, versioned_views/1, views
- Redmine 1.4.2 or 1.4.3...: versioned_views/1.4.2, versioned_views/1.4, versioned_views/1, views
- Redmine 2.x: versioned_views/2, views

Say you're running Redmine 1.4.2, the view will be loaded from versioned_views/1.4.2 or versioned_views/1.4 or versioned_views/1 or views (first has priority).

#3 - 2012-06-09 12:37 - Terence Mill

Btw. is is tersting to see here the compatibility seems to be , also in redmine 1.x. Would b great to have wiki site just to point this compatibility breaks out.
An i like JP's apporach more, cause it's KISS pattern.

#4 - 2012-06-13 17:48 - Vitaly Klimov

Jean-Philippe Lang wrote:

> Interesting but I think that the revisions mapping is a bit too complex and not so usefull as most users run stable versions of Redmine (plus Redmine revision is not necessarily known at runtime). I'd like to propose a different scenario that would not require users to run a rake task for substitution.

Yes, this would be great and it will load off a lot of cross-version plugin development issues.

#5 - 2012-07-11 02:49 - Jean-Baptiste Barth

+1 for Jean-Philippe's proposal, looks like a good balance between the need for versioned views and the added complexity. I also think we need something like this, I'm currently migrating some plugins to Redmine 2.x, and it rapidly becomes hard to manage local differences in views.

When it comes to users running specific revision, we may assume they run latest trunk when they install or upgrade a plugin, so having a "devel" sub-directory in "versioned_views", inserted before numeric ones if Redmine::VERSION::BRANCH == 'devel', may be enough to cover most cases.

#6 - 2012-07-11 21:00 - Mischa The Evil

Jean-Baptiste Barth wrote:

> +1 for Jean-Philippe's proposal, looks like a good balance between the need for versioned views and the added complexity. [...]
>
> When it comes to users running specific revision, we may assume they run latest trunk when they install or upgrade a plugin, so having a "devel" sub-directory in "versioned_views", inserted before numeric ones if Redmine::VERSION::BRANCH == 'devel', may be enough to cover most cases.

+1 from me too...