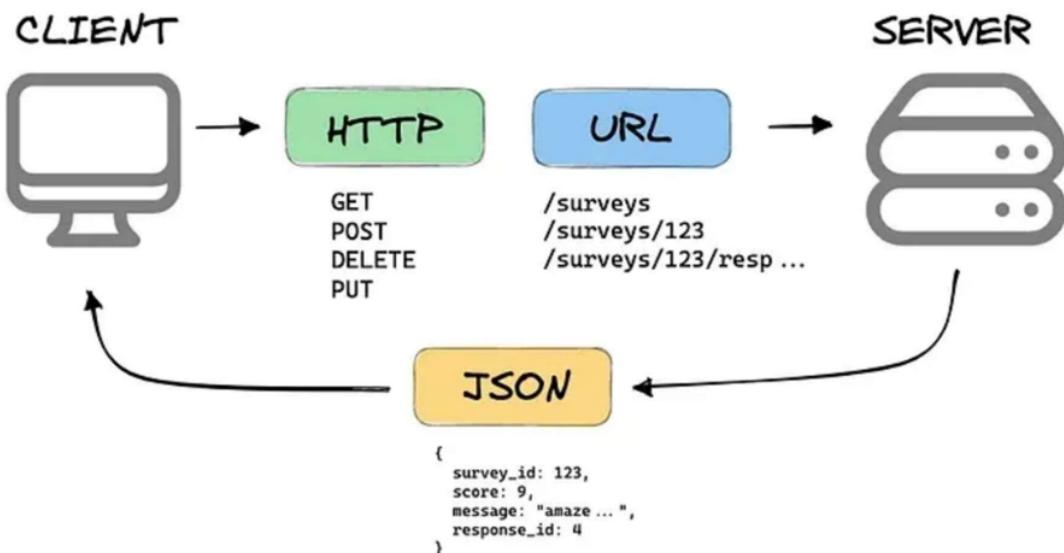


REST API

What is REST API :-

- REST API (**R**epresentational **S**tate **T**ransfer API) is a way for **two systems to communicate with each other** over the internet using the **HTTP protocol**.

WHAT IS A REST API?



➤ REST APIs allow clients (like Postman, mobile apps, web apps) to perform actions such as:

- Saving data
- Reading data
- Updating data
- Deleting data

Client sends a request → Server sends a response in JSON format.

Key Concepts of REST API :-

1. Client–Server Architecture
2. Stateless
3. Resource-Based
4. HTTP Methods
5. JSON Format
6. Uniform Interface
7. Layered System

HTTP Methods :-

Method	Meaning	Example
GET	Read Data	GET / students
POST	Create Data	POST / students
PUT	Update full data	PUT / students / 5
PATCH	Update part of data	PATCH / students / 5
DELETE	Delete data	DELETE / students / 5

Short Summary :-

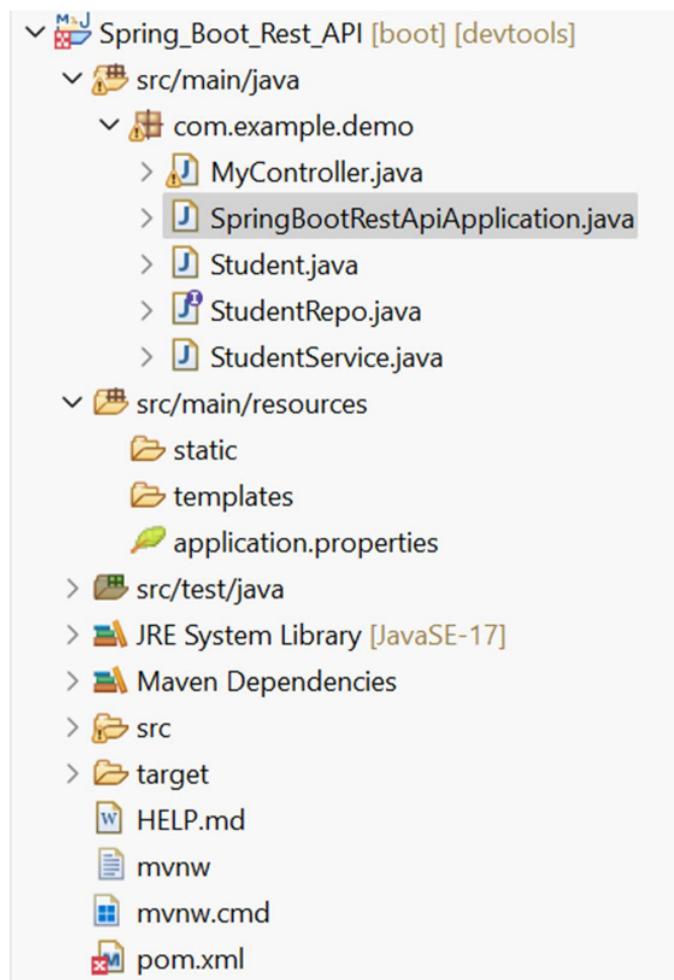
- **REST API:** A way to communicate between client and server using HTTP.
- **Stateless:** Every request is independent.
- **Resource-based:** Data = resources accessed via URLs.
- **Uses HTTP Methods:** GET, POST, PUT, DELETE.
- **Uses JSON:** For sending data.
- **Simple, scalable, fast.**

Spring Boot + REST API

Project Description :-

- This project is a **Spring Boot REST API** for handling **Student Registration**.
- It allows a client (Postman, React, Android, etc.) to **send student details**, and the API will store student information in the **database using Spring Data JPA**.

Project Structure :-



Step 1 :- pom.xml

```
https://maven.apache.org/xsd/maven-4.0.0.xsd (xsi:schemaLocation)
1 <?xml version="1.0" encoding="UTF-8"?>
2<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3   xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">
4     <modelVersion>4.0.0</modelVersion>
5     <parent>
6       <groupId>org.springframework.boot</groupId>
7       <artifactId>spring-boot-starter-parent</artifactId>
8       <version>4.0.0</version>
9       <relativePath/> <!-- lookup parent from repository -->
10    </parent>
11    <groupId>com.example</groupId>
12    <artifactId>Spring_Boot_Rest_API</artifactId>
13    <version>0.1-SNAPSHOT</version>
14    <name>Spring_Boot_Rest_API</name>
15    <description>Demo project for Spring Boot</description>
16    <url/>
17    <licenses>
18      <license/>
19    </licenses>
20    <developers>
21      <developer/>
22    </developers>
23    <scm>
24      <connection/>
25      <developerConnection/>
26      <tag/>
27      <url/>
28    </scm>
29    <properties>
30      <java.version>17</java.version>
31    </properties>
32    <dependencies> Add Spring Boot Starters...
33      <dependency>
34        <groupId>org.springframework.boot</groupId>
35        <artifactId>spring-boot-starter-webmvc</artifactId>
36      </dependency>
37
38      <dependency>
39        <groupId>org.springframework.boot</groupId>
40        <artifactId>spring-boot-devtools</artifactId>
41        <scope>runtime</scope>
42        <optional>true</optional>
43      </dependency>
44      <dependency>
45        <groupId>org.springframework.boot</groupId>
46        <artifactId>spring-boot-starter-webmvc-test</artifactId>
47        <scope>test</scope>
48      </dependency>
49
50
51      <dependency>
52        <groupId>org.springframework.boot</groupId>
53        <artifactId>spring-boot-starter</artifactId>
54      </dependency>
55
56      <dependency>
57        <groupId>org.springframework.boot</groupId>
```

```
58         <artifactId>spring-boot-devtools</artifactId>
59         <scope>runtime</scope>
60         <optional>true</optional>
61     </dependency>
62
63     <dependency>
64         <groupId>org.springframework.boot</groupId>
65         <artifactId>spring-boot-starter-test</artifactId>
66         <scope>test</scope>
67     </dependency>
68
69     <!-- New Added -->
70     <dependency>
71         <groupId>org.springframework.boot</groupId>
72         <artifactId>spring-boot-starter-web</artifactId>
73     </dependency>
74
75     <dependency>
76         <groupId>org.springframework.boot</groupId>
77         <artifactId>spring-boot-starter-data-jpa</artifactId>
78     </dependency>
79
80     <dependency>
81         <groupId>com.mysql</groupId>
82         <artifactId>mysql-connector-j</artifactId>
83     </dependency>
84
85 </dependencies>
86
87     <build>
88         <plugins>
89             <plugin>
90                 <groupId>org.springframework.boot</groupId>
91                 <artifactId>spring-boot-maven-plugin</artifactId>
92             </plugin>
93         </plugins>
94     </build>
95
96 </project>
```

Step 2 :- SpringBootRestApiApplication.java

```
1 package com.example.demo;
2
3@import org.springframework.boot.SpringApplication;
4 import org.springframework.boot.autoconfigure.SpringBootApplication;
5
6 @SpringBootApplication
7 public class SpringBootRestApiApplication {
8
9    public static void main(String[] args) {
10        SpringApplication.run(SpringBootRestApiApplication.class, args);
11    }
12
13 }
```

Step 3 :- Student.java

```
1 package com.example.demo;
2
3@import jakarta.persistence.Column;
4 import jakarta.persistence.Entity;
5 import jakarta.persistence.GeneratedValue;
6 import jakarta.persistence.GenerationType;
7 import jakarta.persistence.Id;
8 import jakarta.persistence.Table;
9
10 @Entity //Table Create
11 @Table(name="student") //Table Name
12 public class Student {
13
14     @Id
15     @GeneratedValue(strategy = GenerationType.IDENTITY) //Autoincrement
16     private Long id;
17
18     @Column //Create Column
19     private String fullname;
20
21     @Column(unique=true)
22     private String email;
23
24     @Column
25     private String password;
26
27     @Column
28     private String address;
29
30     public Long getId() {
```

```
31         return id;
32     }
33
34     public void setId(Long id) {
35         this.id = id;
36     }
37
38     public String getFullscreen() {
39         return fullname;
40     }
41
42     public void setFullscreen(String fullname) {
43         this.fullname = fullname;
44     }
45
46     public String getEmail() {
47         return email;
48     }
49
50     public void setEmail(String email) {
51         this.email = email;
52     }
53
54     public String getPassword() {
55         return password;
56     }
57
58     public void setPassword(String password) {
59         this.password = password;
60     }
61
62     public String getAddress() {
63         return address;
64     }
65
66     public void setAddress(String address) {
67         this.address = address;
68     }
69
70     @Override
71     public String toString() {
72         return "Student [id=" + id + ", fullname=" + fullname + ", email=" + email + ", password=" + password
73             + ", address=" + address + "]";
74     }
75 }
```

Step 4 :- StudentRepo.java

```
1 package com.example.demo;
2
3 import org.springframework.data.jpa.repository.JpaRepository;
4 import org.springframework.stereotype.Repository;
5
6 @Repository
7 public interface StudentRepo extends JpaRepository<Student, Long>{
8
9
10 }
```

Step 5 :- StudentService.java

```
1 package com.example.demo;
2
3 import java.util.List;
4
5 import org.springframework.beans.factory.annotation.Autowired;
6 import org.springframework.stereotype.Service;
7
8 @Service
9 public class StudentService {
10
11     @Autowired
12     StudentRepo sr;
13
14     public void registration(Student s)
15     {
16         sr.save(s); //Insert Query Directly
17     }
18
19     public List<Student> getallstudents()
20     {
21         return sr.findAll();    //select * from student
22     }
23 }
```

Step 6 :- MyController.java

```
1 package com.example.demo;
2
3 import org.springframework.beans.factory.annotation.Autowired;
4 import org.springframework.http.HttpStatus;
5 import org.springframework.http.ResponseEntity;
6 import org.springframework.web.bind.annotation.CrossOrigin;
7 import org.springframework.web.bind.annotation.GetMapping;
8 import org.springframework.web.bind.annotation.PostMapping;
9 import org.springframework.web.bind.annotation.RequestBody;
10 import org.springframework.web.bind.annotation.RestController;
11
12 @RestController
13 @CrossOrigin("*")
14 public class MyController {
15
16     @GetMapping("/")
17     public ResponseEntity<?> test()
18     {
19         return ResponseEntity.status(HttpStatus.ACCEPTED).body("Tested Ok");
20     }
21
22     @Autowired
23     StudentService ss;
24
25     @PostMapping("/register")
26     public ResponseEntity<?> registration(@RequestBody Student s)
27     {
28         try
29         {
30             ss.registration(s);
31             return ResponseEntity.status(HttpStatus.ACCEPTED).body("Registration Successfully Completed");
32         }
33         catch(Exception e)
34         {
35             return ResponseEntity.status(HttpStatus.BAD_REQUEST).body("Fail");
36         }
37     }
38 }
```

Step 7 :- application.properties

```
1 spring.application.name=Spring_Boot_Rest_API
2
3 spring.datasource.url=jdbc:mysql://localhost:3309/springbootdb1
4 spring.datasource.username=root
5 spring.datasource.password=
6
7 spring.jpa.hibernate.ddl-auto=update
8 spring.jpa.show-sql=true
9 spring.jpa.properties.hibernate.format_sql=true
```

Output :-

Server Output :-

The app runs at <http://localhost:8080>

Tested Ok

Postman :-

The screenshot shows the Postman interface with two requests:

- GET Request:** URL: `http://localhost:8080`. Method: `GET`.
- POST Request:** URL: `http://localhost:8080/register`. Method: `POST`.

Below the requests, the **Body** tab is selected, showing the raw JSON payload:

```
1 {
2   "fullname": "Nikhil",
3   "email": "nikhil77@gmail.com",
4   "password": "749977",
5   "address": "Solapur"
6 }
7
```

Body Cookies Headers (8) Test Results | ⌂

Raw ▾ ▷ Preview ⚡ Visualize ▾

1 Registration Successfully Completed

Database :-

id	address	email	fullname	password
3	Solapur	abc@gmail.com	abc	56789
4	Pune	xyz@gmail.com	xyz	749976
6	Pune	nikhil1234@gmail.com	Nikhil 1	123456
18	pune	abc2121@gmail.com	abcd xyz	12345
21	Solapur	nikhil77@gmail.com	Nikhil	749977