Redmine - Defect #29855
add_working_days returns wrong date

2018-10-27 08:46 - Yutaka Hara

Status:	Confirmed	**Start date:**	
Priority:	Normal	**Due date:**	
Assignee:	Go MAEDA	**% Done:**	0%
Category:	Issues	**Estimated time:**	0.00 hour
Target version:	Candidate for next minor release		
Resolution:		**Affected version:**	3.3.7

Description

Redmine::Util::DateCalculation#add_working_days(date, n) returns wrong date when date is holiday and n is a multiple of 5.

Example:

```
irb(main):004:0> Setting.non_working_week_days
=> ["6", "7"]
irb(main):001:0> include Redmine::Utils::DateCalculation
irb(main):002:0> add_working_days(Date.new(2018, 10, 27), 5)
=> Mon, 05 Nov 2018   # Expected Fri, 02 Nov 2018
irb(main):003:0> add_working_days(Date.new(2018, 10, 28), 5)
=> Mon, 05 Nov 2018   # Expected Fri, 02 Nov 2018
```

Tested with trunk@17598

Related issues:

Related to Redmine - Defect # 14846: Calculation of the start date of followi...	**Closed**

History

#1 - 2018-10-28 02:46 - Go MAEDA

- Related to Defect #14846: Calculation of the start date of following issues ignores the "non-working days" setting added

#2 - 2018-10-28 02:47 - Go MAEDA

- Category set to Issues

#3 - 2018-10-28 02:53 - Go MAEDA

- Description updated

- Status changed from New to Confirmed

- Affected version set to 3.3.7

I have confirmed that 3.3-stable and 3.4-stable are also affected.

#4 - 2018-10-30 02:29 - Go MAEDA

- Description updated

#5 - 2018-11-07 07:56 - Mizuki ISHIKAWA

- File fix-29855.patch added

I think that applying this patch will solve the problem.
The code of the add_working_days method changes quite a bit, but all the tests succeed.

Any feedback is welcome.

#6 - 2018-11-25 07:54 - Go MAEDA

The suggested fix works fine but it is much slower than the current code. I think we need to consider whether this will affect the performance of Redmine.

```
$ bin/rails r bench-29855.rb
Warming up --------------------------------------
            before    12.236k i/100ms
             after   997.000  i/100ms
Calculating -------------------------------------
            before    159.524k (± 4.7%) i/s -    807.576k in   5.073660s
             after     10.597k (± 3.4%) i/s -     53.838k in   5.086474s

Comparison:
            before:   159524.1 i/s
             after:    10597.0 i/s - 15.05x  slower
```

```
require 'benchmark/ips'

include Redmine::Utils::DateCalculation

Benchmark.ips do |x|
  x.report('before') do
    add_working_days(Date.today, 30)
  end

  x.report('after') do
    result = Date.today
    30.times do
      result = next_working_date(result + 1)
    end
    result
  end

  x.compare!
end
```

#7 - 2018-12-01 09:50 - Go MAEDA

- Assignee set to Jean-Philippe Lang
- Target version set to 3.3.9

Jean-Philippe, do you think we can accept this performance deterioration?

I think it is OK because 'add_working_days' method will not be executed hundreds of times by the user's single operation. So, it does not affect the

performance of Redmine.

#8 - 2018-12-02 08:53 - Jean-Philippe Lang

- Assignee changed from Jean-Philippe Lang to Yutaka Hara

Mizuki ISHIKAWA wrote:

> *Any feedback is welcome.*

DateCalculation#working_days should be fixed in a similar way to be consistent with the proposed fix. These new assertions should pass:

```
Index: test/unit/lib/redmine/utils/date_calculation.rb
===================================================================
--- test/unit/lib/redmine/utils/date_calculation.rb    (revision 17671)
+++ test/unit/lib/redmine/utils/date_calculation.rb    (working copy)
@@ -41,6 +41,8 @@
       assert_working_days  8, '2012-10-11', '2012-10-23'
       assert_working_days  2, '2012-10-14', '2012-10-17'
       assert_working_days 11, '2012-10-14', '2012-10-30'
+      assert_working_days  5, '2012-10-20', '2012-10-26'
+      assert_working_days  5, '2012-10-21', '2012-10-26'
     end
   end
```

#9 - 2018-12-02 08:55 - Jean-Philippe Lang

- Assignee changed from Yutaka Hara to Go MAEDA

#10 - 2018-12-02 12:39 - Marius BALTEANU

- Assignee changed from Go MAEDA to Jean-Philippe Lang

I took a look and there are some strange (or wrong) test cases the we should review before changing anything else.

Taking the following test scenario:

```
  def test_working_days_with_non_working_week_days
    with_settings :non_working_week_days => %w(6 7) do
      assert_working_days 14, '2012-10-09', '2012-10-27'
      assert_working_days  4, '2012-10-09', '2012-10-15'
      assert_working_days  4, '2012-10-09', '2012-10-14'
      assert_working_days  3, '2012-10-09', '2012-10-12'
      assert_working_days  8, '2012-10-09', '2012-10-19'
      assert_working_days  8, '2012-10-11', '2012-10-23'
      assert_working_days  2, '2012-10-14', '2012-10-17'
      assert_working_days 11, '2012-10-14', '2012-10-30'
    end
  end
```

assert_working_days 4, '2012-10-09', '2012-10-15'
2012-10-09 was Tuesday
2012-10-15 was Monday
The number of the expected working days according to the test is 4. But in my opinion, it should be 5 days (Tuesday, Wednesday, Thursday, Friday and Monday). 4 could be only if we exclude the end date from the count. if we do this, than the number of the expected days for the 2 assertions proposed by Jean-Philippe should be 4 because we need to exclude Friday (2012-10-26).

Also, it sound incorrect to say that between '2012-10-09 - 2012-10-15 (Tuesday - Monday)' and '2012-10-09 - 2012-10-14 (Tuesday - Sunday)' are the same number of working days (4).

Jean-Philippe, what do you think? I'm in favour of including the end date in the count.

#11 - 2018-12-02 17:23 - Jean-Philippe Lang

Marius BALTEANU wrote:

> *The number of the expected working days according to the test is 4. But in my opinion, it should be 5 days (Tuesday, Wednesday, Thursday, Friday and Monday). 4 could be only if we exclude the end date from the count. if we do this, than the number of the expected days for the 2 assertions proposed by Jean-Philippe should be 4 because we need to exclude Friday (2012-10-26).*

#working_days and #add_working_days are used to reschedule an issue when the start date is changed. Its duration is calculated with #working_days and the new due date is calculated with #add_working_days. If there is no "non working day", they should behave like Date#- and Date#+.

#12 - 2018-12-02 17:35 - Marius BALTEANU

- Assignee changed from Jean-Philippe Lang to Go MAEDA

Jean-Philippe Lang wrote:

> *#working_days and #add_working_days are used to reschedule an issue when the start date is changed. Its duration is calculated with #working_days and the new due date is calculated with #add_working_days. If there is no "non working day", they should behave like Date#- and Date#+.*

Thanks, but are still not clear for me the expected results so I'll leave Go Maeda or Mizuki ISHIKAWA to fix this issue.

#13 - 2018-12-08 07:33 - Jean-Philippe Lang

- Target version deleted (3.3.9)

#14 - 2018-12-08 09:28 - Go MAEDA

- Target version set to Candidate for next minor release

Files

fix-29855.patch	1.64 KB	2018-11-07	Mizuki ISHIKAWA