



## Agenda

→ API REST

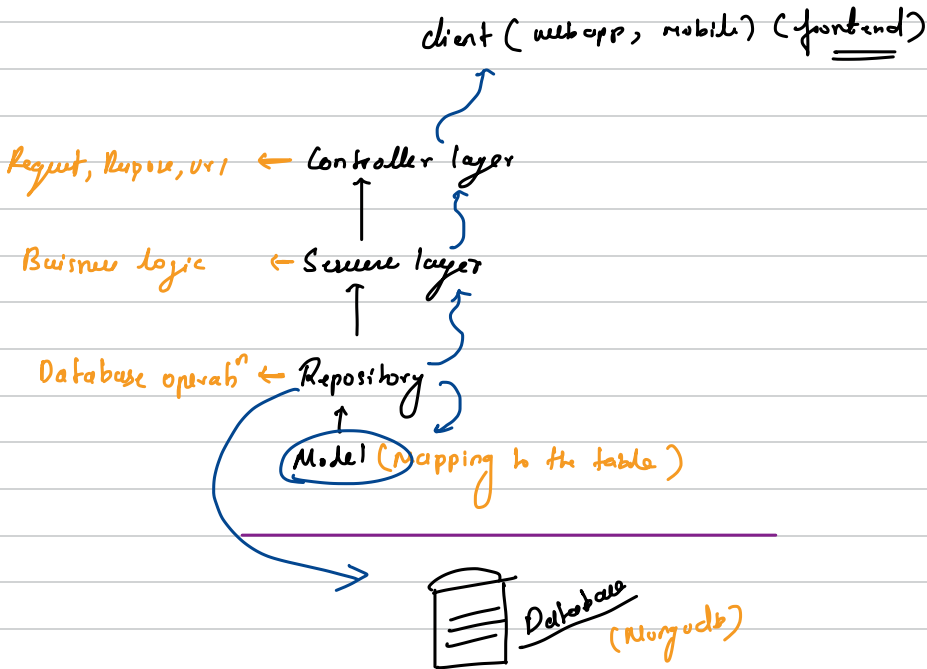
→ DTO layer

→            |

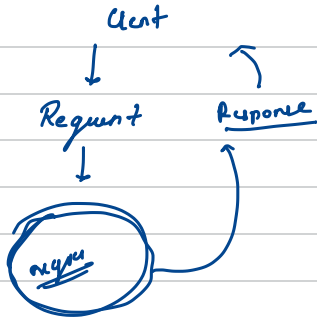
2) created spring boot (spring starter.io) + IntelliJ

2) Controller ; RestController.

3) . . .



Controller



Q Rest controller  $\rightarrow$  class is Ref. -

(c) Request Mapping  $\rightarrow$  base api path

GET → @GetMapping

POST  $\rightarrow$   $\odot$  Post Mapping

PUT → @ PathMapping

DELTA  $\rightarrow$  @ Delta Mapping.

get All Vehicle  $\rightarrow$  List of Vehicle model density expanding model

get Vehicle By Id → Response Entity < Vehicle > " /vehicle/1  
( @PathVariable ) 1 argument.

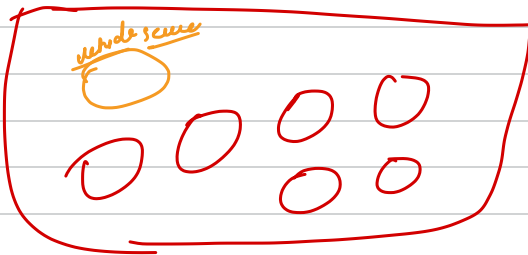
createVehicle → " " "

[ @RequestBody ] ← argument.

② Auto wired.

private Vehicle Service service;

## IOC Container



Component scanner

Service days      Banner logic

Source:

```
public class VehicleService {
```



Repository layer : Database operations using help of Model class.

dependency -> spring boot state data  
mongodb.

@Repository

public class VehicleRepository {

private MongoTemplate tmp;

- ↳ find() -> Vehicle.class
- ↳ findById (id, Vehicle.class)
- ↳ save (vehicle)
- ↳ remove()
- ...

## API Design principle

Open api

Nouns

GET

/api/v1/vehicles.

/api/v1/vehicles/1

/api/v1/vehicles?brand=bmw

/api/v1/vehicle?brand=bmw & year=2015.

x Verb x

x /api/v1/ get All Vehicles x

x /api/v1/ get Vehicle By Id x

/api/v1/ get Vehicle By Brand.

"/ get Vehicle By Brand And  
year.

scenario	status code	Spring way of writing it
GET ✓	<u>200</u>	<u>ResponseEntity.ok().build()</u>
CREATE ✓	<u>201</u>	RE status(201).body( <u>object</u> )
GET/Update	<u>404</u>	RE. not found().build();
POST: 2 ≡ ≡ ≡ s	<u>400</u>  <u>500</u>  <u>503</u>	RE. BadRequest().build();

→ DTO layer

Data Transfer object

User Model

id  
name

email

username

↑

password

address

pin

createVehicle ( Vehicle <sup>request</sup> )

Vehicle Request DTO

Vehicle Response DTO

Vehicle Response

