**Spring Boot**

Spring boot is an open source Java-Based framework used to create stand-alone application. It speeds up the development.

**Advantages:**

* Easy development
* Minimal configuration – automatic configured
* More productivity – less time
* Embedded Servers – No need to configure manually
* Embedded database support

**Disadvantages:**

* Spring boot is best suitable for only scratch development project. It means if we want to merger our new spring boot project with the existing one then it’s difficult and time consuming.

**Creating Spring Boot Project**

We can create Spring boot project in IDE like Eclipse, STS (Spring Tool Suite), or in VS Code as well.

To create project in VS Code we need to install some extensions for that (Spring boot Extension), after that we can create project.

To create packages in VS Code simply we need to create folder and inside folder we can create our java files with the extension (.java).

**Use of application.properties**

We can configure the localhost port and database connection in application.properties file, it located inside resource folder

server.port=8080

spring.datasource.url=jdbc:mysql://localhost:3306/gopaldb

spring.datasource.username=root

spring.datasource.password=password

spring.jpa.show-sql=true

spring.jpa.hibernate.ddl-auto=update

spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL8Dialect

**Spring Data JPA (Java Persistence API)**

Spring Data JPA or JPA stands for Java Persistence API, so before looking into that, we must know about ORM (Object Relation Mapping). So Object relation mapping is simply the process of persisting any java object directly into a database table. Usually, the name of the object being persisted becomes the name of the table, and each field within that object becomes a column. With the table setup set up, each row corresponds to a record in the application. Hibernate is one example of ORM. In short, JPA is the interface while hibernate is the implementation.

The java persistence API provides a specification for persisting, reading, and managing data from your java object to your relational tables in the database. JPA specifies the set of rules and guidelines for developing interfaces that follow standards. Straight to the point: JPA is just guidelines to implement ORM and there is no underlying code for the implementation. Spring Data JPA is part of the spring framework. The goal of spring data repository abstraction is to significantly reduce the amount of boilerplate code required to implement a data access layer for various persistence stores. Spring Data JPA is not a JPA provider, it is a library/framework that adds an extra layer of abstraction on the top of our JPA provider line Hibernate

**API Creation in Spring Boot**

We can create API using Spring boot and use the API in out front end or any android application or where ever we want.

Either we can fetch data from database table and create the API or we can create a class in java and create our dummy APi, to check the API data we need to one application called ‘Postman’ for GET/POST request(We can also use VS Code Postman extension)

To create dummy API using java class we need to create some files inside some folder.

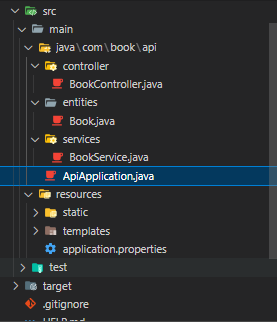
Controller, Entities and Services folder we need to create and inside the folder we need to create java files.

**Services** : It holds the file containing the class which has the data for creating API.

**Entites** : Here we can create the java file which holds the column/field or variable and also holds all the getter and setter methods.

**Controller**: A controller java file is normally used to set mapping( By using the URL we can show our API), here we generally return the object of our class which containing the API data.

Bellow is the folder structure and all the files:



**Book.java(Inside entities folder)**

package com.book.api.entities;

public class Book

{

    private int id;

    private String title;

    private String author;

    public Book(int id, String title, String author) {

        this.id = id;

        this.title = title;

        this.author = author;

    }

    public Book() {

    }

    public int getId() {

        return id;

    }

    public void setId(int id) {

        this.id = id;

    }

    public String getTitle() {

        return title;

    }

    public void setTitle(String title) {

        this.title = title;

    }

    public String getAuthor() {

        return author;

    }

    public void setAuthor(String author) {

        this.author = author;

    }

    @Override

    public String toString() {

        return "Book [id=" + id + ", title=" + title + ", author=" + author + "]";

    }

}

**BookController.java(GET request)**

package com.book.api.controller;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.PathVariable;

import org.springframework.web.bind.annotation.RestController;

import com.book.api.entities.Book;

import com.book.api.services.BookService;

@RestController

public class BookController

{

    @Autowired

    private BookService bookService;

    // @RequestMapping(value = "/books", method = RequestMethod.GET)

    @GetMapping("/books")

    // return "List of new Books"

    public List<Book> getBooks()

    {

        return this.bookService.getAllBooks();

    }

    @GetMapping("/books/{id}")

    public Book getOneBook(@PathVariable("id") int id)

    {

        return this.bookService.getBookById(id);

    }

}

**BookService.java**

package com.book.api.services;

import java.util.ArrayList;

import java.util.List;

import org.springframework.stereotype.Service;

import com.book.api.entities.Book;

@Service

public class BookService

{

    private static List<Book> BList = new ArrayList<>();

    static

    {

        BList.add(new Book(101,"Java Programming","Gopal Sarkar"));

        BList.add(new Book(102,"C Programming","Tanushree Sarkar"));

        BList.add(new Book(103,"Python Programming","Avik Saha Roy"));

        BList.add(new Book(104,"Artificial Intelligence","Sujoy Sarkar"));

    }

    //Get all books

    public List<Book> getAllBooks()

    {

        return BList;

    }

    //Get a prticualr book by ID

    public Book getBookById(int id)

{

        for (Book book : BList) {

            if (book.getId() == id) {

                // Match found, return the book

                return book;

            }

        }

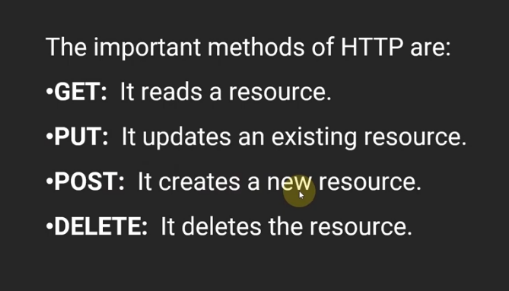
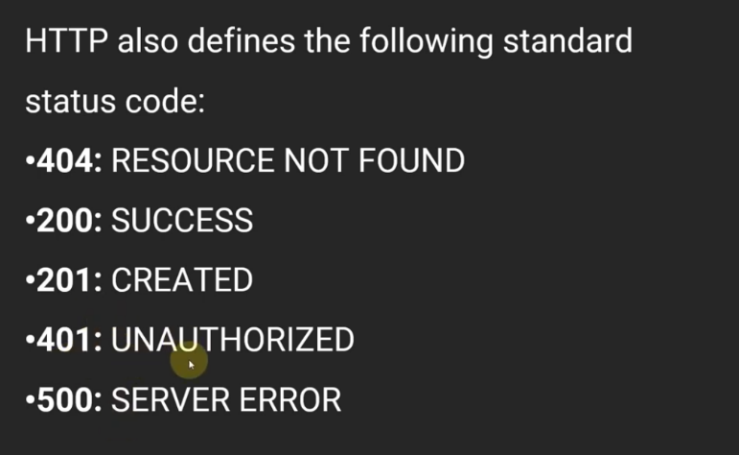
        // If no match is found, return null )

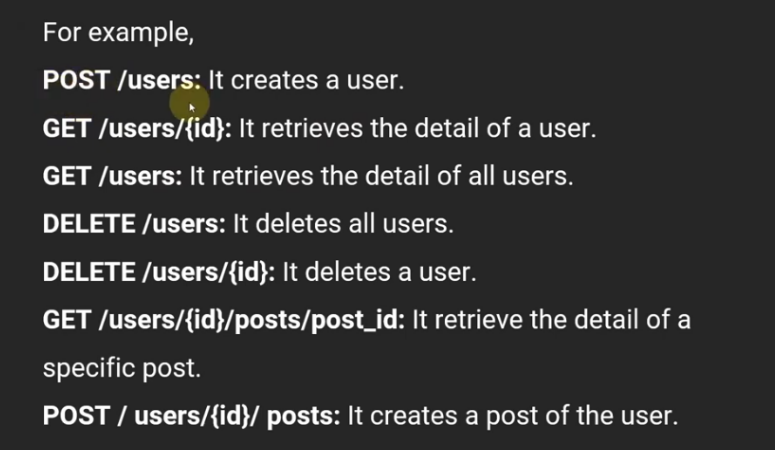
        return null;

    }

}

**Important Points :**

****

****