nested Json object:

```
class Student{

private int roll;
private String name
private int marks;
private Address addr;

}


class Address{

private String city;
private String state;
private String pincode;

}
```

--json representation of Student object:

```
{
"roll": 100,
"name": "Ram",
"marks": 600,
"addr": {

        "city": "pune",
        "state": "Maharashtra",
        "pincode": "432434"

}
}
```

List of Student Json representation:

```
[
{
"roll": 10,
"name": "Ram",
"marks": 780
},
{
"roll": 20,
"name": "Ramesh",
"marks": 880
}
]
```

returning List of Student objects:
--

```
    @RequestMapping("/students")
    public List<Student> getStudentHandler() {

        List<Student> students = new ArrayList<>();


         students.add(new Student(10, "R1", 780));
         students.add(new Student(20, "R2", 580));
         students.add(new Student(30, "R3", 680));
         students.add(new Student(40, "R4", 880));



        return students;
    }
```

Note: following all the mapping details are same:

```
//@RequestMapping("/student") // method GET is default type need not mention
//@RequestMapping(value="/student", method = RequestMethod.GET) // JSON is the
default produce type
//@RequestMapping(value="/student", method = RequestMethod.GET, produces =
MediaType.APPLICATION_JSON_VALUE )
//@GetMapping( value="/student" ,produces = MediaType.APPLICATION_XML_VALUE)
@GetMapping("/student") // shortcut
```

```
public Student getStudentHandler() {

                    return new Student(10, "Ram", 880);

}
```

Note: while returning the response from the webservice methods spring f/w uses some of the **"message converters"** to convert the Java object into the domain format like JSON object, XML format, etc.

The default conversion type is JSON type, here spring f/w uses "message converters" with the help of **Jackson API** internally. so defining **MediaType.APPLICATION_JSON_VALUE** is optional.

http://localhost:8888/student

uri for root resource:

http://localhost:8888/studentapp/student

Devtools:
========

--add the devtools related dependencies to hot module reloading.

Sending the Data to the server by the client using Path variable (data without key) http://localhost:8888/students/12   // here 12 is the path variable

example:

```
	@GetMapping("/students/{roll}")
	public Student getStudentHandler(@PathVariable("roll") Integer rollno) {



		return new Student(rollno, "Ramesh", 880);

	}
```

Sending multiple Path variable:

```
	@GetMapping("/students/{roll}/{name}/{marks}")
	public Student getStudentHandler(@PathVariable("roll") int rollno,
@PathVariable("name") String	name, @PathVariable("marks") int marks) {


		return new Student(rollno, name, marks);

	}
```

http://localhost:8888/studentapp/students/500/Ratan/780

Student.java:

```
package com.masai.model;

public class Student {

	private Integer roll;
	private String name;
	private Integer marks;

//getters and setters
}
```

StudentController.java:

```
---------------------------

package com.masai.controller;

import java.util.Arrays;
import java.util.List;

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

import com.masai.model.Student;

@RestController
public class StudentController {


	@GetMapping("/hello")
	public String sayHello() {
		return "welcome";
	}


	@GetMapping("/students/{roll}")
	public Student getStudentDetailsHandler(@PathVariable("roll") Integer roll) {

		Student student = new Student(roll, "Ram", 780);

		return student;


	}


	@GetMapping("/students")
	public List<Student> getAllStudentHandler() {
        return  Arrays.asList(

                new Student(10, "N1", 780),
                new Student(12, "N2", 720),
                new Student(14, "N3", 740),
                new Student(15, "N4", 750)

                );


    }

}
```

****Note: duplicate Uri with duplicate Http method will throw an exception. same uri with different action(http method) is possbile.

Request parameter:
==================

/students?roll=123&name=ram&marks=500

```
    @GetMapping("/getStudent")
    public Student getStudentDetailsHandler2(@RequestParam("r") Integer roll) {

        Student student = new Student(roll, "Ramesh", 780);

        return student;


    }
```

http://localhost:8888/getStudent?r=50

Note:- By default @RequestParam is mandatory, to make it optional :

```
@GetMapping("/getStudent")
```

```
public Student getStudentDetailsHandler2(@RequestParam(value = "r", required = false)
Integer roll) {

                Student student = new Student(roll, "Ramesh", 780);

                return student;


        }
```

http://localhost:8888/getStudent   //here roll will be null
http://localhost:8888/getStudent?r=50

Sending multiple request paramenter:

```
@GetMapping("/getStudent")
        public Student getStudentDetailsHandler2(@RequestParam Integer roll,
                                                 @RequestParam String name,
                                                 @RequestParam Integer marks) {

                Student student = new Student(roll,name,marks);

                return student;


        }
```

http://localhost:8888/getStudent?roll=10&name=ravi&marks=900

request parameter with pathvariable we can combine also: here pathvariable will come first then we need to use request parameter.

example:

```
//http://localhost:8888/students/10?m=800


        @GetMapping("/students/{roll}")
```

```
	public Student getStudentHandler(@PathVariable("roll") Integer roll,
@RequestParam(value = "m", required = false) Integer marks) {


		return new Student(roll, "Ravi", "Adr1", marks);
	}
```

sending data from the request body.

sending Post request from client:
==============================

```
@PostMapping(value = "/students",consumes =
MediaType.APPLICATION_JSON_VALUE)
	public String saveStudentHandler(@RequestBody Student student) {

		//here we can communicate with the Service layer or Data Access Layer
classes to
		//persist the Student object in the Database.

		return  "Student stored ,"+student;

	}
```

or

```
@PostMapping("/students")
	public String saveStudentHandler(@RequestBody Student student) {

		//here we can communicate with the Service layer or Data Access Layer
classes to
		//persist the Student object in the Database.

		return  "Student stored ,"+student;

	}
```

sending put request with pathvariable and request body as well:
===
GET   : does not have the body, data can be send from pathvariable or request parameter
POST : does have the body, data can be send from pathvarible or request parameter and also with request body
PUT :does have the body, data can be send from pathvarble or request parameter and also with request body
DELETE : does not have the body, data can be send from pathvariable or request parameter