Redmine - Defect #26055
Three issues with Redmine::SyntaxHighlighting::CodeRay.language_supported?

2017-05-28 15:40 - Mischa The Evil

Status:	Closed	**Start date:**	
Priority:	Normal	**Due date:**	
Assignee:	Jean-Philippe Lang	**% Done:**	0%
Category:	Code cleanup/refactoring	**Estimated time:**	0.00 hour
Target version:	3.2.7		
Resolution:	Fixed	**Affected version:**	

Description

Issues

While reviewing/researching #25634 (and in its extend, r16501 & r16502 for #25503), I noticed three issues with the implementation - both pre and post r16568:

1. Redmine::SyntaxHighlighting::CodeRay.language_supported? includes internal CodeRay scanners (default, debug, raydebug & scanner)
2. Redmine::SyntaxHighlighting::CodeRay.language_supported? has multiple responsibilities:
    1. retrieval of the array of supported languages symbols from CodeRay library
        1. storing of a reference to the resulting array in a local variable
    2. checking if the passed language argument is included in the array referenced by the local variable and return the result
3. the array containing supported languages symbols is rebuild for every invocation of Redmine::SyntaxHighlighting::CodeRay.language_supported?, ie. for each and every pre-formatted, syntax-highlighted code block (this seems a uselessly CPU-intensive approach since the array's values won't change unless Redmine is restarted)

Fixes/changes

I've applied the following four changes to solve each of the three issues:

- *change 1 (0001-Remove-internal-CodeRay-scanners.patch)*: remove the four internal CodeRay scanners by subtracting a hard-coded array containing the symbols of these scanners
    - It does not seem to be possible to do this in a more dynamical way.
- *change 2 (0002-Pull-up-retrieve_supported_languages-private-class-m.patch)*: pull-up a new Redmine::SyntaxHighlighting::CodeRay.retrieve_supported_languages private class method (that doesn't store the result)
- *change 3 (0003-Use-stored-ref.-to-array-holding-supported-languages.patch)*: use a stored reference to the resulting array — retrieved by Redmine::SyntaxHighlighting::CodeRay.retrieve_supported_languages, holding the supported languages symbols — via a constant[1]
    - I think there are two ways of solving this: storing a reference to the resulting array 1) in a *memoized instance variable* or 2) in a *constant*. After some benchmarking[1] I decided to go with the constant-storage approach as it seems a tiny bit faster than the memoized instance variable storage approach. Though, both approaches are ~360-400 times faster than the approaches where the array is rebuild on every invocation of Redmine::SyntaxHighlighting::CodeRay.language_supported?.
- *change 4 (0004-Tests-for-Redmine-SyntaxHighlighting-CodeRay.retriev.patch)*: add test-coverage for Redmine::SyntaxHighlighting::CodeRay.retrieve_supported_languages and remove obsolete ApplicationHelperTest#test_syntax_highlight_by_coderay_alias (there's no need to test through textilizable)

This patch serial, against current source:/trunk@16580, is produced using git format-patch, which makes the individual patches apply-able using "patch -p1 < 0001-...".

Environment

```
Environment:
  Redmine version             3.3.3.devel@r16580
  Ruby version                2.3.3-p222 (2016-11-21) [x86_64-linux]
  Rails version               4.2.8
  Environment                 production
  Database adapter            Mysql2
SCM:
  Subversion                  1.8.8
  Git                         1.9.1
  Filesystem
Redmine plugins:
  no plugin installed
```

Footnotes

[1] I passed the following benchmark script to rails runner -e production:

```
puts "####################"
puts "Using benchmark:"
puts "####################"

puts

require 'benchmark'

Benchmark.bmbm do |x|
x.report(".language_supported_core_pre_25634?")  { 100.times do Redmine::SyntaxHighlighting::CodeRay
.language_supported_core_pre_25634?('ruby') end }
x.report(".language_supported_core?")            { 100.times do Redmine::SyntaxHighlighting::CodeRay
.language_supported_core?('ruby') end }
x.report(".language_supported_core_ext?")        { 100.times do Redmine::SyntaxHighlighting::CodeRay
.language_supported_core_ext?('ruby') end }
x.report(".language_supported_core_ext_split?")  { 100.times do Redmine::SyntaxHighlighting::CodeRay
.language_supported_core_ext_split?('ruby') end }
x.report(".language_supported_ivar_ext_split?")  { 100.times do Redmine::SyntaxHighlighting::CodeRay
.language_supported_ivar_ext_split?('ruby') end }
x.report(".language_supported_const_ext_split?") { 100.times do Redmine::SyntaxHighlighting::CodeRay
.language_supported_const_ext_split?('ruby') end }
end

puts

puts "####################"
puts "Using benchmark/ips:"
puts "####################"

puts

require 'benchmark/ips'

Benchmark.ips do |x|
  # Configure the number of seconds used during
  # the warmup phase (default 2) and calculation phase (default 5)
  x.config(:time => 5, :warmup => 2)

  x.report(".language_supported_core_pre_25634?")  { 100.times do Redmine::SyntaxHighlighting::CodeRay
.language_supported_core_pre_25634?('ruby') end }
  x.report(".language_supported_core?")            { 100.times do Redmine::SyntaxHighlighting::CodeRay
.language_supported_core?('ruby') end }
  x.report(".language_supported_core_ext?")        { 100.times do Redmine::SyntaxHighlighting::CodeRay
.language_supported_core_ext?('ruby') end }
  x.report(".language_supported_core_ext_split?")  { 100.times do Redmine::SyntaxHighlighting::CodeRay
.language_supported_core_ext_split?('ruby') end }
  x.report(".language_supported_ivar_ext_split?")  { 100.times do Redmine::SyntaxHighlighting::CodeRay
.language_supported_ivar_ext_split?('ruby') end }
  x.report(".language_supported_const_ext_split?") { 100.times do Redmine::SyntaxHighlighting::CodeRay
.language_supported_const_ext_split?('ruby') end }

  # Compare the iterations per second of the various reports
  x.compare!
end
```

on a modified ./lib/redmine/syntax_highlighting.rb file[2], which gave me the following results:

```
####################
Using benchmark:
####################

Rehearsal ------------------------------------------------------------------
.language_supported_core_pre_25634?    0.010000   0.010000   0.020000 (  0.017590)
```

```
.language_supported_core?              0.010000   0.000000   0.010000 (  0.018621)
.language_supported_core_ext?          0.020000   0.010000   0.030000 (  0.018680)
.language_supported_core_ext_split?    0.010000   0.000000   0.010000 (  0.018822)
.language_supported_ivar_ext_split?    0.000000   0.000000   0.000000 (  0.000302)
.language_supported_const_ext_split?   0.000000   0.000000   0.000000 (  0.000069)
---------------------------------------------------------------- total: 0.070000sec

                                    user     system      total        real
.language_supported_core_pre_25634?    0.030000   0.000000   0.030000 (  0.025548)
.language_supported_core?              0.010000   0.000000   0.010000 (  0.018236)
.language_supported_core_ext?          0.020000   0.000000   0.020000 (  0.018472)
.language_supported_core_ext_split?    0.020000   0.000000   0.020000 (  0.018835)
.language_supported_ivar_ext_split?    0.000000   0.000000   0.000000 (  0.000068)
.language_supported_const_ext_split?   0.000000   0.000000   0.000000 (  0.000049)

####################
Using benchmark/ips:
####################

Warming up --------------------------------------
.language_supported_core_pre_25634?
                         5.000  i/100ms
.language_supported_core?
                         5.000  i/100ms
.language_supported_core_ext?
                         4.000  i/100ms
.language_supported_core_ext_split?
                         4.000  i/100ms
.language_supported_ivar_ext_split?
                         1.763k i/100ms
.language_supported_const_ext_split?
                         2.084k i/100ms
Calculating -------------------------------------
.language_supported_core_pre_25634?
                         53.819  (± 5.6%) i/s -    270.000  in   5.035559s
.language_supported_core?
                         53.027  (± 5.7%) i/s -    265.000  in   5.015003s
.language_supported_core_ext?
                         50.711  (± 5.9%) i/s -    256.000  in   5.071990s
.language_supported_core_ext_split?
                         49.493  (± 8.1%) i/s -    248.000  in   5.055206s
.language_supported_ivar_ext_split?
                         17.443k (±10.2%) i/s -     86.387k in   5.011711s
.language_supported_const_ext_split?
                         19.710k (±11.7%) i/s -     97.948k in   5.045098s

Comparison:
.language_supported_const_ext_split?:  19710.5 i/s
.language_supported_ivar_ext_split?:   17443.3 i/s - same-ish: difference falls within error
.language_supported_core_pre_25634?:      53.8 i/s - 366.23x  slower
.language_supported_core?:                53.0 i/s - 371.71x  slower
.language_supported_core_ext?:            50.7 i/s - 388.68x  slower
.language_supported_core_ext_split?:      49.5 i/s - 398.24x  slower
```

[2] including six different implementations looking like:

Nr.	Method	Description	Abbreviated implementation code
1. {{collapse(View abbreviated implementation...) class << self ... def language_supported_core_pre_25634?(language)	Redmine::SyntaxHighlighting::CodeRay.language_supported_core_pre_25634?	state of trunk, pre #25634	

::CodeRay::Scanners.list.include?(language.to_s.downcase.to_sym) rescue false end end }}				
2. {{collapse(View abbreviated implementation...) class << self ... def language_supported_core?(language) supported_languages = ::CodeRay::Scanners.list + ::CodeRay::Scanners.plugin_hash.keys.map(&:to_sym) supported_languages.include?(language.to_s.downcase.to_sym) rescue false end end }}	Redmine::SyntaxHighlighting::CodeRay.language_supported_core?	current state of trunk, post #25634		
3. {{collapse(View abbreviated implementation...) class << self ... def language_supported_core_ext?(language) supported_languages = ::CodeRay::Scanners.list + ::CodeRay::Scanners.plugin_hash.keys.map(&:to_sym) - %w(debug default raydebug scanner).map(&:to_sym) supported_languages.include?(language.to_s.downcase.to_sym) rescue false end end }}	Redmine::SyntaxHighlighting::CodeRay.language_supported_core_ext?	current state of trunk, post #25634, with change 1 applied		
4. {{collapse(View abbreviated implementation...) def self.retrieve_supported_languages ::CodeRay::Scanners.list + # Add CodeRay scanner aliases ::CodeRay::Scanners.plugin_hash.keys.map(&:to_sym) - # Remove internal CodeRay scanners %w(debug default raydebug scanner).map(&:to_sym) end private_class_method :retrieve_supported_languages class << self ... def language_supported_core_ext_split?(language) supported_languages = retrieve_supported_languages supported_languages.include?	Redmine::SyntaxHighlighting::CodeRay.language_supported_core_ext_split?	current state of trunk, post #25634, with change 1 and 2 applied		

(language.to_s.downcase.to_sym) rescue false end end }}			
5. {{collapse(View abbreviated implementation...) def self.retrieve_supported_languages ::CodeRay::Scanners.list + # Add CodeRay scanner aliases ::CodeRay::Scanners.plugin_hash.keys.map(&:to_sym) - # Remove internal CodeRay scanners %w(debug default raydebug scanner).map(&:to_sym) end private_class_method :retrieve_supported_languages class << self ... def language_supported_ivar_ext_split?(language) @supported_languages ||= retrieve_supported_languages @supported_languages.include?(language.to_s.downcase.to_sym) rescue false end }}	Redmine::SyntaxHighlighting::CodeRay.language_supported_ivar_ext_split?	current state of trunk, post #25634, with change 1, 2 and 3 (memoized ivar variant) applied	
6. {{collapse(View abbreviated implementation...) def self.retrieve_supported_languages ::CodeRay::Scanners.list + # Add CodeRay scanner aliases ::CodeRay::Scanners.plugin_hash.keys.map(&:to_sym) - # Remove internal CodeRay scanners %w(debug default raydebug scanner).map(&:to_sym) end private_class_method :retrieve_supported_languages SUPPORTED_LANGUAGES = retrieve_supported_languages class << self ... def language_supported_const_ext_split?(language) SUPPORTED_LANGUAGES.include?(language.to_s.downcase.to_sym) rescue false end }}	Redmine::SyntaxHighlighting::CodeRay.language_supported_const_ext_split?	current state of trunk, post #25634, with change 1, 2 and 3 (constant variant) applied	

Related issues:

Related to Redmine - Defect # 25634: Highlight language aliases are no more s...	**Closed**
Related to Redmine - Feature # 35676: Optimize performance of syntax highligh...	**New**

Associated revisions

Revision 16622 - 2017-06-06 23:54 - Jean-Philippe Lang

Remove internal CodeRay scanners (#26055).

Patch by Mischa The Evil.

Revision 16623 - 2017-06-06 23:55 - Jean-Philippe Lang

Pull-up retrieve_supported_languages private class method (#26055).

Patch by Mischa The Evil.

Revision 16624 - 2017-06-06 23:55 - Jean-Philippe Lang

Use stored ref. to array holding supported languages symbols via a constant (#26055).

Patch by Mischa The Evil.

Revision 16625 - 2017-06-06 23:56 - Jean-Philippe Lang

Tests for Redmine::SyntaxHighlighting::CodeRay.retrieve_supported_languages (#26055).

Patch by Mischa The Evil.

Revision 16630 - 2017-06-07 21:35 - Jean-Philippe Lang

Merged r16622 to r16625 (#26055).

Revision 16631 - 2017-06-07 21:35 - Jean-Philippe Lang

Merged r16622 to r16625 (#26055).

History

#1 - 2017-05-28 15:43 - Mischa The Evil

- Related to Defect #25634: Highlight language aliases are no more supported added

#2 - 2017-05-28 15:58 - Mischa The Evil

- File 0001-Remove-internal-CodeRay-scanners.patch added

- File 0002-Pull-up-retrieve_supported_languages-private-class-m.patch added

- File 0003-Use-stored-ref.-to-array-holding-supported-languages.patch added

- File 0004-Tests-for-Redmine-SyntaxHighlighting-CodeRay.retriev.patch added

Added the patches.

#3 - 2017-05-28 16:03 - Mischa The Evil

- File deleted (0003-Use-stored-ref.-to-array-holding-supported-languages.patch)

#4 - 2017-05-28 16:03 - Mischa The Evil

- File deleted (0004-Tests-for-Redmine-SyntaxHighlighting-CodeRay.retriev.patch)

#5 - 2017-05-28 16:03 - Mischa The Evil

- File 0003-Use-stored-ref.-to-array-holding-supported-languages.patch added

- File 0004-Tests-for-Redmine-SyntaxHighlighting-CodeRay.retriev.patch added

#6 - 2017-05-30 19:20 - Holger Just

Thanks Mischa for digging into that.

The Redmine::SyntaxHighlighting::CodeRay.language_supported? method is currently mostly used to restrict the CSS classes which are allowed to be used for syntax highlighted blocks. As such, it is not a huge problem that these "internal" scanners are included there. Still, for the sake of providing a clean interface without any surprises, your patches are still desirable. The performance increase also helps.

I just had a look over your patches and they look very clear and straight forward. From my point of view, they can (and should) be applied as is.

#7 - 2017-06-06 23:58 - Jean-Philippe Lang

- Category changed from Text formatting to Code cleanup/refactoring

- Status changed from New to Resolved

- Assignee set to Jean-Philippe Lang

- Resolution set to Fixed

Committed, thanks Mischa for these patches and thanks to Holger for the review.

#8 - 2017-06-07 21:35 - Jean-Philippe Lang

- Status changed from Resolved to Closed

#10 - 2017-06-28 20:31 - Mischa The Evil

FTR: I've proposed to add this functionality to the CodeRay API itself, see https://github.com/rubychan/coderay/pull/210.

#11 - 2021-08-04 17:38 - Mischa The Evil

- Related to Feature #35676: Optimize performance of syntax highlighting implementation added

Files

0001-Remove-internal-CodeRay-scanners.patch	967 Bytes	2017-05-28	Mischa The Evil
0002-Pull-up-retrieve_supported_languages-private-class-m.patch	1.56 KB	2017-05-28	Mischa The Evil
0003-Use-stored-ref.-to-array-holding-supported-languages.patch	1.31 KB	2017-05-28	Mischa The Evil
0004-Tests-for-Redmine-SyntaxHighlighting-CodeRay.retriev.patch	3.66 KB	2017-05-28	Mischa The Evil