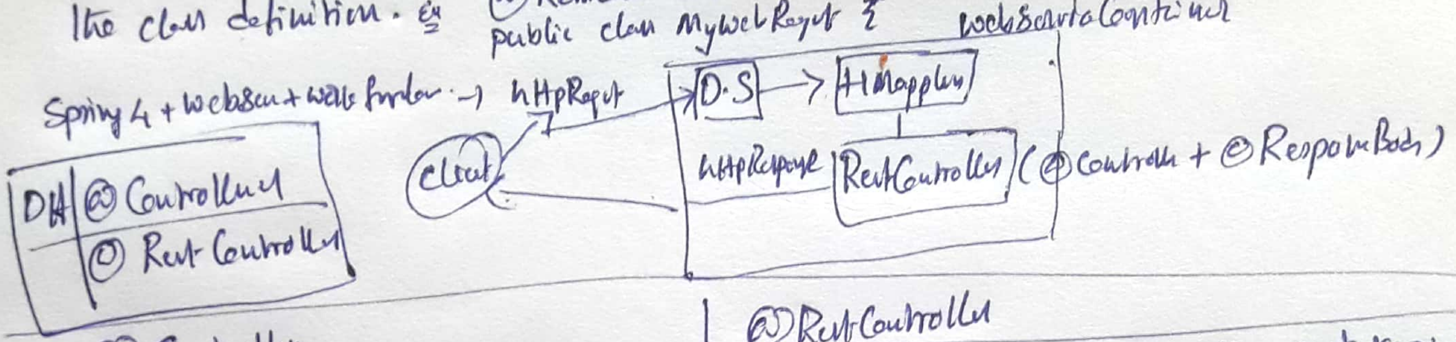
Spring REST :-

@RestController → org.springframework.web.bind.annotation.RestController (Spring 4.0)

→ Spring REST webser., HTTP requests are handled by a Controller. To make a java class as Controller for handling restful webser., just add @RestController annotation just above the class definition. Ex

```
@RestController
public class MyWebReport {
```



@Controller	@RestController
→ Each method in the Controller class must be annotated with @ResponseBody	→ In Case of @RestController not required to annotate @ResponseBody in class methods because by default this active.

@ResponseBody → org.springframework.web.bind.annotation.ResponseBody.

→ when you use the @ResponseBody an. on a method, spring converts the return value and writes it to the http response automatically.

conl :- if @Controller class response methods need to be annotated with @ResponseBody.

Ex :-

```
        @Controller
        @RequestMapping("employee")
public class EmployeeController {

    public @ResponseBody Employee getJson(@Path){
        =
    }
```

if @RestController → by Default then will be active @ResponseBody. not required explicit declaration on method.

@ RequestMapping :- [org.springframework.web.bind.annotation.RequestMapping]

annotation is used to map web requests onto specific handler [Classes | and@ | methods]

it can be applied to the Controller [Class | methods]

① @RequestMapping in class level
method —

```
@RestController
@RequestMapping("/hello")  } [class].
public class SpringRestController {          [method level].
    @RequestMapping(value="/{name}", method = RequestMethod.GET)
    public String helloMethod(@PathVariable String name) {
        String result = "Hello" + name;
        return result;
    }
}
```

URL Uniform resource locators [URI + URN] URI - Uniform resource Information
URN = Uniform Resource Name

→ Defined as absolute address of a web page/resource on the Internet. It is also called the web address.

URL - Structure :- protocol + DomainName + Port + Path to file/resource

Ex http://www.build-your-website.co.uk:80/html/index.htm
protocol — DomainName — port — path to file/filename

protocol = http | domainname = www.build | port-number = TCP/UDP socket @port-Number. normally use port 80 as default.

path to file = html/index.htm.

→ Absolute link → executes ~~by~~ including all essential elements
ex: http://build-your-website/starting-html.

→ A Relative link @ Relative path → tells the browser the resource position relative to the current position in web page. Ex /starting-html/

③ @Requestmapping with multiple URI

```
@RequestMapping(value={"/method1", "/method2/second"})
@ResponseBody
public String method1() {
    return "method1";
```

(note) if you are used @Controller mandatory to write @ResponseBody, in case of @RestController not req.

③ **RequestMapping with HTTP method** :- if diffrent Operations based on HTTP Methods used, even though request URI remains Same.

Ex
① @RequestMapping (Value = "/method2", method = RequestMethod.POST)
public String method2() { return "method2"; }

② @RequestMapping (Value = "/method3", method = {RequestMethod.Post, RequestMethod.GET})
Public String method3() { return "method3"}

④ **@Request-Mapping with Headers** :-

//Header @ multi-Header
@Request Mapping (value = "/method4", method = RequestMethod.GET,
headers = {"type = ACTIVITY", "quantity=3"})
public String getLatestCustomerInfo() {
return "method4"; }

⑤ **With Consumes and Produces / multi-produces**

Produces
It's a method @ field level annotation, this tells which MIME type is delivered by the method annotated with @GET. it means whenever we send a HTTP GET request to our Restful Service, it will invokes particular method and produces the output in diffrent formats.

produces the output in diffrent formats.
We will use @Produces annotation for GET requests only.

(NOTE) We will use @Produces annotation for GET requests only.

⑥ **Consumes** :- it is a class and method level annotation, it uses to define which MIME Type is consumed by the particular method. it means in which formate the method can accept the input from the client.

@RequestMapping (value = "method5", produces = {"application/json", "application/xml"},
consumes = "text/html")
public String method5() { return "method5"; }

In above consume message only with Contented-Type as text/html
produce message type application/json and application/xml

Scanned by CamScanner

@ PathVariable :- for dynamic URI (path value with input Parameters)

→ automatic type Conversion / single path variable
→ auto-detected (same text)
→ regular expression

path resource.

ex URL → http:// localhost : 8080 /spring-rest / ex/ foos/1

path to file / path resource.

→ automatic type / single path

```
@ RequestMapping (value = "/ex/foos/{id}", method = GET)
public String pathMeth( @pathVariable ("id") long id){
        return "Myval" + id ;
```

→ auto-detected / multiple path

```
@ RequestMapping ("/ex/{name}/{id}")
public String getPathAuto (@pathVariable String name,
                           @ Path Variable long id ) {
      return "MyName" + name + "id" + id;
```

→ regular expression

```
@ RequestMapping (value = "/ex/foos/{numericId:[\d]+}",
                  method = GET)
public String getPathReg (@pathVariable long numericId) {
       return "Str" + numericId;
```

Default Mapping

```
@RequestMapping()
public String defaultMapp(){
    return "defaultMap";
```

ex If we want a method for Controller Class URI (/customer) it means without Parameters is helpful.

ex localhost : 8080/ customer/

All class Mapping

It will handle the case no matching handler method is detected for the request.

ex
```
@RequestMapping ("*")
public String fallbackMeth (){
    return "404";
}
```

@ RequestParam -> used to bind request Param values to the handler method arguments in Controller.

@Target(value = PARAMETER)
@Retention(value = RUNTIME)
@Documented
public @interface RequestParam

@ -> attributes

defaultValue -> use as a fallback when the request Parameter is empty value then default value will be using.

name :- it is string type attribute and name of the request Parameter to bind to.

required :- it is boolean type attribute if the request parameter is mandatory @ required.

value :- it is a String type attribute and it is alias for name attribute.

Ex:- https://localhost:8080/ EnDD/login.do? userName="Hi" & Password = xxxx

```
@RestController
public class UserLogin {

@RequestMapping ( "/login")
public String userLoginMethod ( @RequestParam ( default = "RY88419", required = true,
                                name = "userName") string userName)

   @ RequestMapping (value = "/hello")
value -> public String userLoginMethod (@ RequestParam ("id") String id) {
```

URL-will be -> /app/ hello? id = ky 100