Spring Restful

- Web service – is a middle layer that interacts with the front-end(UI) & the back-ends (mostly DBs)
- Web services has service description layer in the web service protocol stack
- Its function is to describe the user interface to a web service
- Web browser – acts as a client (web client)
- Alternate to browser is Restful – acts as a Restful client
- Website – acts as restful provider

Main features of web services:

1. It is available over internet readily
2. It uses a standardized XML messaging system
3. It is not tied to any one operating system or programming language
4. It is self-describing using a common XML grammar
5. It is discoverable via simple find mechanism

Advantages:

1. Interoperability
2. Reusability
3. Modularity
4. Cheaper – for communications

Two types of web services:

SOAP – It is an XML-based protocol for accessing web services

REST – It is an architectural style, not a protocol

➢ REST – Representational State Transfer
➢ It is an architectural style of developing web services that takes advantage of HTTP protocols & leverages the HTTP methods easily to define the actions
➢ Webservices developed using REST-style is known as Restful web service
➢ Components of Restful web services:

(a) Resource – it is a fundamental concept of the Restful architecture; It is an object
(b) Object has three features:
    a. Type
    b. Relationship

c. Methods

(c) Resources are identified using the following:
    a. URI (purpose of URI is to locate a resource on the server of web service)
    b. HTTP Methods
    c. Request / Response data type

(d) Four Operations (CRUD):
    a. Create
    b. Read
    c. Update
    d. Delete

(e) CRUD Operations are performed using
    a. HTTP methods, which are called as verbs
    b. URI, which are called as nouns

HTTP Methods:

(a) GET – Read a resource (Read-only access)
(b) POST – Create a new resource
(c) PUT – update an existing resource
(d) DELETE – remove an existing resource
(e) PATCH – update a resource with conditions
(f) HEAD

HTTP Status Codes:

- ✓ 200 series – Success
- ✓ 300 series – Network bandwidth
- ✓ 400 series – Error or Bad Request
- ✓ 500 series – Internal Server Error

Components of HTTP Request:

- HTTP version – the version of request
- Request Body – Represents the message content
- Request Header – contains meta-data such as setting, client type, data type etc
- URI – Identify the resource
- Verb – HTTP methods

Components of HTTP Response:

- HTTP Version – version of response
- Response Body – Message content with data

- Response Header – Meta data with info like content length, server size etc
- Status Code – indicates the status of response

Steps to create project:

Step 1: Create a web application (SpringBootStarterWeb)

Step 2: @RestController – to create a Rest Endpoint

Step 3: @RequestMapping- to map the path

Step 4: Define methods in the controller

Step 5: Set the respective request

Step 6: Launch the Restful client & trigger the request

Step 7: Validate the result

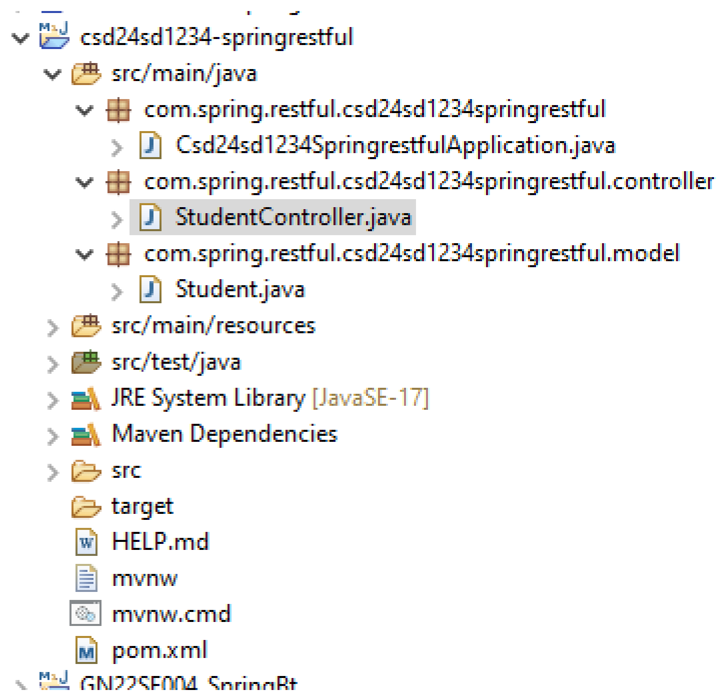
Example:

Pom.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<project
xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"
    xsi:schemaLocation="http://maven.apache.o
rg/POM/4.0.0
https://maven.apache.org/xsd/maven-
4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <parent>

    <groupId>org.springframework.boot</groupI
d>
        <artifactId>spring-boot-starter-
parent</artifactId>
        <version>3.2.3</version>
        <relativePath/> <!-- lookup parent
from repository -->
    </parent>
    <groupId>com.spring.restful</groupId>
    <artifactId>csd24sd1234-
springrestful</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <name>csd24sd1234-springrestful</name>
    <description>Demo project for Spring
Boot</description>
    <properties>
        <java.version>17</java.version>
    </properties>
    <dependencies>
        <dependency>

    <groupId>org.springframework.boot</groupI
d>
            <artifactId>spring-boot-starter-
web</artifactId>
        </dependency>

        <dependency>

    <groupId>org.springframework.boot</groupI
d>
            <artifactId>spring-boot-starter-
test</artifactId>
            <scope>test</scope>
        </dependency>
    </dependencies>

    <build>
```

```
        <plugins>
            <plugin>

    <groupId>org.springframework.boot</groupI
d>
                <artifactId>spring-boot-maven-
plugin</artifactId>
            </plugin>
        </plugins>
    </build>

</project>
```

Folder Structure:



```
csd24sd1234-springrestful
  src/main/java
    com.spring.restful.csd24sd1234springrestful
      Csd24sd1234SpringrestfulApplication.java
    com.spring.restful.csd24sd1234springrestful.controller
      StudentController.java
    com.spring.restful.csd24sd1234springrestful.model
      Student.java
  src/main/resources
  src/test/java
  JRE System Library [JavaSE-17]
  Maven Dependencies
  src
  target
  HELP.md
  mvnw
  mvnw.cmd
  pom.xml
GN22SE004_SpringBt
```

Main Application:

```
package
com.spring.restful.csd24sd1234springrestful;
```

```
import
org.springframework.boot.SpringApplication;
import
org.springframework.boot.autoconfigure.Spring
BootApplication;

@SpringBootApplication
public class
Csd24sd1234SpringrestfulApplication {

    public static void main(String[] args) {

    SpringApplication.run(Csd24sd1234Springre
stfulApplication.class, args);
    }

}
```

Student Class:

```
package
com.spring.restful.csd24sd1234springrestful.m
odel;

public class Student {

    private String id;
    private String name;

    public String getId() {
        return id;
    }
    public void setId(String id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
}
```

Controller Class:

```
package
com.spring.restful.csd24sd1234springrestful.c
ontroller;

import java.util.HashMap;
import java.util.Map;

import org.springframework.http.HttpStatus;
import
org.springframework.http.ResponseEntity;
import
org.springframework.web.bind.annotation.Reque
stBody;
import
org.springframework.web.bind.annotation.Reque
stMapping;
import
org.springframework.web.bind.annotation.Reque
stMethod;
```

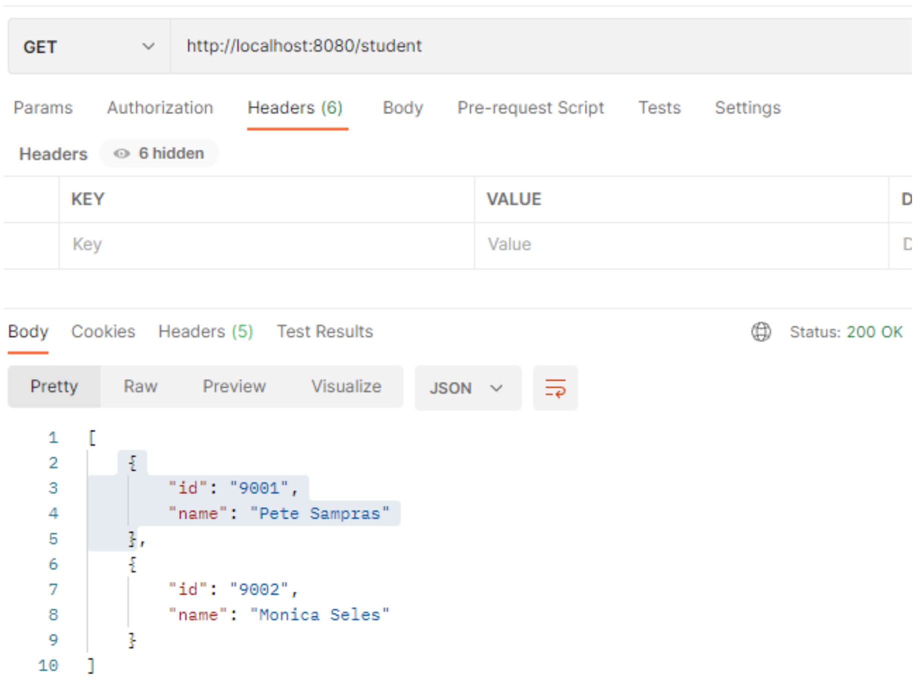
```
import
org.springframework.web.bind.annotation.RestC
ontroller;

import
com.spring.restful.csd24sd1234springrestful.m
odel.Student;

@RestController
public class StudentController {

    private static Map<String, Student>
studentRecord = new HashMap<>();
    static {
        Student s1 = new Student();
        s1.setId("9001");
        s1.setName("Pete Sampras");
        studentRecord.put(s1.getId(), s1);

        Student s2 = new Student();
        s2.setId("9002");
        s2.setName("Monica Seles");
        studentRecord.put(s2.getId(), s2);
    }

    @RequestMapping(value="/student")
    public ResponseEntity<Object>
getStudent() {
        return new
ResponseEntity<>(studentRecord.values(),
HttpStatus.OK);
    }
    @RequestMapping(value="/student", 
method=RequestMethod.POST)
    public ResponseEntity<Object> 
createStudent(@RequestBody Student student){
        studentRecord.put(student.getId(), 
student);
//    
    System.out.println(studentRecord.values()
);
//      return new ResponseEntity<>("Student 
Record Created successfully", 
HttpStatus.CREATED);
        return new 
ResponseEntity<>(studentRecord.values(), 
HttpStatus.CREATED);
    }
}
```

Run the Program & ensure TomCat server started

Request URL:

http://localhost:8080/student

Postman Request & Response (GET Method)

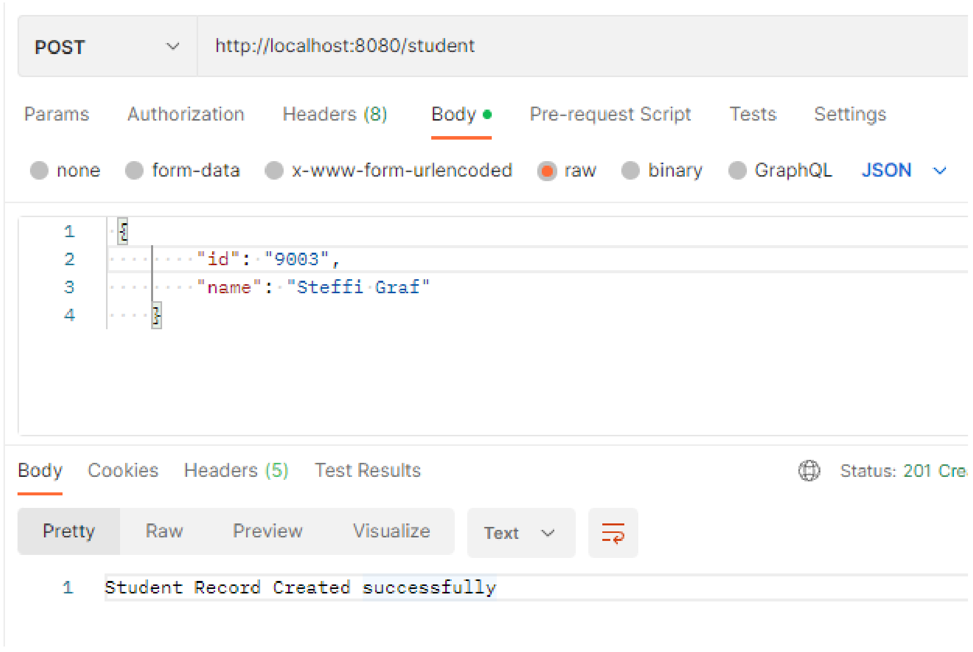


```
http://localhost:8080/student
```

Status: 200 OK

```
[
    {
        "id": "9001",
        "name": "Pete Sampras"
    },
    {
        "id": "9002",
        "name": "Monica Seles"
    }
]
```

Postman Request & Response (Post Method with Response Message)



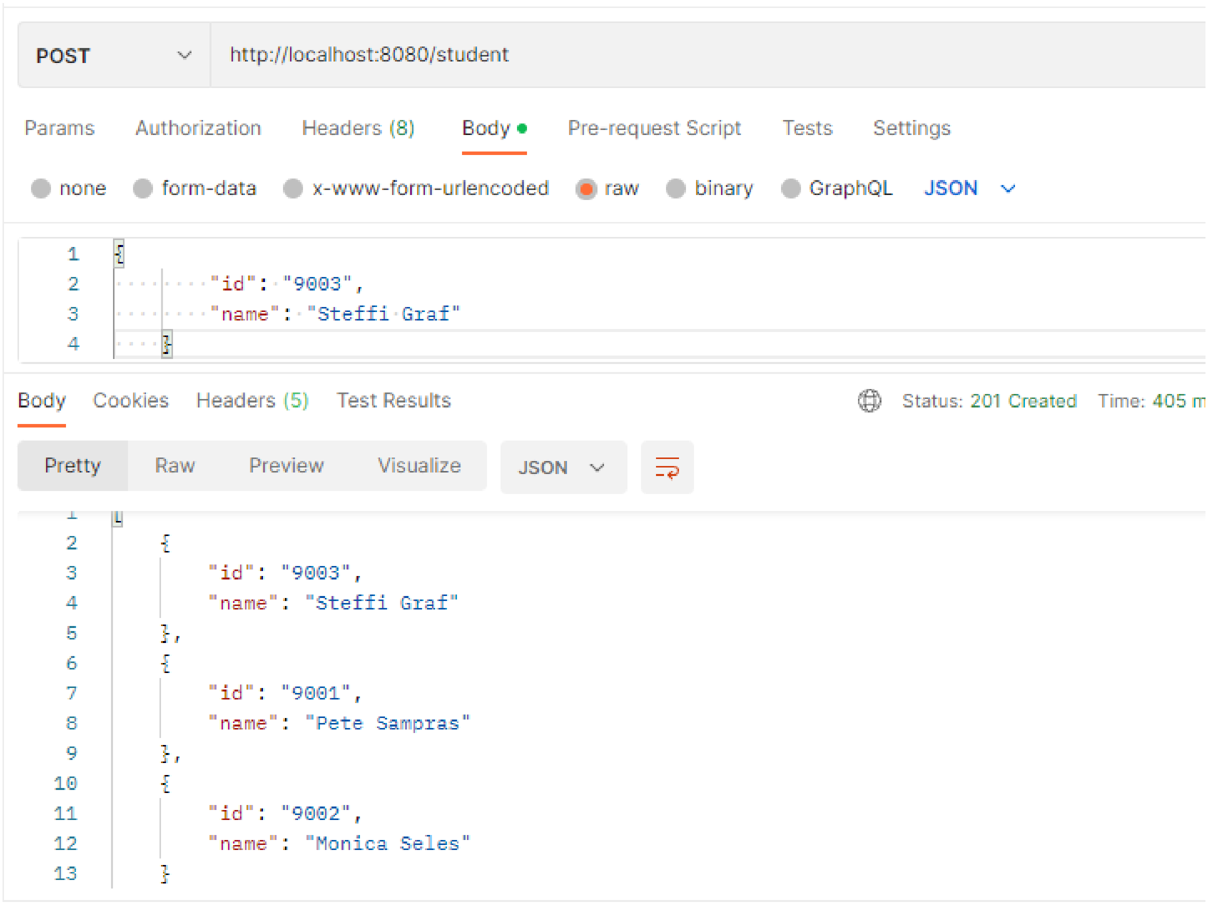
```
http://localhost:8080/student
```

```
{
    "id": "9003",
    "name": "Steffi Graf"
}
```

Status: 201 Cre

```
Student Record Created successfully
```

Postman Request & Response (Post Method with Response Data)



http://localhost:8080/student

```
{
    "id": "9003",
    "name": "Steffi Graf"
}
```

Status: 201 Created   Time: 405 m

```
[
    {
        "id": "9003",
        "name": "Steffi Graf"
    },
    {
        "id": "9001",
        "name": "Pete Sampras"
    },
    {
        "id": "9002",
        "name": "Monica Seles"
    }
```

Delete & Put Methods:

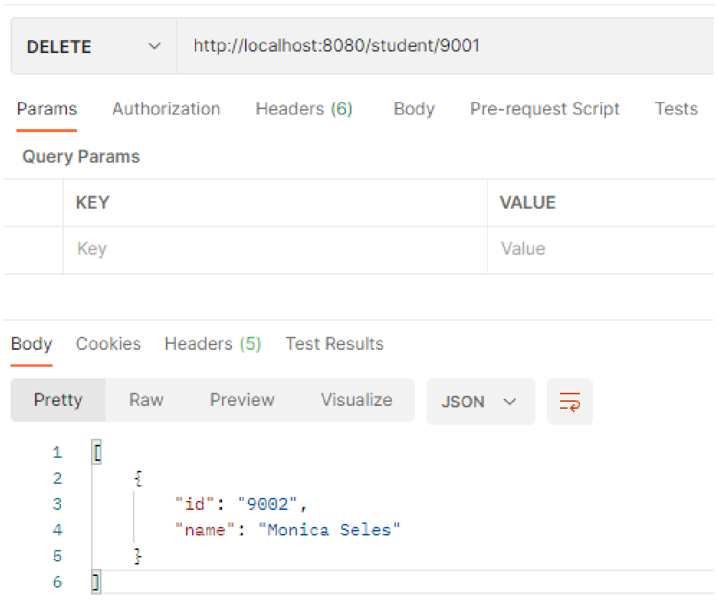
(Add following methods in Controller class)

```
@RequestMapping(value="/student/{id}",
method=RequestMethod.DELETE)
    public ResponseEntity<Object>
deleteRecord(@PathVariable("id") String id){
        studentRecord.remove(id);
        return new
ResponseEntity<>(studentRecord.values(),
HttpStatus.OK);
    }

    @RequestMapping(value="/student/{id}",
method=RequestMethod.PUT)
```

```
    public ResponseEntity<Object>
updateStudent(@PathVariable("id") String id,
@RequestBody Student student){
        studentRecord.remove(id);
        student.setId(id);
        studentRecord.put(id, student);
//      return new ResponseEntity<>("Student
Record updated successfully", HttpStatus.OK);
        return new
ResponseEntity<>(studentRecord.values(),
HttpStatus.OK);
    }
```

Run the Program & ensure TomCat server started

Postman Request & Response (Delete Method with Response Data)

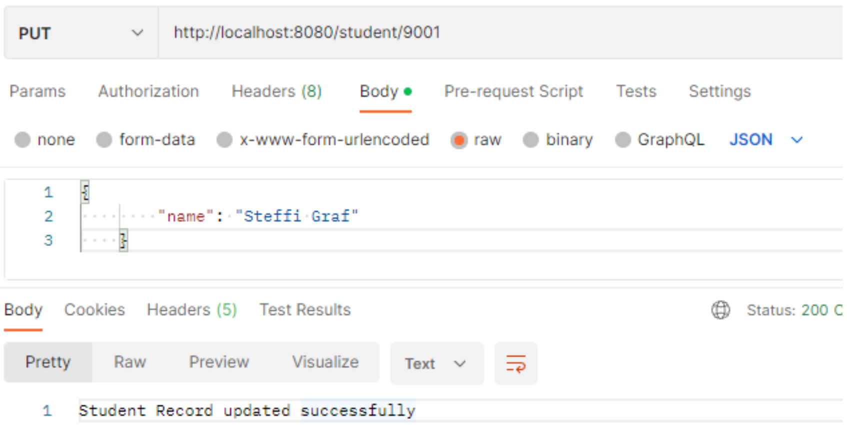


```
http://localhost:8080/student/9001
```

```
[
    {
        "id": "9002",
        "name": "Monica Seles"
    }
]
```

Run the Program & ensure TomCat server started

Postman Request & Response (PUT Method with Response Message)



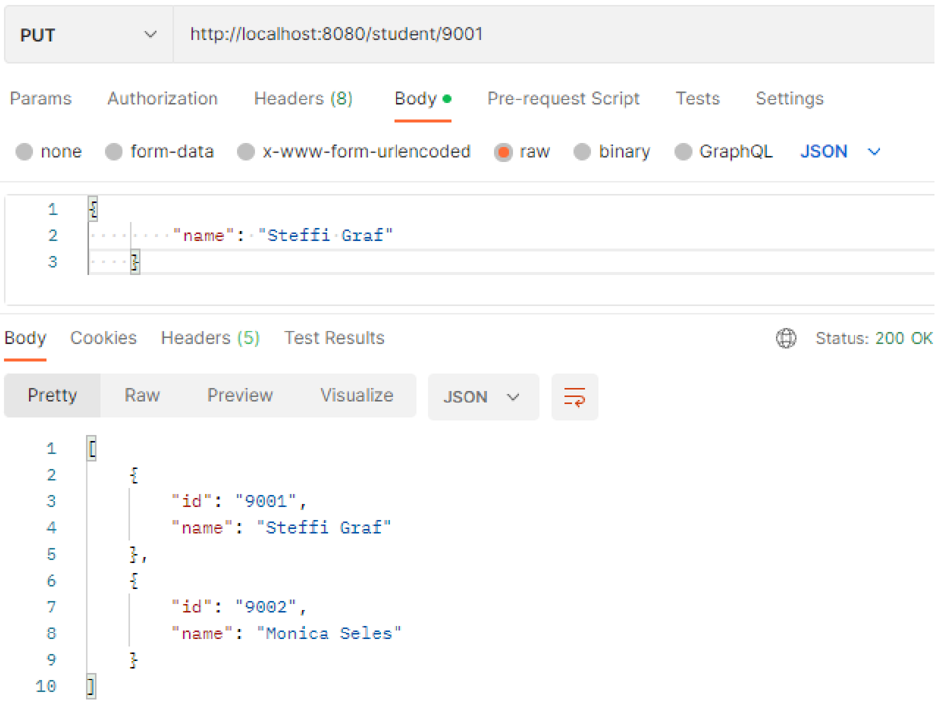
http://localhost:8080/student/9001

```
{
    "name": "Steffi Graf"
}
```

Status: 200 C

```
Student Record updated successfully
```

Postman Request & Response (PUT Method with Response Data)



http://localhost:8080/student/9001

```
{
    "name": "Steffi Graf"
}
```

Status: 200 OK

```
[
    {
        "id": "9001",
        "name": "Steffi Graf"
    },
    {
        "id": "9002",
        "name": "Monica Seles"
    }
]
```