Passing parameters to a REST controller (@PathVariable, @RequestParam, @RequestBody)

@RequestMapping is one of the most widely used **Spring MVC** annotation. **org.springframework.web.bind.annotation.RequestMapping** annotation is used to map web requests onto specific handler classes and/or handler methods.

@RequestMapping can be applied to the controller class as well as methods. Today we will look into various usage of this annotation with example and other annotations **@PathVariable** and **@RequestParam.**

@RequestMapping with Class

We can use it with class definition to create the base URI. For example:

```
@Controller
@RequestMapping("/home")
public class HomeController {

}
```

Now /home is the URI for which this controller will be used. This concept is very similar to servlet context of a web application.

@RequestMapping with Method

We can use it with method to provide the URI pattern for which handler method will be used. For example:

```
@RequestMapping(value="/method0")
@ResponseBody
public String method0(){
        return "method0";
}
```

Above annotation can also be written as @RequestMapping("/method0"). On a side note, you can using @ResponseBody to send the String response for this web request, this is done to keep the example simple. You will use these methods in Spring MVC application and test them with a simple program or script.

@RequestMapping with Multiple URI

We can use a single method for handling multiple URIs, for example:

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

If you will look at the source code of RequestMapping annotation, you will see that all of it's variables are arrays. We can create String array for the URI mappings for the handler method.

@RequestMapping with HTTP Method

Sometimes we want to perform different operations based on the HTTP method used, even though request URI remains same. We can use @RequestMapping method variable to narrow down the HTTP methods for which this method will be invoked. For example:

```
@RequestMapping(value="/method2", method=RequestMethod.POST)
@ResponseBody
public String method2(){
        return "method2";
}

@RequestMapping(value="/method3",
method={RequestMethod.POST,RequestMethod.GET})
@ResponseBody
public String method3(){
        return "method3";
}
```

@RequestMapping with Headers

We can specify the headers that should be present to invoke the handler method. For example:

```
@RequestMapping(value="/method4", headers="name=pankaj")
@ResponseBody
public String method4(){
        return "method4";
}

@RequestMapping(value="/method5", headers={"name=pankaj", "id=1"})
@ResponseBody
public String method5(){
        return "method5";
}
```

@RequestMapping with Produces and Consumes

We can use header **Content-Type** and **Accept** to find out request contents and what is the mime message it wants in response. For clarity, @RequestMapping provides **produces** and **consumes** variables where we can specify the request content-type for which method will be invoked and the response content type. For example:

```
@RequestMapping(value="/method6",
produces={"application/json","application/xml"}, consumes="text/html")
@ResponseBody
public String method6(){
        return "method6";
}
```

Above method can consume message only with **Content-Type** as **text/html** and is able to produce messages of type **application/json** and **application/xml**.

@RequestMapping with @PathVariable

RequestMapping annotation can be used to handle dynamic URIs where one or more of the URI value works as a parameter. We can even specify Regular Expression for URI dynamic parameter to accept only specific type of input. It works with **@PathVariable annotation** through which we can map the URI variable to one of the method arguments. For example:

```
@RequestMapping(value="/method7/{id}")
@ResponseBody
public String method7(@PathVariable("id") int id){
        return "method7 with id="+id;
}

@RequestMapping(value="/method8/{id:[\\d]+}/{name}")
@ResponseBody
public String method8(@PathVariable("id") long id, @PathVariable("name") String name){
        return "method8 with id= "+id+" and name="+name;
}
```

@RequestMapping with @RequestParam for URL parameters

Sometimes we get parameters in the request URL, mostly in GET requests. We can use @RequestMapping with @RequestParam annotation to retrieve the URL parameter and map it to the method argument. For example:

```
@RequestMapping(value="/method9")
@ResponseBody
public String method9(@RequestParam("id") int id){
        return "method9 with id= "+id;
}
```

For this method to work, the parameter name should be “id” and it should be of type int.

@RequestMapping default method

If value is empty for a method, it works as default method for the controller class. For example:

```
@RequestMapping()
@ResponseBody
public String defaultMethod(){
        return "default method";
}
```

As you have seen above that we have mapped **/home** to **HomeController**, this method will be used for the default URI requests.

@RequestMapping fallback method

We can create a fallback method for the controller class to make sure we are catching all the client requests even though there are no matching handler methods. It is useful in sending custom 404 response pages to users when there are no handler methods for the request.

```
@RequestMapping("*")
@ResponseBody
public String fallbackMethod(){
        return "fallback method";
}
```