1) Download spring initializr as setting the Group Name com.techproed, Spring Web & Spring Boot DevTools, all the other options are default

2) Copy your downloading under the eclipse-workspace

3) Open Eclipse

4) File / Import / Maven / Existing Maven Projects / Browse / eclipse-workspace / springboot\_demo.xml

5) open src.pom.xml

6) Biz techproed isimli projeyi indirdik ancak kendisi otomatik olarak springframework.boot isimli parent projeyi de indirmiş dedi. Inheritence kavramından bahsetti sonrasında.- parent class handles the basic features. You can work on those basic features and improve them-

7)-Spring Boot için inherit edilecek parent features- When you work with springboot projects, all projects

1. should connect to the database

Create U.I: all projects need to have user interface(eg. amazon.com is a user interface)

Security

- This parent project has those. When you put your own springboot project, Java put those features automatically.

8) Springboot framework is like getting franchise. You need to choose and decide basic things such as logo, brand or color of your store unless you use franchise. If you use franchise, you don't need to think basic things. You can start to create value immediately.

9) When you use springboot, all dependencies are created automatically. You dont need to create dependencies by opening the maven & working on sth else.

10) There is no version on the dependencies section of pom.xml. Springboot 'll find the relevant version for you.

11) src/main/java => com.techproed.springboot\_demo => altındaki şeye tıkla

Normally, when I create a project on Eclipse, I need to create the main method by myself. But when I work with the SpringBoot, it creates for me.

In springboot, we'll use annotations.(@SpringBootApplication gibi) By using them, thing will become very easy

In springboot, **dependency injection** is very important concept and in your interviews,most probaby, they will ask you as what is the dependency injenction? or what is the version of controll IOS.

Slaytlara geçti.. Tight Coupling: Parts that you cannot seperate from the main(e.g. laptop) Loose Coupling: Parts that you can seperate from the main(e.g. laptop) Which one is better? Loose Coupling.

1)So that every company can do their own task-- laptop makers can make laptops and battery companies can make batteries.

2) If battery is broken, you don't need to change your laptop, you can just change your battery.

In maintanence and testing, loose coupling is better as well.

Slayttan devam ediyor

So far, when we create objects, we use **new** keyword. If you use new keyword, it means tight coupling. -- Laptop class is tightly coupled to BatteryA class. If you want to create BatteryB class, you need to either create new laptop class or new BatteryB object within this class. Thus, it is not possible to use BatteryA and BatteryB on the same class. But the thing is, springboot found a solution for this problem as using the loose coupling in the class and objects

Koda geçti

11) Open the eclipse -- Slaytlara ara-- Create a laptop and battery class on com.techproed.springboot\_demo --sağ kliğe tıklayarak yapıyoruz bunu-- Don't select the main method because springboot gave me the main method

12) SpringFramework send your application to your localhost automatically by TomCat. TomCat i de springboot\_demo/pom.xml de ilk dependency olarak bulabiliriz

13) Lets create two variables from the Laptop class: Battery(b) is my battery object and price

14) To create getters and setters: Source / create getters and setters => ikisini de seç => create

Let’s continue with SpringbootDemoApplication

Lets create the Battery Class also (constructor olmadan)

Lets create the Battery Class also (constructor u inşaa et)

SpringbootDemoApplication ile devam et

Laptop class için constructor inşaa et

(BatteryS I sonra inşaa edeceksin, unutma!)

Run the code and see the result from the console

TIGHT COUPLING BITTI, LOOSE COUPLING E GEÇTIK

15) Loose Coupling example is worked without “**new”** method; why ?

When you create springboot project, springboot gives you an object container.

Bir şey daha ekledik(l2), programı durdurduk ve tekrar çalıştırdık. Can you see something weird over here ? (deyip kod çıktısının en sonundaki 6 satırı işaret ettik). When I create 2 objects(l1,l2) constructor(Çıktısı Laptop is created... olan) should run twice, but it used once. If there is an ongoing laptop object(l1), springboot goes only with existent one, doesn't create a new one -- bu iki de olsa 3 te olsa fark etmiyor. This feature is very beneficial in the manner of memory and speed(**singleton concept**: from a class, you can only have a one object).

New keyword'ünü her kullandığında yeni bir object yaratırsın.

Kod bloğunun başında @Scope("prototype")(LaptopLoose’da) container ın içindeki object üretimini durdurur. We don't use it with Sİngleton

Şimdiye kadar “@Component ve @Scope annotation” larını, “loose ve tight coupling” i ve “singleton concept”'i öğrendik

15) BatteryS isimli başka bir sınıf daha oluşturduk.

16) Laptop'u tekrar manipüle ediyoruz. Orjinali bozmamak için Laptop3 adlı yeni bir sınıf oluşturdum ve onunla devam ediyorum.

17) Laptop3 ün içinde BatteryS isimli yeni bir variable oluşturduk. Bu variable için de getters & setters oluşturmamız gerekiyor ki oluştururken Insertion point'ini After 'setPrice(int)' olarak set ettik.

18) bS ve BatteryS in aynı şey olduğunu springboot nasıl anlıyor? By **autowire;** autowire don't look at the names, it looks at the datatypes

19) If you want to select object by their names, you can do it by using @Qualifier. In this case, first, SpringBoot changes BatteryS as bS, then goes with it.

@Component("bS") // To use @Qualifier("bS"), we need to change @Component with Component("bS")

20) Our ObjectContainer and SpringBootDemoApplication are seperate, not fixed. But they work perfectly fine. This is called **loose coupling**

21) If you do object creation in another process, this is called **inversion of control(IOC)**

As you see,in SpringBoot Object Creation is done in Object Container and the application is created in another part. This is called "Inversion of Control"

22) Dependency Injection : When you run the application, objects come from Object Container. Object Container is outside the application, it means our application is getting sth from outside to inside, this is "Injection".Because of that, this process is called "Dependency Injection". Dependency Injection made SpringBoot pretty famous.

23) So far, we made dependency injection by using getters and setters. We can also make dependency injection by using consturctors.

In order to do that, let's create com.techproed.springboot\_demo2 package and lets rename com.techproed.springboot\_demo as com.techproed.springboot\_demo1.

Now, lets create one more Laptop class

Lets create an interface name it as BatteryI. As you know, interfaces have abstract methods.

Lets create BatteryA and BatteryB Classes

Let's create use and turnOnLaptop() methods.

After completing Laptop Class, please come to BatteryA and put the "price" variable

Let's create a constructor inside the Laptop class as "Laptop"

Come to BatteryA & B and put their constructor

Let's create an object for Laptop Class

Dont forget to pt the @Component annotation in Laptop class, on the top of the public class Laptop { ...

If I create 2 more objects, how many messages that I'll see on the console? Just 1 due to Singleton

If I create any object, will I see any message? - When I use Component annotation, SpringBoot thinks that I'll use the object so it makes ready for me.

Let's write a use method inside the Laptop class. // Which use method is going to be used ? The one within the BatteryA or the one within the BatteryB?

Let's run and see from the console

It gave me an error, why ? Because I have to declare which use method that I am going to use

Laptop class'ını açtık ve private BatteryA bA yı tanımladık. bA.use() u da çalıştırdık

Don't forget to put Component for BatteryA

We need to kill the port:

netstat -ano | findstr :8080(8080 benim port numaram)

taskkill /PID 3696 /F(3696 benim listenining değerim)

Don't forget to use @Autowired annotation