EDIT - feature request #9152

PreferencesService caches CdmPreferences

07/15/2020 02:32 PM - Andreas Kohlbecker

Status:	Closed	**Start date:**	
Priority:	Highest	**Due date:**	
Assignee:	Andreas Kohlbecker	**% Done:**	100%
Category:	cdmlib	**Estimated time:**	0:00 hour
Target version:	Release 5.17		
Severity:	normal		

Description

implement as static private Map

once the methods public void set(CdmPreference preference) or public void remove(CdmPreference preference) are used the cache should be updated or deleted respectively

Related issues:

Related to EDIT - feature request #9146: filter image metadata by include and...	**Closed**

Associated revisions

Revision 922b1ce9 - 07/20/2020 11:55 AM - Andreas Kohlbecker

ref #9152 implementing service level cache for cdm preferences with locking mechanism

Revision 8b3c6dab - 08/13/2020 03:18 PM - Andreas Müller

ref #9152 set cacheIsComplete to true after loading preference values

Revision 5301f5b5 - 08/13/2020 03:22 PM - Andreas Müller

ref #9152 add separator to computed cache key

Revision db9e9aab - 08/18/2020 09:29 AM - Andreas Kohlbecker

ref #9152 disabling taxon node preference caching unused methods deprecated and left by now for potential use later on

History

#1 - 07/15/2020 02:33 PM - Andreas Kohlbecker

- Related to feature request #9146: filter image metadata by include and exclude lists of key words added

#2 - 07/20/2020 11:30 AM - Andreas Kohlbecker

- % Done changed from 0 to 10

Caching CdmPreferences at the service class level is not straight forward and picked with a couple of pitfalls:

1. there are service methods for which caching is not feasible e.g.: public List<CdmPreference> list(IPreferencePredicate<?> predicate)
2. adding a cache to the service class breaks the general principle of spring beans which always should be stateless.

Solutions to the concurrency problem (2.) :

1. use a ThreadLocal variable to store the cache map in it ==> increases memory consumption
2. synchronize all methods which access and modify the cache ==> may cause blocked threads even in case of parallel read operations
3. set a lock on the cache while it is being modified ==> may cause blocked threads only during cache modification

#3 - 07/20/2020 11:57 AM - Andreas Kohlbecker

- Assignee changed from Andreas Kohlbecker to Andreas Müller

- % Done changed from 10 to 50

I implemented the service level cache for cdm preferences with locking mechanism.

Please review carefully.

#4 - 07/20/2020 11:57 AM - Andreas Kohlbecker

- Status changed from New to Resolved

#5 - 08/13/2020 03:05 PM - Andreas Müller

This generally seems to work. One issue that is kind of unclear to me:

- is there a reason why you use the treeindex as key for TaxonNode in e.g.

```
private String cacheKey(TaxonNode taxonNode, String predicate) {
    return taxonNode.treeIndex() + predicate;
}
```

The treeindex is not necessarily as stable as the uuid. But I see that this guarantees that if the position of the node in the tree changes the cached value is not valid anymore which might be good. However, it is generally not clear if caching of taxon node related preferences works. The documentation of find(TaxonNode taxonNode, String predicate) says "Returns the best matching preference that matches the given predicate and the taxon node filter". That means if in the meanwhile a better matching preference was stored this preference is not correct anymore.
A solution could be to not cache this case like we do for "list(IPreferencePredicate<?> predicate)". Another solution could be to recompute the full cache or all taxon node related cache values once a taxon node related preference is stored.

Which solution to take also depends on how often the cache is currently used for these cases (I guess it is not often used and therefore we can choose solution 1)

#6 - 08/13/2020 03:18 PM - Andreas Müller

- Status changed from Resolved to Feedback

- Assignee changed from Andreas Müller to Andreas Kohlbecker

There was a critical thing missing as the cacheIsComplete was never set to true (and therefore the cache was not really in use). I added an according line of code 8b3c6dabe89e

#7 - 08/13/2020 03:22 PM - Andreas Müller

I would suggest to add a separator in the cacheKey(key) method to clearly distinguish the subject and the predicate. Otherwise in rare cases it could happen that the computed key is maybe not unique. I did this with 5301f5b54d8 . Please revert if you think this is not correct.

#8 - 08/18/2020 09:24 AM - Andreas Kohlbecker

Andreas Müller wrote:

> This generally seems to work. One issue that is kind of unclear to me:
>
> - is there a reason why you use the treeindex as key for TaxonNode in e.g.
>
> ```
> private String cacheKey(TaxonNode taxonNode, String predicate) {
>     return taxonNode.treeIndex() + predicate;
> }
> ```
>
> The treeindex is not necessarily as stable as the uuid. But I see that this guarantees that if the position of the node in the tree changes the cached value is not valid anymore which might be good. However, it is generally not clear if caching of taxon node related preferences works. The documentation of find(TaxonNode taxonNode, String predicate) says "Returns the best matching preference that matches the given predicate and the taxon node filter". That means if in the meanwhile a better matching preference was stored this preference is not correct anymore.

I implemented it in this way since the PreferenceDaoImpl.find(TaxonNode taxonNode, String predicate) also uses the taxon node index as key. But you are right there is a risk of missing recently added better matching preferences. BTW: It would be good to extend the documentation of this find method so that the strategy of finding matching pref becomes clear without reading the whole code. THe doc should also answer the question why the last split is omitted. Is this by purpose?

> A solution could be to not cache this case like we do for "list(IPreferencePredicate<?> predicate)". Another solution could be to recompute the full cache or all taxon node related cache values once a taxon node related preference is stored.
>
> Which solution to take also depends on how often the cache is currently used for these cases (I guess it is not often used and therefore we can choose solution 1)

I would also choose this solution for the same reason.

#9 - 08/18/2020 09:34 AM - Andreas Kohlbecker

Andreas Kohlbecker wrote:

> Andreas Müller wrote:
>
> ...
>
> > A solution could be to not cache this case like we do for "list(IPreferencePredicate<?> predicate)". Another solution could be to recompute the full cache or all taxon node related cache values once a taxon node related preference is stored.
> >
> > Which solution to take also depends on how often the cache is currently used for these cases (I guess it is not often used and therefore we can choose solution 1)
>
> I would also choose this solution for the same reason.

Done, see db9e9aab

#10 - 08/18/2020 09:34 AM - Andreas Kohlbecker

- Assignee changed from Andreas Kohlbecker to Andreas Müller

Andreas Müller wrote:

> I would suggest to add a separator in the cacheKey(key) method to clearly distinguish the subject and the predicate. Otherwise in rare cases it could happen that the computed key is maybe not unique. I did this with 5301f5b54d8 . Please revert if you think this is not correct.

Good point, even if I think that this problem will occur extremely seldomly.

You are using @ as separator. I is guaranteed that this character can't be used in predicate or subject?

#11 - 08/19/2020 02:02 PM - Andreas Müller

- Assignee changed from Andreas Müller to Andreas Kohlbecker

- % Done changed from 50 to 90

Andreas Kohlbecker wrote:

> Andreas Müller wrote:
>
> > I would suggest to add a separator in the cacheKey(key) method to clearly distinguish the subject and the predicate. Otherwise in rare cases it could happen that the computed key is maybe not unique. I did this with 5301f5b54d8 . Please revert if you think this is not correct.
>
> Good point, even if I think that this problem will occur extremely seldomly.
>
> You are using @ as separator. I is guaranteed that this character can't be used in predicate or subject?

I am not sure if it is guaranteed but from the subjects and predicates we use I do not remember that there is a @ at all. And even it is theoretically possible in combination with the fact that already without separator the non unique key will occurr extremly seldom so no the probability is still much smaller.

But feal free to use an even better separator which is used even more seldom.

Generally, we can close this ticket.

#12 - 08/19/2020 02:10 PM - Andreas Müller

Andreas Kohlbecker wrote:

> Andreas Müller wrote:
>
> > The treeindex is not necessarily as stable as the uuid. But I see that this guarantees that if the position of the node in the tree changes the cached value is not valid anymore which might be good. However, it is generally not clear if caching of taxon node related preferences works. The documentation of find(TaxonNode taxonNode, String predicate) says "Returns the best matching preference that matches the given predicate and the taxon node filter". That means if in the meanwhile a better matching preference was stored this preference is not correct anymore.
>
> I implemented it in this way since the PreferenceDaoImpl.find(TaxonNode taxonNode, String predicate) also uses the taxon node index as key. But you are right there is a risk of missing recently added better matching preferences. BTW: It would be good to extend the documentation of this find method so that the strategy of finding matching pref becomes clear without reading the whole code. THe doc should also answer the question why the last split is omitted. Is this by purpose?

Hmm, I can't really see what you mean by "last split omitted". Can you explain a bit more?

#13 - 08/19/2020 02:15 PM - Andreas Müller

> I implemented it in this way since the PreferenceDaoImpl.find(TaxonNode taxonNode, String predicate) also uses the taxon node index as key. But you are right there is a risk of missing recently added better matching preferences. BTW: It would be good to extend the documentation of this find method so that the strategy of finding matching pref becomes clear without reading the whole code.

I think I haven't done this yet because the final strategy is not yet fully clear. The current implementation is still a bit preliminary as you can see from the FIXME.

I added "The rules how to do this best is still under discussion (see FIXME in implementing code)" to the doc (not committed yet)

#14 - 08/19/2020 05:24 PM - Andreas Müller

- Status changed from Feedback to Closed

- Assignee changed from Andreas Kohlbecker to Andreas Müller

- % Done changed from 90 to 100

#15 - 08/19/2020 08:41 PM - Andreas Müller

- Target version changed from Release 5.18 to Release 5.17

#16 - 08/19/2020 11:02 PM - Andreas Kohlbecker

- Assignee changed from Andreas Müller to Andreas Kohlbecker