Advanced explanations

Content

Advanced explanations 1
EmployeeController 1
EmployeeRepositry 2
Employee 2
EmployeeServiceImpl 2

EmployeeController

(1) @Controller defines the class as an MVC controller, this means that this class controls the flow of logic between frontend and backend.
(2) @RequestMapping(“/employees”) defines this class’s request mapping as {domain}/employees/ instead of {domain}/ which is the standard. This is very useful if your site contains multiple controllers.
(3) @GetMapping(“/list”) defines that method’s mapping as /list, making it accessible through {domain}/employees/list. This also defines that the method uses a GET method.
(4) The Model is a functionality that allows you to send information to a html file for it to be used in some way there. You attach data to the model by giving it an attribute .addAttribute(“attributeName”, attributeValue). When the method returns a html file, this model is also sent to the html file to be utilized.
(5) @RequestParam(“employeeId”) int theId looks for a parameter named “employeeId” and converts it to an integer for use in the method. This parameter can be seen in the URL of the application. Example: {domain}/employees/showFormForUpdate?**employeeId=2**
(6) @PostMapping(“/delete”) is very similar to (3), the difference is that this is a POST mapping which means that the method receives information whilst GET mapping means that the method provides information.

EmployeeRepositry

(1) The employee repository is a data access object that allows the application to access a database. In this case we're using JPA repository which works well with the spring-boot framework and comes with many built in applications.

Employee

(1) @Entity defines the class as an entity, this means that I represent a table in a database. The entity tag is followed by the @Table(name="tableName") annotation that defines exactly which table in the database that it represents.
(2) @GeneratedValue(strategy=generationType.IDENTITY) is an annotation that defines a variable as a generated variable, in this case it means that the variable is gets it value generated by the database automatically so there is no need to give it an id in the logical part of the application.

EmployeeServiceImpl

(1) An Optional variable is a variable that doesn't need to be initialized, it kind of acts as a try-catch. You can try to define it, but if an error occurs and it doesn't get defined, nothing happens. It just remains uninitialized. You can check if the optional variable was defined by using optional.isPresent(), this will return a true or false value. You can also extract the variable in the optional variable by doing optional.get(). You need to do this because Optional<variableType> is **not** the same as variableType.