

94: Accepting Query Params using @RequestParam Annotation

Whenever we are making a request using get method, which means I'm making an HTTP Get request to the backend, we can append some parameters inside our URL itself.

As you can see here there is an URL localhost:8080 /holidays.

So as of now, inside our web application, whenever we say /holidays, it is going to display all type of holidays like federal and festival.

But now I have a new requirement that I want to build inside our web application, which is if someone is sending a request to the backend by mentioning festival is equal to true and federal is equal to true. In that scenario, I will display both federal and festival holidays, whereas if they send only festival is equal to true but not federal is equal to true, I will not display federal holidays.

I will only display festival holidays in that scenario, and the same vice versa also can be possible.

Here we have two examples where I'm making a request to the backend path which is /products, but to that I can send query params like I want my products to be ordered by popularity.

That's the very first URL indicates.

Whereas the second URL indicates I want the products to be displayed based upon the popularity and also I want only the new products to be displayed, not the old products.

https://localhost:8081/products?order=popular

https://localhost:8081/products?order=popular&filter=new

```
https://localhost:8081/products?order=popular
https://localhost:8081/products?order=popular&filter=new
```

Path

path is up to products, so after path. we need to start the very first query param(key=value) by using question mark. We can separate multiple query params using & Symbol

```
<div class="col-lg-2 col-md-3 col-6 ps-lg-5 ps-lg-4 footer-list-29 mt-md-0 mt-4">
  <ul>
    <li>
      <h6 class="footer-title-29">Explore</h6>
      <a th:href="#/holidays/festival=true,federal=true" >Holidays</a>
      <a th:href="#/contact" >Privacy policy</a>
      <a th:href="#/contact" >Contact Us</a>
    
  

```

Even though we have not added `?`, `&` Thymeleaf will internally add these things.

```
@GetMapping("/holidays")
public String displayHolidays(@RequestParam(required=false) boolean festival
    @RequestParam(required=false) boolean federal, Model model) {
    List<Holiday> holidays = Arrays.asList(
        new Holiday(" Jan 1 ", "New Year's Day", Holiday.Type.FESTIVAL),
        new Holiday(" Oct 31 ", "Halloween", Holiday.Type.FESTIVAL),
```

In spring MVC or in spring boot we can use `@RequestParam` annotation either to accept the query params that we are getting in the `getURL()` or we can also use it to accept the form data that we are going to receive HTML pages.

```
@GetMapping("/holidays")
public String displayHolidays(@RequestParam(required=false,name="festival") boolean fest,
                              @RequestParam(required=false) boolean federal, Model model) {
    List<Holiday> holidays = Arrays.asList(
```

← ↻ ⓘ localhost:8081/holidays?festival=true&federal=true

```
<div class="row">
  <div class="col-lg-6" th:if="${festival} == true">
    <h5 class="sub-title timeline"><i class="fas fa-snowman"></i> Festival Ho
  <div class="timeline">
```

localhost:8081/holidays?festival=false&federal=true

 **George School**
 **Federal Holidays**
 **Martin Luther King Jr. Day**

```
public String displayHolidays(@RequestParam(required=false) boolean festival,
                              @RequestParam(required=true) boolean federal, Model model) {
    model.addAttribute("festival", festival); // Adding the Values to the Model so
    model.addAttribute("federal", federal);
}
```

localhost:8081/holidays?festival=true

This application has no explicit mapping for /error, so you are seeing this as a fallback.

There was an unexpected error (type=Bad Request, status=400).
Required parameter 'federal' is not present.

As the federal value is mandatory but we not specified the value it is resulting in an Exception.

@RequestParam Annotation

easy
bytes

- In Spring @RequestParam annotation is used to map either query parameters or form data.
- For example, if we want to get parameters value from a HTTP GET requested URL then we can use @RequestParam annotation like in below example.

http://localhost:8080/holidays?festival=true&federal=true

```
@GetMapping("/holidays")
public String displayHolidays(@RequestParam(required = false) boolean festival,
                              @RequestParam(required = false) boolean federal) {
    // Business Logic
    return "holidays.html";
}
```

✓ The @RequestParam annotation supports attributes like name, required, value, defaultValue. We can use them in our application based on the requirements.

- ✓ The name attribute indicates the name of the request parameter to bind to.
- ✓ The required attribute is used to make a field either optional or mandatory. If it is mandatory, an exception will throw in case of missing fields.

- ✓ The value attribute is similar to name elements and can be used as an alias.
- ✓ defaultValue for the parameter is to handle missing values or null values. If the parameter does not contain any value then this default value will be considered.

96:Accepting Path Params using @PathVariable Annotation

@PathVariable Annotation

easy
bytes

- The @PathVariable annotation is used to extract the value from the URI. It is most suitable for the RESTful web service where the URL contains some value. Spring MVC allows us to use multiple @PathVariable annotations in the same method.
- For example, if we want to get the value from a requested URI path, then we can use @PathVariable annotation like in below example.

http://localhost:8080/holidays/all
http://localhost:8080/holidays/federal
http://localhost:8080/holidays/festival

```
@GetMapping("/holidays/{display}")
public String displayHolidays(@PathVariable String display) {
    //Business Logic
    return "holidays.html";
}
```

- ✓ The @PathVariable annotation supports attributes like name, required, value similar to @RequestParam. We can use them in our application based on the requirements.

```
@GetMapping("/holidays/{display}")
public String displayHolidays(@PathVariable String display, Model model) {

    if(display!=null&&display.equals("all"))
    {
        model.addAttribute("festival",true);
        model.addAttribute("federal",true);
    }
    else if(display!=null&&display.equals("festival"))
    {
        model.addAttribute("festival",true);
        model.addAttribute("federal",false);
    }
    else
    {
        model.addAttribute("festival",false);
        model.addAttribute("federal",true);
    }
}
```

all coming from the front end will be stored as a String.

If path variable is
all -> then both the festival and federal holidays will be displayed.
festival -> Then Only Festival holidays would be displayed
federal --> Then Only Federal Holidays would be displayed.

<a th:href="@{/holidays/all}" >Holidays
<a th:href="@{/contact}" >Privacy policy
<a th:href="@{/contact}" >Contact Us

As we could see here currently it is /all indicating that both the festival and federal holidays should be displayed.
From the above path all will be taken as PathVariable.

Here as we could see display is variable into which the path variable is mapped and based upon the value of display corresponding functionality will get invoked.

when we were passing input values as Parameters or taking the data from FORM we can use @RequestParam whereas if we are passing using path then we can go with Path Variable.

From the path we can get the variable value and from that we go ahead with the further functionality.