

Forum: [Hack Forums](#)

Thread: [Spring Data Projections - ClassCastException](#)

Aussie - October 3, 2019, 9:04 pm

Using projections to get specific fields for Entity class JobEntity, using JobExcerpt.

JobEntity class: <https://pastebin.com/GPeQDiRp>

JobExcerpt class:

```
Code @Projection(  
    name = "jobProjection",  
    types = { JobEntity.class })  
public interface JobExcerpt {  
    Long getId();  
  
    JobEntity.Status getStatus();  
  
    List<BatchEntity> getBatches();  
  
    Date getCreatedOn();  
  
    Long getRetailerId();  
}
```

Very simple JobExcerptRepository class:

```
Code @RepositoryRestResource(excerptProjection = JobExcerpt.class)  
public interface JobExcerptRepository extends JpaRepository<JobEntity, Long> {  
  
    JobExcerpt findByIdAndRetailerId(Long jobId, Long retailerId);  
  
}
```

The line of code it fails on:

```
Code JobExcerpt jobExcerpt = jobExcerptRepository.findByIdAndRetailerId(jobId, retailerId);
```

The thing is... There is no visible (cast) like operator in this line of code. It shouldn't be displayed, right?

Stack Trace: <https://pastebin.com/iNG8h5dn>

pointandlaugh - October 4, 2019, 5:13 am

(Oct 3, 2019 09:04 PM)Aussie Wrote: [\[CITE\]](#) Code @RepositoryRestResource(excerptProjection = JobExcerpt.class)

```
public interface JobExcerptRepository extends JpaRepository<JobEntity, Long> {  
  
    JobExcerpt findByIdAndRetailerId(Long jobId, Long retailerId);  
  
}
```

Not 100% positive, but it's probably this. Your repository interface is providing the JobEntity class as a type parameter, and you're trying to pull out a JobExcerpt in your method there. It would explain your cast exception, it's probably having issues loading the right object. I notice that there

isn't a createOn field in the JobEntity class, but there is a method for it on the projection interface. Don't know if that'd make a difference

MavenGradle - October 4, 2019, 1:04 pm

First of all, you're missing a field in the Entity class:

Code Date getCreatedOn()

The projection might not work because of the mapping issues this might cause but really I'd have to see the entire source code and be able to debug in order to find the exact cause. I wouldn't at all be surprised if it's simply because of the missing createdOn variable

(Oct 4, 2019 05:13 AM)pointandlaugh Wrote: [\[CITE\]](#) Not 100% positive, but it's probably this. Your repository interface is providing the JobEntity class as a type parameter, and you're trying to pull out a JobExcerpt in your method there. It would explain your cast exception, it's probably having issues loading the right object.

Under normal circumstances, this shouldn't be a problem because of Spring's automatic projection which he correctly annotated. I too believe that the issue primarily lies in the missing attribute createdOn

But then again it might also be something in regards to the JobEntity that is extending the DomainObject class.

5oak - October 6, 2019, 11:59 am

Couldn't you use Lombok for this sort of task? If I'm not mistaken

MavenGradle - October 6, 2019, 1:26 pm

(Oct 6, 2019 11:59 AM)5oak Wrote: [\[CITE\]](#) Couldn't you use Lombok for this sort of task? If I'm not mistaken

This has nothing to do with lombok annotations.

Aussie - October 6, 2019, 5:55 pm

(Oct 6, 2019 01:26 PM)MavenGradle Wrote: [\[CITE\]](#) This has nothing to do with lombok annotations. Hi MvnGradle,

Yup, it's to do with the DomainObject class which has the getCreatedOn() method:

<https://pastebin.com/Y5CkkgJX> (DomainObject source code).

--

Also, is JobViewRepository's findXXX method supposed to be returning a JobEntity or the Projection itself?

MavenGradle - October 6, 2019, 6:06 pm

(Oct 6, 2019 05:55 PM)Aussie Wrote: [\[CITE\]](#) Hi MvnGradle,

Yup, it's to do with the DomainObject class which has the getCreatedOn() method:

<https://pastebin.com/Y5CkkgJX> (DomainObject source code).

--

Also, is JobViewRepository's findXXX method supposed to be returning a JobEntity or the Projection itself?

Glad you got that figured out. Well, I think you meant to say `JobExceptRepository` and yes it's supposed to be returning a `JobEntity` object because you're telling it to do so:

Code extends `JpaRepository<JobEntity, Long>`

However, it's probably failing to cast it as a `JobExcerpt` due to Spring not being clever enough to figure out that your interface projection should be of type `DomainObject`. I'd recommend trying a use case in which you create a dummy object instead of the `JobEntity` that does not extend the `DomainObject` to figure this one out.

If it's an entirely different repository you're now talking about, disregard all of the above and then I'm not sure, it's not included in your code snippets anywhere.

Aussie - October 6, 2019, 6:14 pm

(Oct 6, 2019 06:06 PM)MavenGradle Wrote: [\[CITE\]](#) Glad you got that figured out. Well, I think you meant to say `JobExceptRepository` and yes it's supposed to be returning a `JobEntity` object because you're telling it to do so:

Code extends `JpaRepository<JobEntity, Long>`

If it's an entirely different repository you're now talking about, then I'm not sure, it's not included in your code snippets anywhere.

`JobExcerptRepository`, yes the type parameter is specified as `JobEntity`, but I would like for it's method inside to be `JobExcerpt findByIdAndRetailerId(Long jobId, Long retailerId)`; rather than `JobEntity findByIdAndRetailerId(Long jobId, Long retailerId)`; The class cast exception doesn't show up when I use `JobEntity` as the return type. But, Baeldung and online tutorials recommend using the projection (`JobExcerpt` in this case) as the return type.

Someone on StackOverflow mentioned using a `JobEntity` would work fine in that case because "an excerpt is a projection that is applied automatically to a resource collection"

MavenGradle - October 6, 2019, 6:18 pm

(Oct 6, 2019 06:14 PM)Aussie Wrote: [\[CITE\]](#) `JobExcerptRepository`, yes the type parameter is specified as `JobEntity`, but I would

like for it's method inside to be `JobExcerpt findByIdAndRetailerId(Long jobId, Long retailerId)`; rather than `JobEntity findByIdAndRetailerId(Long jobId, Long retailerId)`; The class cast exception doesn't show up when I use `JobEntity` as the return type. But, Baeldung and online tutorials recommend using the projection (`JobExcerpt` in this case) as the return type.

Someone on StackOverflow mentioned using a `JobEntity` would work fine in that case because "an excerpt is a projection that is applied automatically to a resource collection"

Yes, you can do that, although as you can see, you're running into issues. Like I said above, Spring's projection mechanisms are probably not clever enough to figure out that your `JobExcerpt` interface projection should extend the `DomainObject` class and it fails to cast it as the appropriate object. Which means you'll have to either convert it yourself in whichever service uses the repository (DTO) or use a different projection technique in your particular use case. For example, instead of using an interface, you could create a `JobExcerpt` class in which you explicitly extend the `DomainObject` class and see if that works for you instead of using an interface for the projection. You could also write the query yourself in a repository in order to bypass this.

Also, Baeldung isn't entirely up-to-date on most of their 'tutorials' and quite frequently completely disregards best practices.

Quote:an excerpt is a projection that is applied automatically to a resource collection

Yes, unless the thing that's doing the automatic stuff behind the scenes isn't clever enough to say it in layman's terms.

In any case, if some of the methods above worked, you might want to file an official bug report or ask for insights over at the spring team.

Aussie - October 9, 2019, 3:18 am

(Oct 6, 2019 06:18 PM)MavenGradle Wrote: [\[CITE\]](#) Yes, you can do that, although as you can see, you're running into issues. Like I said above, Spring's projection mechanisms are probably not clever enough to figure out that your JobExcerpt interface projection should extend the DomainObject class and it fails to cast it as the appropriate object. Which means you'll have to either convert it yourself in whichever service uses the repository (DTO) or use a different projection technique in your particular use case. For example, instead of using an interface, you could create a JobExcerpt class in which you explicitly extend the DomainObject class and see if that works for you instead of using an interface for the projection. You could also write the query yourself in a repository in order to bypass this.

Also, Baeldung isn't entirely up-to-date on most of their 'tutorials' and quite frequently completely disregards best practices.

Yes, unless the thing that's doing the automatic stuff behind the scenes isn't clever enough to say it in layman's terms.

In any case, if some of the methods above worked, you might want to file an official bug report or ask for insights over at the spring team.

Thanks for the response. Posting what worked here for future people in case anyone has this issue.

Two solutions worked:

(1) Using native queries @Query() which is an equally viable solution.

(2) Creating my own custom Entity called SimpleJobEntity and making a JpaRepository for this. SimpleJobEntity being a duplicate of JobEntity with less fields.

What didn't work:

- Changing Projection interface into an abstract class to extend DomainObject.

(Oct 9, 2019 03:18 AM)Aussie Wrote: [\[CITE\]](#) (1) Using native queries @Query() which is an equally viable solution.

Apologies, I was a bit drunk. It's not native queries that I used, it's JPQL.

MavenGradle - October 10, 2019, 9:48 am

You should provide an example project and test datasource to the Spring dev team to confirm your findings.

<https://github.com/spring-projects/spring-boot/issues>

Aussie - October 10, 2019, 3:28 pm

(Oct 10, 2019 09:48 AM)MavenGradle Wrote: [\[CITE\]](#) You should provide an example project and test datasource to the Spring dev team to confirm your findings.

<https://github.com/spring-projects/spring-boot/issues>

Yeah, fair point. That's a good idea, I'll do it after work today.