Select specific columns from a database table using Spring Data JPA

faun.pub/select-specific-columns-from-a-database-table-using-spring-data-jpa-d4eb0a24a2c4

November 8, 2021

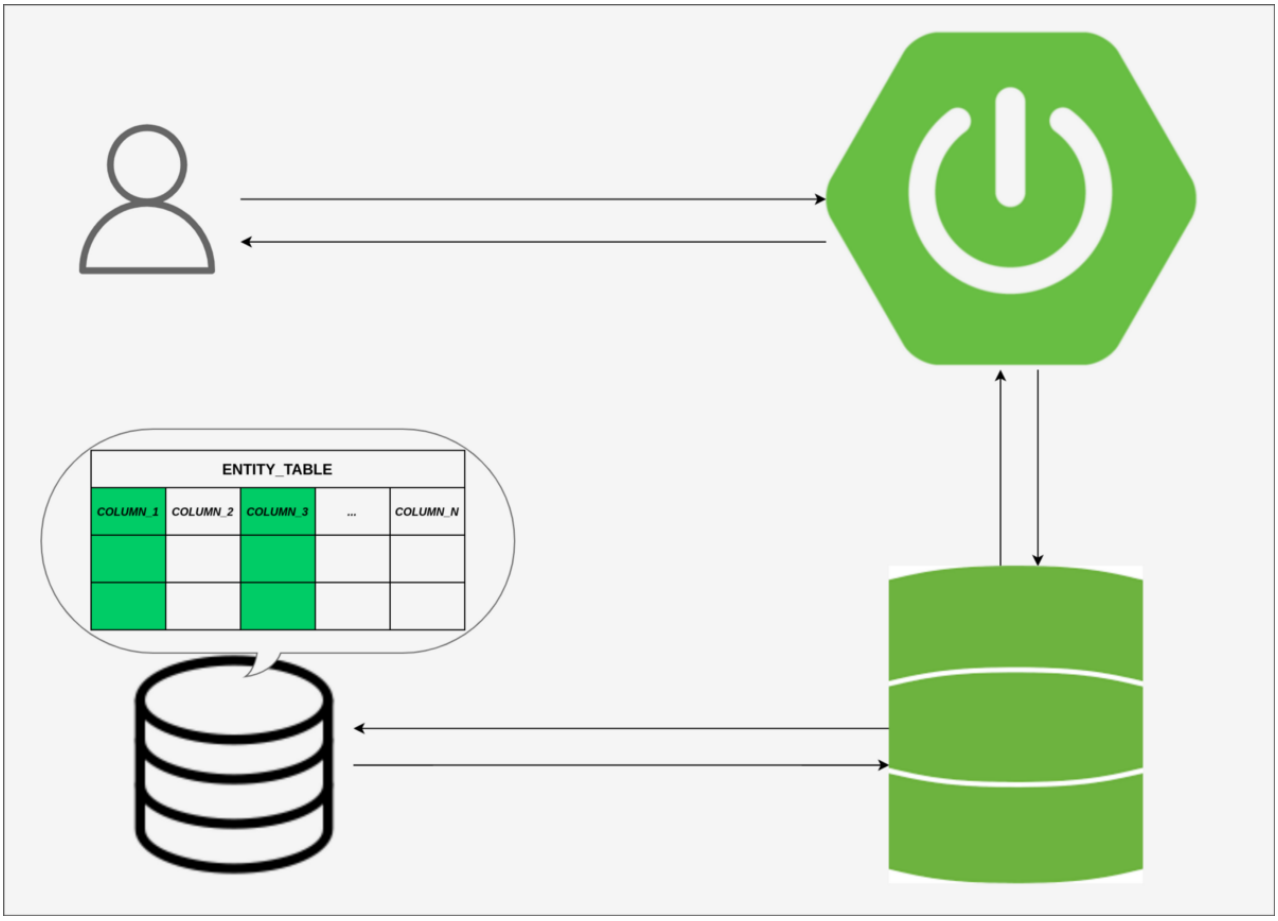
Ivan Polovyi
Nov 8, 2021

.

3 min read

.

There are situations where we need to retrieve only selected properties of an entity. This tutorial explains how to achieve that in different ways.



Let's consider a use case where we have a SpringBoot application connected to a database using Spring Data JPA. Here I use H2, but it can be any SQL database. This database has a table called Customer which contains information about customers such as an id, a full

name, a phone number, an address, a job, and a creation date. The entity class CustomerEntity is mapped to this table. Spring Data JPA interface repository was created to return this entity from a database.

The app exposes REST APIs and for simplicity, each API returns a list of responses. To obtain data all APIs call methods in the repository interface and then returns it. Before returning, as a good practice, each entity is converted to a DTO. The source code you can find here. It is very simple and this tutorial will be focused mainly on the repository interface, where basically all explained technics are.

One API returns a list of all customers with all properties included. But often there are situations when getting all properties is not necessary. And it is always a good practice to retrieve from the database only the required data. The main reason for that is to improve the performance of an application. Bellow, I will show how to retrieve only a specific set of data from a database in different ways.

1. Select only one property of an entity

The app exposes one endpoint, that returns only the phone numbers of the customers. The custom method is created in the repository interface when only one property (field) is needed. This method is annotated with a @Query annotation. This annotation has a custom JPQL statement, which selects only specified property.

```
@Query("SELECT phoneNumber FROM CustomerEntity customer")List<String>
findPhoneNumbers();
```

2. Select specific properties of an entity using custom DTO object

Most of the time we need to return more than one property of an entity. The app has an API that returns only a full name of a customer and an address. The class (DTO) is created with selected fields and a constructor which accepts these fields as parameters. A custom method in the repository interface returns a list of objects created using this class. The method has a @Query annotation. This annotation has a JPQL query that creates the object from a custom class with selected fields. Fields are passed to the constructor of this class. The fully qualified class name has to be specified in the query.

```
@Query("SELECT new com.polovyi.ivan.tutorial.entity.dto.MailingAddressDTO
(fullName, address) FROM  CustomerEntity customer")List<MailingAddressDTO>
findMailingAddresses();
```

3. Select specific properties of an entity using interface-based projections

One more option for retrieving more than one property is to use an interface-based projection. In this case, the interface is created with accessor methods for the properties to be retrieved. An example app has one API that returns the full name and the job of each customer. This API calls repository which returns a list of objects created from the class,

that implements the above-mentioned interface. Spring handles this “on the fly”. Then in the @Query annotation of this method, the JPQL query is specified. Make sure to use aliases to the fields, as without those it won’t work.

```
@Query("SELECT fullName AS fullName, customer.job AS job FROM  CustomerEntity
customer")List<CustomerJobProjection> findAllJobs();
```

Conclusion

The performance of an application is very important. Every developer has to optimize the application so that it is fast and efficient. One of the many ways is to minimize data transfer between an app and a database. It can be achieved by using technics explained in this tutorial.

Thank you for reading! If you have any questions or suggestions, please feel free to write me on my LinkedIn account.

If this post was helpful, please click the clap 👏 button below a few times to show your support for the author 👇