**BİR ÖNCEKİ DERSTEKİ KODU GÜNCELLEME**

(SpringbootDemoApplicationLaptop3\_DependencyInjection’daki)Use method within laptop class for battery A(“Laptop” sınıfına bakarak anlayabilirsin) . This means that evertime you run your code, yur code will use batteryA. It is not bad, but very restricted. Kodu çalıştırdık. Can yo see, everytime it runs, it uses BatteryA.

If yo want t use BatteryA and BatteryB if you want to make your code flexible, what can you do ? Please come to BatteryI. Can you see? I created two more methods (public default void use(BatteryA bA and BatteryB bB). Remember, BatteryI is an interface. How can I put concrate methods inside the interface? I used “default” keyword. When yo use “default” keyword, you can put body inside your interface.

Come to mainApplication class. Write this code (BatteryA bA = con.getBean(BatteryA.**class**);) and Inject "bA" inside the use method(l1.use(bA);) to use BatteryA as parameter. Code is complaining, why? Because that method is in “BatteryI” class, it is not in “Laptop” Class. How can I fix that one ? Come to Laptop Class.I can implement BatteryI inside the “Laptop” class. By typing that code(public class Laptop implements BatteryI), I make the “Laptop” class child of the interface(“BatteryI”). If “Laptop” class is child, it can use methods from the parent(In our case “BatteryI” interface). Come to Application class . Let’s do the same for BatteryB. 1’st Laptop will use BatteryA, 2’nd Laptop will use BatteryB. Let’s run the code. Can you see, it used BatteryA first, and then BatteryB as second.

**RESTFULWEBSERVİCE01**

We learned “Inversion of Control”, “Dependency Injection”, and a couple of annotations. Now , we’ll create one more SpringBoot project. In this project, we’ll use requests;mainly, “get” request, “pull” request, “post” request, “put” request. We’ll create the a new SpringBoot Project—Package Name: resfulwebservice01, Name: springboot\_restfull\_webservice, Dependencies: Spring Web, Spring Boot DevTools, Spring Data JDBC , Spring Data JPA, Oracle Driver.

When you look at the pom.xml file, you can see SpringBoot again gave me the original projects and dependencies ( Burada biraz pom’u keşfet). Everbody, as you see, I have two jdbc over here. First one’s groupId is “org.springframework.boot”, second one’s groupId is “com.oracle.database.jdbc”. No need to use the first one, you can remove it. We removed first one instead of second one because first one is more general, whereas the second one is more spesific. In this project, we need the spesific one. Don’t forget to File/ Save all and Right Click on springboot\_restfull\_webservice / Maven / Update Project.

In this project, we’ll work with the database(Oracle). Please be sure that your database is running. Under the package(springboot\_restfull\_webservice/ src/main/java/restfulwebsevice01) we’ll create a controller(dedi ama class oluşturdu) yani create SF01Controller class. What is controller? When you(user) type sth into the browser(www.google.com), this is request. Browser understands your request in the controller layer, and it sends ou a response. When you send request, there is a controller layer over here. Browser understands your request in the controller layer, namely, what kind of request you sent, for example

\* **Get Request**. If you are sending a Get Request, it means you want to read something(such as the website you typed-www/google.com—which means that you want to read google.com).

\* **Post Request**, you want to create new data on the application(such as entering your basic info and clicking on the Register button on Facebook).

\* **Put Request**, you want to update the data you created(such as changing info—location-- on your Facebook profile). In put request, you have to include the all parameters.

\* **Patch Request**, Put Request is fully updated, whereas the Patch Request is partially updated. In Patch Request, you can only include the parameter that was being modified. *If there is a question about the difference btw Put & Patch Request, go here :* [*https://stackoverflow.com/questions/28459418/use-of-put-vs-patch-methods-in-rest-api-real-life-scenarios*](https://stackoverflow.com/questions/28459418/use-of-put-vs-patch-methods-in-rest-api-real-life-scenarios)

\* **Delete Request**, you want to delete the data you created. You want to delete your account on Facebook, a.k.a. you want to remove your data on Facebook.

All those requests are in the **controller layer.**

After the controller layer, you’ll have the **“Service Layer”**. It contains the logic of the Controller Layer. If you want, you can put the logic inside the Service Layer, but as you know, in coding, making everything seperated is better. What is “logic” anyway ? It is the set of rules you defined for the requests, such as when you want to tweet on Tweeter, it should be 280 character long maximum.

After the service layer, you’ll have the **“Security”.** After the security, lastly, you’ll have the **“Database**”. From the Database, data will come to the user.

Thus, firstly, I created the Controller Class.(SF01Controller’la devam ediyoruz) How can you create a controller class? At the beginning, you need to use the @Controller or @RestController annotation. When you create the @Controller or @RestController, SpringBoot understands that you’ll use the requests on that code.By the way, you can either use @RestController or @Controller, but since @RestController has more feature on it, it is better.

* After putting @RestController, let’s go with the get request. We created a get method(request)(public String getMethod() {return “Get Request is taken…”; }), but it'll return just a string, nothing more. We’ll go easy on the beginning, since you need to understand the structure first. Next, we created the "return"--Get Request is taken...-- to understand getMethod() is taken.

spring.jpa.database-platform.org.hibernate.dialect.Oracle10gDialect

spring.jpa.hibernate.ddl-auto=update

spring.datasource.url=jdbc:oracle:thin:@localhost:1521/orcl

spring.datasource.username=hr

spring.datasource.password=oracle

spring.pa.show-sql=true

spring.jpa.properties.hibernate.format\_sql=true

spring.datasource.driver-class-name=oracle.jdbc.OracleDriver

server.error.include-message=always

* How can SpringBoot understand getMethod() is for Get Request? You should declare it. How can you declare it? By
  + 1) @RequestMapping(method = RequestMethod.GET, path=”/getRequest”) Come to src/main/resources/application\_properties.In order to see the message("Get Request is taken...") on return command, After that, run the SpringbootRestfullWebServiceApplication. Burada hata alıyorum, Nuriye Hocaya yazdım; çözüm için dönütünü bekliyorum
  + 2) Or, you can just use @GetMapping annotation, which is shorter, thus recommended
* You can insert some HTML code inside the getMethod() and make the things more colorful.
* Can you see the return message from your localhost? Yes, you can.. Check it out from here: localhost:http://localost:8080/getRequest
* getMethod() just returns string, sometimes we want to return objects.
* Come to restfulwebservice01 package. Create a SF02ControllerBean Class. By the way, Bean means object. I’ll create an object and work with it. Let’s create the private String msg; Variable. If I make it private, I need to add the getters and setters. Then, let’s create a constructor(SF02ControllerBean(String msg){this.msg = msg;}). To create constructor, we need to use the class name. When I create an object, I'll send a message, and this part(**this**.msg = msg) will assign the message to msg variable. Then, I’ll create toString() method because I want to see the details after I create the object. Source/Generate toString… Fields will be enough. I’ll change that one(SF02ControllerBean [msg) with Message because no need to see the classname.
* Not, come to the SF01Controller Class. I’ll use @GetMapping(path=“/getBean”) public SF02ControllerBean getMethod() {return new SF02ControllerBean(“This is get request to return message object…);}after 2nd way. We used different URL(“/getBean”). You cannot use the same URL more tha once. Inside the same class, if you want to use same methods name, the parameters should be different. Thus, we need to set the first method’s name getMethod1() and second method's name as getMethod2()Come to main method, ya da diğer bir ismiyle the SpringbootRestfullWebServiceApplication,and run the code. Yukarıdaki hatadan yine patladı.Çözüm için Nuriye Hanımın cevabını bekliyorum. There is no error, let’s check the localhost.Type localhost:8080/getBean to your search engine. That one should be displayed the message. Did you see? It displayed. Let’s look at the other one, Type localhost:8080/getRequest. See, it worked perfectly fine. So far, we worked only with Controller Layer. We haven’t worked with Service Layer yet.
* A picture containing timeline

  Description automatically generatedI’ll create one more get method (@GetMapping(path="/getParameterized/{name}")), and it ‘ll return the page according to the URL. If the Url ends with Tony, it’ll return Tony and Tony related things. How can we do it?(public SF02ControllerBean getMethod3(@PathVariable String name{return new SF02ControllerBean(String.format(“Hey, %s this is your page…”, name))) Inside the method paranthesis, you need to use getMethod2(@PathVariable String name). If I type localhost:880/getParameter/Tony.

Run the code, and type localhost:880/getParameter/Tony to your browser.

Burada da G/Ç hatasını aldık Can you see, it says Hey (Hangi ismi koyarsan), this is your page

Now, it is time to create getmethod that returns the list of objects. Use @GetMapping(path = "getList/{name}") annotation.Use (public List<SF02ControllerBean> getMethod4(@PathVariable String name) {return List.of(new SF02ControllerBean(String.format(“Hi %s”, name)), new SF02ControllerBean(String.format(“How are you %s”, name)), new SF02ControllerBean(String.format(“ %s do you want to drink coffee”, name))); We added List<> so that it can return list of objects. Don’ forget to import your list from java.util. Run the code. Burada da G/Ç hatasını aldık

Type it to your browser: localhost:8080/getList/Tony

Can you see? It has []. It means that it is a list. We have also {}. This means that we are creating an API, and output is a JSON data.

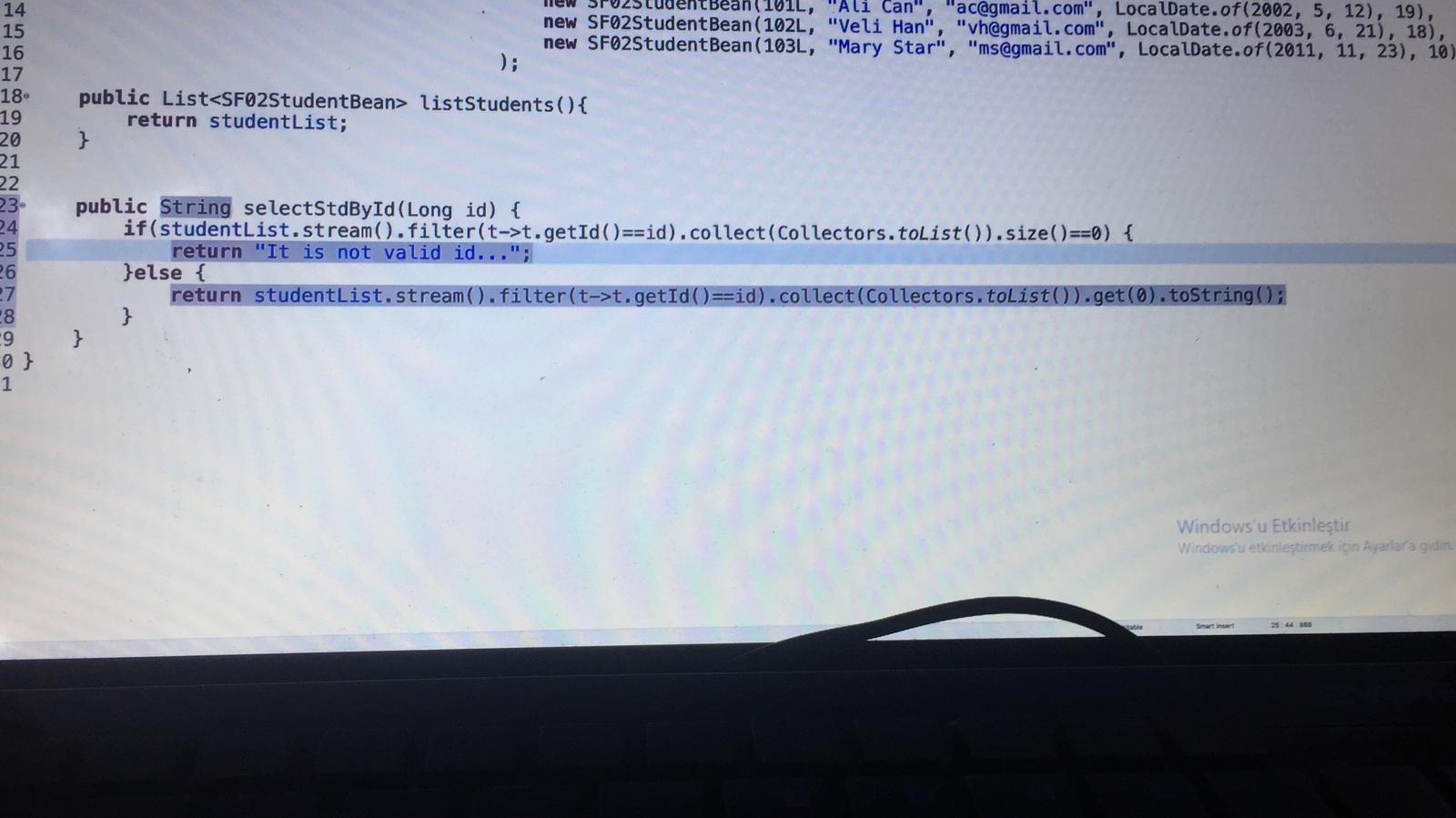
Actually we just worked with Controller Layer. That code above for example, should be in the Service Layer. Craete one more package and name it as restfulwebservice02 . Inside the package, we’ll create classes and use Controller and Service Layers. Firstly, create the SF02StudentBean Class. Within this class, we can create (private Long id;). To use different datatypes, I made id’s type as “Long”. Student should have a name, so let’s create “private String name;”. We can also put “private String email;” and “private LocalDate dob;” –Local Date- , and “private Integer Age;” . So, we have 5 variables. Since they are private variables, we should create getters and setters for them. But before that, let’s generate a constructor. Use all fields within the constructor. Don’t forget to pick insertion point as “Last Member”. You can remove super(), we won’t use it. Now, it is time to generate getters and setters. At the end of the page, let’s create toString method() according to the code . When we create an object, it should be displayed on the console.

After creating StudentBean Class, let’s create the Service Layer. Since the logic will be in the Service Layer, I have to create the Service Layer first. Restfulwebservice02/SF02Service Class . Let’s create the logic of getMethod() named as “listStudents(İÇİNDE ID OLAN VERSİYON)” Let’s set the rules as follows; id = 101, as id's data type is long, we add L. name = "Tommy”. email = [tommy@gmail.com](mailto:tommy@gmail.com). day = dob = LocalDateof(Year,Month, day). age = 27 (public List<SF02StudentBean> ListStudents(){return List.of(new SF02StudentBean(101L, “Ali Can”, [ac@gmail.com](mailto:ac@gmail.com), LocalDate.of(2002, 5, 12), 19), new SF02StudentBean(102L, “Veli Han”, [vh@gmail.com](mailto:vh@gmail.com), LocalDate.of(2003, 6, 21), 18), new SF02StudentBean(103L, “Mary Star”, [ms@gmail.com](mailto:ms@gmail.com), LocalDate.of(2011, 11, 23), 10))};). listStudents returns list of students. . Normally, listStudents() method goes the database and fetches the data from the database. Then, you see the data on the browser.But for now, to make the things simple and more understandable,we’ll go easy and focus on to learn how the Controller and Service Layers work.

Let’s create the Controller Class SF02Controller. For Controller Class, we use annotation @RestController(above the public class SF02Controller{})—we chosed it instead of @Controller since @RestController has more features. Since we used annotation for “Controller Class”, we should use annotation for “Service Class” as well otherwise how can SpringBoot know SF02Service is a Service Layer, and SF02Service has logic that we’ll use in the Controller Layer.Open the SF02Service We use @Service(public class SF02Service in üstüne) annotation for Service Class.We can also use @Controller as Service Class annotation but since @Service is more readable & more spesific, we’ll go with @Service annotation.

How can Controller Class talk with Service Class?? How can I access to Service Class from Controller Class? By creating an Service Class Object private SF02Service studentService; inside the SF02Controller Class.. We should also create a constructor inside the studentService; this.studentService = studentService; Now let’s create the getMethod(@GetMapping(path=”api/v1/students”)). I need to use @GetMapping annotation. path = "api" because the request that we have talked about today are for API. v1 = version1 as we create API right now. After you created API and want to update it, you can use "v2" or "v3". That API is for "students". Maybe in the same API, I 'll get "parents" as well. It’ll return the list of students by **public** List<SF02StudentBean> getStudents1(){ **return** studentService.listStudents();}

See, there is no logic in Controller Class since we seperated it by Service Class. That way is much better.We cannot run this code because as you see, inside the package there is no main method. Go to the restflwebservice01, copy SpringbootRestfullWebserviceApplication and paste to the restflwebservice02. Run the main class as usual. **Yine G/Ç hatası aldık** Type localhost:8080/api/v1/students See, everyting works well. Come to the Service Layer(SF02Service). Let’s create a different method, that will get the students according to their names. Come back to SF02Sservice and listStudents(){} ın dışına create public SF02StudentBean selectStdBId(Long id){}; studentList() isimli yeni bir list oluşturdu ve listStudents’ın içindekileri(101L,102L,103L) studentList’İn içine yapıştırdı. listStudents() ın içine de return studentList; i yazdı.. Let’s put a logic inside the public SF02StudentBean selectStdBId(Long id).{studentList.stream().filter(t->t.getId()==id).collect(Collectors.toList()).get(0);} In studentList, I have 3 students. Those are Tommy,Arthur, John. When you use stream() it put them from top to down. By using filter(t->t.getId()==id), I’m saying to Java that Hey Java, get the id if id equals to given id,and return it. Then, I put them into a list with collect(Collectors.toList()). Inside the list, I’ll have just a single student, because of that, I return the first one by get(0). From the list, if you want to choose element, lambda is the best way. If you don’t want to use lambda, you can use for loop for each element as well.

Come to the Controller(SF02ControllerName) now everybody. I’ll create one more method called as getStudentById.( @GetMapping(path = “api/v1/students/{id}”) public SF02StudentBean getStudentById(@PathVariable Long id) {return studentService.selectStdById(id);}) Run the main method. I shall se just Tommy, not the others on the console. See, Tommy is return. Open localhost:8080/api/v1/students/101. I shall see just Tommy, not the others on the console. See, it worked. If you gow with localh2ost:8080/api/v1/students/102, you’ll see Arthur. If the tr 104, you’ll receive an error. In order not to receive error, type that code:

Graphical user interface, text, application, email

Description automatically generated

Don’t forget to update SF02Controllername as follows:

That’s it for today. In this batch(geçen 2 ders) we learned IOC, Dependency Injection, Scope Annotation, Autowired Annotation, how to create API by using just controller and Controller & Service Layers.

selectStdById(id) olmasına rağme Java görmüyor..