Redmine - Defect #40914

Fix precision issues in TimeEntry#hours calculation by returning Rational instead of Float

2024-06-25 14:12 - Boris Brodski

Status:	Closed	**Start date:**	
Priority:	Normal	**Due date:**	
Assignee:	Go MAEDA	**% Done:**	0%
Category:	Time tracking	**Estimated time:**	0.00 hour
Target version:	6.0.0		
Resolution:	Fixed	**Affected version:**	5.0.7

Description

On the page my/page, if the *timelog* is displayed, the total is shown at the top of the table. The summing is not correctly implemented, leading to rounding errors.

Reproduction and Example:

1. Add time entries for one day, one issue:
   - 0:10
   - 0:40
   - 0:10
2. Navigate to my/page
3. Add the 'timelog' widget if it is not already present.
4. The time entries should be shown. The total displayed is 1:01.

Implementation and Easy Fix:

Current implementation (app/views/my/blocks/_timelog.html.erb:45):

```
html_hours(format_hours(entries_by_day[day].sum(&:hours)))
```

Summing hours introduces rounding errors. To fix this, convert to minutes first:

```
html_hours(format_hours(entries_by_day[day].map{|t| (t.hours * 60).to_i}.sum / 60))
```

Related issues:

Related to Redmine - Defect #36897: The minutes part of a time entry is displ...	**Closed**
Related to Redmine - Defect #41819: Time entry API returning `hours` as Ratio...	**Closed**
Related to Redmine - Defect #41895: Spent time CSV report returning `hours` a...	**Closed**
Related to Redmine - Defect #42339: Value for "hours" incorrect when creating...	**Closed**

Associated revisions

Revision 23000 - 2024-08-31 11:26 - Go MAEDA

Fix precision issues in TimeEntry#hours calculation by returning Rational instead of Float (#40914).

Patch by Go MAEDA (user:maeda).

History

#1 - 2024-06-25 14:20 - Boris Brodski

In the same way the <h3>..</h3> tag should be fixed:

Current implementation (app/views/my/blocks/_timelog.html.erb:7):

```
l_hours_short entries.sum(&:hours)
```

Fix:

```
l_hours_short(entries.map{|t| (t.hours * 60).to_i}.sum / 60)
```

#2 - 2024-06-27 11:19 - Go MAEDA

- Status changed from New to Confirmed

#3 - 2024-06-30 09:56 - Go MAEDA

- File clipboard-202406301643-wtqrs.png added

Here is another solution. This patch will likely also fix the issue reported in #36897.

```
diff --git a/lib/redmine/i18n.rb b/lib/redmine/i18n.rb
index 83ecb05eb..52221029f 100644
--- a/lib/redmine/i18n.rb
+++ b/lib/redmine/i18n.rb
@@ -93,8 +93,9 @@ module Redmine
       return "" if hours.blank?

       if Setting.timespan_format == 'minutes'
-        h = hours.floor
-        m = ((hours - h) * 60).round
+        rational_hours = hours.rationalize(Rational('1/60'))
+        h = rational_hours.truncate
+        m = ((rational_hours - h) * 60).round
         "%d:%02d" % [h, m]
       else
         number_with_delimiter(sprintf('%.2f', hours.to_f), delimiter: nil)
```

clipboard-202406301643-wtqrs.png

#4 - 2024-06-30 10:32 - Go MAEDA

Go MAEDA wrote in #note-3:

> Here is another solution. This patch will likely also fix the issue reported in #36897.
>
> [...]

Sorry, using rationalize(Rational('1/60')) (approximation in minutes) breaks some tests. rationalize(Rational('1/3600')) (approximation in seconds) works fine.

```
Failure:
TimelogReportTest#test_report_all_time_by_day [test/functional/timelog_report_test.rb:87]:
Expected: "162:54"
  Actual: "162:53".
Expected 0 to be >= 1.

bin/rails test test/functional/timelog_report_test.rb:84

diff --git a/lib/redmine/i18n.rb b/lib/redmine/i18n.rb
index 83ecb05eb..40b991f8a 100644
--- a/lib/redmine/i18n.rb
+++ b/lib/redmine/i18n.rb
@@ -93,8 +93,9 @@ module Redmine
       return "" if hours.blank?

       if Setting.timespan_format == 'minutes'
-        h = hours.floor
-        m = ((hours - h) * 60).round
+        rational_hours = hours.rationalize(Rational('1/3600'))
+        h = rational_hours.truncate
+        m = ((rational_hours - h) * 60).round
         "%d:%02d" % [h, m]
       else
         number_with_delimiter(sprintf('%.2f', hours.to_f), delimiter: nil)
```

#5 - 2024-06-30 10:35 - Go MAEDA

- Related to Defect #36897: The minutes part of a time entry is displayed as 60 instead of being carried over added

#6 - 2024-06-30 17:09 - Go MAEDA

- File 40914.patch added

The attached patch fixes rounding errors when displaying spent hours. Converting float values to rational numbers ensures accurate representation and avoids cumulative rounding errors.

This patch also fixes #36897.

#7 - 2024-07-01 01:41 - Go MAEDA

- File 40914.patch added

#8 - 2024-07-01 01:42 - Go MAEDA

- File deleted (40914.patch)

#9 - 2024-07-03 16:19 - Go MAEDA

- Target version set to Candidate for next major release

#10 - 2024-07-16 11:14 - Go MAEDA

- Target version changed from Candidate for next major release to 6.0.0

Setting the target version to 6.0.0.

#11 - 2024-07-23 10:37 - Go MAEDA

- File 40914-v2.patch added

- Subject changed from Rounding error in my/page -> timelog to Fix precision issues in TimeEntry#hours calculation by returning Rational instead of Float

I have updated the patch.

In the new patch, I have removed the change to the Redmine::I18n#format_hours method. The change has been posted as a separate patch in issue #36897.

The previous patch included tests for the behavior of the My Page blocks. However, the fix in the TimeEntry#hours method affects not just the My Page but the entire application. Therefore, I have moved the tests to test/unit/time_entry_test.rb.

#12 - 2024-07-23 15:30 - Holger Just

The previous code hours = hours.to_f in the l_hours method would also accept strings such as "0.5" and format them correctly. With your proposed patch, this would break. I'm not sure if we actually do pass strings there, but I would assume that this can happen, e.g., when e.g. round-tripping raw params.

Also, while I thing the logic to use rationals is a nice and elegant solution, we likely have to check / update other places with similar logic where we calculate sums. This often happens in the database now with a SUM(hours) statement in SQL. Here, the database would again use floats right now and thus show inconsistent sums, depending on where they are calculated.

It might turn out that the only actual and consistent solution for this is to to replace the current fractional hours column with an integer minutes column and calculate the times based on that. Unfortunately, this would be a rather large change though...

#13 - 2024-07-28 06:45 - Go MAEDA

Holger Just wrote in #note-12:

> The previous code hours = hours.to_f in the l_hours method would also accept strings such as "0.5" and format them correctly. With your proposed patch, this would break. I'm not sure if we actually do pass strings there, but I would assume that this can happen, e.g., when e.g. round-tripping raw params.

I have fixed this issue in the new patch. The method now properly handles string inputs such as “0.5”.

> Also, while I thing the logic to use rationals is a nice and elegant solution, we likely have to check / update other places with similar logic where we calculate sums. This often happens in the database now with a SUM(hours) statement in SQL. Here, the database would again use floats right now and thus show inconsistent sums, depending on where they are calculated.

You are correct that applying this patch will not resolve the inconsistency between results calculated using Ruby and results calculated using SQL. However, the overall inconsistencies in Redmine will be reduced with this patch.

The reason is that, before applying this patch, the TimeEntry#hours method rounds the return value to two decimal places (e.g. 0.17), while SQL SUM uses higher precision values (e.g. 0.16666666666666666). Therefore, errors in calculation will occur less frequently in SQL calculations compared to when the sum is calculated in Ruby code.

Test data preparation:

```
Issue.create!(project_id: 1, author_id: 1, tracker_id: 1, subject: 'test')
TimeEntry.create!(issue: issue, hours: '10m', user_id: 1, author_id: 1, spent_on: DateTime.now)
TimeEntry.create!(issue: issue, hours: '10m', user_id: 1, author_id: 1, spent_on: DateTime.now)
```

```
TimeEntry.create!(issue: issue, hours: '40m', user_id: 1, author_id: 1, spent_on: DateTime.now)
```

Calculating the sum using Ruby code (without the patch):
Float values are rounded to two decimal places (e.g. 0.17), so calculations using these values often return an incorrect value.

```
irb(main):001> TimeEntry.where(issue: Issue.last).map(&:hours).sum
  Issue Load (0.1ms)  SELECT "issues".* FROM "issues" ORDER BY "issues"."id" DESC LIMIT ?  [["LIMIT", 1]]
  TimeEntry Load (0.1ms)  SELECT "time_entries".* FROM "time_entries" WHERE "time_entries"."issue_id" = ?  [["
issue_id", 16]]
=> 1.01
```

Calculating the sum using Ruby code (with the patch):

```
irb(main):003> TimeEntry.where(issue: Issue.last).map(&:hours).sum
  Issue Load (0.1ms)  SELECT "issues".* FROM "issues" ORDER BY "issues"."id" DESC LIMIT ?  [["LIMIT", 1]]
  TimeEntry Load (0.1ms)  SELECT "time_entries".* FROM "time_entries" WHERE "time_entries"."issue_id" = ?  [["
issue_id", 16]]
=> (1/1)
```

Calculating the sum using SQL:
Since higher precision values (e.g. 0.16666666666666666) are used than the return values of TimeEntry#hours, it returns a correct value in many cases.

```
irb(main):002> TimeEntry.where(issue: Issue.last).sum(:hours)
  Issue Load (0.1ms)  SELECT "issues".* FROM "issues" ORDER BY "issues"."id" DESC LIMIT ?  [["LIMIT", 1]]
  TimeEntry Sum (0.1ms)  SELECT SUM("time_entries"."hours") FROM "time_entries" WHERE "time_entries"."issue_id
" = ?  [["issue_id", 16]]
=> 1.0
```

In conclusion, this patch does not introduce new inconsistencies or worsen existing ones. While the SQL calculation results remain occasionally inaccurate both before and after applying the patch, the Ruby calculation results, which is often inaccurate, will always be accurate. As a result, overall inconsistencies will be reduced.

	Sum by Ruby	Sum by SQL
Without the patch	Often inaccurate	Occasionally inaccurate
With the patch	Accurate	Occasionally inaccurate

#14 - 2024-07-31 10:18 - Go MAEDA

- File 40914-v3.patch added

I forgot to attach an updated patch in #note-13.

#15 - 2024-08-31 11:27 - Go MAEDA

- Status changed from Confirmed to Closed

- Assignee set to Go MAEDA

- Resolution set to Fixed

Committed the fix in r23000.

#16 - 2024-12-01 07:23 - Go MAEDA

- Related to Defect #41819: Time entry API returning `hours` as Rational instead of Float added

#17 - 2024-12-01 07:23 - Go MAEDA

- Related to Defect #41895: Spent time CSV report returning `hours` as Rational instead of Float added

#18 - 2025-03-02 02:20 - Go MAEDA

- Related to Defect #42339: Value for "hours" incorrect when creating a time entry using the API added

Files

clipboard-202406301643-wtqrs.png	69.6 KB	2024-06-30	Go MAEDA
40914.patch	3.99 KB	2024-06-30	Go MAEDA
40914-v2.patch	2.53 KB	2024-07-23	Go MAEDA