Class → 11

Agenda →

1. IOC [Inversion of control)
2. DI (Dependency Injection)
3. Beans
4. Annotations
5. MVC (Model View Controller)

Scenario →

1. You want to commute to your ofc.

(a). Drive via your car

(b). Someone else drop you to ofc everyday

Object Creation

① You can create a JAVA obj yourself

① You can

(new)

② Spring will do that job for you

we want spring to handle the obj lifecycle
(Initialisation, deinitialisation etc)

① IOC →

C1
C2
C3
C4
C5

Bean → ~~class~~
→ Object

object which spring will create

@ component

IOC → phenomenon by which spring
creates an obj for us instead

of we manully creationg that obj

IOC container / Spring Container / Spring context

Memory

Application context

Beans which are created by spring will be stored here

Synchronised {

Critical section

}

~~Hashmap~~ → X

IOC → Bean → IOC container

@Component

Application startup

100 classes

- ↳ P1
  - ↳ C1 – C50
- ↳ P2
  - ↳ C51 – C100

10 – 15 → @ component

compiled

MVC — Model View Controller

↳ Design Pattern

↳ Java Framework

→ Dispatcher Servlet to accept requests

↳ Map

Right resources

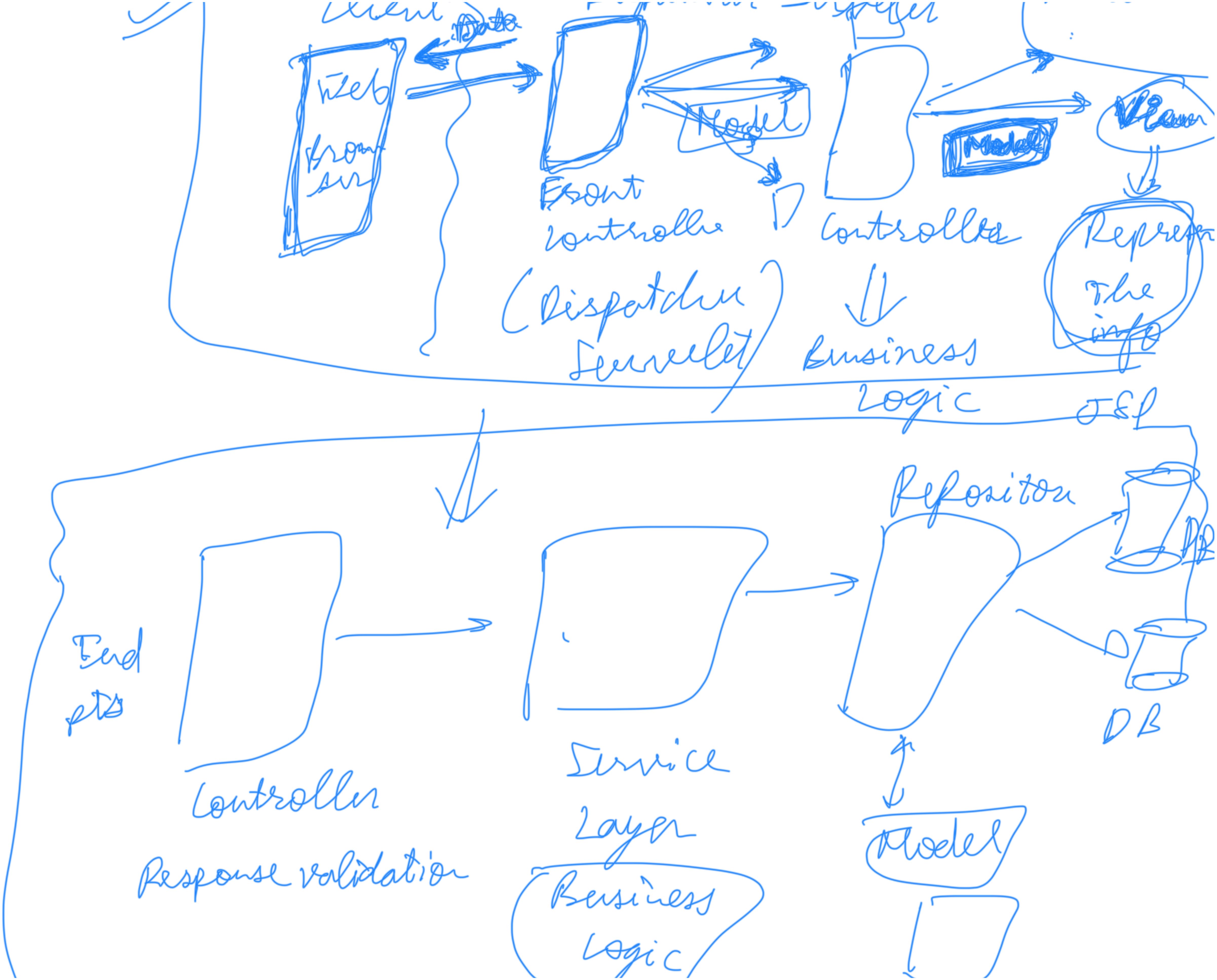
Controller, Model or View

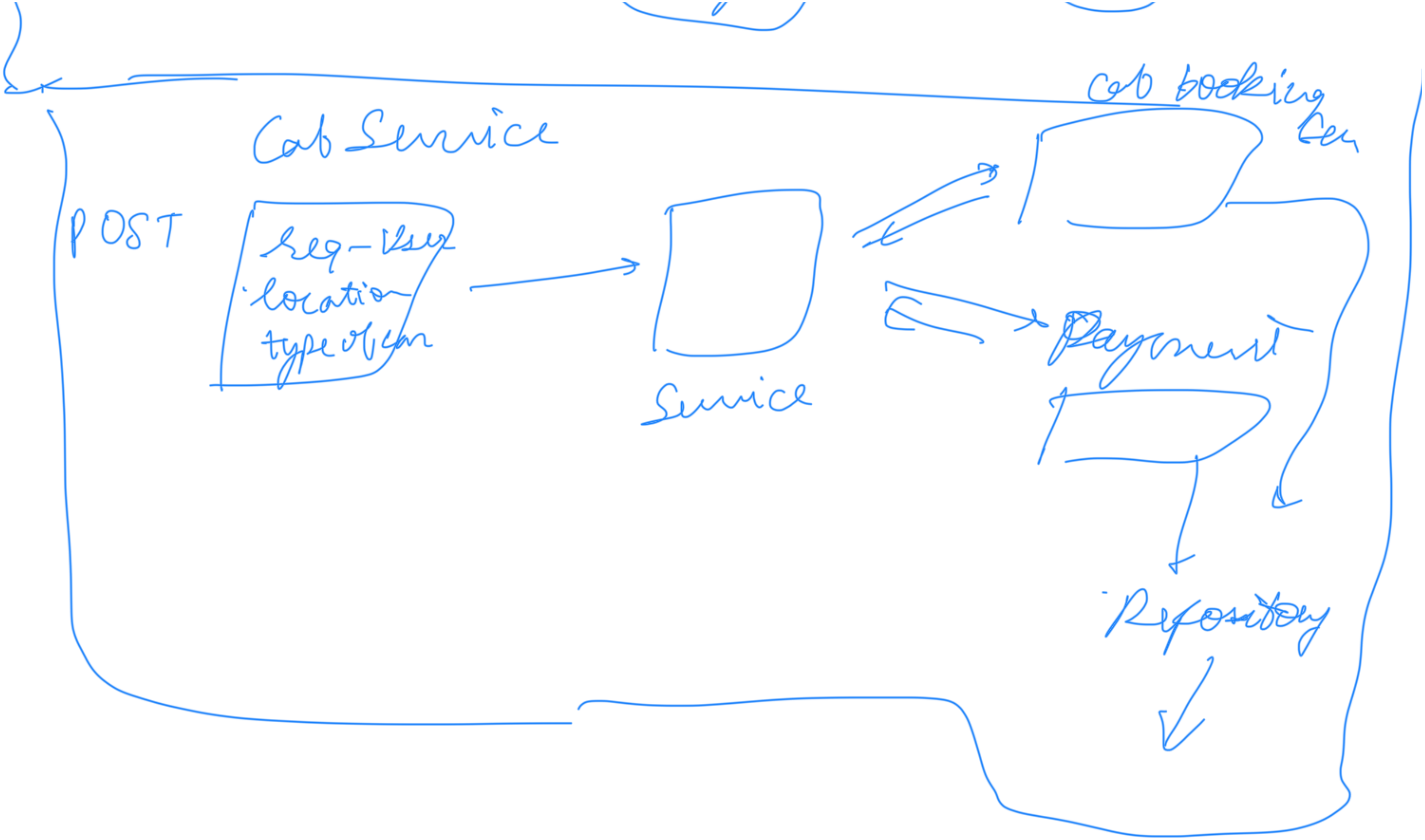
MVC

Client

Dispatcher Servlet

View


Client
Data
Web Browser
Front controller
(Dispatcher Servelet)
Model
Controller
Business logic
Model
View
Represn The info
JSP
End pts
Controller
Response validation
Service Layer
Business logic
Repositon
Model
DB
DB


Cab Service
POST
Req-User
location
type of car
Service
Cab booking Ser
Payment
Repository