EDIT - bug #5250

feature request # 4931 (New): [Master] Fix Performance Issues

Derivate search is slow

09/15/2015 03:20 PM - Andreas Müller

Status:	Resolved	**Start date:**	
Priority:	Highest	**Due date:**	
Assignee:	Andreas Müller	**% Done:**	0%
Category:	taxeditor	**Estimated time:**	0:00 hour
Target version:	Release 3.12		
Severity:	major	**Found in Version:**	

Description

In the latest caryophylalles 2015 workshop database we have ~ 25 SpecimenOrObservations. These take >5 sec to load by a "*" search.

History

#1 - 09/15/2015 04:00 PM - Andreas Müller

- Priority changed from New to Highest

It takes <1 sec in the standalone version, so it is mostly a remoting issue

#2 - 09/15/2015 06:46 PM - Cherian Mathew

- Status changed from New to Resolved

- Assignee changed from c.mathew - to Andreas Müller

r25686 adds the necessary property paths, which has considerably improved the speed.

#3 - 09/21/2015 03:57 PM - Andreas Müller

- Assignee changed from Andreas Müller to c.mathew -

Looks like this is not fully solved yet. First loading still takes long, second loading for same search is fast then.

#4 - 11/24/2015 10:56 AM - Cherian Mathew

- Status changed from Resolved to In Progress

#5 - 11/25/2015 06:42 PM - Cherian Mathew

The main reason for the performance is the remote loading of each derivate (using its uuid) even after the entire list has already been loaded when search is clicked.

The reloading of the derivates on search should not be required since the derivate composite which contains the search button is part of the view implying that there will be no session conflict.

r26667 refactors the code to use a newly created service method to bulk load entities from a list of uuids, avoids reloading of the derivates for the search use-case and adds relevant property paths in the search manager.

#6 - 11/25/2015 06:46 PM - Cherian Mathew

Another main cause of performance issues is the

```
listTypeDesignations
```

call in the DerivateLabelProvider class.

This service method is invoked for each derivate in the view, which will be time consuming for long lists.

Since we already have the list of derivates before the first call to the label provider, it would be more performant to retrive all type designations in a single service call, cache the result and use it in the subsequent calls in the label provider.

#7 - 11/25/2015 06:48 PM - Cherian Mathew

- Status changed from In Progress to Resolved

- Assignee changed from c.mathew - to p.plitzner -

Assigning for review of the performance and also to check if the other use-cases (taxon selection, etc) still work fine after the refactoring.

#8 - 11/30/2015 11:55 AM - Patrick Plitzner

- Assignee changed from p.plitzner - to c.mathew -

Replying to c.mathew:

> Another main cause of performance issues is the
>
> listTypeDesignations
>
> call in the DerivateLabelProvider class.
>
> This service method is invoked for each derivate in the view, which will be time consuming for long lists.
>
> Since we already have the list of derivates before the first call to the label provider, it would be more performant to retrive all type designations in a single service call, cache the result and use it in the subsequent calls in the label provider.

The type designations are only retrieved once for every derivative and cached afterwards so there should be no problem.

Concerning the search performance: Searching with the search bar runs with an OK speed but when linked to taxon selection and a taxon with a lot of specimens is selected it still need a lot of time.

We should do the loading in an asynchronous job. Added this info to ticket #5402

I don't know if the performance can be increased even more. If not we can close this ticket.

#9 - 11/30/2015 02:12 PM - Cherian Mathew

Replying to p.plitzner:

> Replying to c.mathew:
>
> The type designations are only retrieved once for every derivative and cached afterwards so there should be no problem.

Retrieving the type designations (and determination events) for each derivate in the label provider is not scalable from a performance point of view. The performance is then tied to the number of derivates in the list since every derivate will require its own service call (two actually).

In addition to this, the caching mechanism for type designations (as is currently) does not work. Setting a breakpoint on the put call of the cache map reveals, that it is in fact never called. This is one of the reason why the updates in the details view fields which contribute to the derivate label are slow.

Another reason for the derivate view fields -> label update being slow is that any update of relevant fields in the details view updates the entire derivate view instead of just the single label in question. This should be ideally moved to a design where listeners are attached to the derivate entity which update specific tree nodes.

#10 - 11/30/2015 02:18 PM - Cherian Mathew

Replying to p.plitzner:

> Replying to c.mathew:
>
> Concerning the search performance: Searching with the search bar runs with an OK speed but when linked to taxon selection and a taxon with a lot of specimens is selected it still need a lot of time.
>
> We should do the loading in an asynchronous job. Added this info to ticket #5402
>
> I don't know if the performance can be increased even more. If not we can close this ticket.

The idea of a asynchronous job does make it more user friendly but does not solve the underlying remote loading performance issue. If there are specific use cases (like the one mentioned above), they should be checked to see which entities are loaded and these should be added to the property paths list as already done for this ticket.

In theory, with the right property paths setting the number of remote calls can be considerably reduced.

#11 - 12/02/2015 11:20 AM - Andreas Müller

- Assignee changed from c.mathew - to p.plitzner -

- Target version changed from Remoting 5.0 to Unassigned CDM tickets

#12 - 12/02/2015 11:40 AM - Patrick Plitzner

- Assignee changed from p.plitzner - to Andreas Müller

To improve the performance the following has been done:

- I implemented the new formatter framework that provides string representations for objects. This replaces the many if-statements in the DerivateLabelProvider.
- I also worked on the type designations which are now loaded in bulk when the label provider is initialized.
- Only the labels of elements that were changed are updated that.

r26714 - r26732

#13 - 02/16/2016 05:18 PM - Andreas Müller

- Target version changed from Unassigned CDM tickets to Release 3.12

#14 - 12/05/2020 01:14 PM - Andreas Müller

- Description updated

- Private changed from Yes to No