Redmine - Defect #14621

AJAX call on the issue form resets data entered during the request

2013-08-06 18:21 - Robert Chady

Status:	Closed	**Start date:**	
Priority:	Normal	**Due date:**	
Assignee:	Jean-Philippe Lang	**% Done:**	0%
Category:	UI	**Estimated time:**	0.00 hour
Target version:	2.4.0		
Resolution:	Fixed	**Affected version:**	2.3.2

Description

Running in to an issue where the ajax call being issued when the status dropdown box is updated is causing data to be lost due to the delay.

To reproduce, make a status change in the status dropdown and then pretend you don't see the faint 'Loading' indicator and update the assigned_to field. When the ajax call completes for the change to status_id it will lose the assigned_to change. The Loading indicator is faint enough that it is very easy to miss it and the delay from that call is significant enough that it easily can trip up people making a lot of changes in the system.

Older Redmine did not have an ajax call on field update so this was never an issue before. Is there a way this can be changed so it does not cause the delay for that ajax call, or have the ajax call removed entirely? (I did not see what its use was with my cursory examination of it)

Related issues:

Related to Redmine - Defect #13587: unpredictable tab position after ajax cal...	**Confirmed**
Related to Redmine - Defect #15125: Task properties are reverted when doing a...	**Closed**

Associated revisions

Revision 12166 - 2013-09-29 12:29 - Jean-Philippe Lang

AJAX call on the issue form resets data entered during the call (#14621).

Revision 12169 - 2013-09-29 17:27 - Jean-Philippe Lang

Removed unused code (#14621).

History

#1 - 2013-08-07 13:34 - Etienne Massip

Robert Chady wrote:

> Older Redmine did not have an ajax call on field update so this was never an issue before. Is there a way this can be changed so it does not cause the delay for that ajax call, or have the ajax call removed entirely? (I did not see what its use was with my cursory examination of it)

The delay is mostly dependent on your setup performance and the ajax call is necessary for dynamic workflow handling.

What can be done is blocking the ui (or better but probably a lot more difficult to implement, only the part of the ui which will be updated) until ajax call is over.

#2 - 2013-08-07 13:37 - Etienne Massip

- Related to Defect #13587: unpredictable tab position after ajax call loading workflow added

#3 - 2013-09-14 00:03 - Ben Hockey

the issue is happening because the request to update_form.js?id=<ticketnumber> is returning a response like this:

```
$('#all_attributes').html(' ... ');
```

where the ' ... ' is the complete contents of the #all_attributes form. this means that the complete contents of that form (including input elements) that exist before this response is received will be lost and replaced with new contents. this means that any changes made during the time it takes for that request will be lost when the response is received.

if there is no way to merge the response with the changes that happened while waiting on that response then the loading indicator should block that

whole form to prevent user interaction that is going to be lost.

#4 - 2013-09-14 15:53 - Jean-Philippe Lang

- Category changed from Issues workflow to UI

- Assignee set to Jean-Philippe Lang

- Target version set to 2.4.0

Etienne Massip wrote:

> What can be done is blocking the ui (or better but probably a lot more difficult to implement, only the part of the ui which will be updated) until ajax call is over.

Here is another approach with a small patch, fields are not disabled and values that were entered during the ajax call are preserved.

```
Index: app/views/issues/update_form.js.erb
===================================================================
--- app/views/issues/update_form.js.erb    (revision 12132)
+++ app/views/issues/update_form.js.erb    (working copy)
@@ -1,4 +1,10 @@
-$('#all_attributes').html('<%= escape_javascript(render :partial => 'form') %>');
+var replacement = $('<%= escape_javascript(render :partial => 'form') %>');
+$('#all_attributes input, #all_attributes textarea, #all_attributes select').each(function(){
+  var object_id = $(this).attr('id');
+  if (object_id) replacement.find('#'+object_id).val($(this).val());
+});
+$('#all_attributes').empty();
+$('#all_attributes').prepend(replacement);

 <% if User.current.allowed_to?(:log_time, @issue.project) %>
   $('#log_time').show();
```

Any feedback is welcome before I check it in.

#5 - 2013-09-17 20:20 - eyal R

I think blocking is better.

In the way it works now , it allows plugin developers more customization.
For example , I can create a plugin that changes assignee to author when changing to fixed state.

With the suggested js change , the assignee will always change back to the value before the ajax.
I think the html from the server after status/tracker change , should be unchanged.

Thanks.

#6 - 2013-09-23 19:10 - Jean-Philippe Lang

I see your point but blocking fields would affect user experience for the vast majority of people who don't have this concern.

#7 - 2013-09-23 19:16 - Jean-Philippe Lang

Another option would be to detect the fields that were manually changed by the user during the ajax call and only restore these fields.

#8 - 2013-09-24 16:07 - Etienne Massip

Jean-Philippe Lang wrote:

> I see your point but blocking fields would affect user experience for the vast majority of people who don't have this concern.

If they don't have this concern this is probably that the call is fast enough to not let them update any other field before it ends, so having some ui blocking could go unnoticed too?

I don't know what's best, I have in mind 2 features requests that are related:

- a form which fields are updated using in-place editor
- more advanced workflow with more dependent fields updated on more fields updates

Just FTR, there's some discussion here about the way Atlassian implemented (or should have implemented) inline editing.

#9 - 2013-09-25 19:21 - Jean-Philippe Lang

> If they don't have this concern this is probably that the call is fast enough to not let them update any other field before it ends, so having some ui blocking could go unnoticed too?

No, I meant that the vast majority of users don't have any plugin that automatically changes the field values. For having tried it, the blocking gets actually noticed even if the ajax call is fast and I personnaly dislike it.

#10 - 2013-09-26 19:39 - Rowan Beentje

I prefer Jean-Philippe's current patch over a blocking solution, but I think what he suggested in comment 7 would be the best of both worlds. In other words, if the form field states are stored on ajax submission, then on ajax return the current values are compared and any changes re-applied to the new form, that would allow both:

- Users without plugins can apply changes to menus as the ajax call proceeds
- Plugins can still change state

And if a user decision would override the plugin state change the user decision would be maintained (which is correct IMHO).

#11 - 2013-09-26 21:05 - eyal R

I agree with Rowan , best solution would be comment #7

#12 - 2013-09-29 12:30 - Jean-Philippe Lang

- Subject changed from Ajax call on status_id update can lose data to AJAX call on the issue form resets data entered during the request

- Status changed from New to Closed

- Resolution set to Fixed

Solution proposed in note-7 committed in r12166, thanks for the feedback.

#13 - 2013-09-29 15:24 - Rowan Beentje

Amazing, thanks Jean-Philippe :)

#14 - 2013-10-10 23:44 - Miles Matthias

This update does make an improvement, however no matter what you do to the values of the elements, if you replace the elements on the page, you have the potential of ripping the carpet out from under the user interaction-wise. For example, you can still do:

1. Change the status
[ajax begins/loading is shown]
2. Open drop down for assignee
[ajax ends/loading is removed and elements are replaced]
3. Attempt to make a selection on your open drop down

And your selection won't be recorded because the element that you're making a selection on is no longer present.

It wouldn't look great at scale, but instead of replacing the elements you could hide them, add a click handler to them, and show the new elements. The old element click handler would then update the new element with your selection or alert the user that that option is no longer available.

#15 - 2013-10-14 10:39 - Toshi MARUYAMA

- Related to Defect #15125: Task properties are reverted when doing a task update from firefox. added