Slide Note

```
{
"product":
  {"productCode": "MOBXX001",
  "productName": "realme 8 (Cyber Silver, 128 GB)  (6 GB RAM)"
  },
"ProductPrice":
{
  "price": 350,
  "discountedPrice": 325
},
 "Seller": {
     "sellerName":"TradePLFIPL",
     "isFAssured": true
 },

 "Customer": {
  "customerName":"xxxx",
  "customerCode": "Px101"
  },
  "DeliveryDetails":
  {
     "dateOfExpectedDelivery":"10-09-2021",
     "isReplacementPolicy":true
  }
}
```

Shopping Cart

- We are familiar with JSON files in front-end application development.
  - Currently, the structure of the data of a shopping cart in JSON format is shown on the left.
  - Do you think we should divide this data and store it in RDBMS tables?
  - What will be an efficient way to store this data as it is?

Slide Note

Think and Tell

- What happens to your application when an error occurs during a transaction?
- What happens when a client sends a request, and your application fails to respond due to a technical glitch?
- Do you think all backend applications must handle exceptions?

Slide Note

Manage Semi-structured and Unstructured Data and Handle Exceptions Within a RESTful Service by Using Mongo Repository

Slide Note

Learning Objectives

- Create a RESTful API using Spring Data MongoDB and Spring Boot.
- Handle exceptions in a RESTful API.

Slide Note

Create a RESTful API Using Spring Data MongoDB and Spring Boot

Menu

00:05:00:00:25

05/ 25

Slide Note

Returns the entity identified by the given ID

MongoRepository

- `MongoRepository` is an interface provided by Spring Data to work with MongoDB.
- It extends `CrudRepository`.
- It provides all the necessary methods that help create a CRUD application, and it also supports the custom-derived query methods.
- Some CRUD Methods:
  - `S save(S entity)` - Saves the given entity.
  - `findById(ID primaryKey)` - Returns the entity identified by the given ID.
  - `S insert(S entity)` - Inserts the given entity.

Click on link to refer to docs.

Slide Note

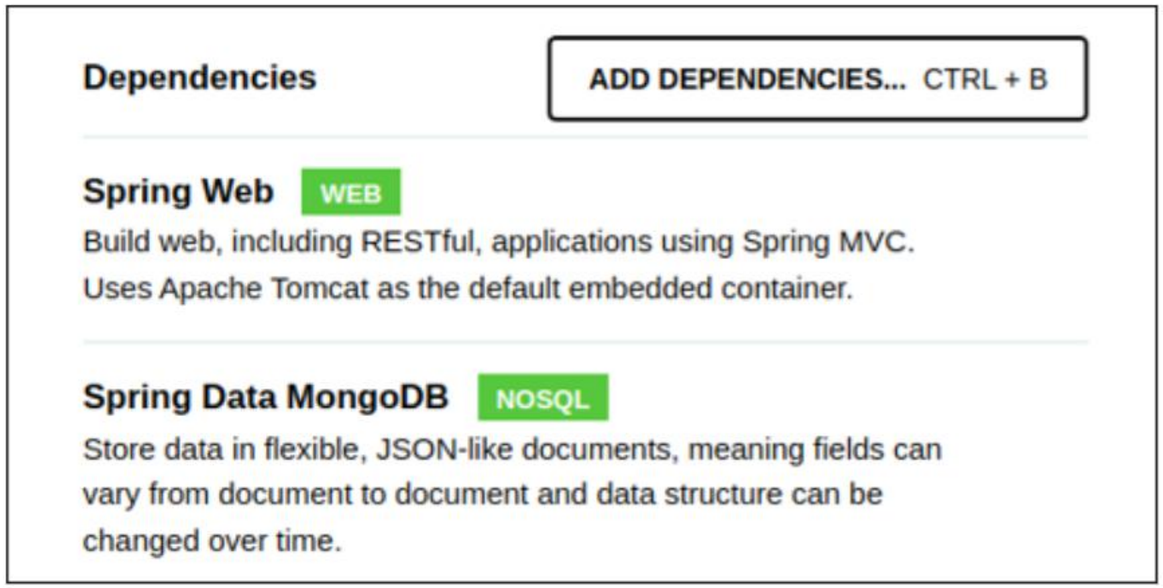
Create a RESTful API

- Set up the Sprint Boot project by using the Spring Initializr.

Slide Note

The Spring Data MongoDB project provides integration with the MongoDB document database. Key functional areas of Spring Data MongoDB are a POJO centric model for interacting with a MongoDB DBCollection and easily writing a Repository style data access layer

Dependency Management



- Spring Data MongoDB dependency provides integration with the MongoDB document database.
- It is a POJO-centric model for interacting with a MongoDB collection, which is its key functionality.

Slide Note

Domain Layer

- The domain layer has two classes - `Address` and `Customer`.
- Here, the `Customer` class is a document in the database, so the annotation we will provide is `@Document`.
- `CustomerId` is annotated with `@Id`, which means this attribute is an objectId in MongoDB.

```java
public class Address {
    private String city;
    private String state;
    //getter and setter
}
```

```java
@Document
public class Customer {
    @Id
    private int customerId;
    private String customerName;
    private Address customerAddress;
    private long customerPhoneNo;

    //getter and setter

}
```

Slide Note

Repository Layer

- Spring Data `MongoRepository` helps access the data in the MongoDB database.
- The `MongoRepository` provides common functionalities that we can easily plug in and use.
- Customer Repository extends the `MongoRepository`.
- The `Customer` is the domain class, which is a document, and the `Integer` is the dataType of the objectId.

```
public interface CustomerRepository extends MongoRepository<Customer,Integer> {

}
```

Document

Data type of the Id

Slide Note

Any data we wish to retrieve based on any other attribute other than the attribute marked with @Id from the database can be retrieved by using the findByAttribute() method. Here, attributes are the attributes of @Document class.

Repository Layer (contd.)

- Any data we wish to retrieve, based on any other attribute, other than the attribute marked with `@Id` from the database, can be retrieved using the `findByAttribute()` method. Here, attributes are the attributes of the @Document class.
- You can only write these methods for the domain class that is annotated with `@Document`.
- The implementing class will be given by the Spring Data.
- Follow the conventions given by Spring Data to write these methods, and the implementation will be taken care of by the Spring Data.

```java
public interface CustomerRepository extends MongoRepository<Customer,Integer> {
    int findByCustomerPhoneNo(long customerPhoneNo);
    String findByCustomerName(String customerName);
}
```

Slide Note

As City is an attribute of the Address class so by using @Query annotation we can create custom query.

As customerAddress is a variable of type Address, so by using customerAddress we can call city.

Repository Layer (contd.)

- Consider a scenario where we want to get information about all the customers who belong to a particular city.
- In this scenario, the city is not the attribute of the document `Customer`, so we cannot use `findByX()` methods of `MongoRepository`.
- As City is an attribute of the Address class so by using @Query annotation we can create custom query.
  - As customerAddress is a variable of type Address, so by using customerAddress we can call city.

```
public interface CustomerRepository extends MongoRepository<Customer,Integer> {
    @Query("{'customerAddress.city' : {$in : [?0]}}")
    List<Customer> findAllCustomerFromCity(String city);
}
```

The placeholder *?0* references the first parameter of the method.

`CustomerAddress` is an attribute in the Customer class.

Attribute of Address class

$in – here `in` is operator

- Click on link for docs

Slide Note

Service Layer

```java
@Override
public Customer saveCustomerDetail(Customer customer) throws CustomerAlreadyExistsException {
    if(customerRepository.findById(customer.getCustomerId()).isPresent())
    {
        throw new CustomerAlreadyExistsException();
    }
    return customerRepository.save(customer);
}
```

- The Service object is injected with the Repository object.
- The Repository layer method findById() is used.
- CustomerAlreadyExistsException will be thrown if a customer already exists in the database.

Slide Note

Controller Layer

```java
@PostMapping("customer")
public ResponseEntity<?> saveCustomer(@RequestBody Customer customer) throws CustomerAlreadyExistsException {
    try {
        customerService.saveCustomerDetail(customer);
        responseEntity = new ResponseEntity(customer , HttpStatus.CREATED);
    } catch (CustomerAlreadyExistsException e) {
        throw new CustomerAlreadyExistsException();
    }
    catch (Exception e)
    {
        responseEntity = new ResponseEntity<>( body "Error  !!!Try after sometime", HttpStatus.INTERNAL_SERVER_ERROR);
    }
    return responseEntity;
}
```

If there is any exception other than CustomerNotFound, we can handle it in this catch block.

- The Controller layer provides handler methods for GET, POST, PUT, and DELETE mapping.
- Access the Service layer to perform the operations.
- Test the RESTful API using POSTMAN.
- If the Service layer throws the exception, it will get handled and be thrown again from the Controller layer.

Slide Note

Handle exceptions in a RESTful API

Slide Note

Exception Handling

- Exception handling is an integral part of any application development process.
- The layers of a RESTful application must handle the exceptions appropriately:
  - `Service Layer`

    The Service layer performs the business logic of the application and communicates with the database; thus, all methods must declare exceptions.

  - `Controller Layer`

    A RESTful application can communicate the success or failure of an HTTP request by returning the right status code in response to the client.

    If the service layer throws an exception due to a database failure, etc., the controller sends an error status code back to the client.

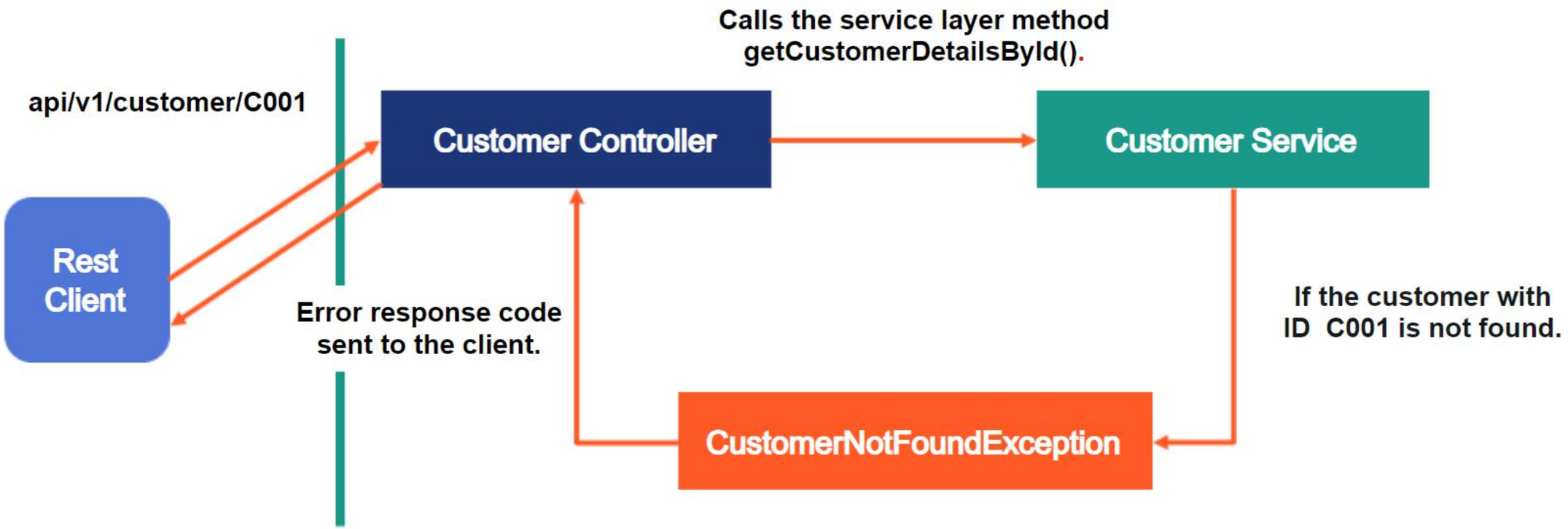
  - `Client side`

    An appropriate status code can help the client identify problems that might have occurred while the application was dealing with the request.

Slide Note

- From POSTMAN, the call is made to the appropriate handler.
- The handler then calls the service methods to process the business logic of the request. (In this case, get a product by the product code).
- If the product is not found, the error handling is done and a response is sent back to the client (POSTMAN).

Exception Propagation in a RESTful Application



Slide Note

Exception Class

- `CustomerNotFoundException` **is a user-defined exception class.**
- `@ResponseStatus` **marks a method or exception class with the status code and the reason a message should be returned.**
- **The status code is applied to the HTTP response when the handler method is invoked, or whenever the specified exception is thrown. It overrides status information set by other means, such as** `ResponseEntity` **or redirect.**

```
@ResponseStatus(code= HttpStatus.NOT_FOUND , reason = "Customer with specified id is not found")
public class CustomerNotFoundException extends Exception{
    public CustomerNotFoundException(String message){
        super(message);
    }
}
```

Menu

00:18:00:00:25

18/ 25

Slide Note

application.properties

```
server.port=8083
spring.data.mongodb.database=customerdb
spring.data.mongodb.uri=mongodb://localhost:27017/
server.error.include-message=always
```

Port on which this service will run

Name of the collection

Path to the MongoDB database.

Keeping this property as always will always show the error messages to the POSTMAN output.

Slide Note

Quick Check

What is the response code for HttpStatus.OK?

1. 302
2. 200
3. 404
4. 500

Slide Note

Quick Check: Solution

What is the response code for HttpStatus.OK?

1. 302
2. 200
3. 404
4. 500

Slide Note

Supermarket

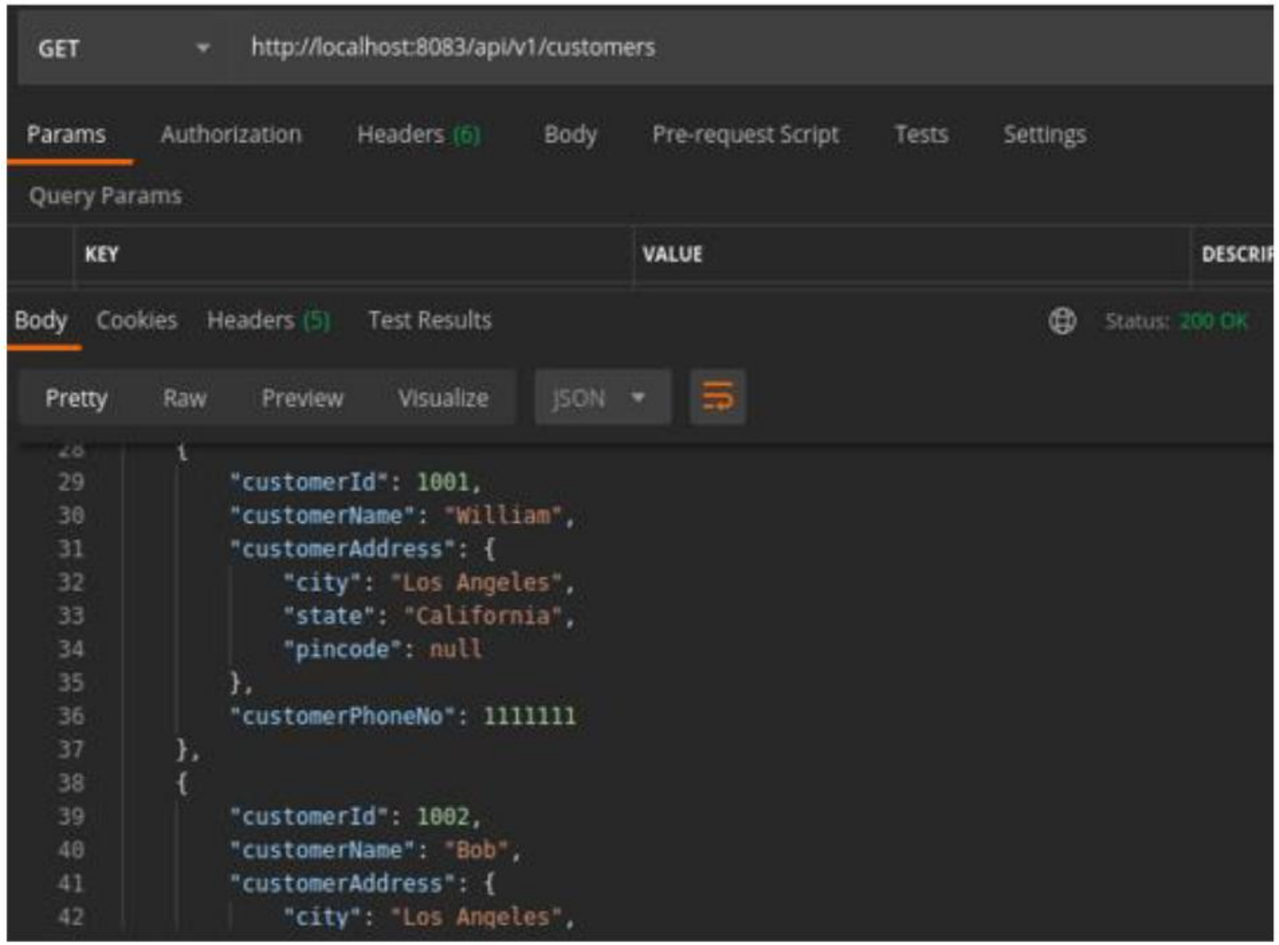
A supermarket chain needs to perform an analysis of customer data. They need to determine how they can expand their business to new localities based on where the customers reside.
To start off, the entire dataset must be stored in the system. The admin of the supermarket chain should be able to add, update, delete, and insert the customer data along with the address of the customers.
Let us build a RESTful API with the MongoDB Repository to perform the task. The application should be able to handle exceptions.

Check the solution here.

DEMO

Slide Note

Postman Output



Menu

00:23:00:00:25

23/ 25

Slide Note

Postman Output With an Exception