Dependency Injection & layering like a Pro

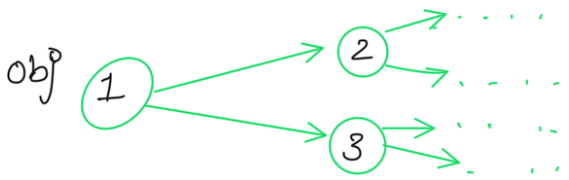
→ Understanding why and how

Spring Boot manages it's dependencies

→ And how to design a clean Backend structure.

Dependency Injection (DI):

→ We know that in java we create objects which may or may not depend on another objet. → called object graph



→ Normally we do this to creati object.

```
class Laptop {

    HardDrive obj1 = new HitachiHDD();
}
```

now, lets say I wish to change my harddrive from HitachiHD to Samsung HD. But for that I'll need to return back to this code and change object name from

new Hitachi() to → new Samsung();

"new Hitachi"; ⟶ "new Samsung";

Therefore this is Hardcoded and we do not wish to use a such tightly coupled hardcode.

Hence,
we use **Dependancy Injection Containers**

↓

They are responsible for creating an object for us.

Then they will be injecting in our class.

→ How do we use them?

we add → @ Component on top of the class.

↳ This makes the class a component of spring framework which is generated as per requirements.

and in the Laptop class, we can add
@ Autowired above the object refrence

Eg → class Laptop {

@ Autowired
HardDrive obj1; }

TESTING → For testing purposes we need to create a mock object to run and test the applicat?, in this case also DI is used to create that mock object.

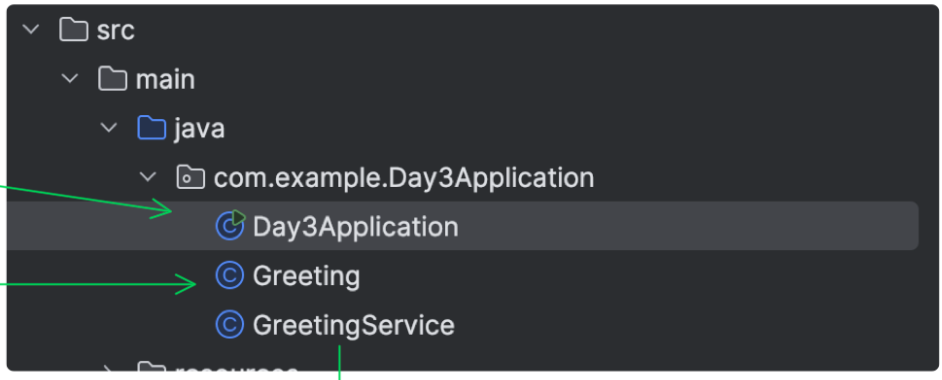
@Autowired → It's spring automatically inject required dependencies

Constructor Injection

→ It makes dependencies immutable (final)
→ Enables unit testing more easily.
→ Avoids hidden dependencies & null pointers.

Used to Acieve DI

Codebase →



(1) App to run file

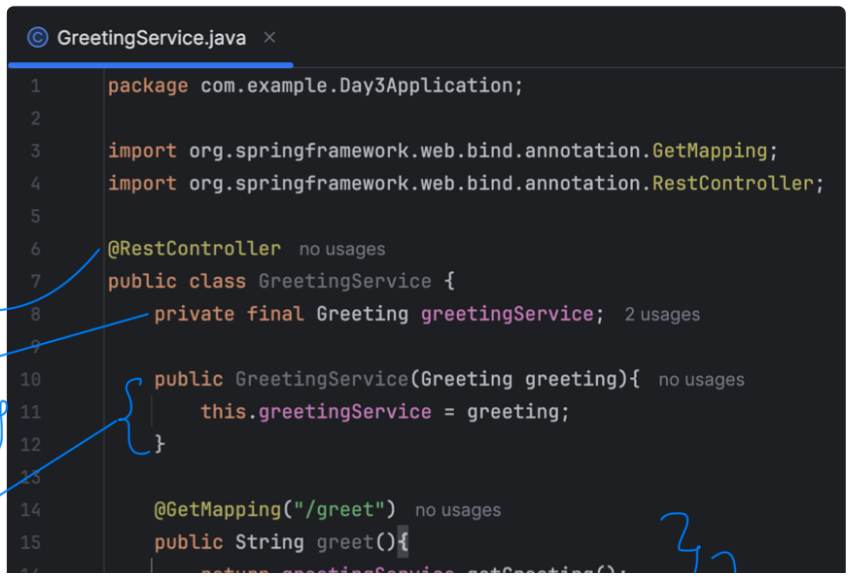
(2) function we wish to serve to user

(3) How the service is provided at Business level.

- Controllers should only call services and handle responses
Never write business logic inside controllers.

immutable refrence to Greeting

Constructor of this class taking



this class taking Greeting class as parameter and assigning objects

```
16          return greetingService.getGreeting();
17      }
18  }
```

mapping greet and for it to be used at "/greet".