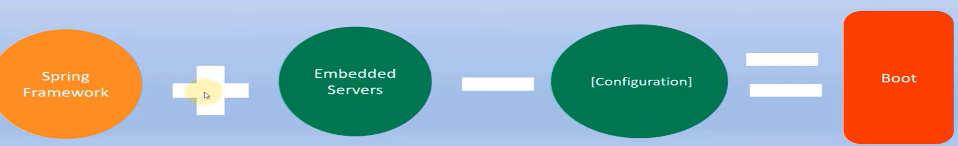
Spring Boot

1. Spring boot is an module of the spring from which we speed up the development
2. Spring Boot makes it easy to create stand-alone, production-grade Spring based application that you can just run
3. It provides an easier and faster way to set up, configure, and run both simple and web-based applications



1. Spring boot follows Convention over configuration software design style
2. Spring boot will scan the class path and find the dependency it will automatically configure the things

🡪Eg. If we added the mysql connector in the dependencies than spring boot will scan it and --will automatically configure all the things related to it and we have to simply use it

1. Spring boot have embedded server, So we do not require to configure the tomcat, And absolutely no code generation and no requirement for XML configuration, And will automatically configure the Spring and 3rd party library whenever possible
2. The All the work starts form the starter POM(Project Object Model), So the here the maven is been used to make the dependency download and configure by the spring
3. The first dependency to keep in the pom.xml is of spring-boot-starter-web
4. And if we want to add the jpa than we have to add the spring-boot-starter-data-jpa, As per our requirement we will add the tags in the xml file and than the spring will do its work of configuring and maven will do work of the downloading of dependencies
5. When we add starter jars then it pull all the jars of the auto-configuration
6. And the new dependencies are downloaded and controlled by the auto-configutration’s jar, Specifically the folder init named META-INF/spring.factories and init all the configurations come by default and this classes are been made to start and stop according to the conditions provided in the inbuilt code
7. There are four ways to start with the spring boot:
   1. Create a maven project and add starter dependencies
   2. Using spring initializr **(Recommended)**

🡪In it go to its website add the details, download zip file extract it and open in the IDE, project is ready to go and work

* 1. Use IDE like STS **(Recommended)**
  2. Spring boot command line interface

1. Method of creating the spring project using spring intializr
   1. First go to the official website of the spring intializr which is start.spring.io
   2. Than an user interface will open, In it select Maven project, java, and the stable version of the spring will be selected by default, So do not make change in it
   3. Than we have to give the group name, artifact name, description, and package name, Than we will choose the packaging as the jar file and the version of java according to the our available java in our system
   4. And after generating the project, We have to add the dependencies on the right side of the page by clicking on add dependencies
   5. We will choose them according to our project requirement initially we will choose the spring web, and than as per our requirement there are many dependencies like security, jpa, mysql, etc
   6. And than click on generate to create the project
   7. Now we will extract that zip file where we want our project and we will get the folder of our project
   8. And than we will open the folder which is having the pom.xml in it in the eclipse or any other IDE where we want to use it(STS IDE is recommended to work with spring boot)
   9. For eclipse we will first open the eclipse than we will go to the file🡪import🡪maven🡪Existing Maven Project🡪Browse the project’s folder which is having the pom.xml file in it🡪Finish🡪Wait till the dependencies are downloaded and make sure internet is on
   10. And than the project would be imported to you workspace And you can make changes in it
2. Method of creating spring project using Spring Spring Tool Suite
   1. Download Spring Tool Suite (STS) from internet
   2. Open STS
   3. Goto File🡪New🡪Spring Starter Project🡪Fill the details required for making the spring project(Make sure you select maven and system version of java)
   4. Than add the dependencies you want to add in your project by searching, For making an web application make sure you include spring web dependency and any other if required like JPA, mysql, etc
   5. And than click on finish and wait until the dependencies are downloaded, And than your project is ready to go for work
3. Introduction to the set up in the spring project
   1. Now the main file in the src/main/java and than in your project there will be your main file and this is starting point of your application
   2. Make sure you write your code in this package only not outside this package
   3. And than there is another important file which is by name **application.properties** in it if we want to make any change in the automatic configuration made by the spring than we will do it there, And this file is present in the **src/main/resources**
   4. And the next main file in the project is pom.xml, In which if we want to add any new dependencies than we will add them in it and than maven and spring will automatically download and configure them
   5. And the spring project would by default embedded by tomcat server as we have selected spring web in the dependencies while making it
   6. We can simply run our project from the our main file which is in the package that we have maded, and the name of that file will be in the format like:

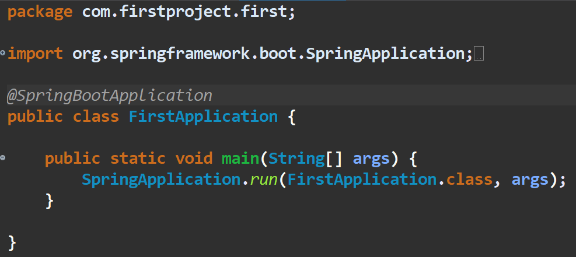
<name-of-project>Application.java

* 1. And for making the application to run just make right click and click on run as application as there is inbuilt tomcat, We do not have to make it run on the server, Server will automatically start while running application
  2. And make sure if you use the dependencies like jpa, mysql than they will give error if they are not been configured, Dependencies like them should required some level of configuration, If they are not required for now then make them comment out, So that they will not require to configure for now
  3. Now lets, make some testing if the spring app is working or not
  4. If we want to change the port on which our app is working than we have to go to the application.properties file that we have mentioned above which is in folder named src/main/resource
  5. And in it we will write the following statement to change the port: server.port=5000

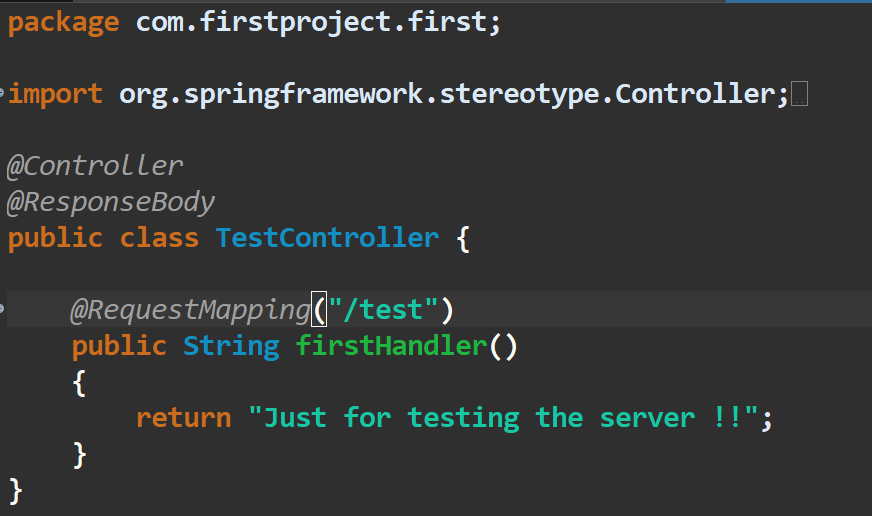
🡪Here the port number can be maked according to our convivence

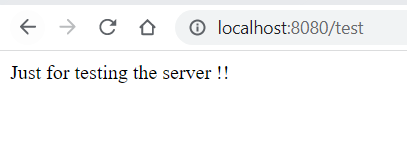
* 1. And while running if any alert comes than make it allow
  2. Now our application is working, We can now make an controller to control our server, We make make controller in the main file also or in the new class also
  3. For example let us make an class named TestController.java and in it we have written following code (Make sure you write code in your main package otherwise spring will unable to scan new class):

🡺Main Class

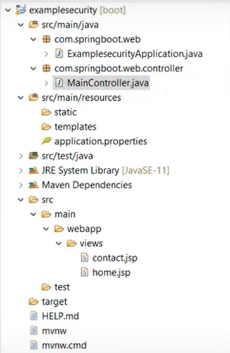


🡺Controller Class that we made



🡺Result we got(running on port:8080 bydefault) 

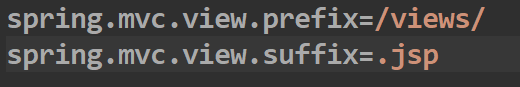
1. Let’s Make an server in which the pages will be changed according to the url we give and in this project we will also use the jsp page to be viewed when an particular url is given:
   1. We can make this possible by help of them-leaf template but we will use the simple method without it
   2. First we have to make an folder in which we will keep all our jsp files
   3. And this folder is to be made in the src/main folder for now lets make an folder named webapp and in it folder named views in which we will make our jsp files which is to be viewed, Example Structure is as follow: (Not of this project but same)



* 1. Now lets make an controller file which will control our requests in the url, And for now we will support to the two url’s home and contact, For which we will write code in the MainController class

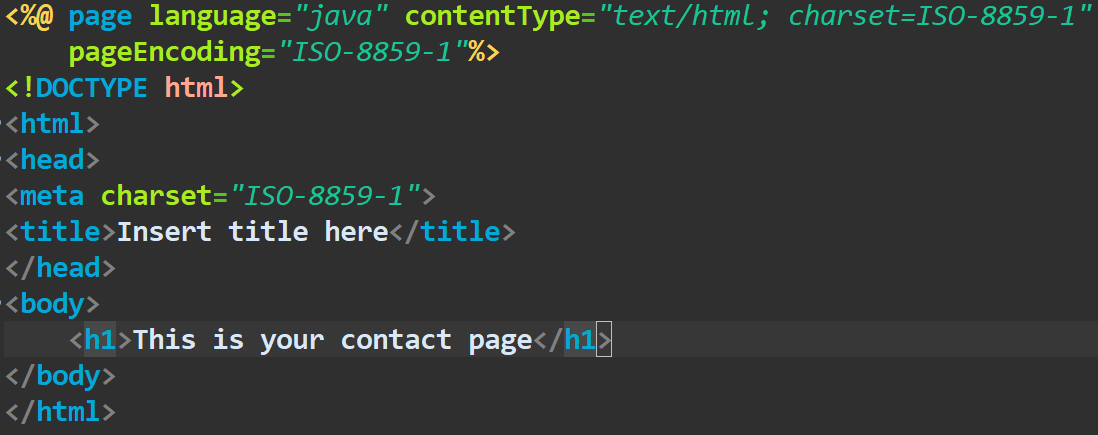


* 1. And as we have return and string from requestMapping and we have to configure its prefix and suffix to make an page to be load, We will work on it in the application.properties file as we are in spring boot and no xml is to be made
  2. In application.properties we will configure following things as followed:



🡪Here the /views is the folder we maked in which we putted .jsp files, If we make the views folder name to be changed than we also have to change it here

* 1. Now as we have made changes, But when we return the jsp file to tomcat than tomcat does not have the complier to make it run, So that we have to make download an compiler’s dependency which will help tomcat to compile jsp files, And for it goto the mvn repository and find the dependency named tomcat embed jasper and put its link in our pom.xml
  2. And than check writing the **localhost:<port-num>/** & **localhost:<port-num>/contact** in the browser and the result would be seen
  3. And we can write any thing in the jsp file, But we have just maked demo of how mechanism works, An Sample jsp file is also as followed:



* 1. **We do not have to muck up the properties** that is to be changed in the application.properties file, **We have to just go to internet and search for that thing** and we will get the property related to it and we have to use it only
  2. If we want to find the property directly than we can also find it by help of the writing keyword of that property and than selecting from the suggestion list

1. Using JPA(Java Persistence Api) in Spring Boot
   1. JPA is an ORM(Object Relational Mapping) tool
   2. ORM helps us to do work with the database via pre-defined method and for which we do not need to write the sql statements
   3. ORM helps us to do work with the table stored in the database, For which we have to make an class, Which have the properties same and in order as that of stored in the table and by help of the class that we made, We can make changes in the database’s table
   4. And process of CRUD made via an class to database is called as the mapping
   5. JPA is an specification(Steps) that is given by oracle to save our object/data in the sql database, But this is only specification but the implementation work is not done in JPA
   6. And for implementation of JPA there is hibernate
   7. JPA provides us two interfaces that are:
      1. EntityManagerFactory
      2. EntityManager

🡪EntityManagerFactory provides us EntityManager and entity manager provides us the method like create, Read, Update, Delete i.e.(CRUD)

* 1. To use the JPA in the spring to make all the things automate, We use dependency named:

**spring-boot-starter-data-jpa**

* 1. So now we will make an Class(DAO) in which we will perform operations related to table of same configuration at the user level, Than at the database level we will make an userRepository and we will extend this repository by the CRUDRepository which is an interface and CRUDRepository will provide us all functionality to perform all database operations
  2. Now instead of the CRUDRepository we can also use the JPARepository, But JPARepository provides us some extra functionality as compared to the CRUDRepository
  3. **And for making an repository we will make it an interface as it have to extends more than one classes or interfaces**

🡺Practical Of Creating New Entity and Repository

1. For now we will make an standalone application via jpa and mysql and if we want to make an web project than there will be no change in the usage of the jpa and mysql, So steps are same
2. And for making an project which connects to the mysql database, we will use the following two dependencoes
   1. MYSql Driver
   2. Spring Data JPA

🡪Make sure that system have working internet, Otherwise the dependencies would not download

1. \*Solution Of Error: If the error comes like this:  , Than there is an error in the system that the **jre of spring is different than the system’s jre version**, And to make it solve we can change the jre used in the system by following the below steps:

**Right click on project 🡪 Properties🡪 Java build path 🡪 Libraries 🡪 Remove jre 🡪 class path 🡪 Add library 🡪 Jre system library 🡪 Next 🡪 Alternate Jre 🡪 Installed jre 🡪 Add 🡪 Directory 🡪 Browse from system 🡪 Select 🡪 Finish 🡪 Apply And Close 🡪 Finish 🡪 Done**

1. As we want to work with the JPA, we will have to make an entity(Model)(DAO Class) for of which object do we want save, do want will update and delete

🡪For example if there is an book than it’s model would contain author, price, topic and pages

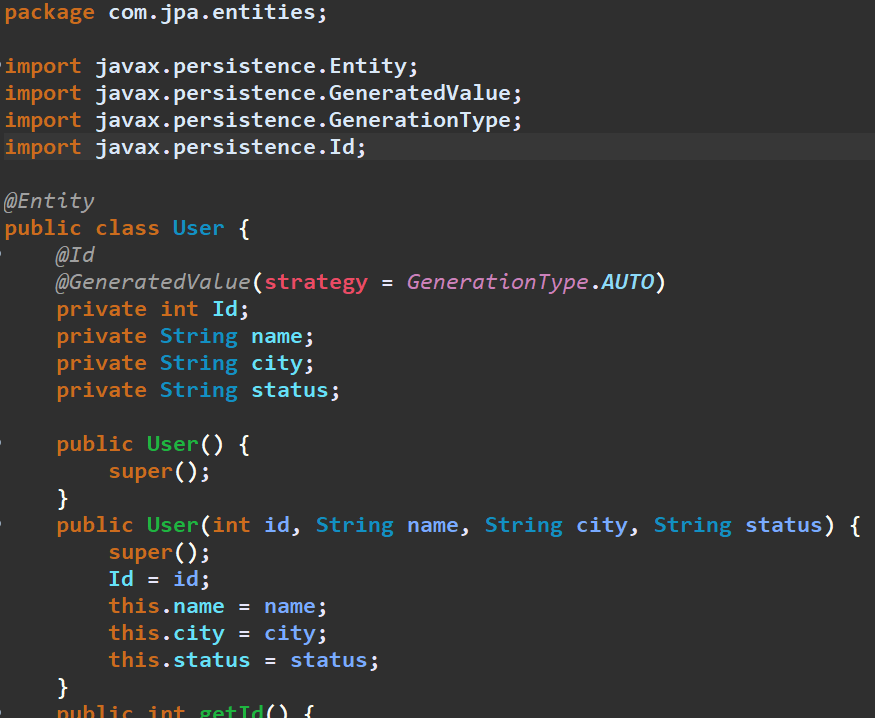
1. And while making we would be not caring about the dependencies like ORM, hibernate, etc which would automatically downloaded as they are included in the spring jpa starter
2. Now lets make an entity which is an model, And by help of which we will manage data
   1. First in the parent package we will make and package by name <package-name>.entities and in it we will store entities of our project, But make sire that the new package is included in the original package otherwise the spring will be unable to store it
   2. Now lets make an entity, For example we will make an entity by the name User and the entity container the properties of the data that are to be stored, getter-setters, constructor(Both), toString method
   3. And we will also use the @Entity annotation on the top of the class name, as the class is an entity
   4. And on the top of the id we will use the @Id annotation as it is an id, and we want to make this id to be auto-incremented, So we will use the another annotation which is as followed:

**@GeneratedValue(strategy = GenerationType.AUTO)**

🡪And if we want to set name of the column than we have to use:

**@Column(name=”<name-you-want>”)**

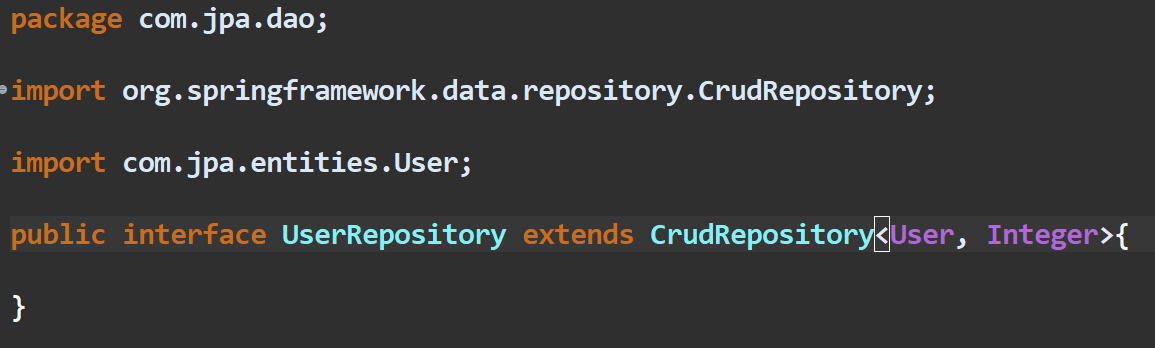
* 1. Example of the entity is as follow:



* 1. Now we will make **Repository** for our entity which is an **interface** that will work at the database level, As we have completed the user level work in the upper step, In this database level the interface we made will extend the CRUDRepository OR JPARespository, And its name will be in format like **<Name-of-entity >Repository.java**, And it will take two arguments the first is entity that we made and second is Type of Id in that entity

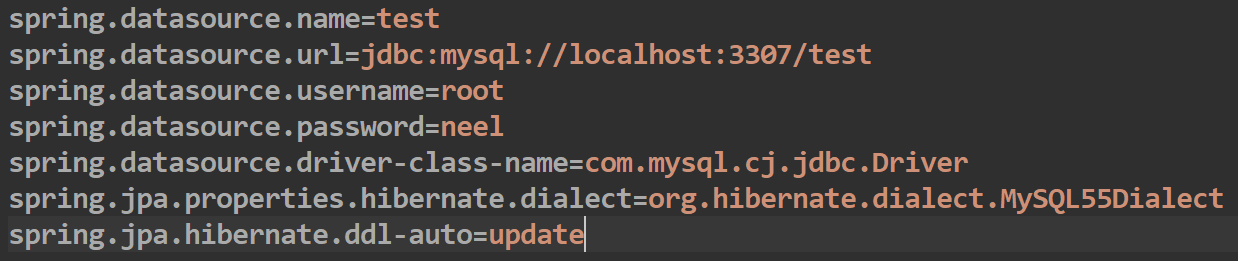
🡪By doing this thigs we will get the basic functionalities to handle database, If we want extra functionalities than we will discus about it later

🡪For example:



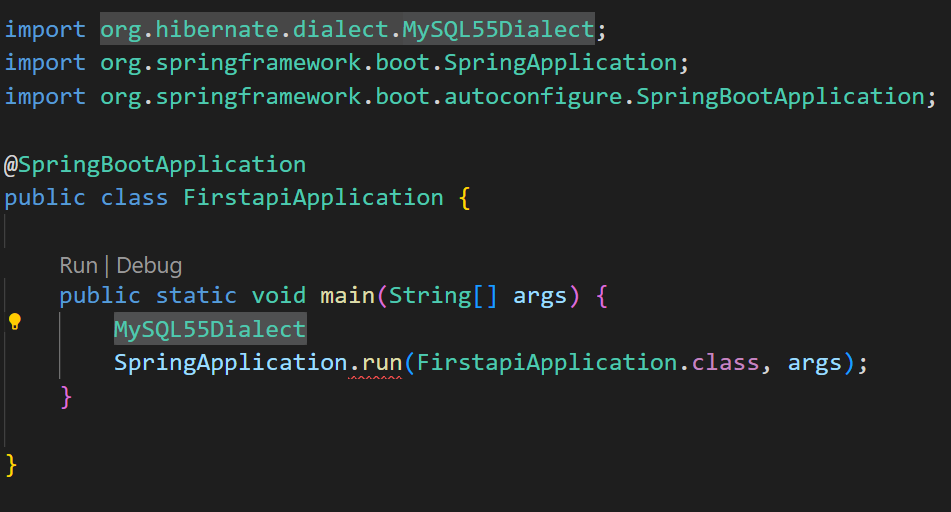
* 1. Now next step is to configure our database in the application.properties file, For it we will have to write the following code:

Database Name



Data Source Name, Give unique name

🡪For getting the class of the mysql dialect in the system in which you are using we have to first go to any file in the project with the .java extension and there we have to write mysql and the option will come with **MYSQL<NUMBER>Dialect** , and an import regarding to that class will be also made, we have to just copy that import address from that file and than make it copy at the hibernate.dialect field and our work will be done, Example of any java file for getting dialect import:



🡪And after using import of this file make sure we delete the import and statement we wrote from that file

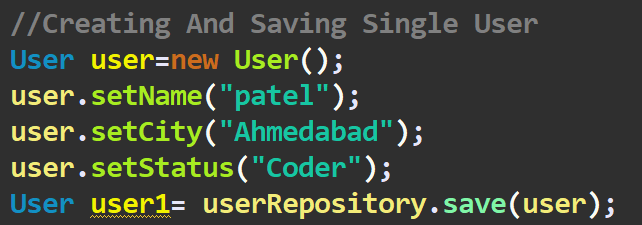
* 1. Now to add data we have to make following change in the main file:

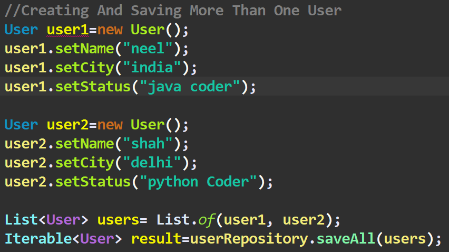


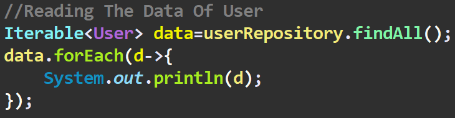
1. If we want to change the name of the table by which we will save information in table than we have to write the following annotation and in it we have to pass the parameter name as shown in the example:

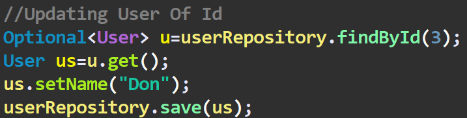
Eg. @Table(name=”<Table-name>”)

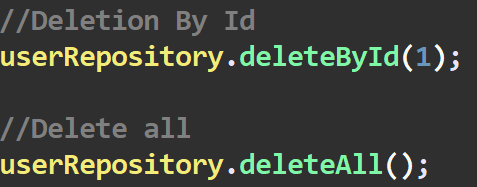
1. Code for crud function in using JPA and Mysql in Spring Boot(As per above example)









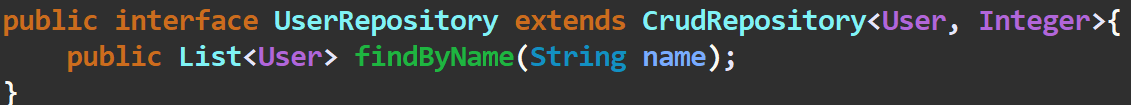


1. Derived Query Methods In Spring Data JPA (Custom Finder Method)

🡪In this method we have to only define an method name and the thing that method will take as argument, and thing that method will return  
🡪**We do not have to write all the code for this method**, We have to just write only name of this method in the **camelCase format** in the repository file that we have made, In this tutorial we have made UserRepository

🡪This finding system works on the **JPQL which is Java Persistence Query Language**

🡪For creating an method we just have to male an public method as shown below:



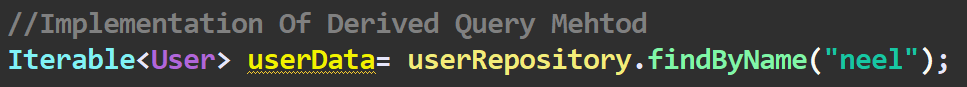
Type Of Primary Id Of Entity

Entity For Which Repository Is Made

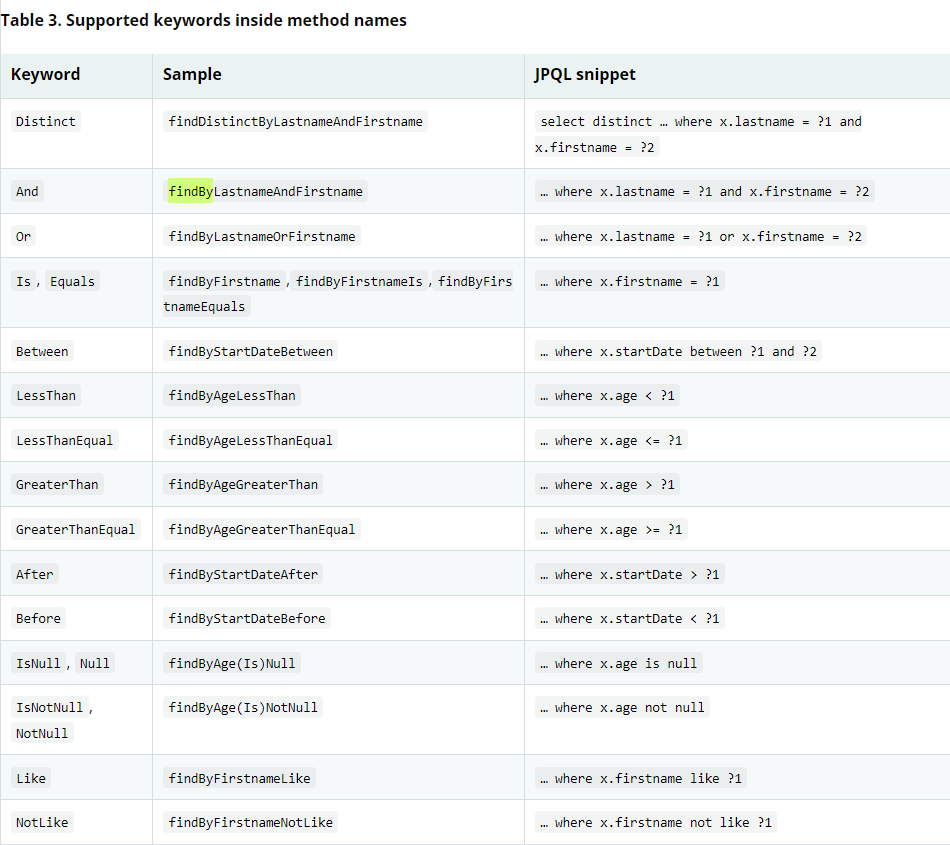
Criteria

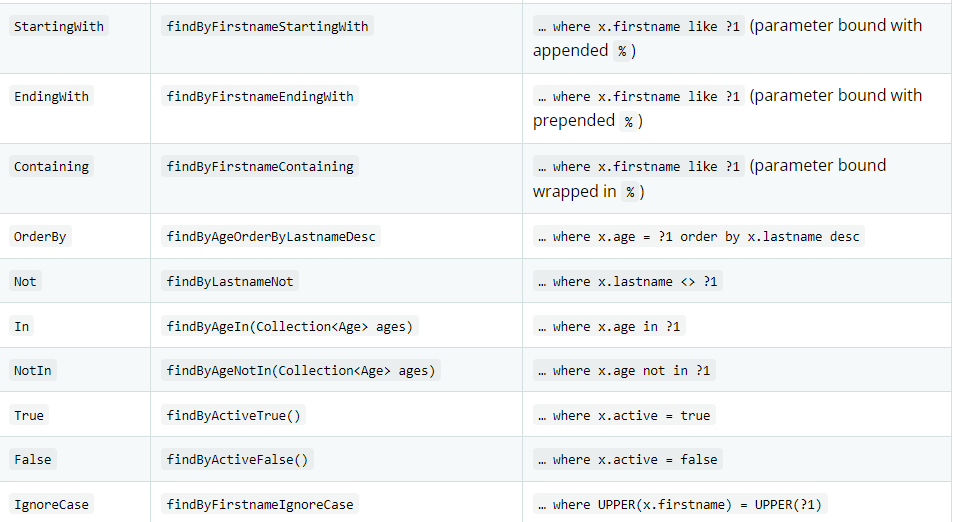
Introducer

🡪And than we can simply implement this method in our main class, And it will also be shown in the recommendation by IDE, Example of implementation in main class is as follow:

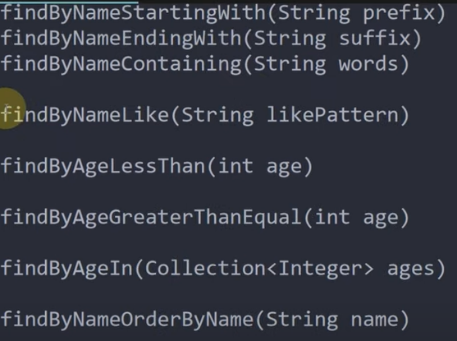


🡪An Important Table in which the keywords OR Rules that we can use while creating the functions is as follow:





🡪Example of another methods that we can also make are:

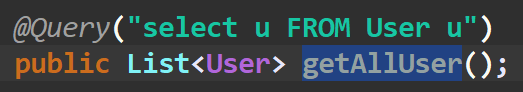


1. Executing JPQL and Native Queries with Spring (Method-Made)

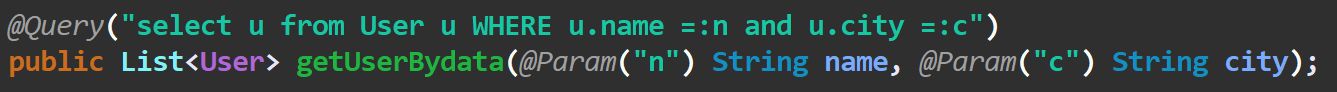
🡪In this type of query we are able to define function’s method of an query that we were unable to do in the upper method’s case

🡪And for it we just have to write an annotation named **@Query** on the top of the method that we have defined, And after that query in rounded bracket we have to define the our query which should in the **JPQL language** which is similar to the HQL language

🡪We have to mention this query in the repository class that we made to connect to database, Same as we added the query in the above method, Example of the custom query in custom language is as follow:



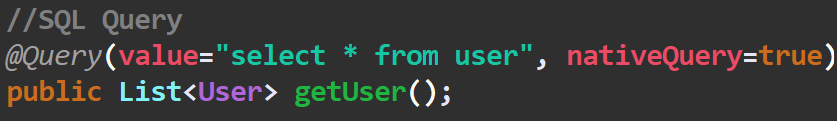
🡪Now we can also pass the parameters in the query by help of the params method which is as shown below:



🡪The process of linking the n and name, c and city is called as binding, And while binding make sure that there is an space before we use special operator **=:** , Otherwise there will be an error in binding

🡪Also make sure there is no space between special operator **=:** and the parameter otherwise it will give binding error

🡪Now we can also make an query using the **SQL language**, whose method is as follow:



🡪Here in this query we have to write the our query which is in sql language in the value field, and have to make the **nativeQuery=true**

1. API(Application Programming Interface) is a set of rules that allow programs to talk to each other. The developer creates the API on the server and allows the client to talk to it
2. REST(Representational State Transfer) is a set of constraints to be used for creating web services

🡪For an API to be an rest api, API must follow the following features:

* 1. Client-Server
  2. Cacheable
  3. Stateless
  4. Layered

🡪Important method for http are:

1. GET: It reads a resource
2. PUT: It updates an existing resource
3. POST: It creates a new resource
4. DELETE: It deletes the resource

* HTTP also defines standard for status code:

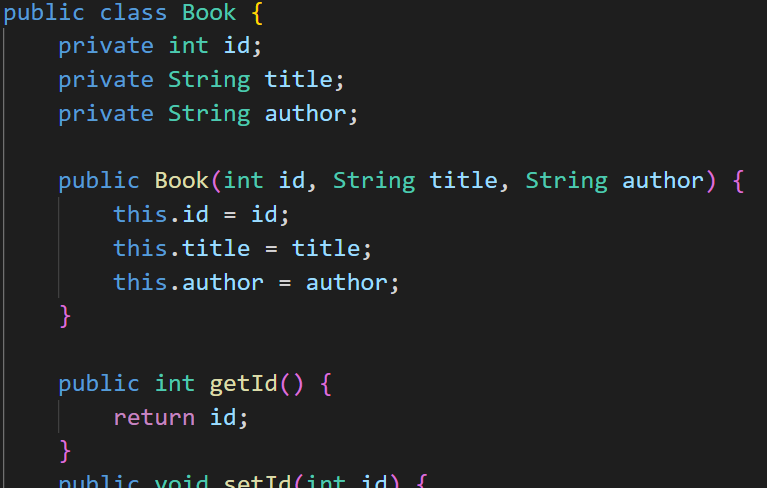
1. 404: RESOURCE NOT FOUND
2. 200: SUCCESS
3. 201: CREATED
4. 401: UNAUTHORIZED
5. 500: SERVER ERROR
6. **Spring Dev Tool**: DevTools stands for developer tool. The aim of the module is to try and improve the development time while working with the spring boot application. Spring boot DevTools pick up the changes and restart the application

🡪For using the dev tool we will select the **Spring Boot DevTools in dependencies** while creating an spring project

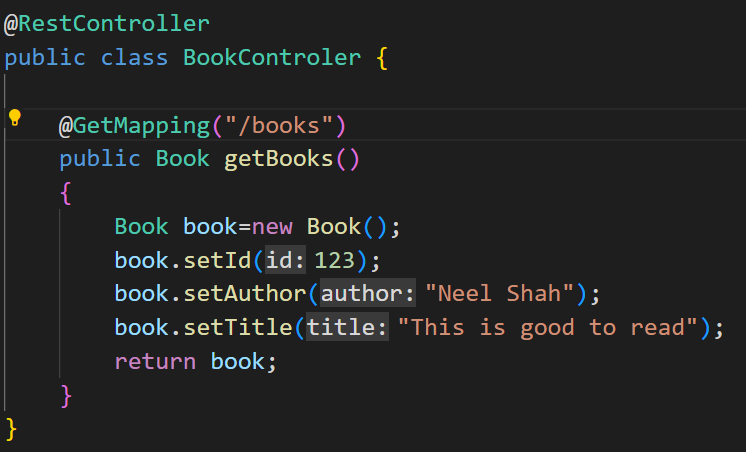
1. Making of an rest api using in spring boot
   1. Make an spring project having the spring web dependency and spring dev tool dependency

🡪We can make new spring project in the vs code by clicking **ctrl+shift+p** to get choice what to make and than we will select an spring maven project by **spring Initializr**

* 1. And than if the vs code asks to open a new tab for a new project than make yes and work in new project tab and close the old tab
  2. We will make an entity(Model) by which we can make the structure of our data, The entity will be maked in the new folder which will be of name entity, Which would be inside the main package, In it we will make an entity class, For Example we will make an book entity
  3. In which we will have the different fields, constructor, getter-setter, toString methods, Example of it is as follow:



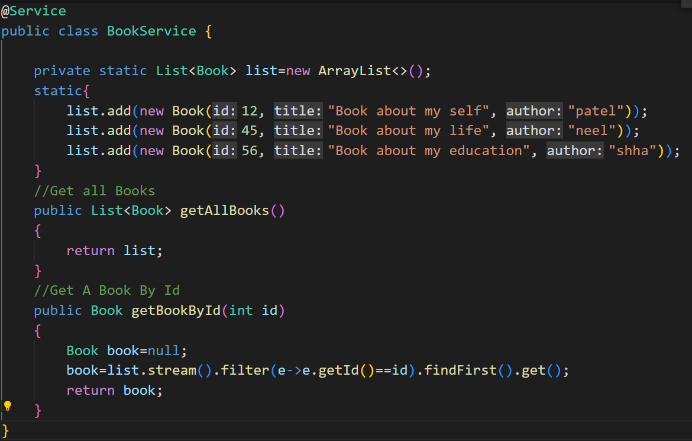
* 1. Now to store controllers of the api we will make new folder by the named controlers in the package in which we are working which is inside src/main/java/com/<Artifact-name>/<Project-Name> and in it we will make a file which will save the controllers, For demo we will make an BookController, For Example:



🡪Here instead of the @Controller, We have used @RestController as we are working with the rest api

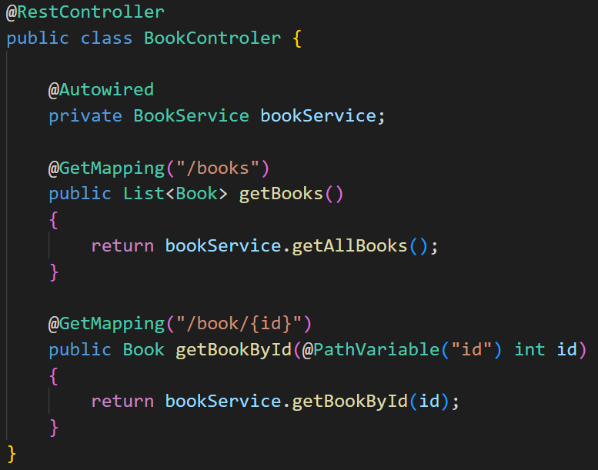
🡪And For mapping we have used the **@GetMapping** as we want to **get the data**, If we want to delete than we would have used @DeleteMapping, Same exists for the put, post

* 1. Now as we are not working with the database, and we want to work with more data we will make an dummy static data in the folder named service, which will provide us fake service, And we will make this folder named service insider our main package in which we are working

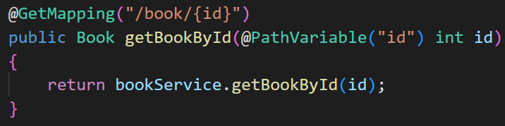


🡪And instead of the @Service annotation, We can also use @Controller annotation

🡪And by help of the @Autowired annotation we can use the method of service class, and than we can use the method of that service class, Example:

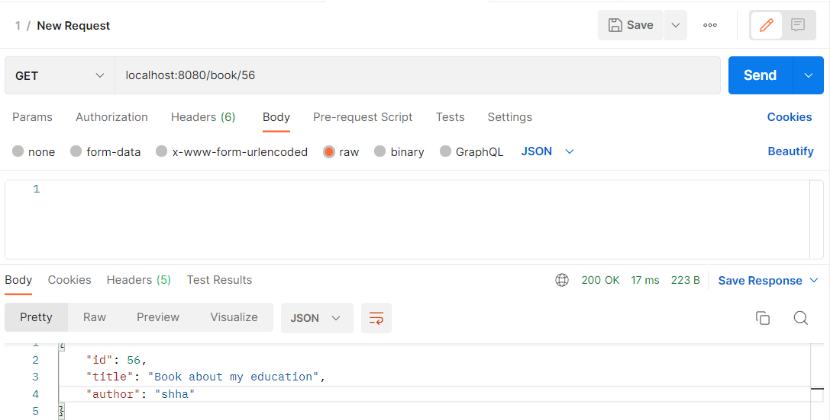
3 

1. We can also retrieve the information from the url, If we write that key in the curly braces as shown in figure:

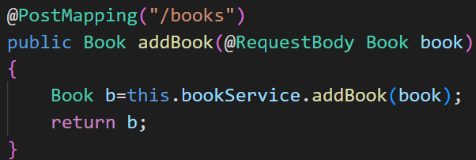


🡪And we have to get the value from the url by help of the @PathVariable annotation and its argument we will write name of variable that we want to get from url as shown in the figure above

1. We can send the data in the url in postman by changing the following settings in the postman:

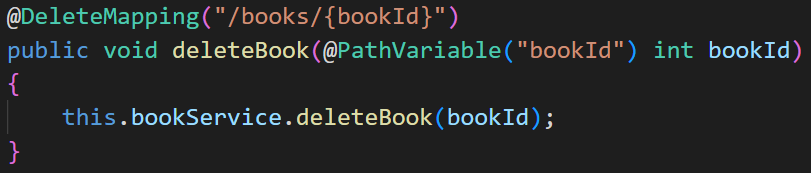


1. **IMPORTANT**: If we have two mapping of same name but having the different method of calling them, Than we can call any one method as the spring searches the method on basis of two **parameters name and method**
2. Now lets see how we can send the data to api via body in json format**(@PostMapping**)
   1. For this method we have to make the addMehtod in the service page OR if we use the database than we have to use the method of database, Which we are not discussing here
   2. We will discuss about how we can get the data from the url request and process it
   3. Now we have to make an method in the controller class for this purpose having the @PostMapping on the top of it
   4. And we can get the body which is in the json format, and make it to be used via an varible, It can be cleared by seeing the code written below:



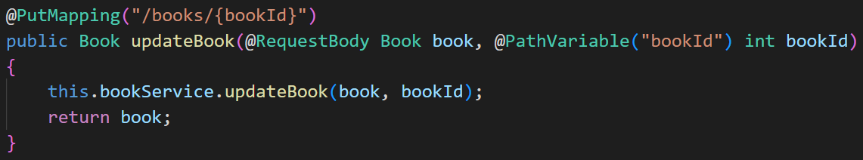
🡪Here we get the data in the Book’s format, So that after the @RequestBody we have to write the class OR entity name in which’s format do we will get the data and than we have to allot it an variable and than we can use that variable in our method as show in figure

1. For deleting, We will use **@DeleteMapping** annotation, as shown below:



🡪And than it will call deleteBook function which is in the bookService

1. For updating, We will use **@PutMapping** annotation, as shown below:



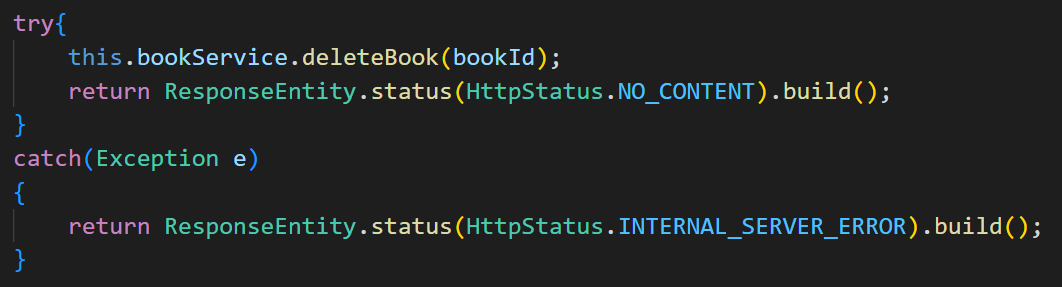
1. For sending the response status code when api is called
   1. For sending the status code in the response we have an class by name response Entity, which is as child class of the HttpStatus
   2. For sending the status in the response we have to make change in the method that is we have to return back the ResponseEntity and it we want send the data with the ResponseEntity than we can change the genric type of the response entity to our required entity or class
   3. Here in this case we have changed the type of response entity as the list of book by writing ResponseEntity<List<Book>>,
   4. We can send the response by writing the following statement as followed:

**ResponseEntity.status(HttpStatus.NOT\_FOUND).build();**

🡪Here the NOT\_FOUND is the type of the code we want to send and there exists so many codes like that to view that see the recommendation given by IDE

🡪Other methods like **NOT\_FOUND** are **INTERNAL\_SERVER\_ERROR, NO\_CONTENT, CREATED**

**🡪**Example:



🡪And if we want to send data in response than we have to write the following code as shown below:

**ResponseEntity.of(Optional.of(list))**

**OR**

**ResponseEntity.ok().body(book);**

🡪Example: 

1. If we want to add any starter dependency of spring after making the project than we have to go to the pom.xml file, And init we have to add the starter dependency by copying its text from mvnRepository OR we can also make the right click in the IDE and than click on Add Starters than find the starter you want to add to existing list, and the make conform, the dependency would be automatically downloaded and configured

🡪For example we have to add JPA dependency, if we want to add directly from IDE than we have to search for **spring data jpa**, And if we want to add the dependency from mvnRepository than we have to search for **spring starter jpa** dependency

🡪If after adding the dependency in the pom.xml if we are not able to use that dependency than,

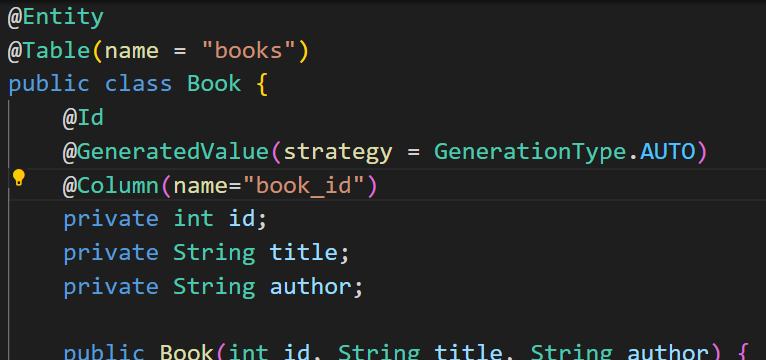
🡪If in eclipse or sts, **go to project make right 🡪 maven 🡪 update project 🡪 select project you are working 🡪 ok 🡪 Dependencies will be downloaded**

🡪If in vs code, **Go to menu 🡪 Go to maven folder 🡪 click on refresh which is on folder tab 🡪 Dependencies will be downloaded**

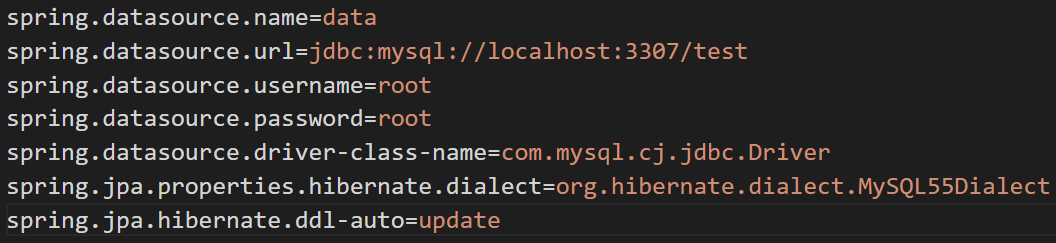
1. Now, We are working on connecting the database to the upper program that we made of books by help of the JPA dependency
   1. For it we will add the dependency of the jpa by following steps as mentioned above
   2. Now we have to make configuration with the database, for it we first have to make an entity(table name) and id(primary key) as we had done in the hibernate, to make spring able to transfer our entity in the table format and work with database
   3. And if we want to change the table name which if we want to make different from the entity name than we have to write @table annotation and in it we have to pass the name as argument
   4. Than we will make the id of book as primary key, by help of @Id annotation, And if we want to make auto increase in the id than we will use the following annotation

**@GeneratedValue(strategy = GenerationType.AUTO)**

* 1. And if we want to change the name of any of our column than we will use the @Column annotation, Example: **@Column(name=”book\_id”)**
  2. Practical example of the above steps is as follow:

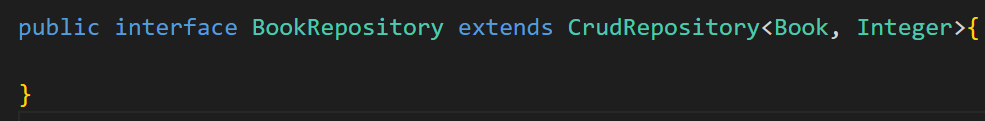


* 1. Now, we will work to make the database to be connected with our program by writing following code in application.properties:



* 1. And we are using the MySQL database, So we have to also include dependency of mysql driver in pom.xml file, And if the mysql driver is not included than make the maven of project to be restarted
  2. Now next step is to built an repository so that by help of which we can manage different functions like CRUD in the database and repository provides us methods for managing it
  3. So, we will make a new package OR folder for storing repository and the folder would be of name DAO and init we will make an repository by name BookRepository and make sure the BookRepository it is an interface as we have to extend many of classes or interface to it
  4. And while making an repository we extend CrudRepository, which will give the by default methods, if we want to make new our own method than we can also make them and access them

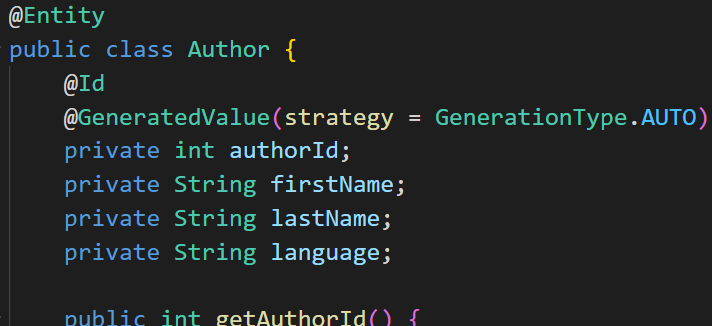
🡪Example:



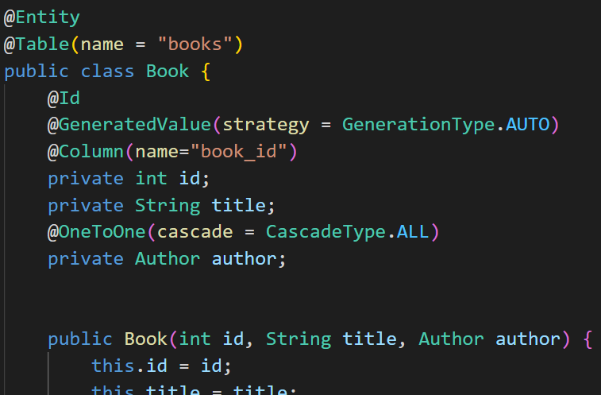
* 1. Now, As we have used the service method in this program, Before we had connected the database, So we will comment out the list and static data in the service method and will make the variable of the bookRepository in it, Example:

**# IMPORTANT: If we want to update any of the thing that exists already and have an id and we want to use the byDefault method only than we have option to use save method, If we send the object with the same id than instead of making new entry in the database, It will save the changes regarding that id in that existing object only i.e. It will perform update function**

1. Now, we will see how we can save an object in an object in database, by oneToOne Mapping, Here we will make an another object named author and will make its reference in our book object
   1. So first we will make an object named author as shown bellow:



* 1. And will also make an OneToOne mapping, of the author as shown below, But the problem arises in this is that we have to first save the id of author, And than we can save the book because the book’s table will store author’s id
  2. Hence in oneToOne mapping we will use the cascade, By help of which the object which are present in the object will be stored first and than the initial or main object will be stored, Due to which database can store id of secondary objects in table of primary object, Example:



1. **Jackson-Bidirectional Relationships(@JsonManagedRefrence, @JsonBackRefrence)**
   1. The @JsonManagedRefrence and @JsonBackRefrence are two annotations which are part of Jackson and are very important for handling bi-directional mapping, Till now we have used oneToOne uni-directional mapping
   2. As till now in our book project we have used OneToOne mapping for the book to author, But now we will use the OneToOne mapping also for the author to book, Also and by doing this we will face an problem of the infinite loop, It is been explained in detail below:
   3. The infinite loop occurs as when we call the book than the book have author and when we call the author than author have book, And this cycle goes on and will make an infinite loop
   4. Till now we have the following code before making any change for the vi-directional mapping
   5. Now we will make the change for bi-directional mapping, And will also add the Book in author having OneToOne mapping, And if we do not use the @JsonManagedRefrence and @JsonBackRefrence than the program will give error instead of infinite loop due to the functionality of spring, And to resolve them we will use the **@JsonManagedRefrence** where the primary entity calls the secondary entity and **@JsonBackRefrence** where secondary entity calls the primary entity, As shown below:

🡪When we use the @JsonBackReference than there is no conversion of the child field into the json and we do not get that data, Hence the infinite loop is been avoided

**IMPORTANT: IN THE RESOURCE FOLDER OF THE SPRING PROJECT, THERE ARE TWO FOLDERS, STATIC AND TEMPLATE, IN TEMPLATE WE HAVE TO KEEP THE HTML FILES AND IN STATIC WE HAVE TO KEEP THE DATA THAT WE WANT TO STORE**

1. Now we will do work on storage of the files by an web application, The storage of the file can be done in two place:
   1. Server
   2. Database

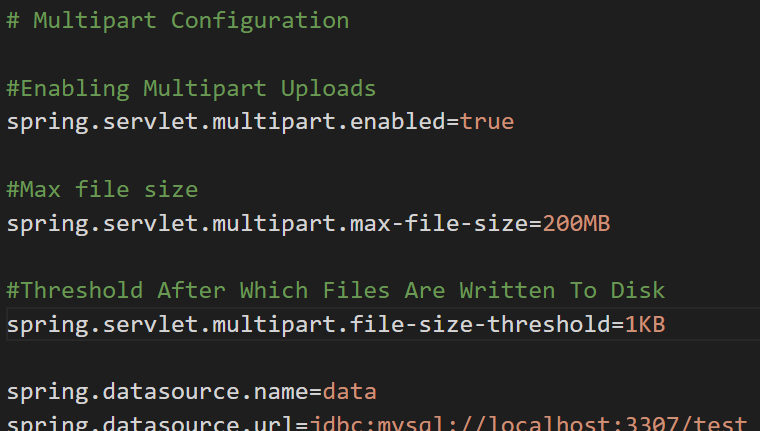
🡪Now we will work on how to upload the file to server, And for it we have to make an folder where we can save the files in server

🡪For storing the files in server, We have an folder named static in the resource, there we can save the files, But in the static folder we have to make an another folder for saving process

🡪For example, We made an folder by name images, and in it we will store files

🡪Than we have to also make some configuration in the application.properties, The configuration made is as shown below

🡪We have to add the following part of code on the top of the code already written in application.properties:



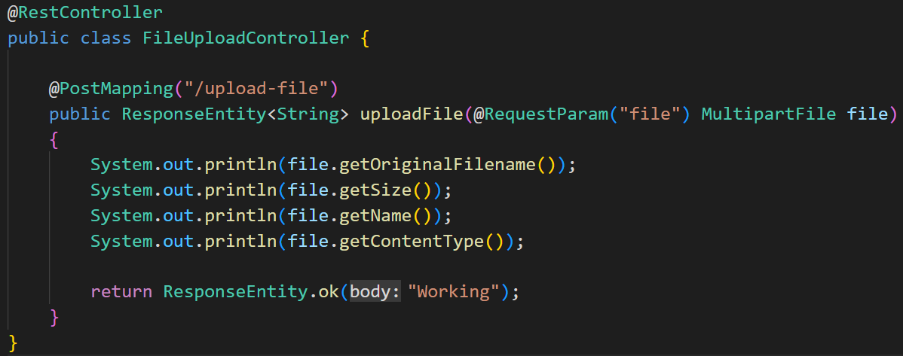
🡪Than we will make new file which will control the file upload by name FileUploadController.java, We can give it any name according to our convince

🡪Than as we are creating the rest api, on the top of the name of controller we will write the annotation by name **@RestController** , And the handler we have to make is as follow:

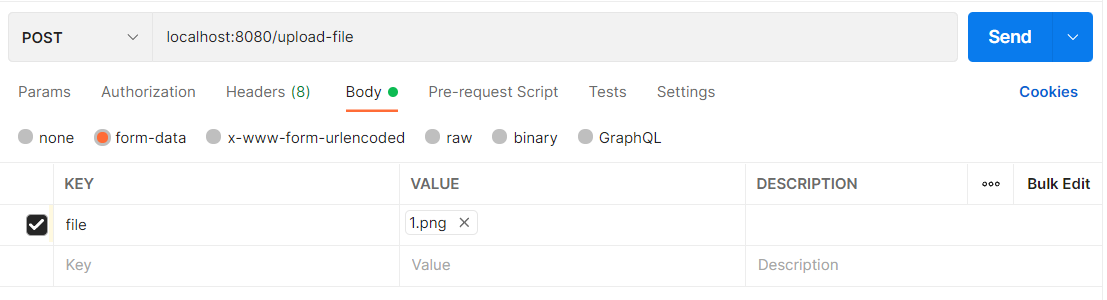
🡪For sending the file via an http we will go to the body in post man or thunder client and init click on form data, than write the value of field and select the file from the dropdown box and then there would be an button by help of which we can upload an file from our system

🡪Now to receive an file in the file controller, And than in argument we will use @RequestParam annotation and in it’s argument we will write the value of field of which’s value we want to take, and than its datatype will come, the datatype we will use is **MultipartFile** and than will assign an variable to it

🡪Now lets try to take out some basic details of the uploaded file, The code for it is as followed:



🡪The Postman which is sending data is also as followed:



🡪And the output is also as followed:



🡪

🡪The flush() function **requests the server to send its currently buffered output to the browser**.

🡪