Redmine - Defect #21074

When changing the tracker of an existing issue, new custom fields are not initialized with their default value

2015-10-26 18:37 - Holger Just

Status:	Closed	**Start date:**	
Priority:	Normal	**Due date:**	
Assignee:	Jean-Philippe Lang	**% Done:**	0%
Category:	Custom fields	**Estimated time:**	0.00 hour
Target version:	3.2.0		
Resolution:	Fixed	**Affected version:**	3.1.1

Description

Given two configured trackers "Support" and "Task" with different enabled custom fields. There is a required boolean custom field named "Bool" which is only enabled on the Task tracker. The custom field defaults to "No".

Given an existing saved issue on the Support tracker, we navigate to the issue update form and select the Task tracker which has the additional custom field defined. When rendering the form, the value of the CustomValue for the "Bool" field is not properly initialized with the default value. This results in the form rendering with no explicitly selected value. The browser thus selects the first value of the list, i.e. "Yes" by default. On saving, the custom field is set to Yes because the browser sends this value, despite the default of the field being set to "No". This can be very surprising as the user typically has not updated the field manually and has the possibility to break implicit workflows.

The cause of this issue is that in CustomValue#initialize, the value is only set as the default if the customized object (the issue in this case) is still new and unsaved. Because of this logic, selecting a tracker works when creating a new issue; in this case, the custom values are properly initialized with the default value. In our case however, the issue is not new anymore but would still need the default value initialized.

Unfortunately, a fix it probably isn't straightforward to implement as it seems nil is a value value today to denote "Not set" for fields which do not require a value. Thus, a possible fix could be to distinguish between an empty string and nil here and set the default value for nil values. This change would have to be checked against the API.

At the very least, the CustomField inititializer could check if the field is required (and thus doesn't allow nil / empty string as a valid value anyway) and force the nil value to the field's default value here. At least this should be changed so that required fields have their default value set.

I'm asking for thoughts here first. If there is a consensus about how to proceed, I can provide a patch, courtesy of Planio. This issue affects every version of Redmine since 0.7 when defaults for custom fields were introduced.

Related issues:

Related to Redmine - Defect #25726: Issue details page shows default values f...	**Closed**

Associated revisions

Revision 14768 - 2015-10-30 13:59 - Jean-Philippe Lang

Adds tests for default custom field value (#21074).

Revision 14769 - 2015-10-30 16:55 - Jean-Philippe Lang

Test failure (#21074).

Revision 14773 - 2015-10-31 10:17 - Jean-Philippe Lang

New custom fields of existing issues are not initialized with their default value (#21074).

Revision 14775 - 2015-10-31 11:52 - Jean-Philippe Lang

Test failure (#21074).

History

#1 - 2015-10-29 14:28 - Jean-Philippe Lang

Couldn't we just do:

```
Index: app/models/custom_value.rb
===================================================================
--- app/models/custom_value.rb    (revision 14752)
+++ app/models/custom_value.rb    (working copy)
@@ -22,7 +22,7 @@

   def initialize(attributes=nil, *args)
     super
-    if new_record? && custom_field && (customized_type.blank? || (customized && customized.new_record?))
+    if new_record? && custom_field
       self.value ||= custom_field.default_value
     end
   end
```

That would fix this issue and set the default value as well for custom fields that are added after an issue is created.
Is there any problem with that?

#2 - 2015-10-30 10:47 - Holger Just

This was my first guess too. I'm slightly concerned however, that this could change the custom fields on save too. Right now, nil and the empty string seems both to be valid values for the "blank" value of non-required custom fields.

While your proposed patch solves the immediate problem, it might have undesired side-effects when people send nil as a custom field value, e.g. via the API or by omitting the field value in a request.

You could argue that this is the desired behavior, i.e. sending nil results in the default value being set. But it would change existing behavior here and might break certain API use-cases where users send nil instead of an empty string and expect the blank value to be set. After having thought about this for a bit longer, it is however probably worth the change, as it makes the final behavior more obvious and consistent.

Thus, finally, +1 from me for your proposed patch.

#3 - 2015-10-31 10:18 - Jean-Philippe Lang

- Status changed from New to Closed

- Assignee set to Jean-Philippe Lang

- Target version set to 3.2.0

- Resolution set to Fixed

Thanks.

#4 - 2017-04-25 19:16 - Jens Krämer

- File 0001-do-not-display-default-values-that-aren-t-actually-s.patch added

Actually I think there is a problem with this.

Steps to reproduce:

- Create a new issue.
- Create a new boolean issue custom field with a default of yes or no and *use as filter* set.
- look at the previously created issue in issues list (have the custom field displayed) and detail view
- note the different values being shown for the custom field - issue details page shows the configured default, while the issue list shows the field as unset (empty).
- When filtering the issue list for the field's value (or simply looking at the database) it turns out that the field is indeed unset (neither filtering for the field being 'yes' or 'no' turns up the issue). So clearly the details view is lying about the field's true value by replacing *unset* with the configured default.

I think this also applies to other field formats, not just booleans.

Imho the default value for unset custom fields (where no corresponding custom_value record exists) should only be preset in the edit form, but not when merely showing the issue details. I came up with a patch that checks if there is actually any custom value present in the database for the given field before calling show_value, but to be honest I'm not very happy with that.

#5 - 2017-04-26 04:06 - Mischa The Evil

Jens, your comment made me re-read this whole issue and it reminded me of an issue I encountered and reported a long time ago (#9849). As of today I don't know whether or not that issue is still present and still reproducible (and I'm currently unable to test this vigorously), but I can't help thinking it might be related to this issue in a sense. So to be sure, I post this comment to bring that issue to the attention within the context of this current issue.
However, since this issue has been closed targeted against 3.2.0, it might be a better idea to create a new issue altogether...

#6 - 2017-04-28 17:14 - Jens Krämer

posted as new issue #25726

#7 - 2017-06-02 12:14 - Mischa The Evil

- Related to Defect #25726: Issue details page shows default values for custom fields that aren't actually set added

#8 - 2017-06-09 06:02 - Mischa The Evil

holger mareck: as being the OP of this issue, do you maybe have have some ideas on #25726? What do you think about the solution of handling custom field default values as proposed by Jens?

Files

0001-do-not-display-default-values-that-aren-t-actually-s.patch	1.64 KB	2017-04-25	Jens Krämer