Spring framework

Is an open source frame work.

- Dependency Injection & Inversion of control.
- Aspect-Oriented programming
- Data access and persistence.
- WEB and RESTful Services.
- Messaging and Integration.
- Testing and debugging.

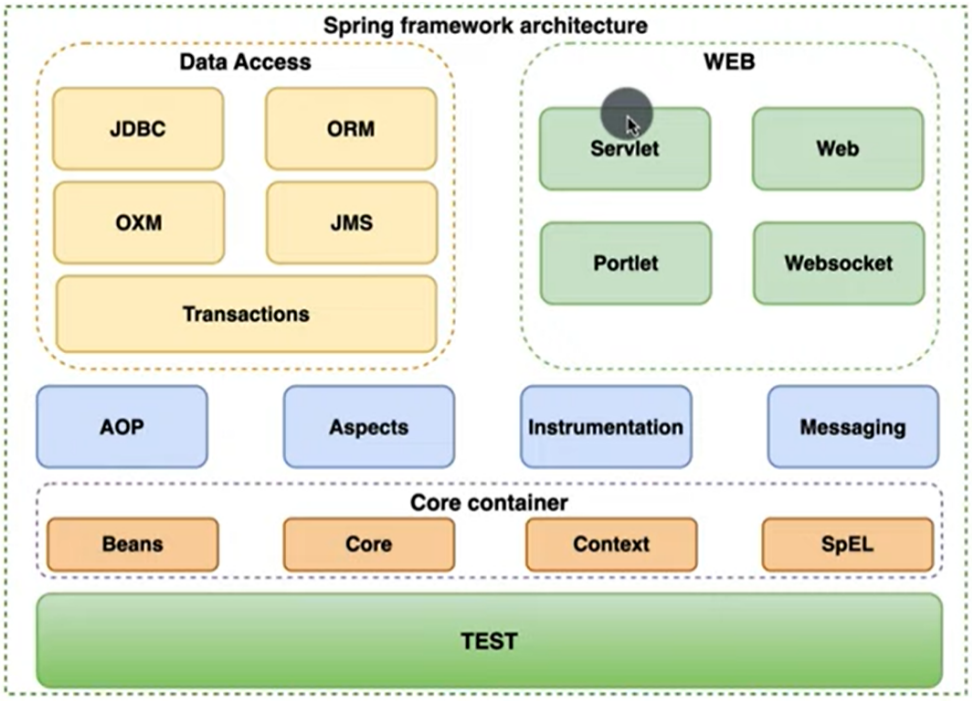
Spring Boot

It is extension of the spring framework.

It will help to build a stand alone application using spring.

- Embedded we server (Tomcat, Jetty).
- Auto configuration of the Spring and 3rd party libraries.
- Production ready.
- Easy integration with popular tool Maven, Gradle, STS.
- Developer friendly.
- It is suited for micro service, cloud native development .

Architecture



Data Access

Providing the to access the different type of data sources in spring application.

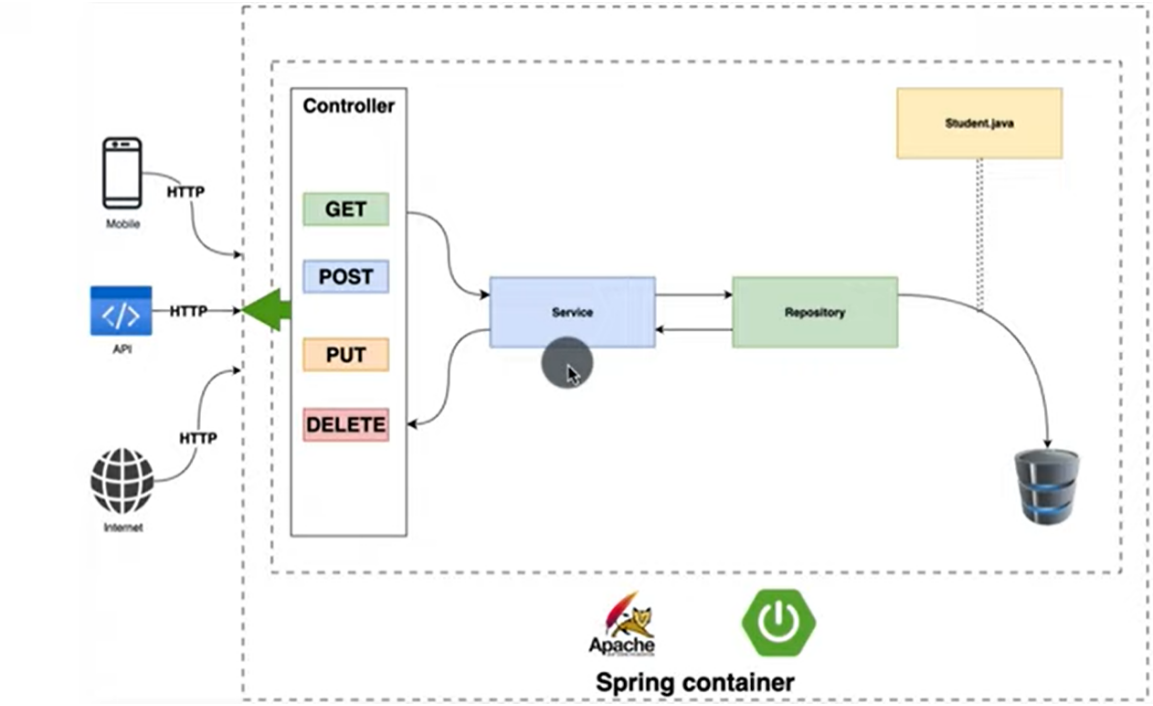
- JDBC → It is help to access and manipulate the data from database and avoid boilerplate code.
- ORM → Object-Relational mapping it connect the Java or any type of object codes to the relational database.
- OXM → Mapping the objects in the XML representation.
- JMS → Is a Java API that provides a common way to send, receive, and process the messages
- Transactions

WEB

- Servlet → In the context of Spring, servlets are used for handling web requests and generating responses.
- Web → It includes features such as handling HTTP requests and responses, managing web sessions, and handling web-specific exceptions.
- Portlet → Portlets are web components used for building dynamic web applications

- Websocket → It is a two way communication between client and server.

Application Architecture



Let’s Build a application

Add the dependencies from Spring Initializer site.

This is help to add create the base for build a Spring project.

https://start.spring.io/

Add the Home page for the application

In a project folder navigate to the static file.

Project folder → src → main → resources → static

- add the sample html file.
- It will show whenever run the application

Rest API

Representational State Transfer.

It a API that provide to perform the CURD operations on application from client side.

Controller

It responsible for the requests that comes from client and forwarded it to server.

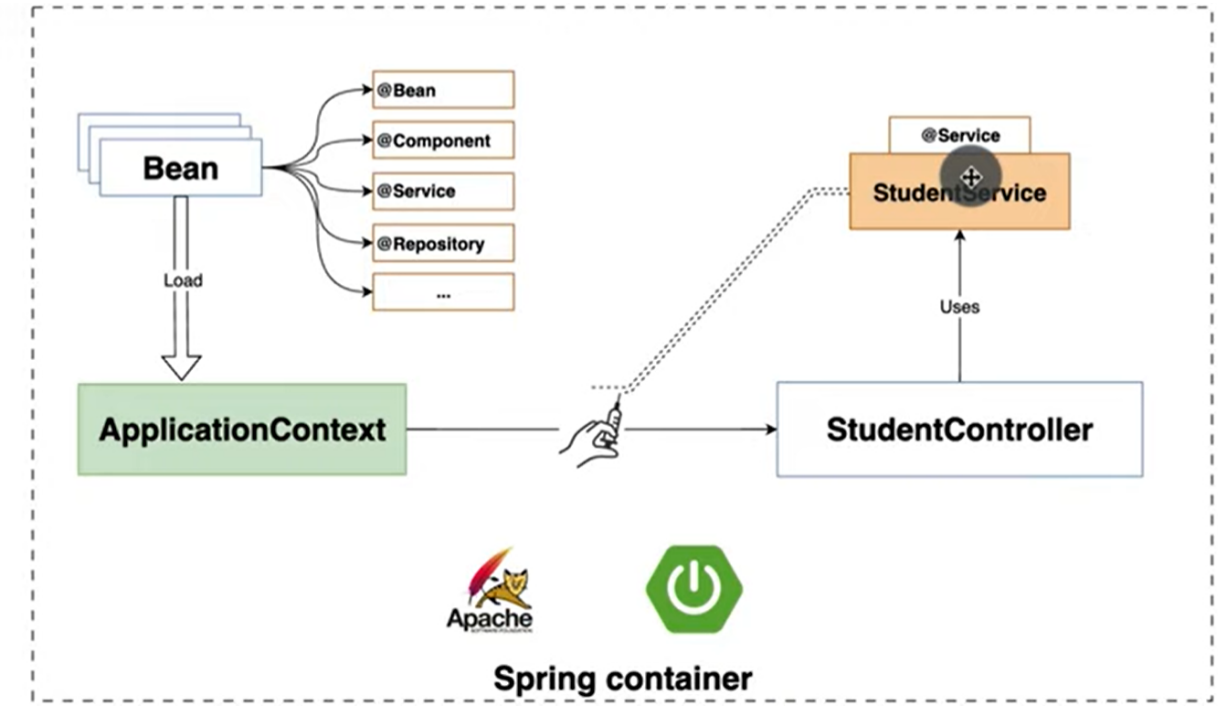
Every time that access the controller it create the object so it will **tightly coupled.**

Service

It mapped to the controller give the service to the controller.

To over come the tightly coupled problem **Dependency Injection** is used.

In a spring container it run all the Bean.



This is the service that injected when the user send the request.

Classes and Interfaces

Student (class)

It contain the basic details to get and store the data from the client.

Only these information will stored and asked to the client.

Key Points

`@Entity` → this is tell that is stored in a entity

`@Table(name = "student")` → this is stored in a entity in this table

`public int getAge() {  return Period.between(dateOfBirth,LocalDate.now()).getYears();}` → this special method calculate the age from the DOB.

Student Service(interface)

This is base service interface it will be implemented by other service classes.

It provide some sort of service based on the information from student class.

Like save the student information, find the student information, update the student info, delete the info and etc

It is interface so it contain body less methods.

example:

`Student save(Student s);` → this is help to save the info about student.

`List<Student> findAllStudents();` → this is give the response to the client with all the student information.

Controller

- It is a class annotated with these two beans

`@RestController'` → It is handler of the requests that come from the client side.

`@RequestMapping("/api/v1/students")`-> This mapped annotation give the URL to the Request that made.

- It should need injection of the service interface.

Now we need to give the request to the API and get the response from that.

there are multi way to send the request API, Browser, Mobile

But we need to **store the data** to the API for get the Response.

- **In Memory** → It will stored in a cache once the application will restart the data will be deleted.

- **Database** → It store the in a memory and give every time we run the API until the table will delete.

In Memory Service

To implement in memory service first create the DAO for the service.

DAO - Data Access Object.

- In DAO class,

  It will annotated by `@Repository` it tell to the spring it is DAO class and access the data from here

  For example:

  -

```
public Student save(Student s) {
 STUDENT.add(s);
  return s;
}
//This is save the info about the student
public List<Student> findAllStudents() {
      return STUDENT;
}//This is give all the student list to the client
```

  this code will perform request is made by the client

- In service class,

  It will annotated as `@Service.` and it work with the DAO class

Database Service

To store the information to the data base service need to create interface and extend the `JpaRepository<Student,Integer>` .

- In service class,

  There is some in built methods in `JpaRepository` give s advantage to create some service easily like `findAllStudents()`.

- Only the manipulation of the service will added in `JpaRepository` interface.

Application . yaml file

It is the main configuration file for the application.

This give some sort of services to the application like

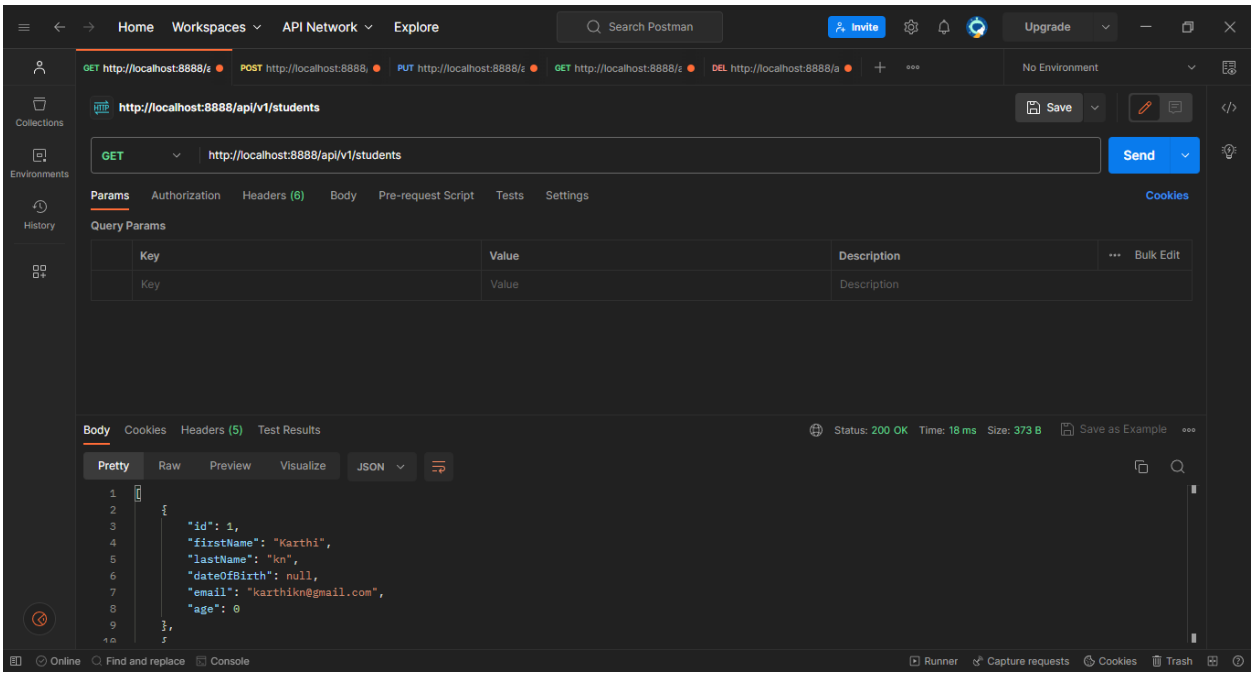
Configuring server settings:

```
server:
  port: 8080
```

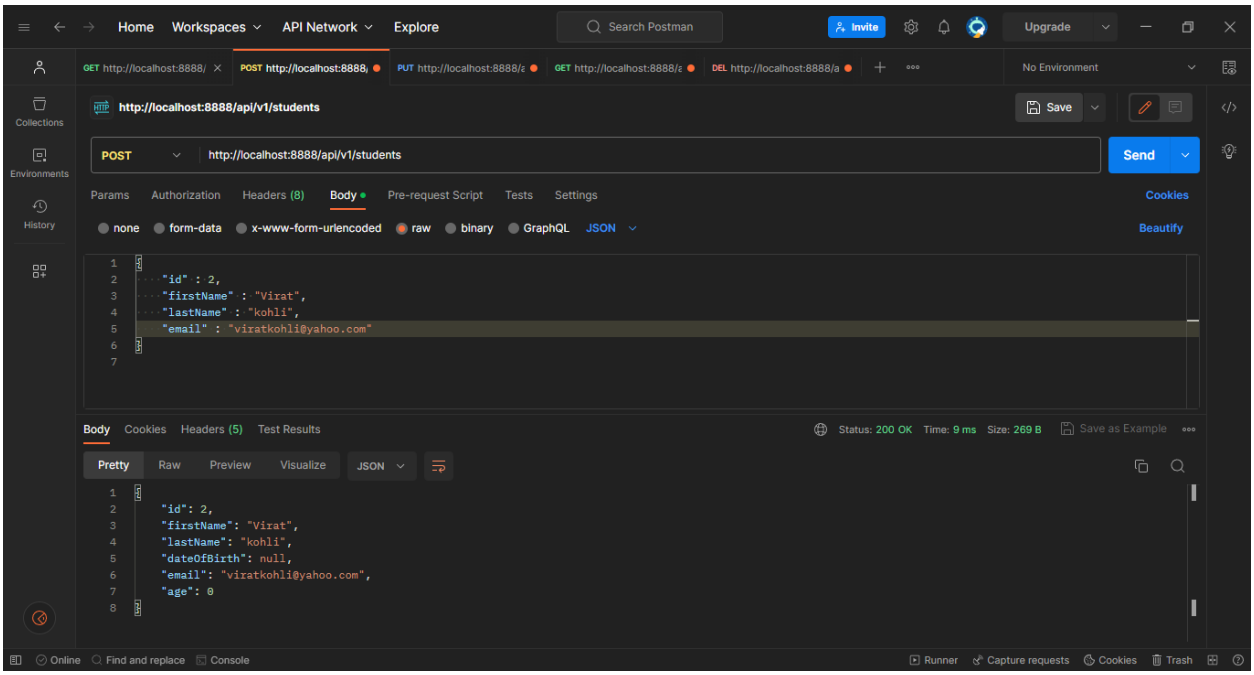
Configuring database connection:

```
spring:
  datasource:
    url: jdbc:mysql://localhost:3306/mydb
    username: myuser
    password: mypass
    driver-class-name: com.mysql.jdbc.Driver
```

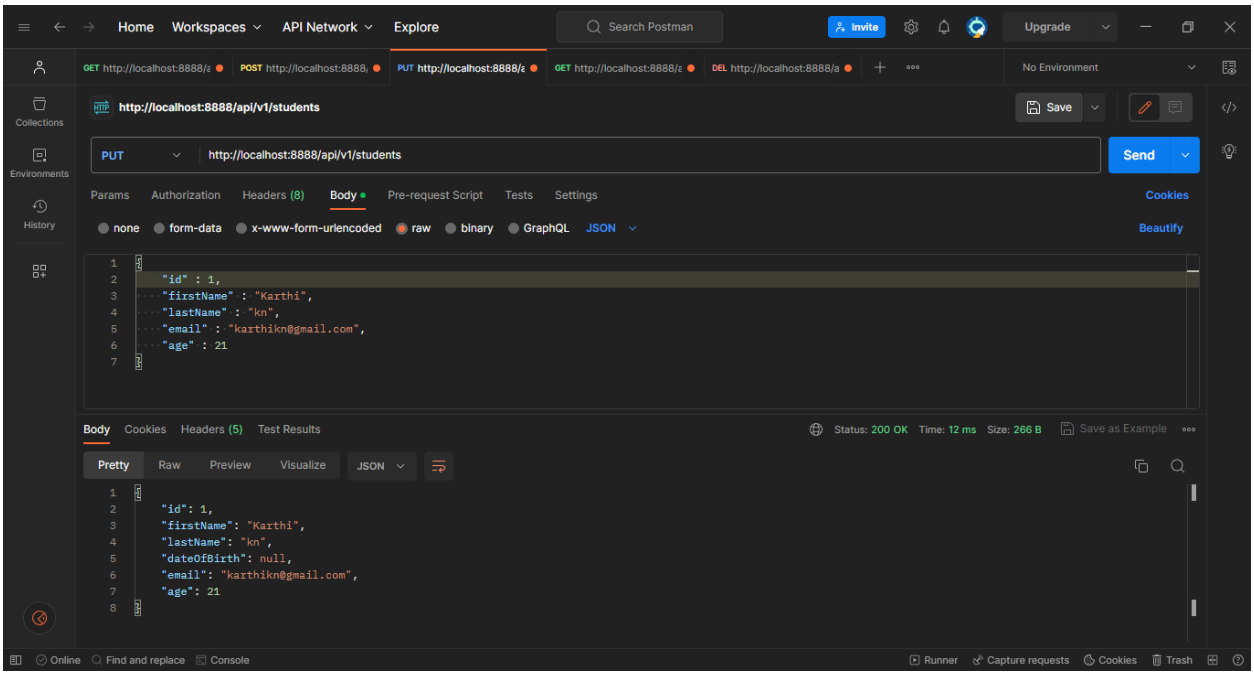
This is sample images of the API running:



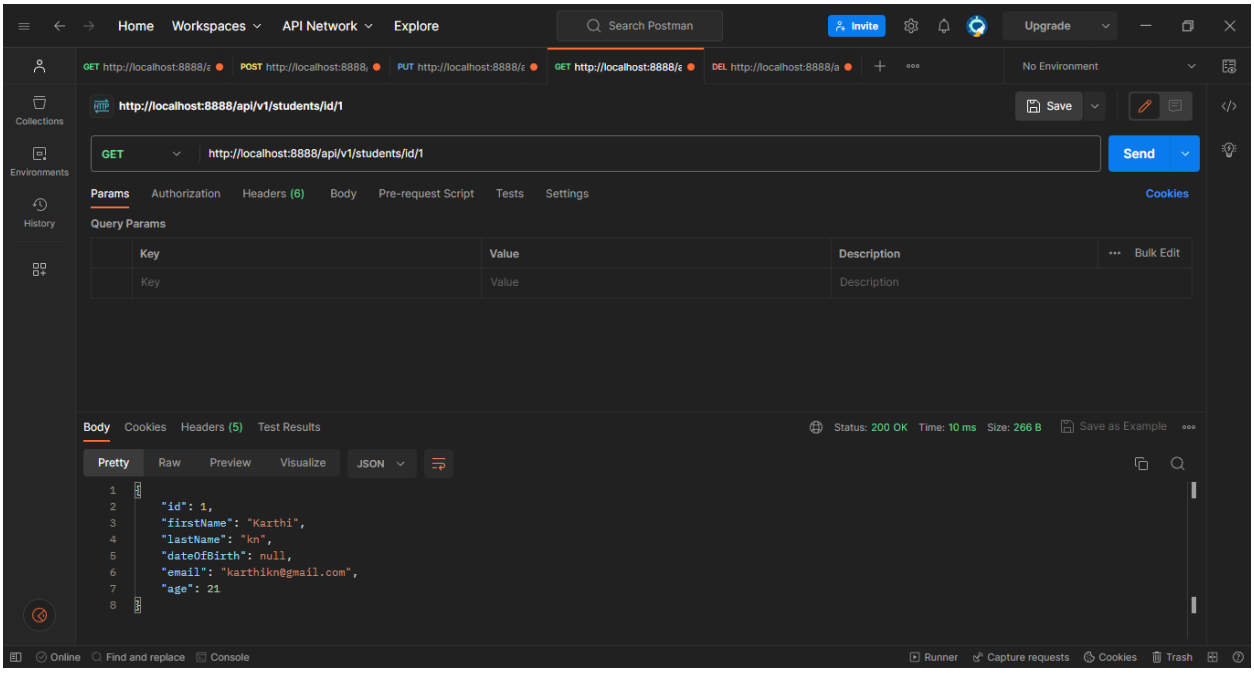
This GET request get all the information in the students database.



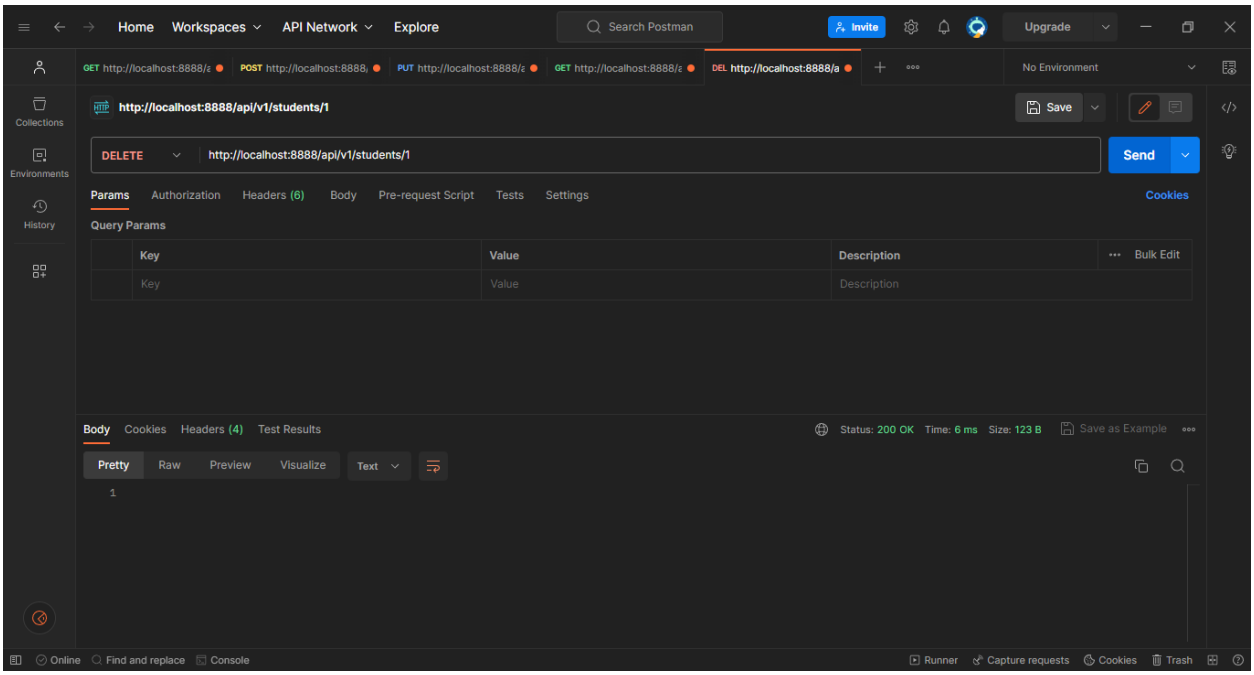
This POST request send the new student bio to the system and it will saved.



This PUT method will update the student with AGE column in system.



This GET request give the detail of the student based on the ID number.



This DELETE request is removed the student [ID =1] from the Database.

For more info take the source code from Github Repo: https://github.com/Karthikn-n/SpringBoot.

This is learned by Youtbue channel:

Spring Boot Tutorial For Beginners

#spring #learning #springboot #springtutorial #springsecurity #developpement #java #arraylist #linkedlist #springdatajpa #querybuilder #aliboucoding #alibou #validation #mongodb

https://www.youtube.com/watch?v=wsq1-m1dy_I&list=PL41m5U3u3wwkJXP69jYLzBnFoldbDr5FR