How to search with JpaRepository and nested list of objects?

Asked 2 years, 1 month ago Active 2 years, 1 month ago Viewed 13k times

10

5

Description

There is a `PersonRepository` and `Person` entity, `Person` class contains `List<Qualification>` . `Qualification` class has 3 simple fields.

I have tried to add `@Query` annotation on custom method and use JPQL to get the results, but `Qualification` class fields were not available for manipulation in JPQL as it repository itself contains `List<Qualification>` instead of just a simple field of `Qualification` .

How can I search by these Qualification's nested fields?

Query

Now I need to find list of person entity where qualification's experienceInMonths is greater than 3 and less than 9 AND qualification's name field = 'java'.

Code

Person.java

```
@Data
@Entity
public class Person {

@Id
@GeneratedValue(strategy = GenerationType.IDENTITY)
private String id;
@NotEmpty
@Size(min = 2)
private String surname;

@ElementCollection(targetClass = java.util.ArrayList.class, fetch = FetchType.EAGER)
private List<Qualification> qualifications = new ArrayList<>();

}
```

PersonRepository.java

```
@Repository
public interface PersonRepository extends JpaRepository<Person, String> {
}
```

Qualification.java

```
@Data
@AllArgsConstructor
public class Qualification implements Serializable {

    @Id @GeneratedValue
    private String id;
    private String name;
    private String experienceInMonths;
}
```

EDIT: not duplicate of this post, as here is the collection of nested objects. Not just single reference.

java jpa spring-boot spring-data

edited Dec 27 '17 at 18:34

asked Dec 27 '17 at 18:14
DevDio
1,165 1 11 23

Possible duplicate of Spring Data JPA find by embedded object property – Robert H Dec 27 '17 at 18:30

1 Answer

15

First, change `experienceInMonths` from `String` to `int` (otherwise you can not compare the string with the number). Then you can try to use this 'sausage':

```
tions_experienceInMonthsGreaterThanAndQualifications_experienceInMonthsLessThanAndName(int
ter, int experienceLess, String name);
```

Or you can try to use this pretty nice method:

```
@Query("select p from Person p left join p.qualifications q where q.experienceInMonths >
?1 and q.experienceInMonths < ?2 and q.name = ?3")
List<Person> findByQualification(int experienceGreater, int experienceLess, String
name);
```

answered Dec 27 '17 at 20:52
Cepr0
15.6k 3 38 68

did it worked for you locally? the JPQL version gaves me `org.hibernate.QueryException: cannot dereference scalar collection element:` and the sausage one: `Caused by: java.lang.IllegalStateException: Illegal attempt to dereference path source [null] of basic type` . Even with changing Qualification#experienceInMonths type to `int` – DevDio Dec 27 '17 at 21:16

@DevDio Try to add `@Entity` to `Qualification` and replace `@ElementCollection` to `@OneToMany` (a similar variant works for me...) – Cepr0 Dec 28 '17 at 7:55

Ok, finally got it working, but it required implementation of separate `Qualification` table in Hibernate. which holds FK to Person (@Entity, additional field person with @ManyToOne) + `QualificationRepository` was mandatory. Also tested proposed methods, the version with JPQL worked fine, yet the 'sausage' name required to be adjusted to: `findByQualifications_experienceInMonthsGreaterThanAndQualifications_experienceInMonthsLessThanAndQualifications_Name` . Otherwise it searched for Person name. Big thanks for help @Cepr0! – DevDio Dec 28 '17 at 11:35

1 @DevDio 'QualificationRepository was mandatory' - it will be not necessary if you use @OneToMany(cascade = CascadeType.ALL, orphanRemoval = true). Then you can use only one Person repo to store both entities. – Cepr0 Dec 28 '17 at 13:43