Edit - bug #7331

AdvancedBeanInitializer fails to initialize properties of preinitialized beans in the graph in very special situations

03/27/2018 01:50 PM - Andreas Kohlbecker

Status:	New	**Start date:**	03/27/2018
Priority:	New	**Due date:**	
Assignee:	Andreas Müller	**% Done:**	10%
Category:	cdmlib	**Estimated time:**	0.00 hour
Target version:	Unassigned CDM tickets		
Severity:	normal	**Found in Version:**	Release 5.0

Description

In the RegistrationWorkingSetService.loadWorkingSetByReferenceID(Integer referenceID) a RegistrationWorkingSet is loaded with the following code:

```
List<String> REGISTRATION_INIT_STRATEGY = Arrays.asList(new String []{
            "blockedBy",
            // typeDesignation
            "typeDesignations.typeStatus",
            "typeDesignations.typifiedNames.typeDesignations", // important !!
            "typeDesignations.typeSpecimen",
            "typeDesignations.typeName.$",
            "typeDesignations.citation",
            "typeDesignations.citation.authorship.$",
            // name
            "name.$",
            "name.nomenclaturalReference.authorship.$",
            "name.nomenclaturalReference.inReference",
            "name.rank",
            "name.homotypicalGroup.typifiedNames",
            "name.status.type",
            "name.typeDesignations", // important !!"
            // institution
            "institution",
            }
    );

Reference reference = repo.getReferenceService().find(referenceID);
Pager<Registration> pager = repo.getRegistrationService().page(Optional.of(reference), null, null
, null, REGISTRATION_INIT_STRATEGY);
return new RegistrationWorkingSet(makeDTOs(pager.getRecords()));
```

This usually works, but with a specific data set the pager contains Registrations where authorship properties of the nomenclaturalReferences are not initializes even if the AdvancedBeanInitializer apparently is doing the initialization properly (also confirmed by debugging):

```
[eu.et.cd.pe.da.in.AdvancedBeanInitializer] -  processing /name.nomenclaturalReference
[eu.et.cd.pe.da.in.AdvancedBeanInitializer] -  invoke initialization on /name.nomenclaturalReferen
ce beans of class TaxonName ...
[eu.et.cd.pe.da.in.AdvancedBeanInitializer] - bulk load /name.nomenclaturalReference
[eu.et.cd.pe.da.in.AdvancedBeanInitializer] - bulk load /name.nomenclaturalReference - DONE
[eu.et.cd.pe.da.in.AdvancedBeanInitializer] -  processing /name.nomenclaturalReference.authorship
[eu.et.cd.pe.da.in.AdvancedBeanInitializer] -  invoke initialization on /name.nomenclaturalReferen
ce.authorship beans of class Reference ...
[eu.et.cd.pe.da.in.AdvancedBeanInitializer] - bulk load /name.nomenclaturalReference.authorship
[eu.et.cd.pe.da.in.AdvancedBeanInitializer] - bulk load beans of class TeamOrPersonBase
[eu.et.cd.pe.da.in.AdvancedBeanInitializer] -  SELECT c FROM TeamOrPersonBase as c  WHERE c.id IN
```

```
(:idSet)
[eu.et.cd.pe.da.in.AdvancedBeanInitializer] - initialize bulk loaded beans of class TeamOrPersonBa
se: [1383]
[eu.et.cd.pe.da.in.AdvancedBeanInitializer] - bulk load - DONE
```

The actual initialization takes place in AdvancedBeanInitializer.bulkLoadLazyBeans(BeanInitNode node). Here a strange behavior can be observed in these special cases:

```
private void bulkLoadLazyBeans(node node) {
        for (Class<?> clazz : node.getLazyBeans().keySet()){
            Set<Serializable> idSet = node.getLazyBeans().get(clazz);
    String hql = " SELECT c FROM %s as c %s WHERE c.id IN (:idSet) ";
    hql = String.format(hql, clazz.getSimpleName(), autoInit.leftJoinFetch);
// hql = "SELECT c FROM TeamOrPersonBase as c  WHERE c.id IN (:idSet)"
    Query query = genericDao.getHqlQuery(hql);
    query.setParameterList("idSet", idSet);
    List<Object> list = query.list();
```

Now the TeamOrPersonBase entities are initialized.
However, checking the parent object contained in the node object passed as parameer reveals that the 'authorship' proxy in the object graph has not been initialized.

Preloading the reference in the RegistrationWorkingSetService.loadWorkingSetByReferenceID(Integer referenceID) method avoids the problem (in the below code only relevant lines are shown)

```
Reference reference = repo.getReferenceService().find(referenceID);
repo.getReferenceService().load(reference.getUuid());
```

Since this behavior is data dependent, I serialized the fully initialized registrations as loaded in the RegistrationDaoHibernateImpl.list(Optional<Reference> reference, Collection<RegistrationStatus> includedStatus, Integer limit, Integer start, List<String> propertyPaths). registrations-fully-initialized.ser can be used to restore the original data by de-serializing and persisting the objects to a cdm-database (version 4.15.0-SNAPSHOT) :

```
 try {
            FileInputStream fin = new FileInputStream("registrations-fully-initialized.ser");
            ObjectInputStream oin = new ObjectInputStream(fin);
            List<Registration> registrationsDeserialized = (List<Registration>) oin.readObject();
        } catch (ClassNotFoundException | IOException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
```

NOTE: Doing the serialization in RegistrationDaoHibernateImpl.list() also removed the problem described in here:

```
        try {
            FileOutputStream fout = new FileOutputStream("registrations-before-initialization.ser"
);
            ObjectOutputStream oos = new ObjectOutputStream(fout);
            oos.writeObject(results);
        } catch (IOException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
```

```
        defaultBeanInitializer.initializeAll(results, propertyPaths);

        // initialization is incomplete

        try {
            FileOutputStream fout = new FileOutputStream("registrations-after-initialization.ser"
);
            ObjectOutputStream oos = new ObjectOutputStream(fout);
            oos.writeObject(results);
        } catch (IOException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }


// fully initialized now! the file registrations-fully-initialized.ser has been created after this
```

Related issues:

Related to Edit - bug #7342: incomplete bean initialization with specific dat...	**New**	**04/04/2018**
Duplicated by Edit - bug #7511: LazyInitializationException (LIE) opening Reg...	**Duplicate**	**06/25/2018**

Associated revisions

Revision 7d60d703 - 03/27/2018 02:44 PM - Andreas Kohlbecker

ref #7331 test created and loging more details at level TRACE

Revision 1d6d9f08 - 03/27/2018 02:52 PM - Andreas Kohlbecker

ref #7331 cicrumventing the bug for RegistrationWorkingSetService.loadWorkingSetByReferenceID()

Revision beb7ef65 - 06/25/2018 12:59 PM - Andreas Kohlbecker

ref #7331 re-introducing load to cirumvent bug

Revision 637becc5 - 07/03/2018 06:12 PM - Andreas Kohlbecker

ref #7331 using load instead of find to avoid the magic bug in the AdvancedBeaninitializer

History

#1 - 03/27/2018 02:05 PM - Andreas Kohlbecker

- Description updated

#2 - 03/27/2018 02:10 PM - Andreas Kohlbecker

- File registrations-after-incomplete-initialization.ser added

- File registrations-before-initialization.ser added

#3 - 03/27/2018 02:16 PM - Andreas Kohlbecker

- Description updated

- Found in Version set to Release 5.0

#4 - 03/27/2018 02:25 PM - Andreas Kohlbecker

- Description updated

#5 - 03/27/2018 02:25 PM - Andreas Kohlbecker

- File deleted (registrations-before-initialization.ser)

#6 - 03/27/2018 02:25 PM - Andreas Kohlbecker

- File deleted (registrations-after-incomplete-initialization.ser)

#7 - 03/27/2018 02:25 PM - Andreas Kohlbecker

- File registrations-fully-initialized.ser added

#8 - 03/27/2018 02:29 PM - Andreas Kohlbecker

- Description updated

#9 - 03/27/2018 02:30 PM - Andreas Kohlbecker

- Description updated

#10 - 03/27/2018 02:36 PM - Andreas Kohlbecker

- Description updated

#11 - 03/27/2018 02:38 PM - Andreas Kohlbecker

- File deleted (registrations-fully-initialized.ser)

#12 - 03/27/2018 02:39 PM - Andreas Kohlbecker

- File registrations-fully-initialized.ser added

- Description updated

#13 - 03/27/2018 02:53 PM - Andreas Kohlbecker

- % Done changed from 0 to 10

#14 - 03/27/2018 02:56 PM - Andreas Kohlbecker

I created a test to reproduce this problem in isolation 7d60d703 without success and I am avoiding the effect of this bug in RegistrationWorkingSetService.loadWorkingSetByReferenceID() by doing an explicit load of the Reference commit:cdm-vaadin|7d60d70.

And now I am leaving this issue in favor of getting some other work done ...

#15 - 04/04/2018 02:48 PM - Katja Luther

- Related to bug #7342: incomplete bean initialization with specific data causes LIEs added

#16 - 06/06/2018 05:14 PM - Andreas Kohlbecker

- Description updated

#17 - 06/25/2018 01:12 PM - Andreas Kohlbecker

- Duplicated by bug #7511: LazyInitializationException (LIE) opening Registration Workingset Editor added

#18 - 07/10/2018 03:46 PM - Andreas Kohlbecker

- File AdvancedBeanInitializer-Breakpoints.bkpt added

Adding exported breakpoints from my eclipse which might help debugging this issue.

Files

registrations-fully-initialized.ser	33.9 KB	03/27/2018	Andreas Kohlbecker
AdvancedBeanInitializer-Breakpoints.bkpt	9.16 KB	07/10/2018	Andreas Kohlbecker