Section 6 Spring MVC and Controller. Relation between them.

🌿What is Spring MVC?

🌿It is the framework within the Spring framework which is used to build the web- applications.

🌿It follows the MVC architecture pattern.

🌿**This pattern separates into three different components:**

1️⃣Model → This represents the business logic.

2️⃣View → This represents the UI and handles it.

3️⃣Controller → Processes user input and update the model and views.

🌿What is Controller?

🌿The controller in the spring mvc is the java class that can handle the incoming HTTP request. It process them and routes them to the respective model of the remaining application program.

🌿It acts as the bridge between model (that is the business logic) and the view (that is the user interface from where the user interacts with the application).

🌿This controller classes are marked with the annotation **@RestController. or @Controller annotations. If we want the response in JSON or XML format then we use @RestController and if we want the returning response in JSP or Thyme leaf then we annotate it with @Controller.**

🌿Below is the example of the sample Controller class:

```
@RestController
@RequestMapping("/api')
public class ControllerClass{

      @RequestMapping("/hello")
      public String GetAllData(){
               return "Hello World";
               }
}
```

🌿Let's see what is happening here.

1️⃣What is @RestController?

🌿@RestController is the annotation which is the combination of two stereotype annotations that are @Controller and @ResponseBody.

🌿Here the @Controller annotation is responsible for processing the incoming HTTP request and the @ResponseBody is used for processing the result in the form of JSON or XML format.

🌿If we use @Controller and forget to use the @ResponseBody then spring will give an error if it couldn't find the view.

🌿But if we directly use the @RestController then spring will automatically return the output in JSON or XML format.

2️⃣What are the different types of request that we make to the application?

🌿Now HTTP request are of few types such as follows:

🚀Get Request → Gets all the data. It is used to fetch the data from the server. It doesn't modify the data.

🚀Post Request → For creating the new resource. This requires the Request Body to create the new resource in the DB.

🚀Put Request → Updates and Replaces the Existing resource.

🚀Delete Request → Delete the Resource.

Type of Request	Idempotent	What is meaning of being Idempotent and Non-Idempotent?
Get	Yes	Making same subsequent Get request will give us the same output without making any changes in DB
Post	No	Making the same subsequent Post request will keep on creating the new resources and it is also the duplication.
Put	Yes	Making the same subsequent Put request will update the existing record only once and wont change anything after that.
Delete	Yes	Once the record is deleted , and we are running the same request again of deleting the same data it won't change anything.

3️⃣How does the application know what type of request is being sent to it ?

@RequestMapping is the annotation which is used to map the request to the specific method inside the controller class.

💡 Imagine that you are going to the big shopping center:

This shopping center has different sections example Section A for trekking gear, Section B for swimming gear, Section C for cycling gear, Section D for customization, Section E for return.

🚀Now you have came there and you want to buy the trekking gear, At the main entrance of the shopping mall there is a man standing and he is directing you to the section A.
🚀Similarly someone is coming for swimming gear the same guard is guiding towards the swimming section.
🚀This goes to the all other section that man is guiding everyone to the section that will serve their interest.

Now compare this customers to the different types of http requests. GET, POST, PUT, DELETE. And the sections are the methods which are serving that types of specific requests. So the man standing there and guiding everyone is nothing but the @RequestMapping annotation.

There is one more catch here. If all the inside sections are the methods then the whole shopping mall which is acting as an entry points for all these customers request is the controller class. Right? So the address to reach this mall is same right? So this address is also handled by the @RequestMapping. And this address prefix is written on the class.

We will see this in an example in next section.