TECH WITH MADDY

Software Engineering | Technical Writing

Follow

Replace a JPA entity with a DTO

Maddy

May 30, 2022 · 4 min read

3 1

TABLE OF CONTENTS

0:00 / 4:04

Have you ever come across a Sonar vulnerability issue such as:

> *Replace this persistence entity with a POJO or DTO object.*

This happens when you pass a persistence entity into the **@ ResponseBody** of a REST call instead of a DTO object.

This article will show you how to replace a persistence entity with a DTO object.

You can clone the Github re

We'll use this application as a reference.

TABLE OF CONTENTS

1. Add Model Mapper

The Model Mapper is an object mapping library. It makes it easy to convert one object model into another object model.

In the `pom.xml`, add the Model Mapper dependency:

COPY

```
<dependency>
   <groupId>org.modelmapper</groupId>
   <artifactId>modelmapper</artifactId>
   <version>2.4.5</version>
</dependency>
```

P.S: if you ever get any problems after adding the above dependency, running the command `mvn clean install` usually helps.

2. Create a Data Transfer Object

Martin Fowler introduced the Data Transfer Object pattern in his book "Patterns of Enterprise Application Architecture".

A Data Transfer Object is an object which carries data between processes.

This object doesn't contain

In the `src/main/java/com/techwithmaddy/CustomerAPI` directory:

TABLE OF CONTENTS

This class has the following content:

COPY

```
package com.techwithmaddy.CustomerAPI.dto;

import lombok.Data;

@Data
public class CustomerDTO {

    private String firstName;
    private String lastName;
    private String email;
    private String phoneNumber;

}
```

3. Create a Customer Converter class

This converter class is responsible for converting an entity into DTO and vice versa.

In the `src/main/java/com/techwithmaddy/CustomerAPI` directory:

1. Create another package called `converter` (all lowercase).
2. Create a class called `CustomerConverter` in the `converter` package.

TABLE OF CONTENTS

```
package com.techwithmaddy.CustomerAPI.converter;

import com.techwithmaddy.CustomerAPI.dto.CustomerDTO;
import com.techwithmaddy.CustomerAPI.model.Customer;
import org.modelmapper.ModelMapper;
import org.springframework.stereotype.Component;

@Component
public class CustomerConverter {

    public CustomerDTO convertEntityToDto(Customer customer) {
        ModelMapper modelMapper = new ModelMapper();
        CustomerDTO customerDTO = modelMapper.map(customer, CustomerDTO.class)
        return customerDTO;
    }

    public Customer convertDtoToEntity(CustomerDTO customerDTO) {
        ModelMapper modelMapper = new ModelMapper();
        Customer customer = modelMapper.map(customerDTO, Customer.class);
        return customer;
    }
}
```

4. Refactor Customer Service class

We want to use the Custom ner entity database object.

TABLE OF CONTENTS

COPY

```
@Autowired
ModelMapper modelMapper;

@Autowired
CustomerConverter customerConverter;
```

The `saveCustomer()` method will now be like this:

COPY

```
public CustomerDTO saveCustomer(CustomerDTO customerDTO) {
    Customer customer = customerConverter.convertDtoToEntity(customerDTO);
    customer = customerRepository.save(customer);
    return customerConverter.convertEntityToDto(customer);
}
```

5. Refactor the Customer Controller class

Add this property to the Rest Controller:

COPY

```
@Autowired
private CustomerConver
```

TABLE OF CONTENTS

COPY

```
@RequestMapping(method = {POST}, path = "/save", produces = MediaType.APPL
public CustomerDTO saveCustomer(@Valid @RequestBody CustomerDTO customerDT
    return customerService.saveCustomer(customerDTO);
}
```

6. Run the application

For the sake of this tutorial, we want to check if the above changes work.

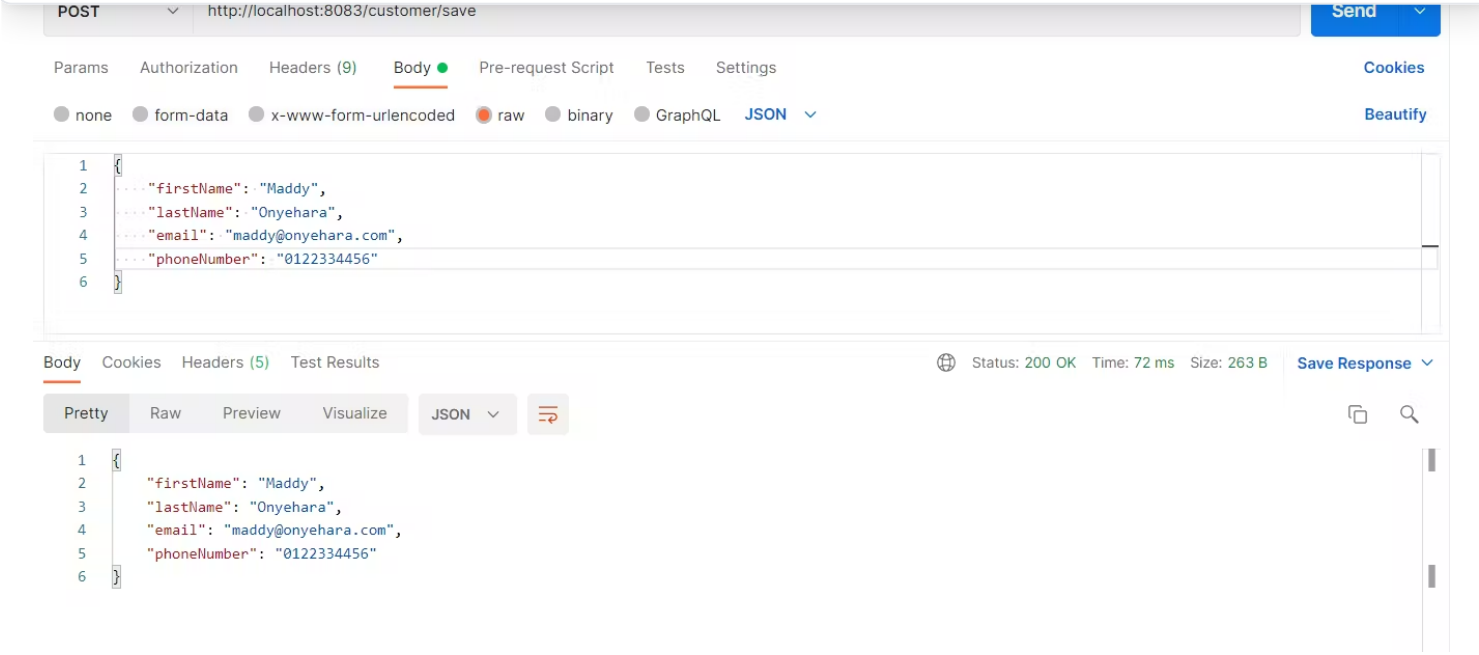
Therefore, temporarily comment out:

1. The Rest Controller's GET, PUT, and PATCH requests.
2. The entire CustomerServiceTest class.

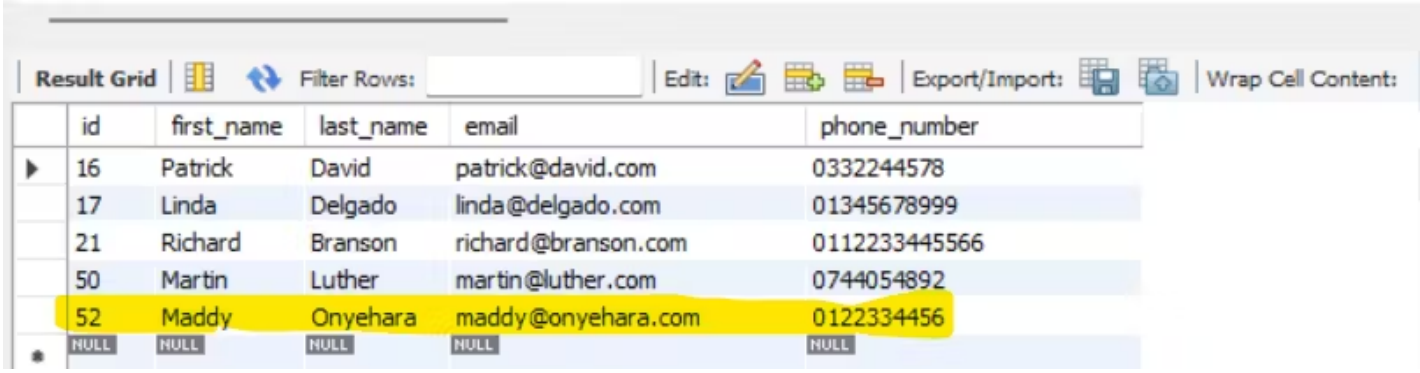
Now you can run the application.

- On Postman, create the following POST request and hit the SEND button:

TABLE OF CONTENTS



- On MySQL, you should see the new customer added to the table.



7. Why use a DTO instead of an entity?

Let's imagine the following scenario:

> *A school wants to save data about its students. A database stores student names, surnames, email[illegible]ation. Teachers only have access to some of the data stored in the database (such as name, surname,*

TABLE OF CONTENTS

Data Transfer Objects work this way: they store some of the data present in the database with no business logic.

The client communicates with the controller layer in the Spring Boot architecture.

If we didn't have Data Transfer Objects, we would have to use an Entity class in the controller class, creating a vulnerability risk in our application.

> ***Related:*** *Spring Boot Architecture*

Key Takeaways

This article has shown you how to replace an entity with a DTO and use a DTO in the ResponseBody of a REST call.

I hope you've found this article helpful.

Do you know of any other approach? Please let me know in the comments.

Until next time! 🤷🏾‍♀️

ADDITIONAL RESOURCES

- StackExchange: What is the use of DTO instead of Entity?
- Viktoria Senuic, Stack Abuse: Data Transfer Object Pattern in Java - Implementation and Ma
- Baeldung: The DTO Patte (Data Transfer Object)

TABLE OF CONTENTS

Subscribe to my newsletter

Read articles from **Tech with Maddy | Tech & Writing** directly inside your inbox. Subscribe to the newsletter, and don't miss out.

Enter your email address SUBSCRIBE

Did you find this article valuable?

Support **Maddy** by becoming a sponsor. Any amount is appreciated!

Sponsor

See recent sponsors | Learn more about Hashnode Sponsors

Java Spring Springboot design patterns best practices

TABLE OF CONTENTS

Maddy

Follow

A software engineer who turned technical writer.

ARTICLE SERIES

Spring Boot

1

How To Update Or Insert A Record In Spring Data JPA

Do you want to know how to update or insert a new record using Spring Boot? This article is for you....

2

Replace a JPA entity with a

TABLE OF CONTENTS

••• Show all 3 posts

6

How to Create a Spring Boot REST API

In my previous article, I wrote about Spring Boot Architecture. I think Spring Boot is great to crea...

TABLE OF CONTENTS

7

Spring Boot Architecture

This article will explain how the Spring Boot architecture operates. Let's start. What Is Spring Boo...

TABLE OF CONTENTS