MVC and Web

Query Parameters:

In Spring Boot, query parameters pass data from the client to the server through the URL. Query parameters are appended to a URL's end and preceded by a question mark. Each query parameter consists of a key-value pair, separated by an equals sign, and an ampersand separates multiple parameters.

Here's an example of a URL with query parameters:

```
http://example.com/search?q=spring+boot&sort=rating
```

In this example, the query parameter "q" has a value of "spring boot", and the query parameter "sort" has a value of "rating".
In a Spring Boot application, you can easily retrieve query parameters using the @RequestParam annotation. For example, let's say you have a REST endpoint that accepts query parameters for a search:

```
@GetMapping("/search")
public String search(@RequestParam("q") String query,
@RequestParam("sort") String sortBy) {
    // Search logic here
    return "Search results for " + query + ", sorted by " + sortBy;
}
```

In this example, the search method accepts two query parameters: "q" and "sort". The values of these parameters are retrieved using the @RequestParam annotation and are passed as arguments to the method. The method then returns a string containing the search results, including the query and sort parameters.

When this endpoint is accessed with a URL like http://example.com/search?q=spring+boot&sort=rating, the search method will be called with the values "spring boot" for the query parameter and "rating" for the sort parameter.

Model VS ModelMap

- "Model" and "ModelMap" are both classes used in the Spring Boot framework for Java to represent data that can be used to render views in web applications.

- The main difference between the two is that Model is an interface that defines a contract for objects that can be used to pass data between a controller and a view. It's a simple key-value store that allows you to store and retrieve values using a string key.

- On the other hand, ModelMap is a concrete implementation of the Model interface and provides additional functionality to store and manipulate data. It is also the default implementation used by Spring Boot when passing data between a controller and a view.

- In practice, both Model and ModelMap can be used interchangeably. However, ModelMap offers more advanced features, such as the ability to set a default value for a key not present in the map or copy the contents of one ModelMap to another.

Here's an example of how to use Model and ModelMap in a Spring Boot controller:

```java
@RequestMapping("/example")
public String example(Model model) {
    model.addAttribute("message", "Hello World!");
    return "example";
}

@RequestMapping("/example2")
public String example2(ModelMap model) {
    model.addAttribute("message", "Hello World!");
    return "example";
}
```

In both cases, the controller is adding a "message" attribute to the model or model map, which can then be used by the view to display the message. The only difference is that the first example uses the Model interface, while the second example uses the ModelMap implementation.

Support Page:

After registering the User we need to create a new page called the support page, and to do that, we need a support controller and support JSP file. We need to have a User id on that support page. We will see all these in the coming path with the following concept.

SignUpController

```
@RequestMapping("/registerUser")
public   String   createdUser(@ModelAttribute(value   =   "user")   StudentUser
studentUser) {
            int      userId      =      userService.signUp(studentUser.getName(),
studentUser.getGender(),studentUser.getLocation(), studentUser.getCollege());
  if(userId != -1)
  {
     ModelAndView modelAndView =new ModelAndView("redirect:welcome?id="+userId);
     return modelAndView.getViewName();
  }
  return "signup";
}
@RequestMapping("/welcome")
public String showWelcomePage(@RequestParam("id") String userID,ModelMap map) {
  map.addAttribute("userID",userID);
  return "welcome";
}
```

Support Controller

```
@RequestMapping("/support")
   public  String  getSupportPage(@RequestParam("id")  String  id,ModelMap
map){
      map.addAttribute("message","Welcome to support page "+id);
      return "support";
   }
}
```

Welcome.jsp

```
<html>
<h1>Thank you, you are now a member</h1>
<a href="/support?id=${userID}">Contact support</a>
<a href="/instructors">my Instructors</a>
</html>
```

Support.jsp

```
<h1>Support Page</h1>
<div>${message}</div>
<% String a = "rahul is using java here"; %>
<%=a%>
```

Model vs ModelAndView vs ModelMap

Here's a brief overview of each:

- **Model:** Model is an interface that provides a simple way to pass data to a view. It's a key-value store that allows you to store and retrieve values using a string key. The Model interface is implemented by the controller and passed as a parameter to the controller method. The values set in the Model object are made available to the view.

- **ModelAndView:** ModelAndView is a class that combines a model with a view name. It provides more flexibility than the Model interface, allowing you to set both the model and the view name in a single object. ModelAndView can also add information to the response, such as HTTP headers or a status code.

- **ModelMap:** ModelMap is a class that implements the Model interface and provides additional functionality. It's similar to a Map and allows adding and retrieving values using a string key. ModelMap provides additional convenience methods, such as the ability to set a default value for a key not present in the map.

In practice, all three classes can pass data to a view, but the choice between them depends on the application's requirements. If you need to pass data to a view and specify the view name separately, you can use ModelAndView. If you just need to pass data to a view, you can use Model. If you need to add additional functionality to the model, such as setting a default value or copying the contents of one model to another, you can use ModelMap.

Path param

A path parameter, also known as a path variable, is a variable that is part of a URL path in a web application. In the context of RESTful web services, path parameters are used to identify a specific resource.

In a RESTful web service, the URL is used to identify a resource, and path parameters provide additional information about the resource. For example, consider a URL that represents a user profile:

```
https://example.com/users/{userId}.
```

In this URL, {userId} is a path parameter that identifies the specific user profile to retrieve. When a request is made to this URL with a specific userId, the server retrieves the data for that user profile and returns it in the response.

Instructor Page:

Signup Controller

```
@RequestMapping("/instructors")
public String showInstructorsPage(ModelMap map) {
  //creating instructors using hash maps
  HashMap<String, Object> instructor1 = new HashMap<String,Object>();
  instructor1.put("name", "Rahul Mohan");
  instructor1.put("age",23);
  instructor1.put("id",243);

  //ArrayList for a list of instructors
      ArrayList<HashMap<String,   Object>>   listOfInstructors    =    new
ArrayList<HashMap<String, Object>>();
  listOfInstructors.add(instructor1);

  // sending key value pairs to the view
  map.addAttribute("instructors",listOfInstructors);
  return "instructors";
}
```

Instructor.jsp

```
<%@page import="java.util.HashMap"%>
<%@page import="java.util.ArrayList"%>
<html>
```

```
<h1>Here are you instructors:</h1>
<%
  ArrayList<HashMap<String,Object>> listOfInstructors =
  (ArrayList<HashMap<String,Object>>)request.getAttribute("instructors");
  for(HashMap<String,Object> instructor: listOfInstructors) {
%>
name <%=instructor.get("name")%>
<a href="profile/<%=instructor.get("id")%>">profile</a>
<%
  }
%>
</html>
```

.

Profile page:

Now, adding the profile page

SignupController

```
@RequestMapping("/profile/{profileID}")
public String ShowProfile(@PathVariable("profileID") String profileID,ModelMap
map) {
  map.addAttribute("profileID",profileID);
  return "profile";
}
```

Profile.jsp

```
<html>
  <h1>Profile Page</h1>
  <div>id from path param:${profileID}</div>
</html>
```

Conclusion

In conclusion, Query Parameters and Path Parameters are two important ways of passing data in a URL for a web application. Query Parameters are used for non-essential data that doesn't change the output of the endpoint, while Path Parameters are used for essential data that alters the output.
A Model is an object that holds data and is used by the controller to interact with the view. It separates the concerns of business logic and presentation logic. ModelView and ModelAndView are two popular approaches for returning data from a controller to a view in Spring MVC.
ModelView is a simple object that contains both the data model and the view name, while ModelAndView contains both the model and view information as separate properties. Both approaches provide flexibility in designing and structuring web applications with Spring MVC.

Overall, understanding the differences and use cases of Query Parameters, Path Parameters, Model, ModelView, and ModelAndView is crucial for building efficient and scalable web applications.

Instructor Codes

- Website Application

References

1. Request Param

2. Model VS ModelView VS ModelAndView