Redmine - Feature #4601

Issue spent time recording

2010-01-18 22:17 - Jiří Křivánek

Status:	Closed	**Start date:**	2010-01-18
Priority:	Normal	**Due date:**	
Assignee:		**% Done:**	0%
Category:		**Estimated time:**	0.00 hour
Target version:			
Resolution:	Duplicate		

Description

Hello,

I have just discovered the Redmine and I am considering whether it fits my needs. Well, 90% yes. I am missing the following feature:

The time recording now just involves entering the number of hours spent on the issue. I would preffer entering the start time and finish time, whilst the number of hours could be calculated automatically (simple subtraction).

For example:
- Working on issue XY.
- Works were performed on a certain date.
- Works started at 13:00 (the granularlity should be configurable - e.g. 15minutes, 30minutes, 1hour).
- Works finished at 15:30.
- The time spent on the issue as recorded is calculated automatically: 2.5 hours.

I scanned the issues list for something like that and I did not find such an feature request.

I tried to define the custom fields but I failed to make them of the TIME format, neither to be used for the automatic calculations.

I also tried to install the plugin "My effort pluging" but I failed to make it working.

Any chance to have such a functionality in a historically close future (say weeks)?

Thank you & best regards,

Jiri.

Related issues:

Is duplicate of Redmine - Feature #440: Time tracking stopwatch	**New**

History

#1 - 2010-01-20 05:21 - Ewan Makepeace

+1

If we are to record time spent on a project we might as well make it useful for time sheet applications?

#2 - 2010-01-20 12:46 - Jiří Křivánek

Sorry, I do not know what you mean with the "time sheet applications"...

#3 - 2010-01-20 13:22 - Felix Schäfer

Well, at the time being, you can enter a time and a date, but you'd want it to be more precise and show when you spent this time, correct?

#4 - 2010-01-20 15:29 - Jiří Křivánek

- File WorkMonitor.png added

Yes, the current support for the spent time recording only involves
- the date,
- the amount of time,
- the comment and
- the kind of activity.

I would like it to be slightly different, the spent time recording would invlove the following data:
- the date (read-write: calendar),
- the start time of day (read-write: dropdown list),
- the end time of day (read-write: dropdown list),
- the amount of time (read-only: automatically calculated as end time of day minus start time of day),
- the comment (read-write: text area) and
- the kind of activity (read-write: dropdown list).

The experience in the behind of this feature request is following: My management requires me (as a team leader) to give them the real effort feedback, per developper and project. Until I started to request the "full time qualified work diary records", the team members (SW and HW designers in my case) used to give me just their estimates of how long they spent on the particular tasks (vague -> non-precise -> useless input). As soon as they MUST enter the date, start time and end time of their works on a particular task, they get must more reliable (as this can much simplerly be checked by me as their boss).

For this purpose, I am currently using my own, very simple "WorkMonitor" For your better understanding, I am attaching the screenshot of the typical usage (it is czech, but I still hope you can understand what I meant). Well, my private system does not support project management (like Redmin does). So I decided to abandon my system and start using Redmine. But I still need the perfectly granular work records.

#5 - 2010-01-20 15:35 - Jiří Křivánek

- File RecordingWorks.png added

#6 - 2010-01-20 16:48 - Felix Schäfer

I'm not sure how nicely existing datasets will migrate to such a scheme though, as currently redmine only tracks the date (not datetime) and the time spent. If we just convert that data to a datetime, we could get the extra bit of information you want (the start time), but existing entries would all start at midnight of the entry date, which could seem pretty weird :-)

I'm also not comfortable with only allowing to choose from a fixed set of times at regular intervals, as other people might want the extra precision, or making any of the three "time" fields read-only (I'd rather have the end-time automatically set to the current time, and enter an amount of time a have the start-time be calculated on-the-fly).

#7 - 2010-01-20 19:31 - Jiří Křivánek

What I am suggesting does not involve changing the date to timestamp. The date would stay as is. There just would be added a two new fields of the time data type - one as a start time, another for an end time.

I believe that the solution is backward compatible - i.e. the existing records can stay untouched - the two new time fields will be just NULL, which is no problem!

Furthermore, this functionality can simply be configurable: The UI can either behave like todate or variantly like I am proposing.

I actually already tried to achieve this with the custom fields (as I already mentioned in my initial spec). However, there is no time data type for the custom fields available. Moreove, it is not really possible to achieve the automatic calculation of the time amount based on the start and end times.

OK, the fixed intervals may be considered limiting for someone - that is why I initially proposed it to be configurable.

#8 - 2010-01-21 08:37 - Jiří Křivánek

I am hardly trying to learn Ruby (and later also Rails). It will take some time, but perhaps, later, I will be able to participate...

BTW: Would you be able to explain what this syntax means:

```
create_table "time_entries", :force => true do |t|
```

In know that it is the method call but the part after the comma is tricky to me and I am not able to learn it from the Ruby doc...

What I am suggesting involves the following change into your data model for time_entries entity.

I do not know whether your SQL server supports triggers (the stored procedures automatically execured on the SQL server before/after insert/update/delete). If so then all the business logic may be implemented in these. I know that some pure SQL servers do not support these high end features (but these are still evolving)... so otherwise, it has to be implemented in Ruby...

- t.time "start_time", :null => true
  t.time "end_time", :null => true
- t.float "hours", :null => false
  t.string "comments"
  t.integer "activity_id", :null => false
  t.date "spent_on", :null => false
  t.integer "tyear", :null => false
  t.integer "tmonth", :null => false
  t.integer "tweek", :null => false
  t.datetime "created_on", :null => false

t.datetime "updated_on", :null => false
end

```
create_table "time_entries", :force => true do |t|
    t.integer  "project_id",  :null => false
    t.integer  "user_id",     :null => false
    t.integer  "issue_id"
```

#9 - 2010-01-21 09:34 - Felix Schäfer

(I'll just make the ruby/rails stuff extra so it doesn't clutter the other answer)

Jiří Křivánek wrote:

> BTW: Would you be able to explain what this syntax means:
>
> > create_table "time_entries", :force => true do |t|
>
> In know that it is the method call but the part after the comma is tricky to me and I am not able to learn it from the Ruby doc…

It might be more clear written this way: create_table("time_entries", :force => true) do |t|. In essence, create_table takes 2 arguments: table_name, options = {}. table_name is mandatory as it is not defaulted in the definition, options however defaults to an empty hash, that's why you'll have seen it without more than one argument most of the time, because the default options are fine.

In ruby, he parentheses for a method call are not mandatory, so some_method arg_one, arg_two, arg_three and some_method(arg_one, arg_two, arg_three) are equivalent. Moreover, having a hash as last argument will glob all extraneous arguments (well, it's a little more complicated than that, but that should describe 95% of all occurrences :-)), so some_method arg_one, arg_two, key_one => value_one, key_two => value_two is the same as some_method(arg_one, arg_two, {key_one => value_one, key_two => value_two}). Last but not least, and I'm sorry I can't remember the name for the :something, but :something has the advantage over other objects that it gets instantiated once and only once for a whole ruby process, and they are more lightweight that other objects, that's why they are widely used as keys for hashes and other similar situations.

I hope this gets you up to speed. You'll find more information about the call on itself in the create_table API reference and the migration API, though I'd also recommend the rails guides if you are new to ruby and rails.

#10 - 2010-01-21 10:00 - Felix Schäfer

Jiří Křivánek wrote:

> What I am suggesting involves the following change into your data model for time_entries entity.
>
> I do not know whether your SQL server supports triggers (the stored procedures automatically execured on the SQL server before/after insert/update/delete). If so then all the business logic may be implemented in these. I know that some pure SQL servers do not support these high end features (but these are still evolving)... so otherwise, it has to be implemented in Ruby…

First of all: rails only uses SQL servers as a data store and most rails programs will just leverage basic SQL functionality. Stored procedures especially are deemed to complexify things as they are not always portable, puts code in two different locations, and makes the overall program more difficult to test and to maintain. As far as redmine is concerned: I'm quite sure code relying on SPs won't be accepted.

This aside, let's talk about the proposed implementation, which unnecessarily duplicates information (yes, I know TimeEntry has extra attributes for weeks, months and year, but I think they are there to make reporting easier, so bare with me here). Current state is to save a date (spent_on) and a duration (hours), the only information you want more is to know *when* this duration was spent on that day. If spent_on were to also save the time at which the TimeEntry starts, you'd have that information, because based on the duration, you could then infer the timespan in which the activity took place. So switching spent_on to a datetime, and maybe rename it to started_at, would solve the data retention part of this problem.

The other topic I discussed with Eric on IRC yesterday was how the UI should look like. First thing he pointed out was that not everyone would want that level of precision for tracking time, as it could result in "started working on XYZ", "worked a little more on XYZ", "finished XYZ" type of time entries, which are summed up to a daily "spent time" entry in the current model. That means we have to support both variants, which in turn might clutter the UI. The best I could come up with for now was to have start_time and end_time fields in addition to the duration field, retain the "old" behavior when start_time and end_time are left blank, and have the fields tied together through some js (you only need 2 of the 3 fields, so filling the 3rd when the first 2 get input seems ok). Maybe we could even have a js listener on the end_time so that it defaults to the current time (or the quarter-hour before current time) when selected, so that it makes entering either simple or precise time entries easy enough.

Last but not least: Eric also suggested to get this feature done as a plugin, which in my mind would be the safest and quickest way for you to get that done in your redmine. You'd duplicate some core functionality, but it might even be possible to reuse some of the code in a plugin.

#11 - 2010-01-21 18:37 - Jiří Křivánek

A. Thank you for the Ruby clarification - it really helped.

B. I think that what you are describing is what I originally proposed, so we are on the same wave.

C. I can feel that you are suggesting me to do a Redmine plugin. Well, I would love to, but I am not quite sure I am experienced enough in

Ruby&Rails&Redmine yet.

I have no time to go through the entire R&R&R learning curve.

I will try to prepare the plugin but I will probably have tons of stupid questions of three ranks:
1. Ruby syntax.
2. Rails technology.
3. Redmine architecture.

First question is here: How to extend the data model with the two discussed new fields (start and aend time)? The header of the schema.rb file states that it was generated by some tool...

D. This is probably not a right place to perform these discussions (I silently expect that you are willing to give me a helpful hand)... Could you point me to better place?

#12 - 2010-01-22 10:51 - Jiří Křivánek

Can I use this to extend the database with my fields:

class TimeAddStartEndFields < ActiveRecord::Migration
def self.up
add_column :time_entries, :start_time, :time, :null => true
add_column :time_entries, :end_time, :time, :null => true
end

```
def self.down
    remove_column :time_entries, :start_time
    remove_column :time_entries, :end_time
  end
end
```

#13 - 2010-01-22 15:04 - Jiří Křivánek

Uffffff, I am still missing the relation between the SQL table name and the Ruby model class name...

#14 - 2010-01-24 15:22 - Jan Schenck

+1 i also would like to have this fields.
Also whats in my mind is a feature to track breaks and holidays. For a first start it would be maybe enought if the amount time field could be negative or if there would be a checkbox so that the time must not be counted in the reporting.

#15 - 2010-01-26 19:17 - Sebastian Cabot

+1. I would like to see this feature as well.
It's true I can calculate everything my self but it will be easier just to log the start time and the end time.

#16 - 2010-02-01 11:51 - Jiří Křivánek

For me, this is not only the simplification of the entering approach but also a protection against some sort of cheating (or may be in a less offense way - guessing the time efforts) the from the side of development teem members (especially those external/contractors - i.e. not company employees).

E.g.: If the external resource states he/she spent on particular problem some 25 hours then I can hardly check it. However, if he/she is due to record the exact date/time-from/time-to then it is better possible to argue that these records cannot be true...

#17 - 2010-02-01 11:55 - Jiří Křivánek

Well, I am not quite sure, I am able to do this as a plugin.

I was able to extend the data model and to create the report as a plugin.

However, I cannot update the existing built-in Redmine pages (for entering the effort from within the issue update form).

Any chance to have this feature in core Redmine?

The script for updating the database is as is:

```
class TimeAddStartEndFields < ActiveRecord::Migration
  def self.up
    add_column :time_entries, :time_from, :time, :null => true
    add_column :time_entries, :time_to, :time, :null => true
  end

  def self.down
    remove_column :time_entries, :time_from
```

```
    remove_column :time_entries, :time_to
  end
end
```

#18 - 2010-02-01 21:46 - Felix Schäfer

The database migration looks good. If you want to modify the views, you won't be able to do it in-place, but will have to copy it to your plugins directory, with the same path structure as in redmine, as the look-up for a corresponding view first looks over plugins, and then in the redmine core.

E.g.: copy $REDMINE_DIR/app/views/timelog/details.rhtml to $REDMINE_DIR/vendor/plugins/$YOUR_PLUGIN/app/views/timelog/details.rhtml and edit away, be sure to restart your app to see the effects of your changes.

#19 - 2010-02-03 02:56 - Jiří Křivánek

Frfgh... please help: How can I localize the day names?

date.strftime('%A')

produces the english day names. I need them to be localized...

Rails help says: Use

l Time.now, :format => '%A'

this does not work at all to me...

#20 - 2010-02-03 09:39 - Felix Schäfer

The localization function is l() in redmine, but I don't know if it'd work for dates. Maybe this would better be asked in the plugins forum.

#21 - 2010-02-03 11:07 - Jiří Křivánek

I have already resolved that problem - the method name is day_name(wday). I still do not understand how could it be, but it is somehow available in my views...

I am sorry for othering you with that.

#22 - 2010-03-19 03:32 - Oleg Volkov

I would be interested in this feature.

#23 - 2010-03-20 16:58 - Oleg Volkov

I offer a compatible version, using the existing fields. End time work in the field indicate created_on. Start time = created_on - hours * 3600.

#24 - 2010-03-22 21:56 - Mischa The Evil

- Category deleted (Time tracking)

- Status changed from New to Closed

- Resolution set to Duplicate

Closed as a duplicate of issue #440. I'll post an update on that one later giving a plugin heads-up...

#25 - 2010-03-28 10:23 - Jiří Křivánek

- File Screen_shot_2010-03-28_at_10.23.33.png added

I went through the #440 which is said to be duplicate of this issue. Well, it is not, at least not completely. However, I agree to close this issue as I prepared my own plugin, which does exactly what I needed (though it is not perfect and misses the report for my managers).

FYI, I attached the screenshot...

#26 - 2010-03-28 10:57 - Oleg Volkov

It is a pity that no one solution is not included in the plan for 0.9.4 or 1.0 version.
:(

Files

WorkMonitor.png	156 KB	2010-01-20	Jiří Křivánek
RecordingWorks.png	137 KB	2010-01-20	Jiří Křivánek
Screen_shot_2010-03-28_at_10.23.33.png	187 KB	2010-03-28	Jiří Křivánek