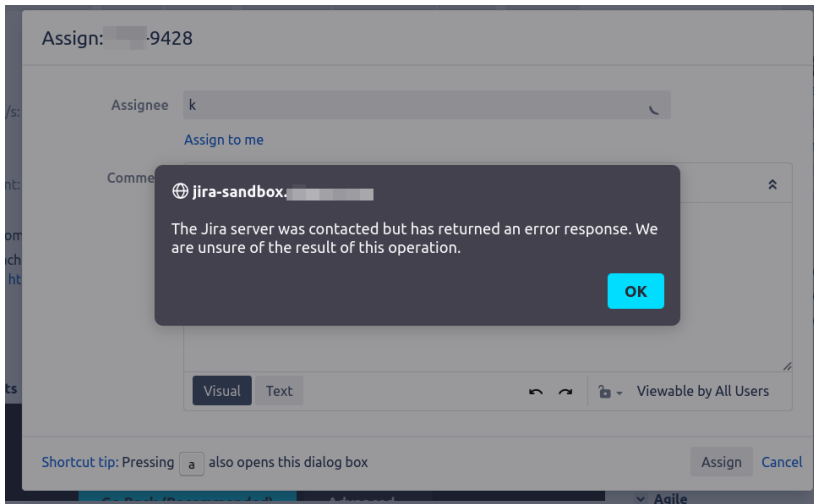


Jira 8.20.x autocomplete breaking with NullPointerException

What a barrel of laughs this Jira 8.20.1 upgrade is proving to be! After yesterday's [Jira 8.20.x gotcha: lost notifications](#), today's problem is that autocomplete on the 'Assignee' field is breaking:



But only for some users on some projects, which is why the problem wasn't picked up in sandbox testing.

Lovely source code

There is at least a stacktrace in `atlassian-jira.log`:

```
Caused by: java.lang.NullPointerException
    at java.base/java.util.Comparator.lambda$comparing$77a9974f$1(Comparator.java:469)
    at java.base/java.util.Comparator.lambda$thenComparing$36697e65$1(Comparator.java:216)
    at java.base/java.util.TreeMap.put(TreeMap.java:550)
    at java.base/java.util.TreeSet.add(TreeSet.java:255)
    at java.base/java.util.AbstractCollection.addAll(AbstractCollection.java:352)
    at com.atlassian.jira.permission.DefaultIssueUserSearchManager.findTopUsersWithPermissionInIssue
(DefaultIssueUserSearchManager.java:253)
    at com.atlassian.jira.permission.DefaultIssueUserSearchManager.findTopUsersInternal
(DefaultIssueUserSearchManager.java:168)
    at com.atlassian.jira.permission.DefaultIssueUserSearchManager.findTopAssignableUsers
(DefaultIssueUserSearchManager.java:115)
    at com.atlassian.jira.bc.user.search.DefaultUserPickerSearchService.findTopAssignableUsers
(DefaultUserPickerSearchService.java:326)
    at java.base/jdk.internal.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
```

This is the point where I'm glad I work with Atlassian products, because although they are sometimes buggy and broken, **the source code is available** to licensees. So we can debug this problem.

The code in question is adding a bunch of user objects to a collection object called `result`:

```
252 if (permissionSchemeLogic.isUserTypeProjectRole()) {
253     result.addAll(schemePermissionsDAO.findTopUsersFromTypeProjectRoleUsers(userSearchName, schemeId,
254     sqlCount++);
255 }
```

The `result` field is defined a bit higher, as a collection object of type `TreeSet`, and the constructor is passed a `Comparator` used to determine the sort order. Here we see the `Comparator` is using `getLowerDisplayName()` and `getLowerUserName()`:

```
242 final TreeSet<UserDTO> result = new TreeSet<>(Comparator
243     .comparing(UserDTO::getLowerDisplayName)
244     .thenComparing(UserDTO::getLowerUserName));
```

So we should be on the lookout of null values of those fields. Sure enough, if we hook up a debugger and inspect the UserDTOs starting with letter 'k', we have one object with a null `lowerDisplayName` field:

Expression: `projectRoleUsers(userSearchName, schemeId, project.getId(), project.getName())`

Use Ctrl+Shift+Enter to add to Watches

Result:

- > {f id = {Long@77221} 1000798
- > {f directoryId = {Long@77187} 1
- > {f userName = "Karen. [REDACTED]"
- > {f lowerUserName = "karen [REDACTED]"
- > {f active = {Integer@77171} 1
- > {f createDate = {Timestamp@77224} "2013-02-27 00:53:45.0"
- > {f updatedDate = {Timestamp@77225} "2021-11-01 02:21:57.942"
- > {f firstName = null
- > {f lowerFirstName = null
- > {f lastName = null
- > {f lowerLastName = null
- > {f displayName = "Karen [REDACTED]"
- > {f lowerDisplayName = null
- > {f emailAddress = "Karen. [REDACTED]"
- > {f lowerEmailAddress = "karen. [REDACTED]"
- > {f credential = "{PKCS5S2}7[REDACTED]"
- > {f deletedExternally = null

Buttons: Evaluate, Close

The class involved, [DefaultIssueUserSearchManager](#), appears to be new in Jira 8.19.x, and clearly wasn't tested with much real-world data.

Why is the user data missing fields?

I don't know why those fields are null. It is only the case in a small subset of user accounts (80 of 3700 accounts), all created before May 2017.

If I find the user's account in the Jira administration section, click 'Edit Details' and simply save, then all the null fields suddenly get values:

Before	After
--------	-------

```
jira-sandbox=> select * from cwd_user where lower_display_name is null and user_name ~ 'Karen.';
```

[RECORD 1]	
id	1000798
directory_id	1
user_name	Karen.
lower_user_name	karen.
active	1
created_date	2013-02-27 00:53:45-08
updated_date	2021-11-01 02:21:57.942-07
first_name	NUL
lower_first_name	NUL
last_name	NUL
lower_last_name	NUL
display_name	Karen.
lower_display_name	NUL
email_address	Karen.
lower_email_address	karen.
credential	{PKCS5S2}
deleted_externally	NUL
external_id	NUL

```
jira-sandbox=> select * from cwd_us
```

[RECORD 1]	
id	1000798
directory_id	1
user_name	Karen.
lower_user_name	karen.
active	1
created_date	2013-02-27
updated_date	2021-11-01
first_name	
lower_first_name	
last_name	Karen.
lower_last_name	karen.
display_name	Karen.
lower_display_name	karen.
email_address	Karen.
lower_email_address	karen.
credential	{PKCS5S2}
deleted_externally	NUL
external_id	NUL

Fixing with SQL

So let's fix up the null fields. Here is some Postgres-flavoured SQL that makes use of the fact that `display_name` is generally of the form "Firstname Lastname", with fallback to just blatting `display_name` into `first_name` and `last_name`.

```
begin;
update cwd_user set lower_display_name=lower(display_name) where lower_display_name is null;
update cwd_user set first_name=split_part(display_name, ' ', 1) where first_name is null and display_name~'^[ ]'
+ '[^ ]+$';
update cwd_user set last_name=split_part(display_name, ' ', 2) where last_name is null and display_name~'^[ ]'
+ '[^ ]+$';
update cwd_user set first_name=display_name where first_name is null and display_name!~'^[ ]+ [^ ]+$' ;
update cwd_user set last_name=display_name where last_name is null and display_name!~'^[ ]+ [^ ]+$' ;
update cwd_user set lower_first_name=lower(first_name) where lower_first_name is null;
update cwd_user set lower_last_name=lower(last_name) where lower_last_name is null;
commit;
```

The nice thing about `cwd_` tables is they aren't cached (AFAIK), so you don't need to restart. Commit the changes to the database, and your autocomplete should start working.