24-SpringBoot Project-ResponseEntity & GetById

In the previous chapter, we successfully obtained a list of products. However, what happens if we click on a specific product on the home page? A request for that product, along with its ID, is sent to the server. Since we don't have an implementation for handling this, we get a request error. Let's resolve this issue by implementing the necessary functionality.

Adding a Method to Handle Product Requests

1. **Create the getProduct() method in the ProductController class:**
    - Annotate the method with @GetMapping("/product/{id}") to obtain a single product by its ID.
    - Delegate the implementation to the ProductService class in the Service layer.
    - Use the ProductRepo interface in the Repository layer to call the findById method from the JpaRepository, which retrieves the product based on its ID and returns an Optional object.

```
 3 import java.util.List;
14
15 @RestController
16 @RequestMapping("/api")
17 @CrossOrigin
18 public class ProductController {
19
20     @Autowired
21     ProductService service;
22
23     @GetMapping("/products")
24     public List<Product> getAllProducts() {
25         return service.getAllProducts();
26     }
27
28     @GetMapping("product/{id}")
29     public Product getProduct(@PathVariable int id) {
30         return service.getProduct(id);
31
32     }
33 }
```

Handling non-existent product IDs

- If a product ID that doesn't exist in the database is passed, we should handle it appropriately by sending a status code and a message instead of returning a default product.
- The initial implementation might use get(), which returns the product if it exists.
- Update this to return a ResponseEntity with the appropriate status code and message.

The get() method

When you retrieve a product by its ID using methods like **findById** in Spring Data JPA, the return type is **Optional<Product>**. The Optional class is a container that may or may not contain a non-null value. This helps avoid **NullPointerException** and allows more expressive handling of absent values.

- **Using get()**: The get() method is one way to retrieve the value from an Optional. If the value is present, get() will return it. However, if the Optional is empty (i.e., the product ID doesn't exist in the database), calling get() will throw a NoSuchElementException.

```
Product product = productRepo.findById(prodId).get();
```

The orElse() Method

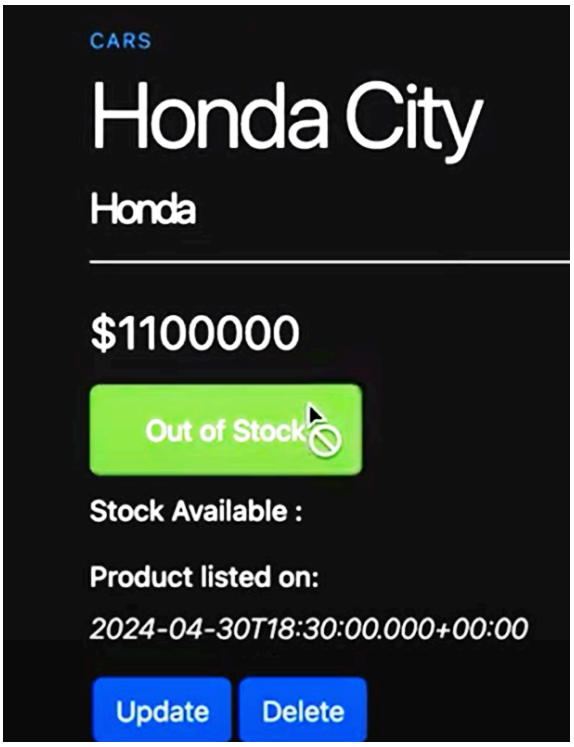
The **orElse()** method is a safer alternative to **get().** It allows you to specify a default value to return if the Optional is empty, preventing an exception from being thrown. However, using **orElse()** in a case where the product doesn't exist might not be appropriate for production environments, as it would provide a default product, which might lead to misleading information being returned to the client.

```
Product product = productRepo.findById(prodId).orElse(defaultProduct);
```

```
 1 package com.example.ecom_proj.service;
 2
 3 import java.util.List;
10
11 @Service
12 public class ProductService {
13
14     @Autowired
15     ProductRepo repo;
16
17     public List<Product> getAllProducts(){
18         return repo.findAll();
19     }
20
21     public Product getProduct(int id) {
22         return repo.findById(id).get();
23     }
24 }
```

```
 1 package com.example.ecom_proj.repo;
 2
 3 import org.springframework.data.jpa.repository.JpaRepository;
 7
 8
 9 @Repository
10 public interface ProductRepo extends JpaRepository<Product, Integer> {
11
12 }
```

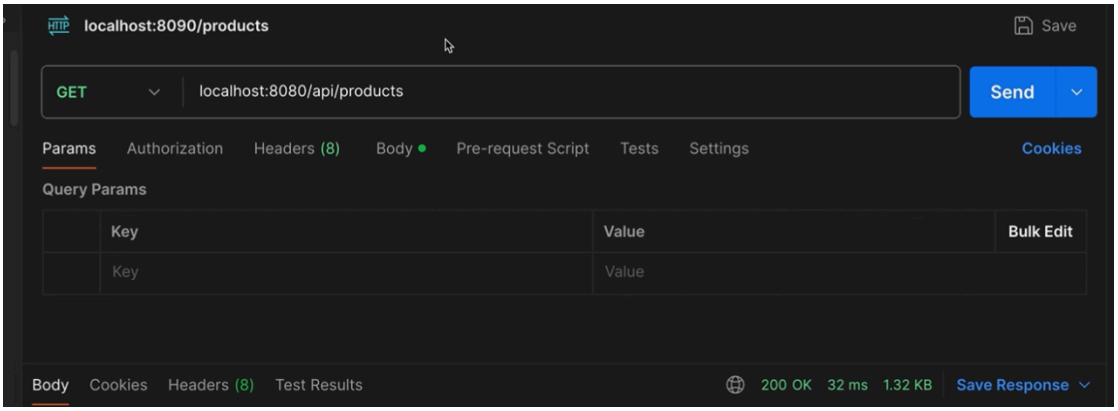
Formatting the Date

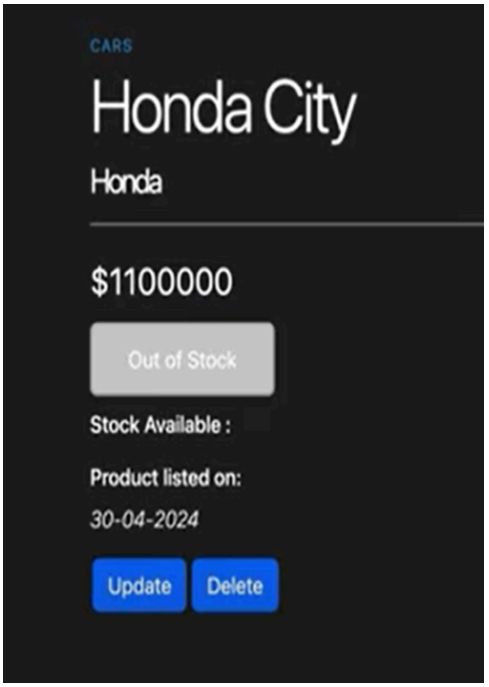


- The date that appears in the product details is not formatted correctly. There are two ways to fix this:
    - Modify the date on the frontend.
    - Update the date format in the backend and send it to the React application in the correct format.
- To update the backend:
    - Add the @JsonFormat(shape=JsonFormat.Shape.STRING, pattern = "dd-MM-YYYY") annotation to the releaseDate variable in the Product model class.

```java
package com.example.ecom_proj.model;

import java.math.BigDecimal;

@Entity
@Data
@AllArgsConstructor
@CrossOrigin
public class Product {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int id;
    private String name;
    private String descp;
    private String brand;
    private String category;

    @JsonFormat(shape=JsonFormat.Shape.STRING,pattern = "dd-MM-YYYY")
    private Date date;
    private boolean available_status;
    private int quantity;
    private BigDecimal price;
}
```

Returning Status Codes





- When using tools like Postman, it's important to observe the status code for each request.
- Status codes are crucial for developers working on frontend and backend projects as they help in understanding and handling exceptions and errors.
- Use the ResponseEntity<> object to return both the product data and the status code.
- Update the getProduct() method in the ProductController class to use ResponseEntity:
- Now, every implementation should use ResponseEntity, which is a good strategy.

Verifying the Solution

- Restart the application.
- Check if the updated code is producing the expected output on both React and Postman.

1) First way to handle it.

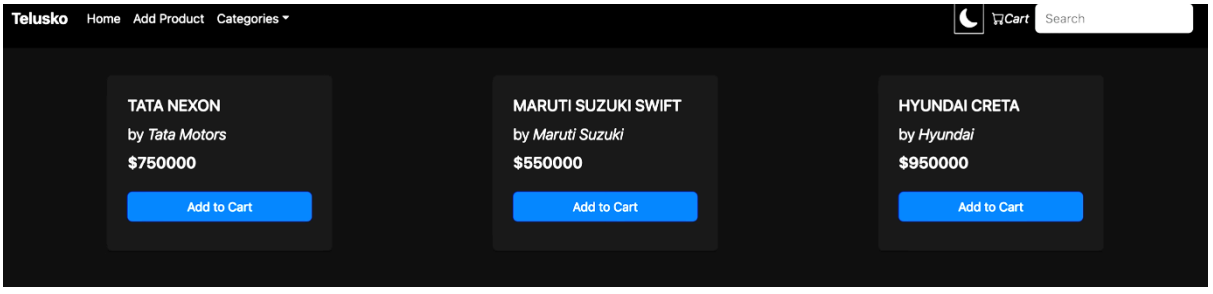
```
@GetMapping("/products")
public ResponseEntity<List<Product>> getAllProducts(){

    return new ResponseEntity<>(service.getAllProducts(), HttpStatus.OK);
}
```

2) In second way we can handle it like

```
@GetMapping("/product/{id}")
public ResponseEntity<Product> getProduct(@PathVariable int id){

    Product product = service.getProductById(id);

    if(product != null)
        return new ResponseEntity<>(product, HttpStatus.OK);
    else
        return new ResponseEntity<>(HttpStatus.NOT_FOUND);
}
```

Additional Features in the Next Chapter

- We will examine how products are added via the client side (React) and stored in the database.



Conclusion

By implementing these changes, we ensure that our application handles specific product requests correctly, formats dates properly, and returns the appropriate status codes.

Stay Curious and Happy Coding! ☺👾

☺ 👾