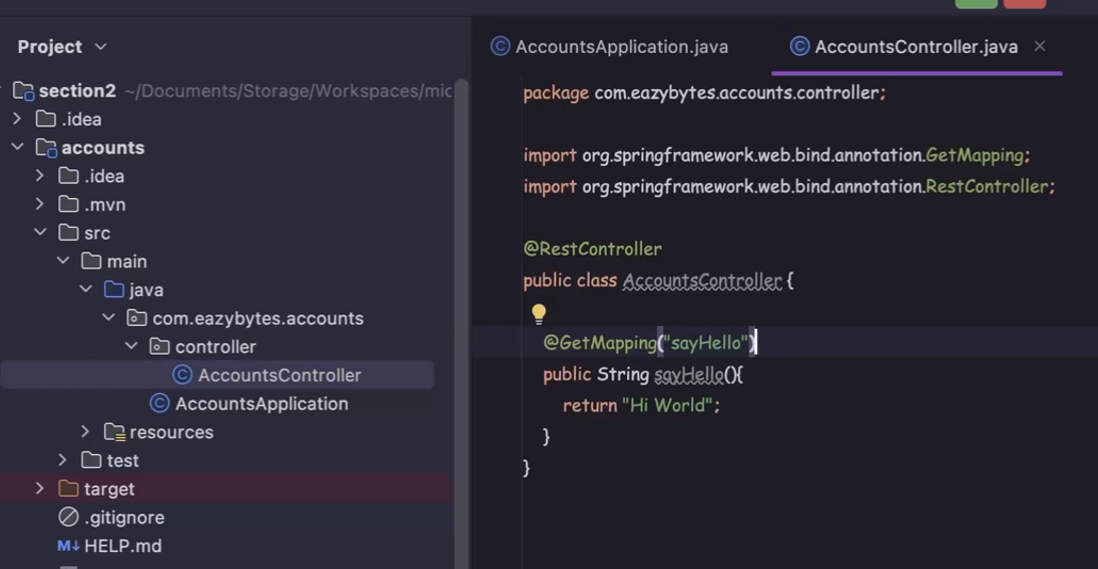
Creating a first microservice using springboot and rest api

Creating a service is a similar to creating any springboot web application it contains a set of rest api services that is consumed by other services or external client. So lets start creating spring boot app using spring.io and necessary dependency as below

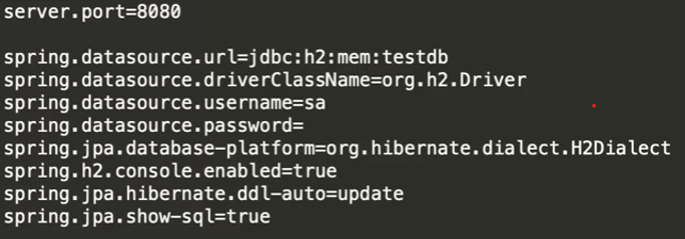
Add the following dependency

* Web
* Devtools
* Actuator
* H2 db
* Data jpa
* Validation
* Lombak

Create the hello world rest api as below for testing purpose whether to check application is working or not

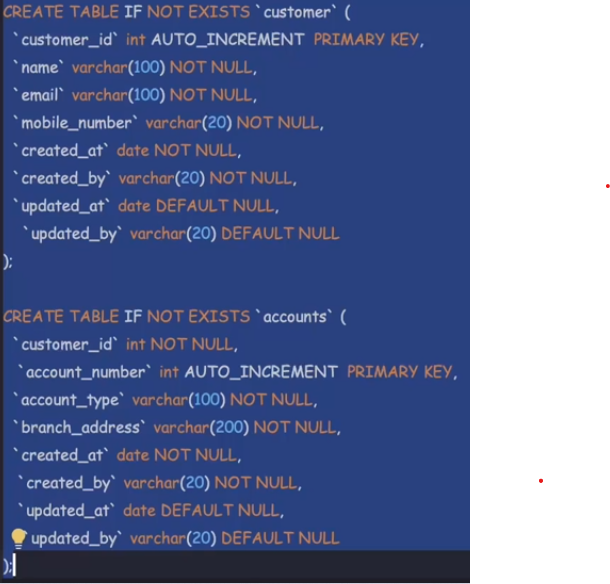


We need to specify data base related properties in application.prop file as below



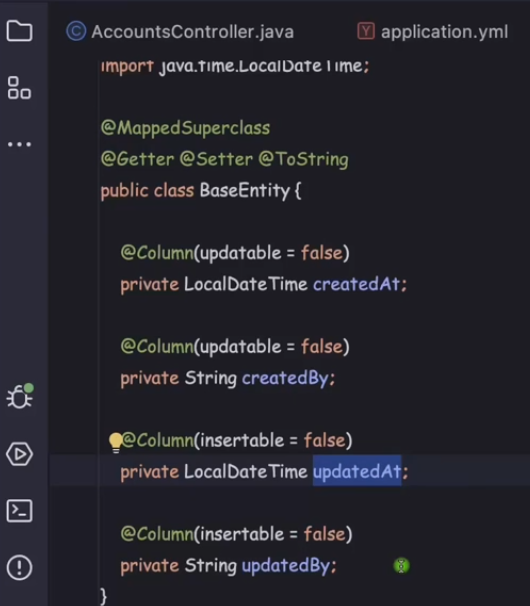
Now a days yml is being used to define properties which is based on indentation hence its very important to properly format the properties as below.



To create tables in H2 DB create schema.sql spring boot will excute from sql files 

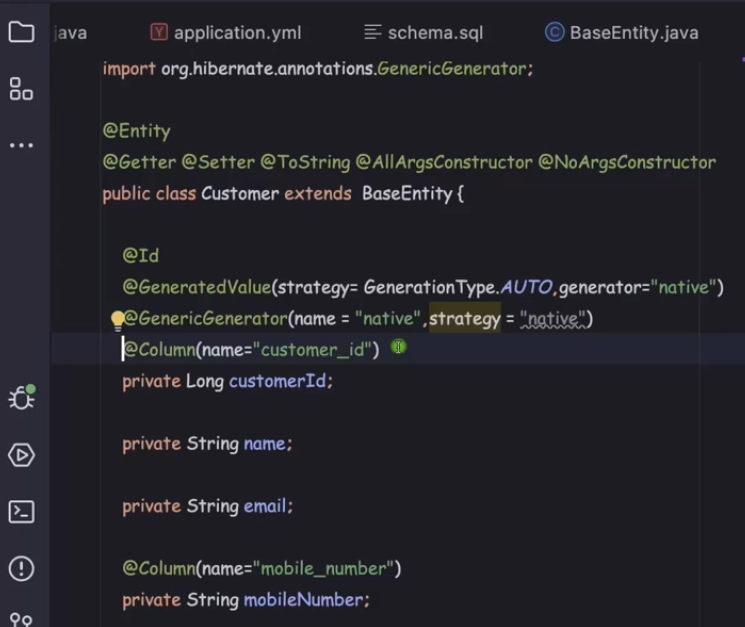
Lets start creating java classes that communicate with DB using hibernate. To do this will create entities classes

BaseEntity serves as superclass for customer and accounts entity as it contains common fields hence mapped with @MappedSuperclass annotation and other annotations to fields

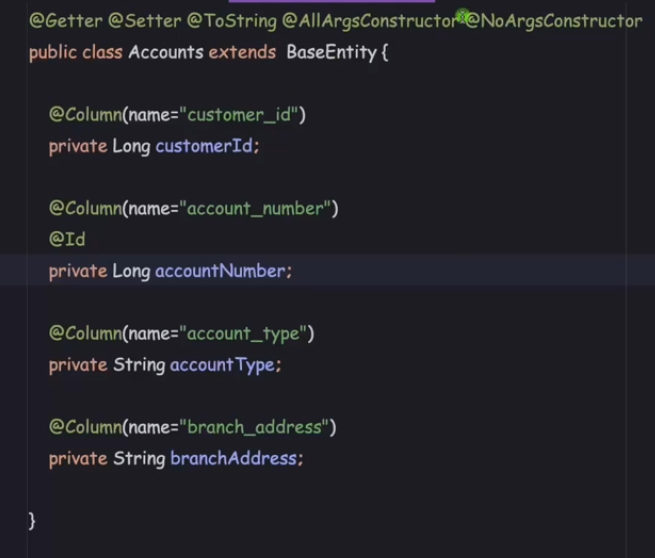


We don’t have to provide getters and supports as lombak will do for us

Now lets create customer and accounts entity

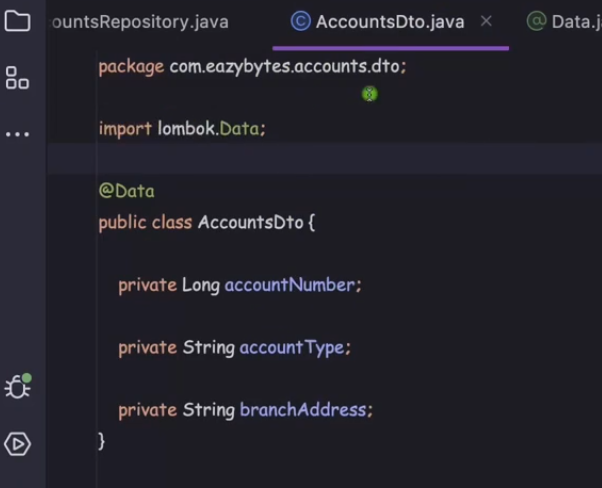


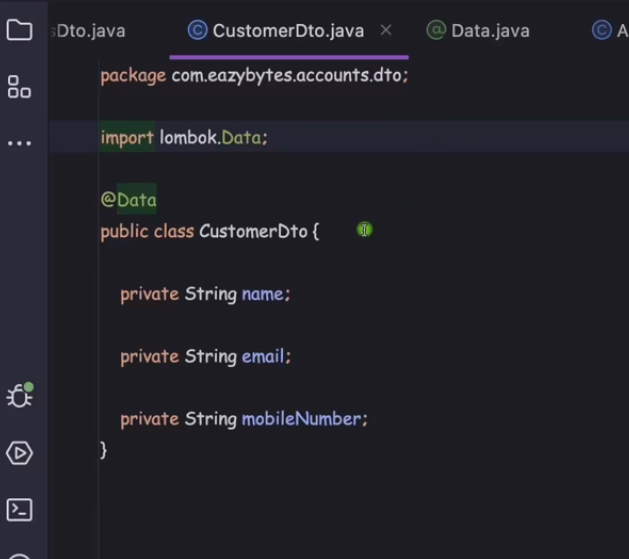
@Column is optional if it matches with db column name but its good practice to add @Column

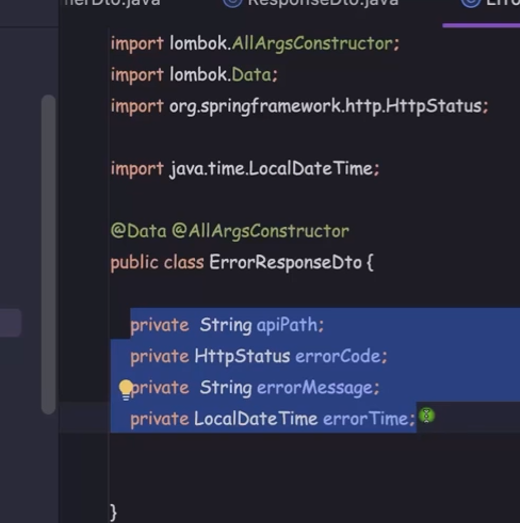
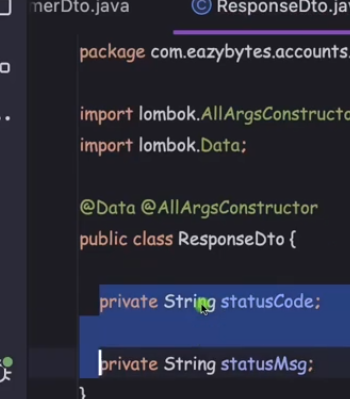


Lets create repositories Accounts and Customer repositories

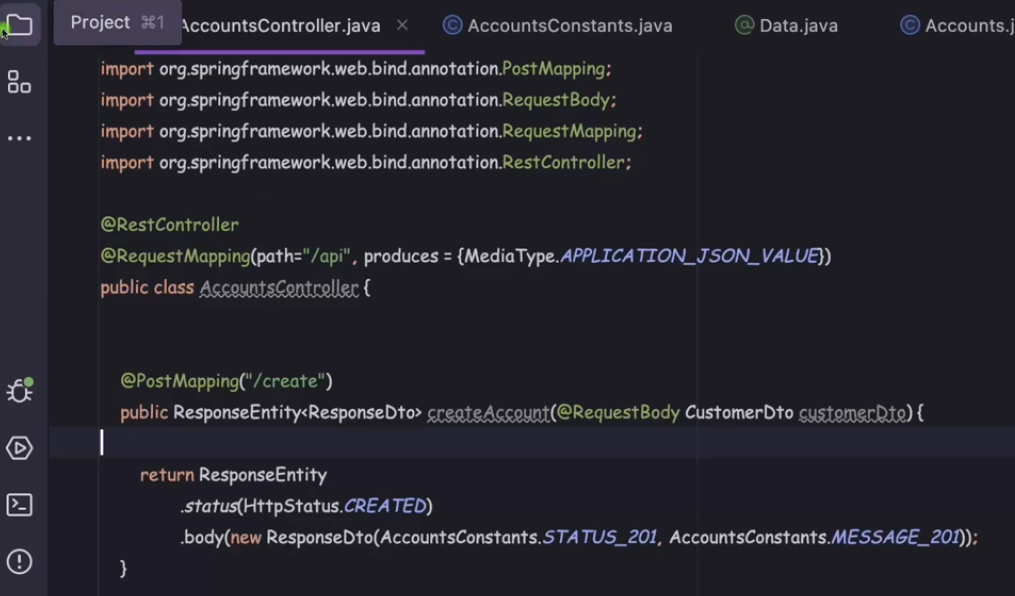
But before that will create DTO which is required to send response to Client.







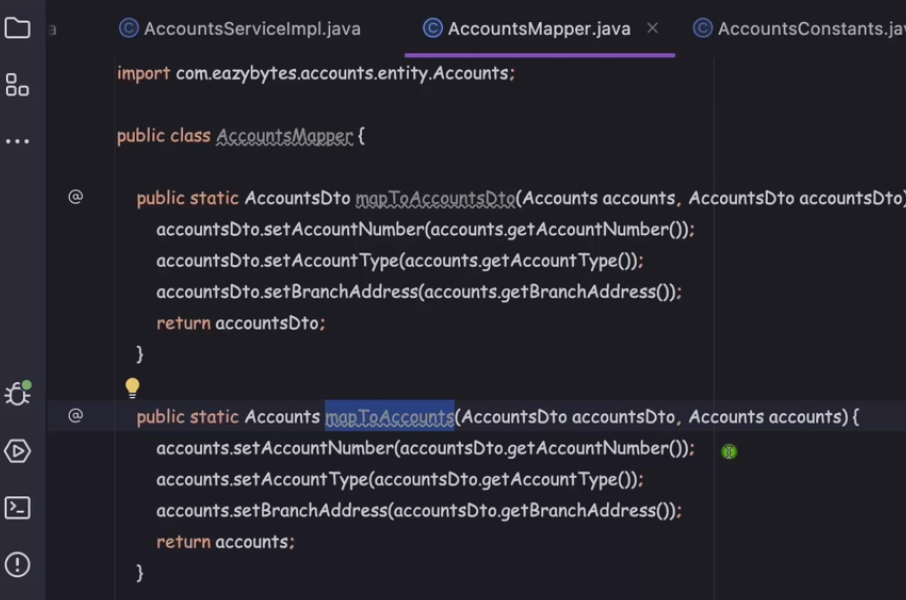
Now lets create restcontrollers

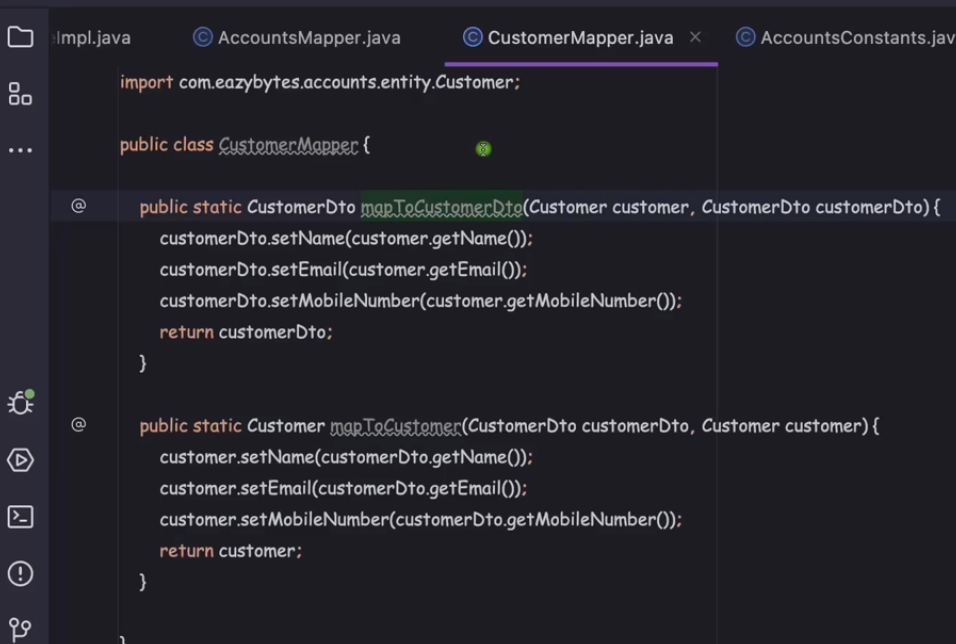


Here accounts constants is constant file pls chk on github

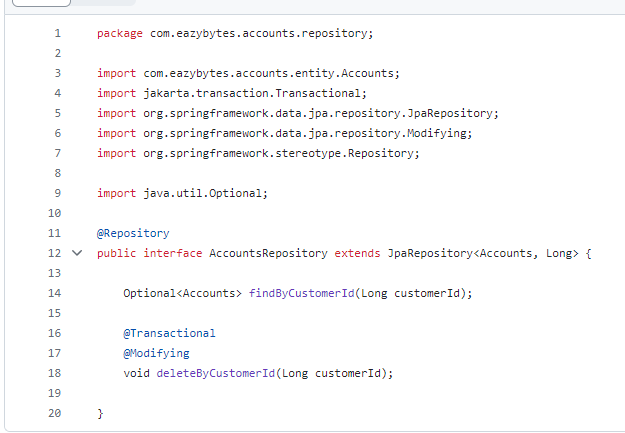
Lets create service class

Accounts mapper and customer mapper is used to convert from DTO to Entity



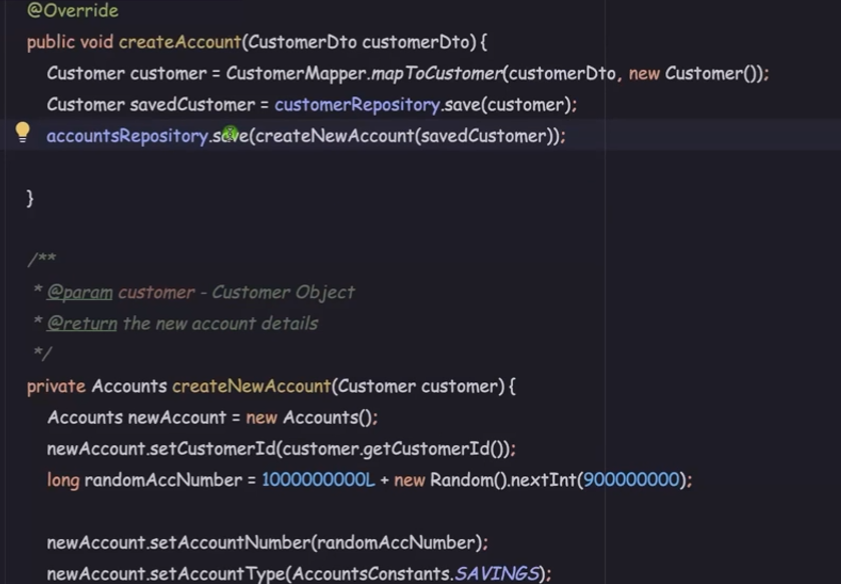


Accounts repository

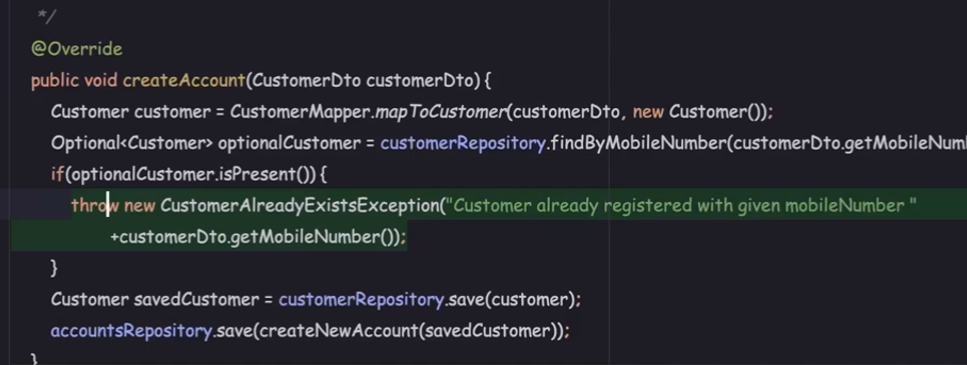


Customer repository

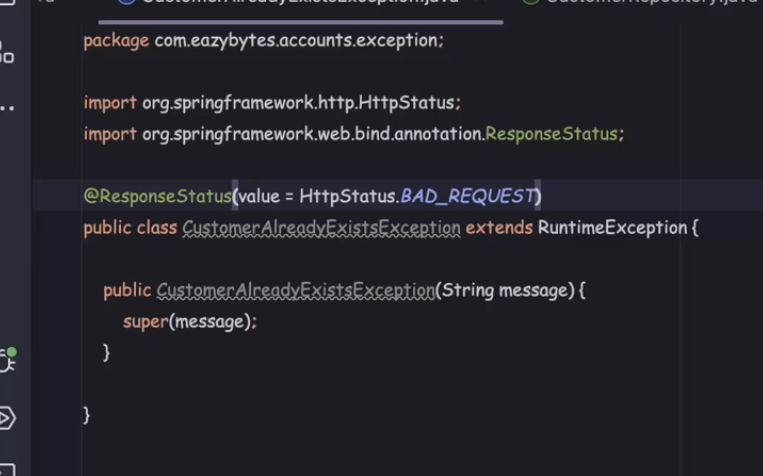
In service create a method that creates and saves customer and accounts details



Lets create exception handler class



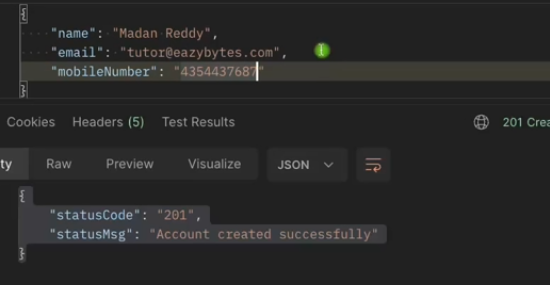
Below is the custom exception

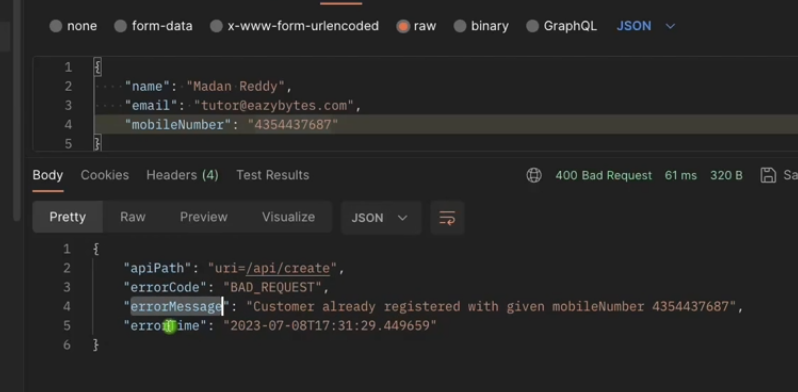


We can handle global exceptions by creating class with annotation @ControllerAdvice



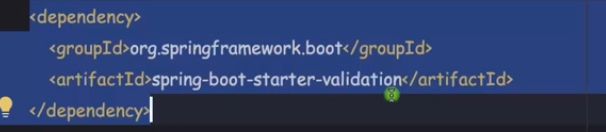
Now will call the create API using post

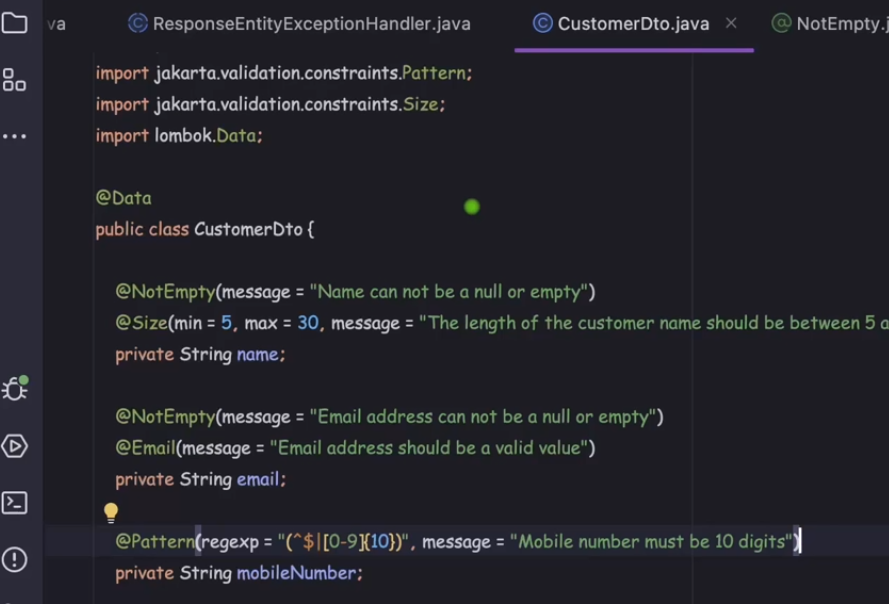




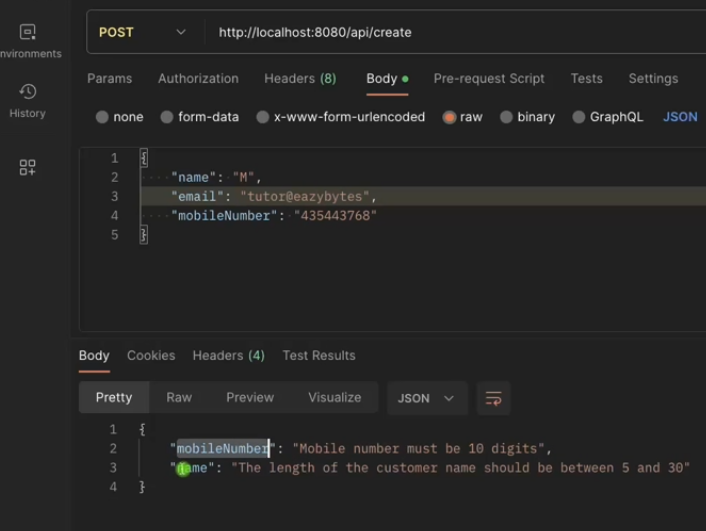
This above code is only for reference pls chk github

Now for validating inputs values we can do it by annotations easily for that we need add dependency statrter-validation



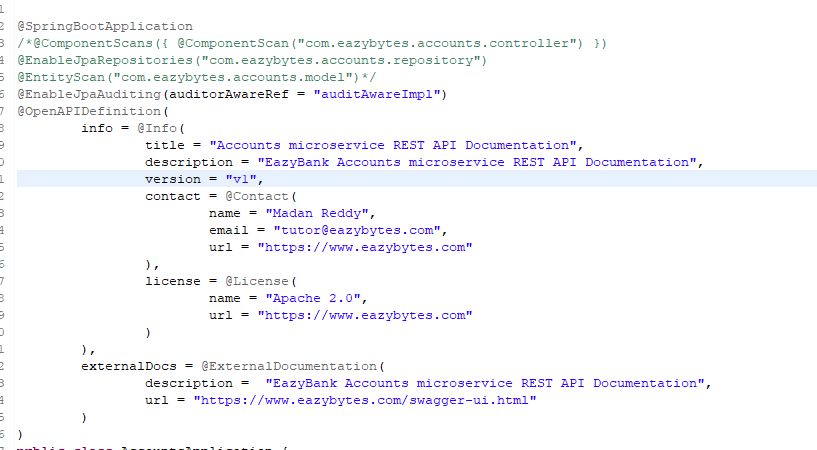


Similarly you can add other annotations for email validation and many are available

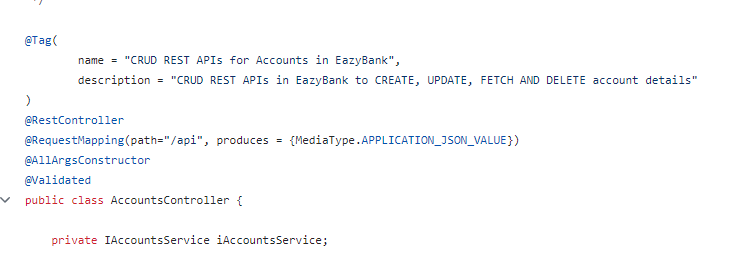


We can also use swagger for API documentation

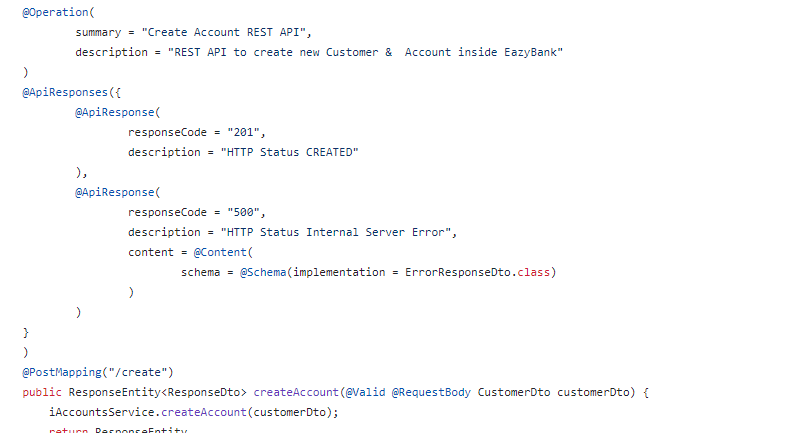
Which is used to provide information about the API to the end users. Swagger API also contains various annotations to display such information few are as below



At the application level



@Tag is applied at the class level



@operation and @ApiResponses is applied at method level that is end points

