

SpringRest ==> SpringMVC++
=====

Note:: @RestController ==> @Controller + @ResponseBody

ResponseBody<T>(return type of handler methods) indicates that the generated output should directly go via DispatcherServlet to the RestClient.

ResponseBody<String> as the return type of handler method / String as the return type of the handler methods indicates that in the Response body the content-type = "text/plain".

RestFulServices
=====

1. RestResource
2. RestClient

What is RestResource?

It is a distributed component which provides business services to other applications over the network.

What is RestClient?

RestClient can be human being who can send the HttpRequest.
RestClient can be Postman, Soapui, or any UIComponent.

Note: Writing url pattern at the method level is optional

1. If we don't write url pattern for a particular request type then default will be taken

eg:

```
@GetMapping
public String welcomeMessage() {
    String msg = "Welcome to Restful Services from Ineuron...";
    return msg;
}
```

2. If we have 2 methods with no url pattern then we need to have a url pattern which is compulsory

```
@GetMapping("/welcome")
public String welcomeMessage() {
    String msg = "Welcome to Restful Services from Ineuron...";
    return msg;
}
```

```
@GetMapping("/greet")
public String greet() {
    String msg = "Good Afternoon";
    return msg;
}
```

3. It is always recommended to write both class level and method level url pattern through which we can achieve unique url pattern and avoid ambiguity problems.

HttpMethods

- 1.GET ----> Selecting a Record from Database(Read operation)
- 2.POST ----> For Creating a Record(Insert Operation)
- 3.PUT ----> For Updating a Record(Update Operation)

4.DELETE --> For Deleting a Record(Delete Operation)
5.PATCH --> For Updating a Record(Update Operation)
6.TRACE
7.HEAD
8.OPTIONS

Note:: PUT for Complete updation/modification of Record.
PATCH for Partial modification of Record.

eg::

```
@PutMapping("/employee")
public <RT> updateEmployee(Employee emp){

}

@PatchMapping("/employee")
public <RT> updateEmployeeByEmail(String email){

}
```

HEAD Request mode HttpServletResponse does not contain body/output/results so it can't be used for Read/Select Operation.
TRACE Request mode is given to trace/debug components involved for SUCCESS or FAILURE of request processing.
It can't be used for CRUD operation.
OPTIONS Mode request gives the possible HttpRequest methods/modes that can be used to generate the request to webcomponents. Its HttpServletResponse purely contains ListModes/methods that are possible to give the request to webcomponents.

eg#1.

```
@WebServlet("/test")
public class TestServlet{

    public void doGet(request,response){

    }

}
```

If we give options mode as the request to this webcomponent, then we will get the response as

Allow :: GET,HEAD,OPTIONS

eg#2.

```
@WebServlet("/test")
public class TestServlet{

    public void doPost(request,response){

    }

}
```

If we give options mode as the request to this webcomponent, then we will get the response as

Allow :: POST,OPTIONS

a. GET -> It is used to get a data from the Database.

It is called as Idempotent Request that is even if we send the request multiple times the output won't change.

When should i bind method to Http GET?

If the client is sending the request to get some information from the resource then we need to use GET.

eg: client wants to get the product data based on id.
client wants to get the bookInfo based on id.
client wants to get the Employee data based on id.

To send the request along with the data we need to use

a. Query parameter

`@RequestParam("variableName")` it is mandatory to send it through url pattern

```
//http://localhost:9999/welcome/msg?id=10&name=sachin
eg:@GetMapping("/msg")
public ResponseEntity<String> greetMessage(@RequestParam("id") Integer
id, @RequestParam("name") String name) {

}
```

The data will go to the server in the form of String, but the data gets converted to respective type by Dispatcher Servlet during PreProcessing phase using filters.

Note: `@Query` Paramter can be made as optional in SpringMVC and also in SpringRest

```
//http://localhost:9999/welcome/msg?id=10&name=sachin
@GetMapping("/msg")
public ResponseEntity<String> greetMessage(@RequestParam(value =
"id", required = false) Integer id,
                                         @RequestParam("name") String
name) {

}
we can send multiple query parameters through a seperation called
"&".
```

b. Path parameter

They are called as "URI Paramteres".

They are used to send the data from client to server in the URL

eg: `www.ineuron.in/course/{nitin}/{springboot}`

Path parameters can be in anywhere in the url, where as Query paramters should be at the end of the URL.

eg: `www.ineuron.in/course/{nitin}/java/{springboot}`

To use PathVariables in RestController we use "`@PathVariable`".

```
eg: @RestController
    @RequestMapping("/welcome")
    public class WelcomeController {

        //http://localhost:9999/welcome/msg/10/sachin
        @GetMapping("/msg/{id}/{name}")
        public ResponseEntity<String> greetMessage(@PathVariable
Integer id, @PathVariable String name) {

        }

    }
```

Note: 1. We can have 1 path parameter

eg: `@GetMapping("/msg/{name}")`

2. We can have 2 path parameter
eg:@GetMapping("/msg/{id}/{name}")
3. We can have path parameter in between also.
eg:@GetMapping("/msg/{id}/course/{name}")

Note:

@RequestParam -> In Spring framework to hold query parameters(http://localhost:9999/test?name=sachin&courseId=10)
 @PathVariable -> In Spring framework to send the data from client to server (/name/{courseId})
 @PathParam -> This annotation is not from SpringFramework,it is from JAX-RS.

Difference b/w QueryParameter and PathParameter?

QueryParameter

- a. It is used to send the data from client to server in the URL
- b. It represents the data in key value pair(K=V)
- c. It should always be at the end, starts with ? and seperated with &
- d. It will be collected in the RestController through @RequestParam,it can be made optional using required=false

PathParamter

- a. It is used to send the data from client to server in the URL.
- b. It does not represents the data in the key value pair(/url/{variable}).
- c. It need not to be always at the end,it can be anywhere.
- d. It will be collected in the RestController through @PathVariable,it is compulsory always.

Case1::

```
@GetMapping("/info/{id}/JRTP/{name}")
URI :: course/info/5/JRTP/navinreddy/telusko/microservices
Output:: 404 Error
```

Case2::

```
@GetMapping("/info/{id}/JRTP/{name}")
public ResponseEntity<String> getCourseDetails(
    @PathVariable Integer id,
    @PathVariable String name) {
```

////////

}

```
URI :: course/info/navinReddy/JRTP/5
Output:: MethodArgumentTypeMismatchException
```

Case3::

```
@GetMapping("/info/{id}/JRTP/{name}")
public ResponseEntity<String> getCourseDetails(
    @PathVariable String id,
    @PathVariable String name) {
```

////////

}

```
URI :: course/info/navinReddy/JRTP/5
Output:: Success(not always, it might disturb the logic)
```

