Spring Boot

08 March 2025 08:59

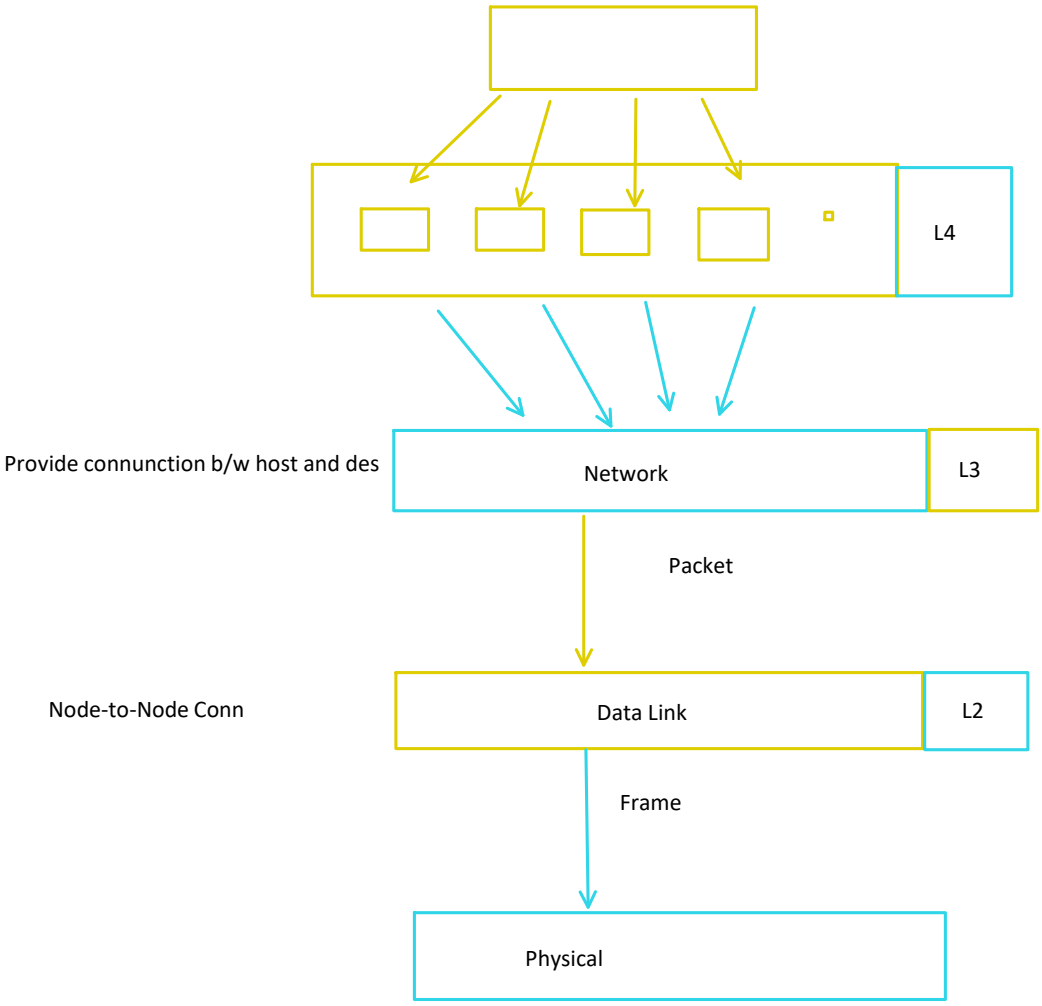
OSI Model

---

1. Physical
2. Data link
3. Network
4. Transport
5. Session
6. Presentation
7. Application

N/w engineers not intract with these layer

Transport: break large piece of data into small segments



Application : Http is an application layer protocol that provide web-based comm

Presentation: Responsible for data formatting (JSON,XML) Jackson for JSON conversion

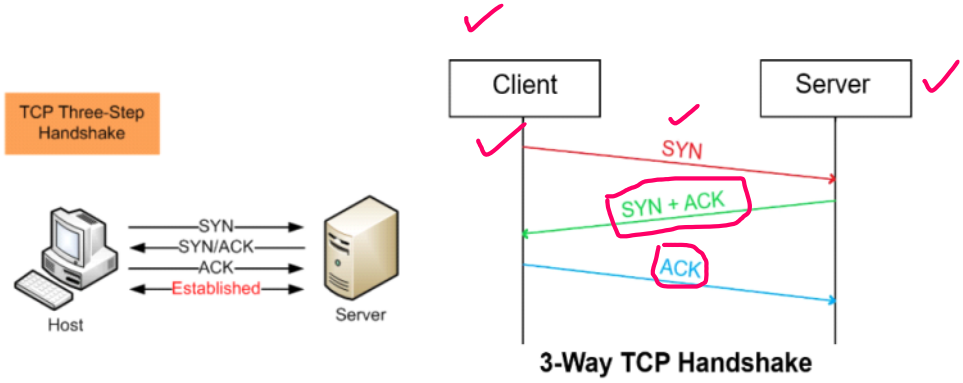
Session Layer: Mostly it is used for security

Transport layer: Uses TCP for reliable connection of HTTP req/res

Network Layer: IP handles addressing and routing http packets

Data Link Layer: handles MAC addressing and physical device comm
Physical Layer: Deals with h/w transmission

3 Way TCP handshake to establish a TCP Connection



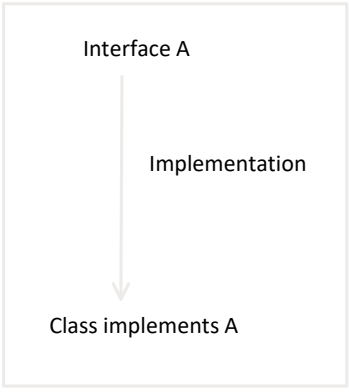
UNICMINDS

JDK 1.1(1997) -> Introduced --> @Deprecated
JDK 1.5(2004) -> introduced full annotation support

Spring, hibernate, JPA adopted annotation heavily

Syntax of making custom annotations

```
@interface MyAnnotation {
    no usages
    String value();
}
```




```
@interface MyAnnotation {
    String value();
}
```

It uses reflection API

```
@MyAnnotation(value = "Custom Annotation | Genie Ashwani")
public class Test
{
    public static void main(String[] args) {
        System.out.println("Hello sir ");
    }
}
```

Spring Boot
---

1. Spring boot is one approach to develop spring bases applications with less configurations

Spring boot = spring framework - xml configurations

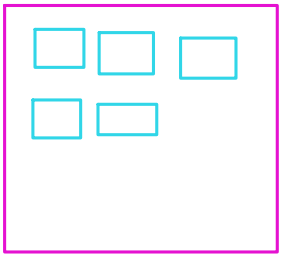
2. Spring boot is not replacement for spring. Spring boot is developed on top spring
3. All core and spring framework concepts can be used in spring boot also

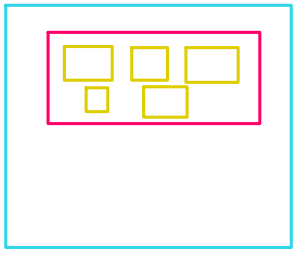
Advantages
---

1. Less configuration & No Xmls configurations
2. Pom Starters to simplify dependencies
3. Auto Configuration
4. Embedded Server
5. Actuator (Production ready features)

What is started ?

1. If you want to make one application then you will need multiple dependencies so spring has combined all these and make one dependency is called started





Eg: web-starter, jpa-starter, security-starter

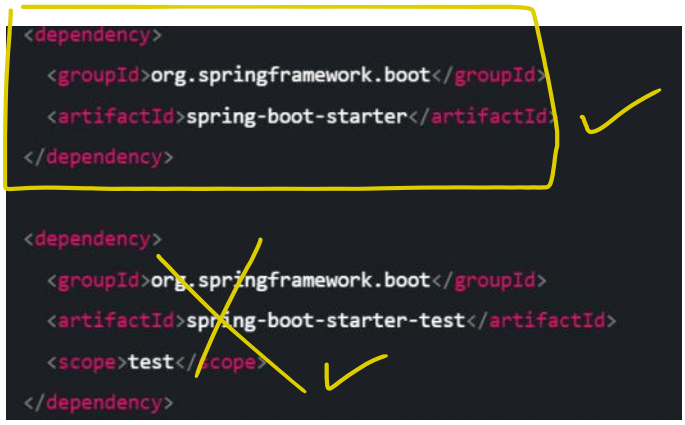
Note : Spring boot makes it easy to create stand-alone, production grade spring based application that you can "just run"

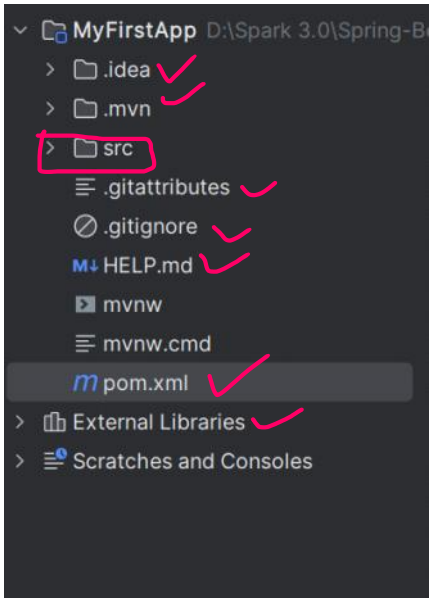
Spring boot 1.0 released in 2014
Current version of spring boot is 3.4.x ----> Nov-2025
Note: Java 17 is mandatory to work with spring boot 3.x version

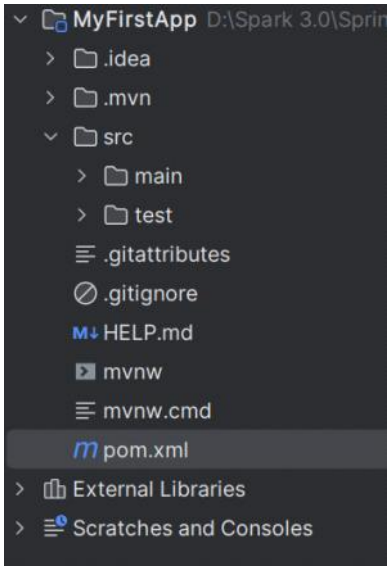
Spring boot Application creation
---

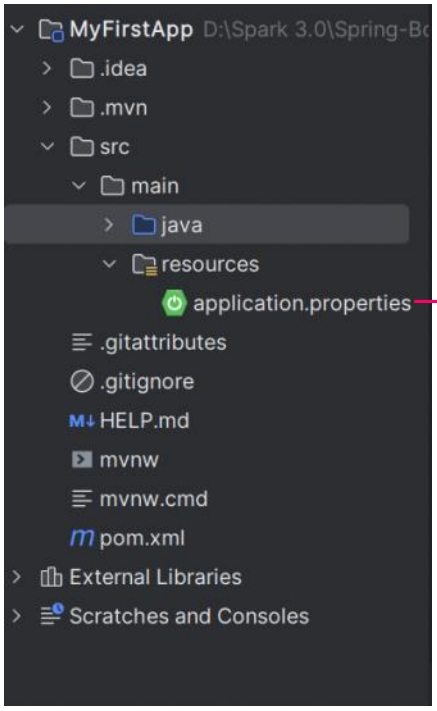
1. we can create SB application in 2 ways
   a. Initializer website
   b. Spring starter project in IDE

Note : if we try to create SB application using IDE then also IDE is communication with Initializer

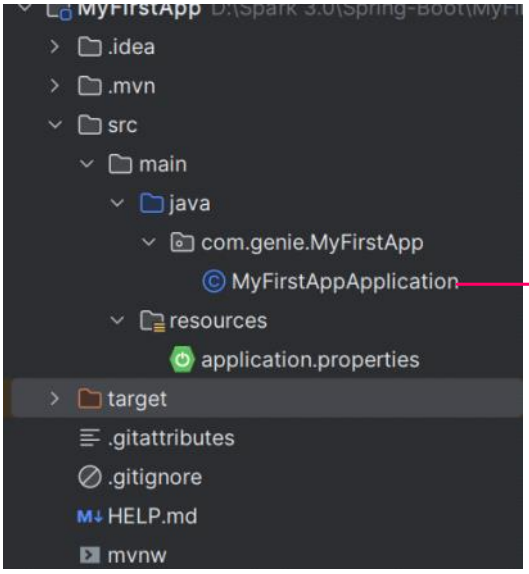








External configuration
Eg: db config



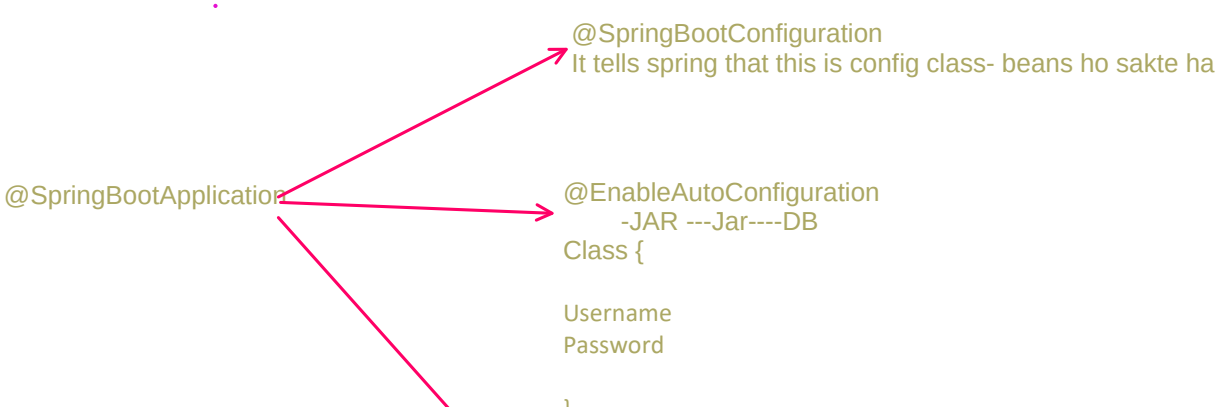
Run

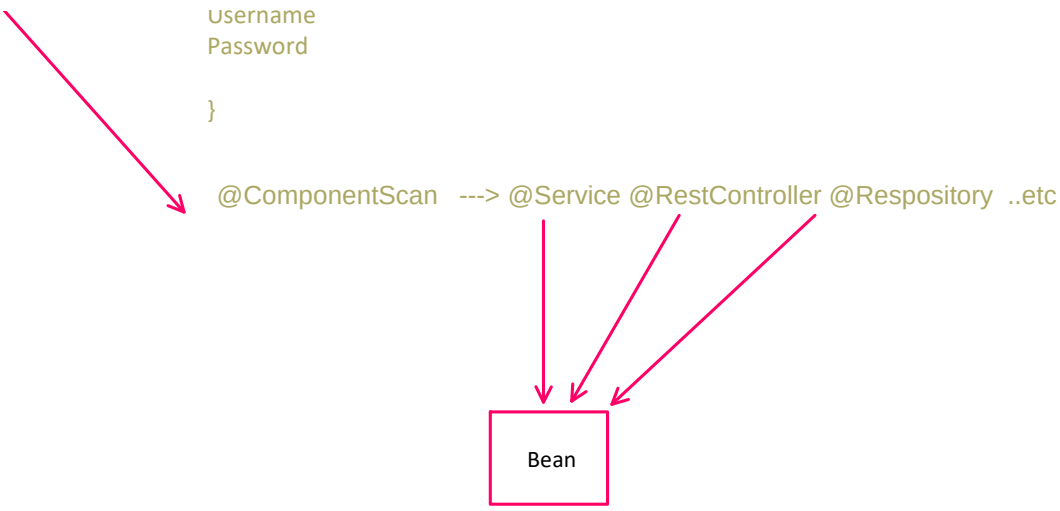
Intelij sirf .class file use krta ha build nhi banata

Build banane k liy maven ka use

Note: Based on type of our application, it will start IOC container

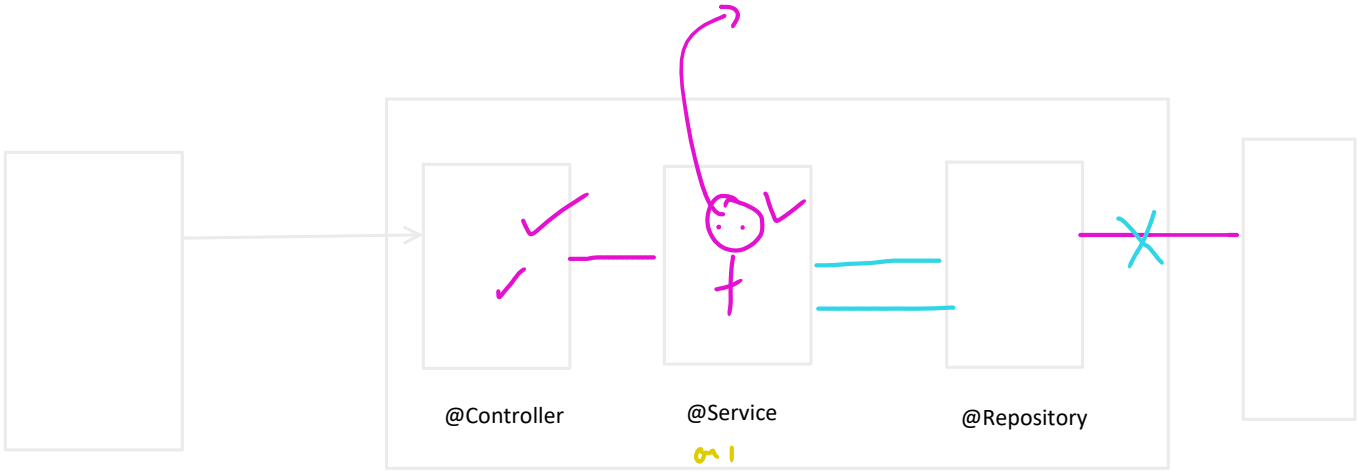
1. Run() method will print banner on console
2. Run() method will start IOC container
3. Run() will return context of IOC container

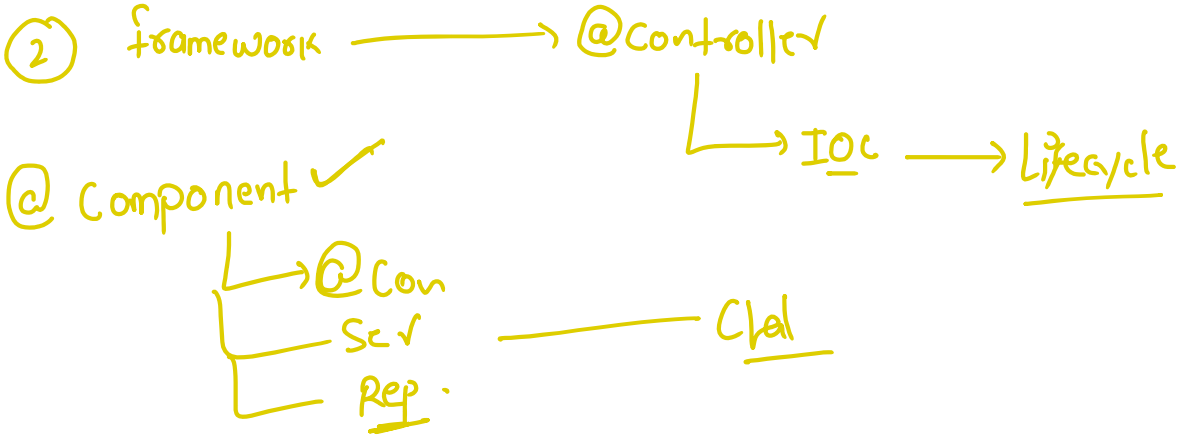


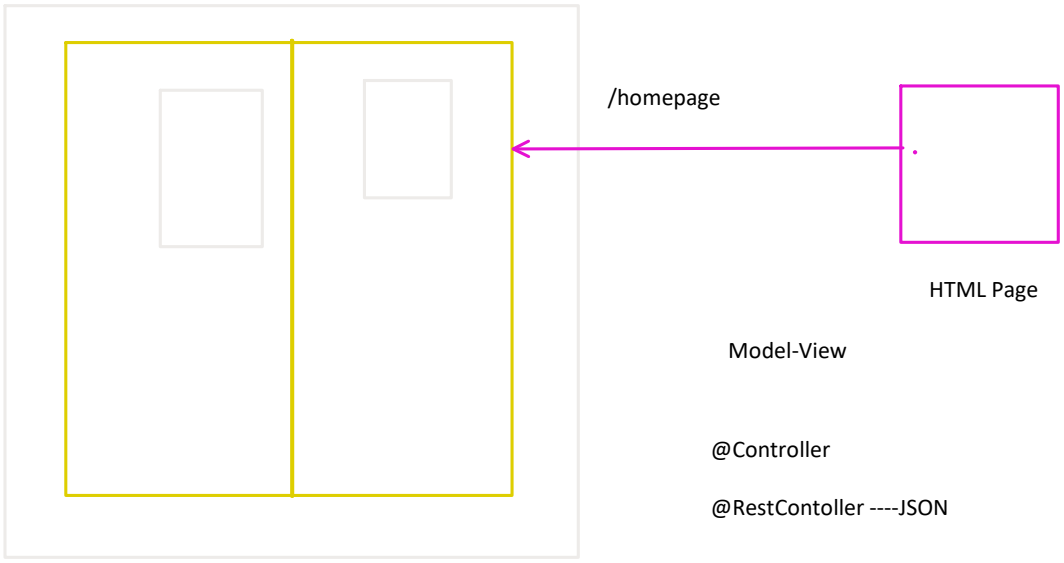


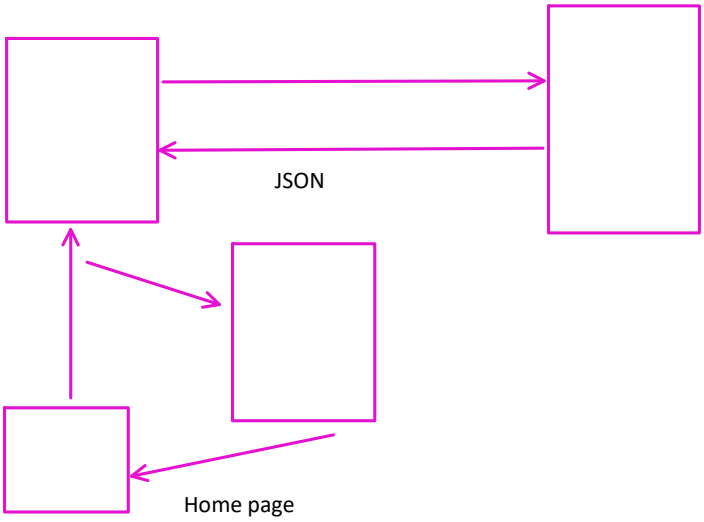
Spring-Boot: DI + Auto Config + Embedded Server + Actuator

Spring Boot + RestAPI(Web-Starter)+(JPA)



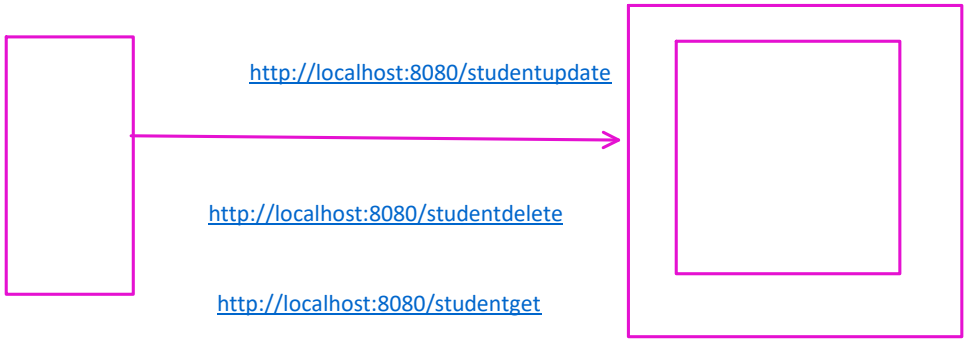


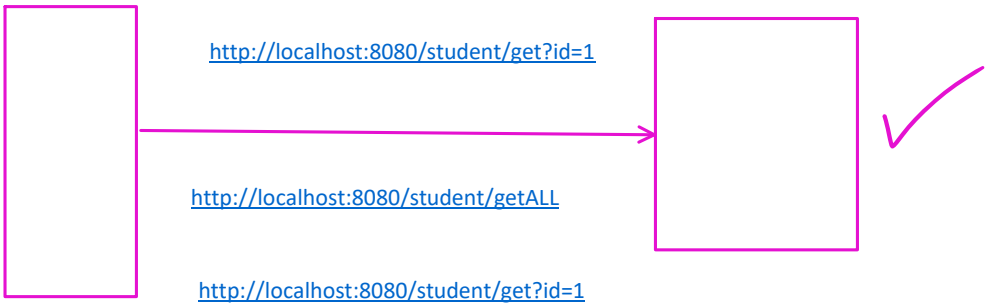




URL:

1. Protocol
2. Domain
3. Resource
4. Res-id

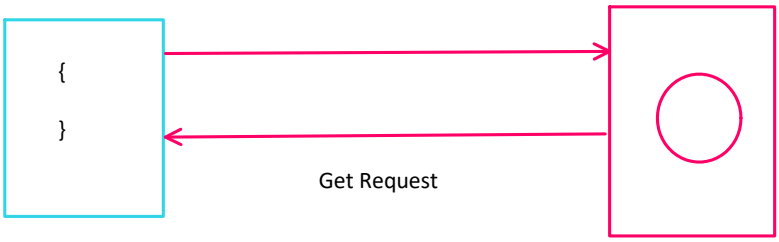


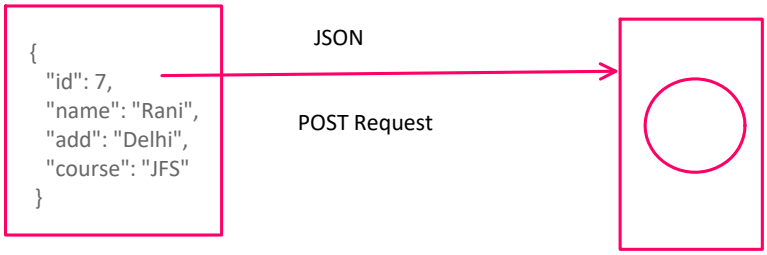


@Controller-----> html page return krta

"home-page"------>webApp--Web-INF--->views--->home-page.html

REST-API

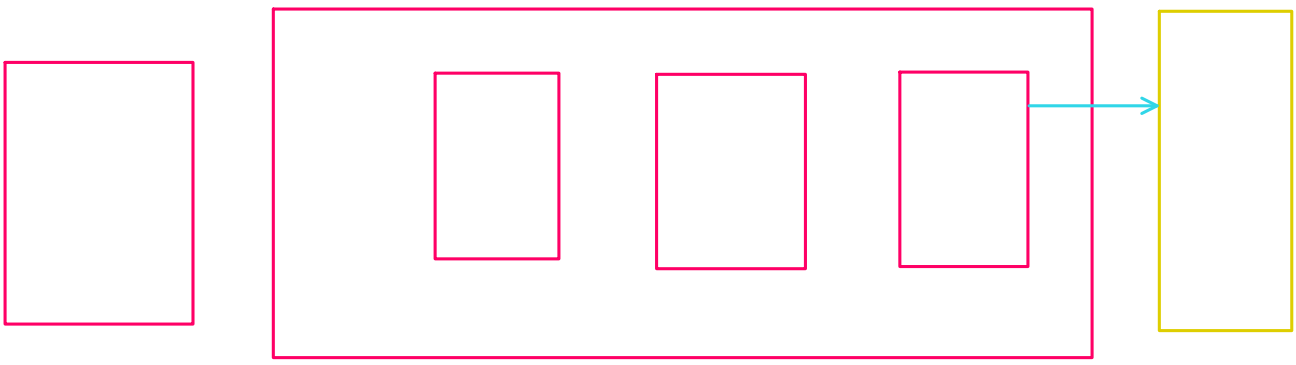




Object----json----serialization
Json----object -deserialization

Homework

Http status code
PUT -->
PATCH -->



Spring Data JPA

1. Application contains several layers
    a. Presentation Layer
    b. Web layer
        i. C
        ii. S
        iii. R
    c. Persistence layer

1. Spring data JPA is used to develop Persistence Layer in app
2. Provides readymade methods to perform CRUD operations
3. JPA provides two interfaces
    a. CrudRepository(I) = CRUD
    b. JpaRepository(I) = CRUD + pagination + sorting methods

Employee

@Entity : class ko entity banane k liy
@Table : class Emp ----> Employee
@Id
@Column

For making connect to db we need driver and db config properties

JPA---> Spring boot ------->driver------>mysql

TABLE : we need make table for primary increment
SEQUENCE, : we need sequence to increase pk
IDENTITY: MySQL auto will increment
UUID:
AUTO: Mysql qill automatically select best one for you

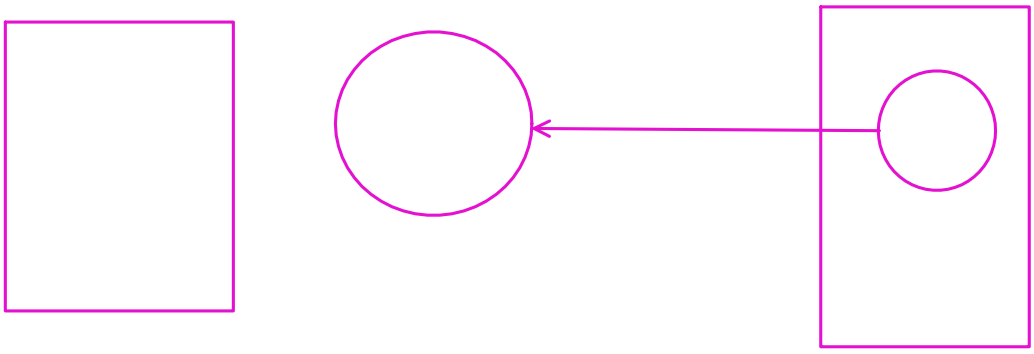
Note:
If you created table employee in database you want to create class in you project

```
create table employees(
empName varchar(30)
}

@Entity
@Table(name="employees")
Class Employee
{
```

```
    @Column(name="empName")
    String name;

}
```

empName------>emp_Name

sir just ek question hai ki agar mai anotation ki use na karu to fer dono mehi same name hoga na?



Methods in jpa

1. Save(object)
2. saveAll(<Iterable<Entity>)
3. findById(id)
4. findAll()
5. Count()
6. existByid(id)
7. deleteById(id)
8. deleteAll()

We can write native query

Security
Unit testing
swagger
redis

Microservices
kafak
Docker

BE -SB
Gmail
Telegram API notification

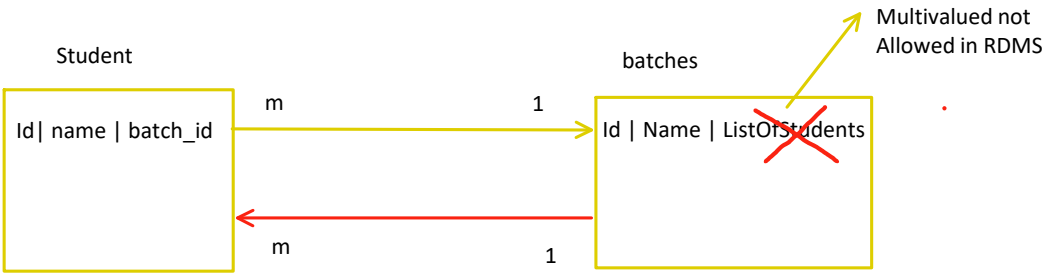
Mapping

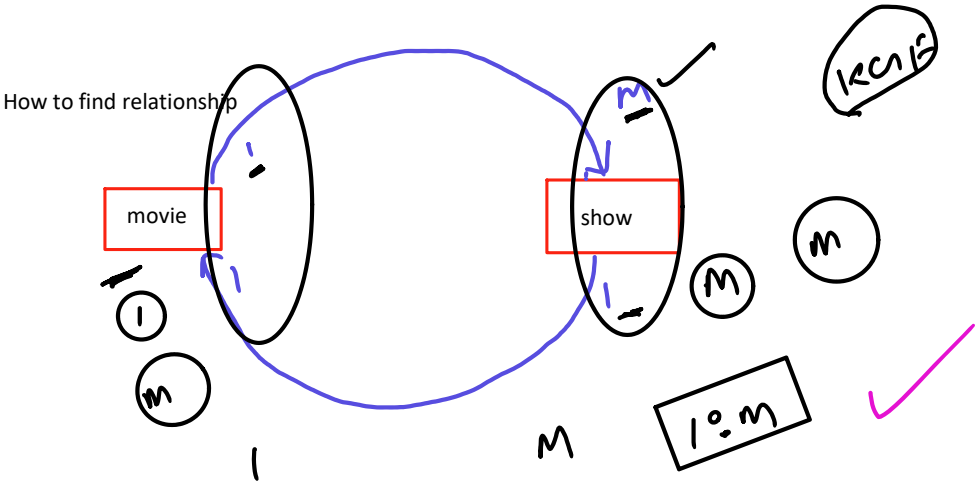
1. One to one
2. One to many
3. Many to one
4. Many to many

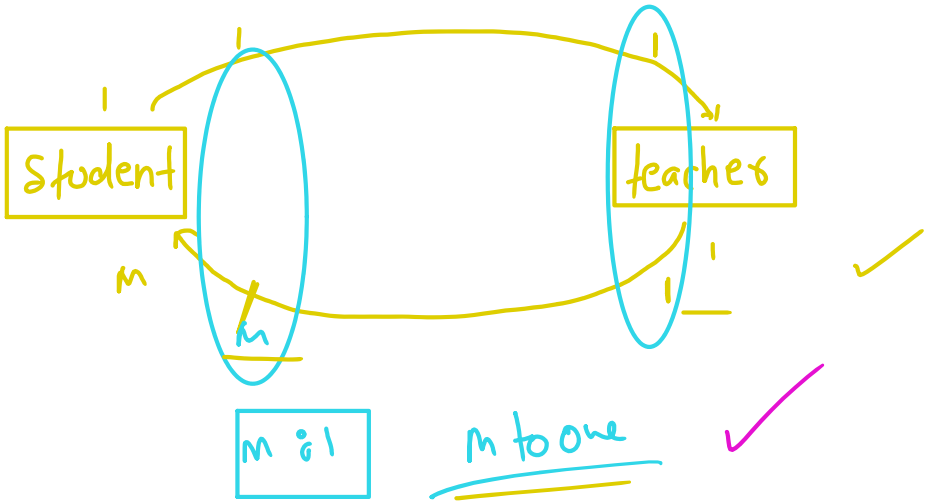
How to design schema in development
---

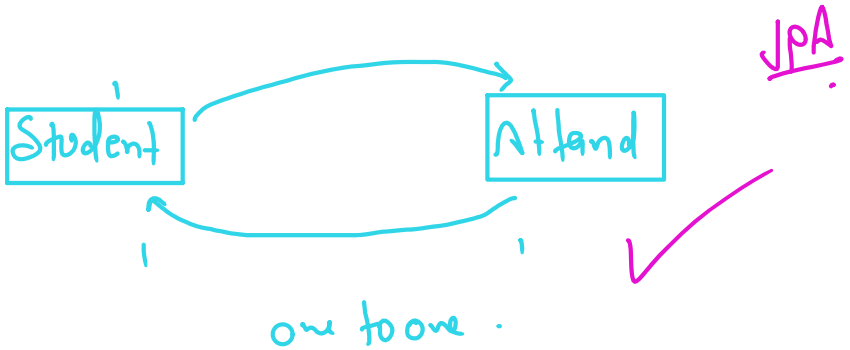
1. Requirement

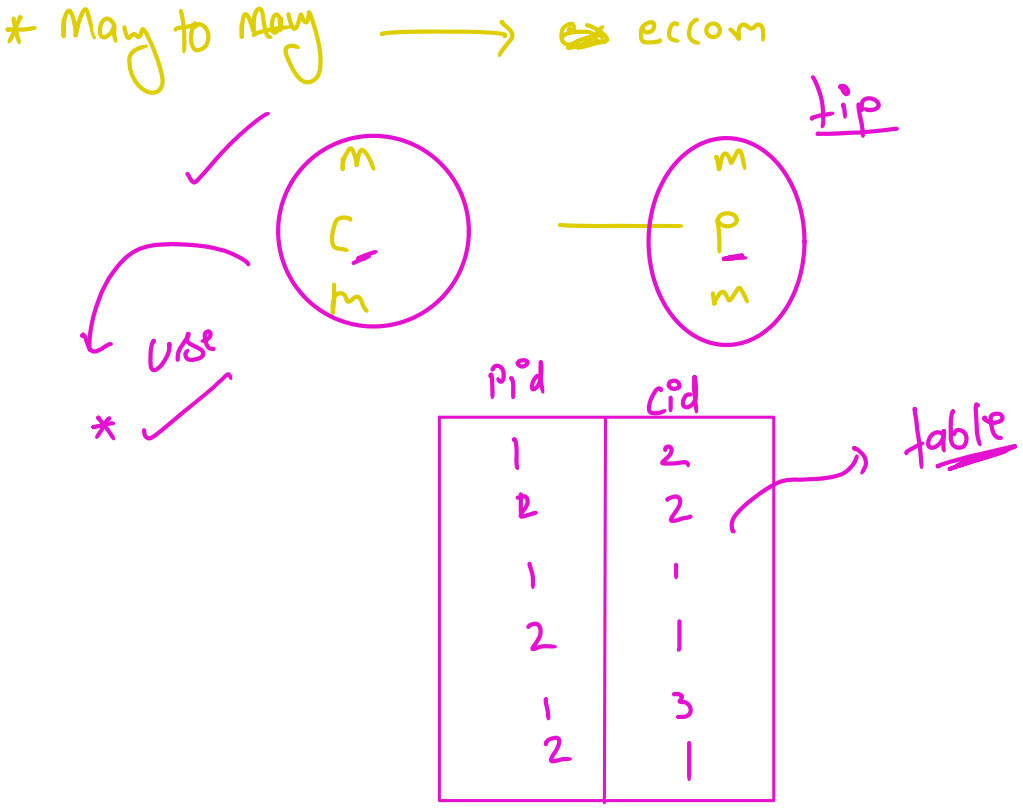
   CFS has multiple students
   CSF has multiple batches
   1 Batch can have multiple students
   Every batch have teacher
   Every batch has multiple class
   For every student in each class we have attendance

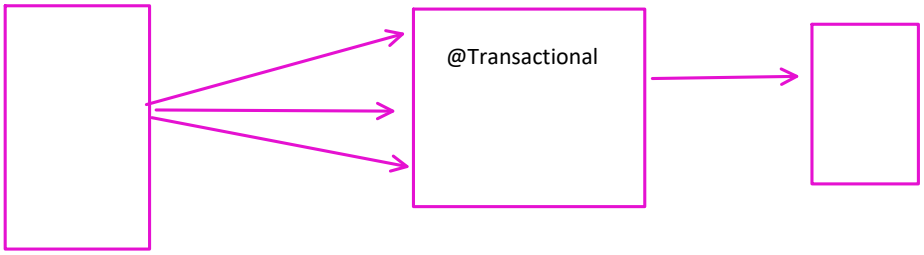

Student
batches
Multivalued not
Allowed in RDMS
Id| name | batch_id
m
1
Id | Name | ListOfStudents
m
1


How to find relationship
movie
show
1
m
1 : m

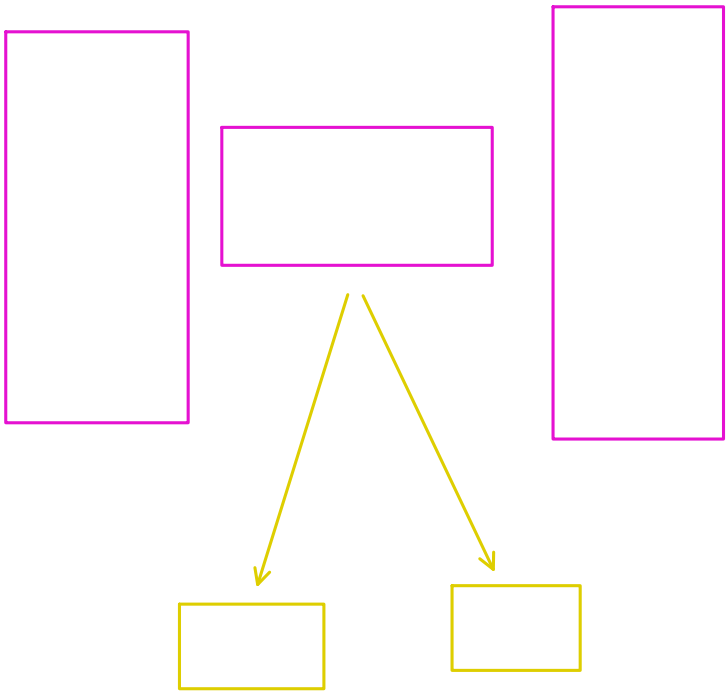

Student
teacher
m
m : 1
m to one

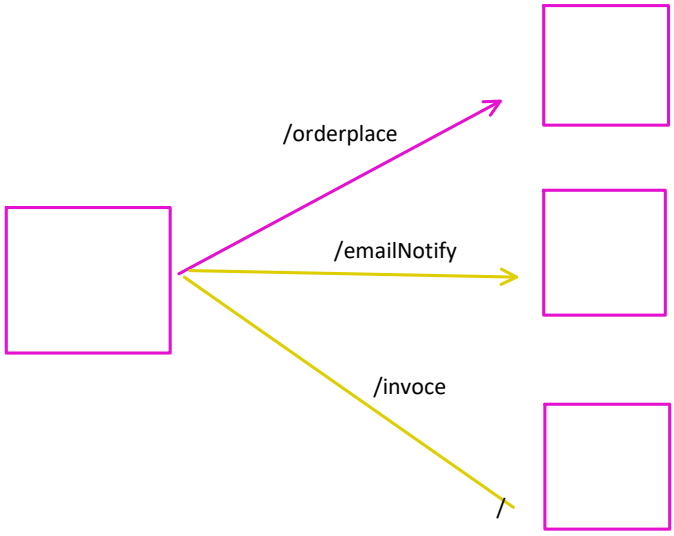

JPA
Student
Attand
one to one

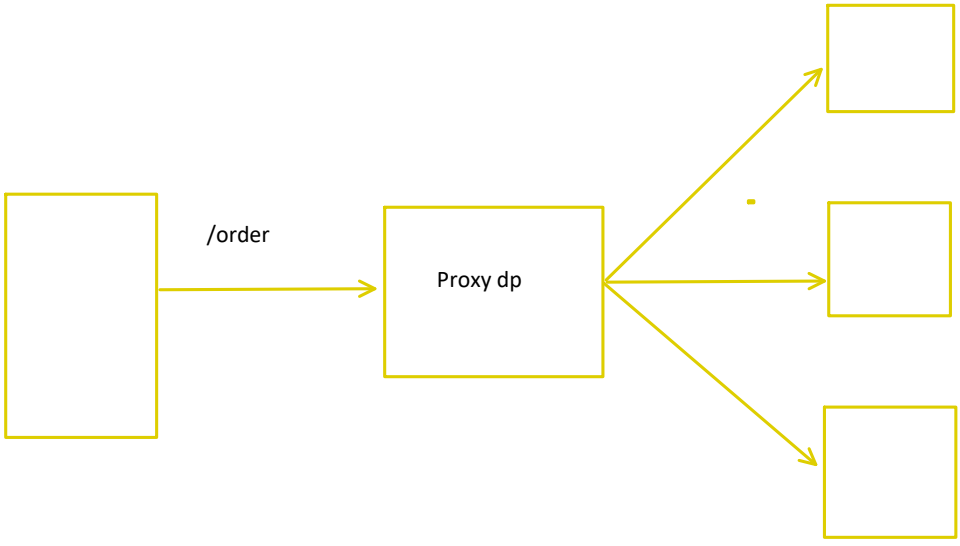


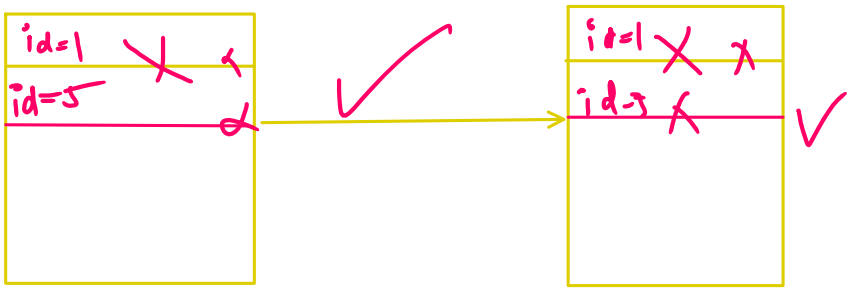


Commit
Rollback




/orderplace
/emailNotify
/invoce
/

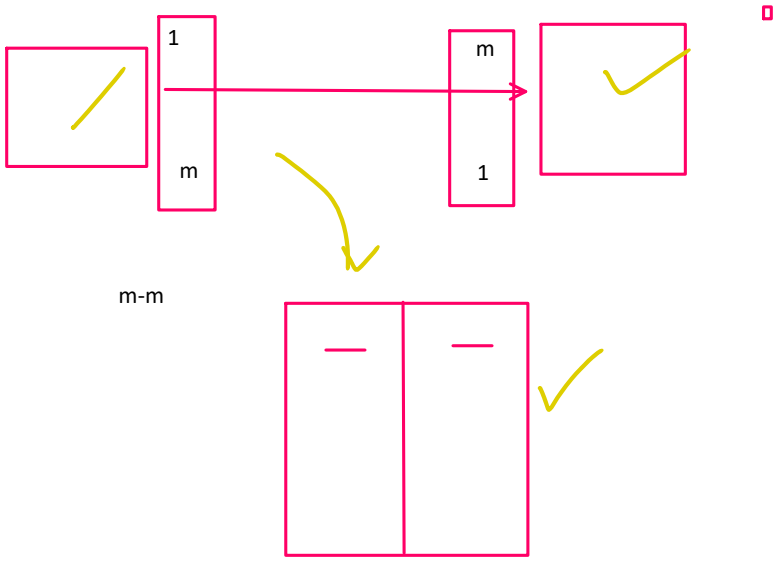

/order
Proxy dp

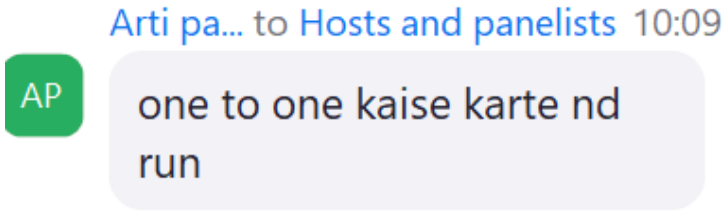


ACID ---->

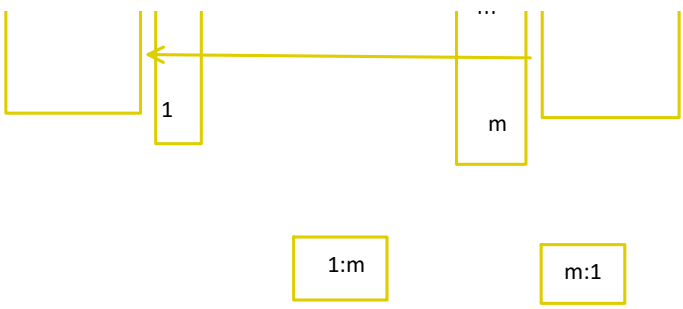
Try catch----->risky code
@Transactional ---> db ope----db open ----> service

Hard rule nhi ha

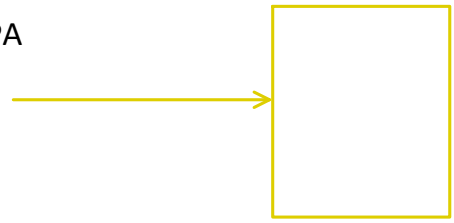





```
SELECT Orders.OrderID, Customers.CustomerName, Orders.OrderDate
FROM Orders
INNER JOIN Customers ON Orders.CustomerID=Customers.CustomerID;
```



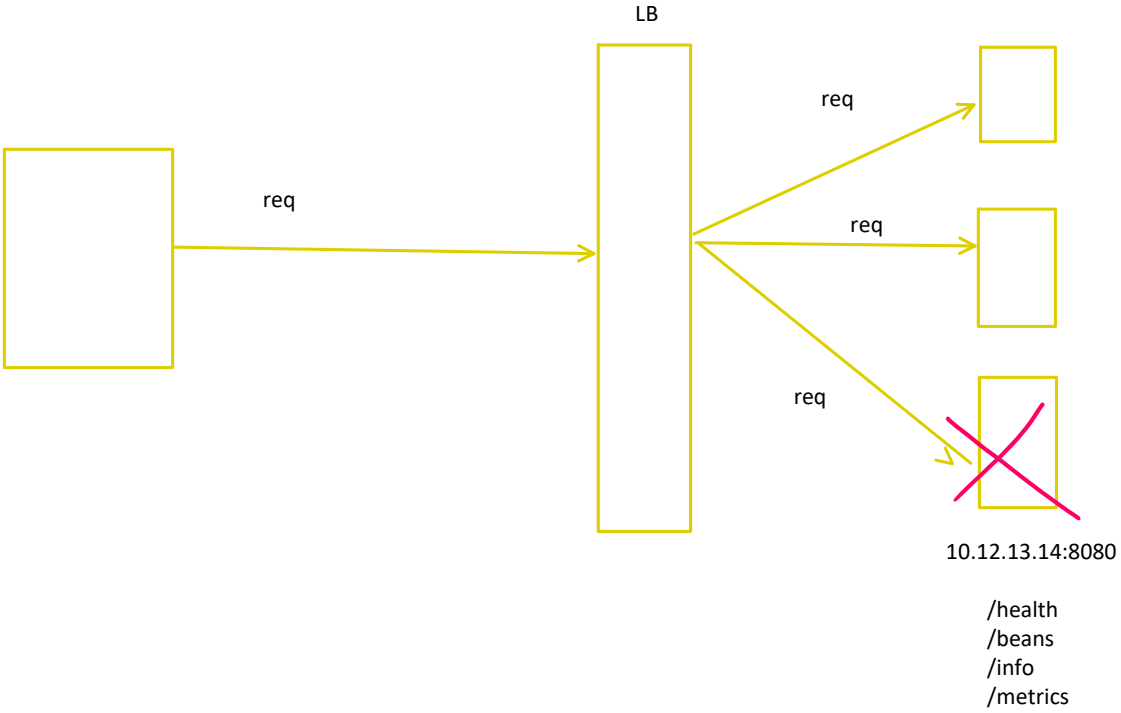
1. Spring core
2. Spring boot
3. Spring data JPA
4. JSON
5. HTML
6. CSS
7. JS
8. Rest API
9. MySQL

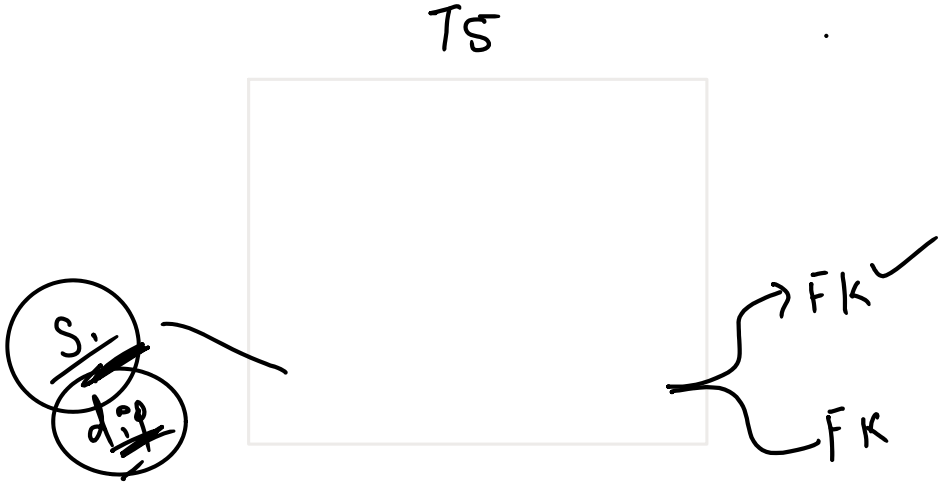


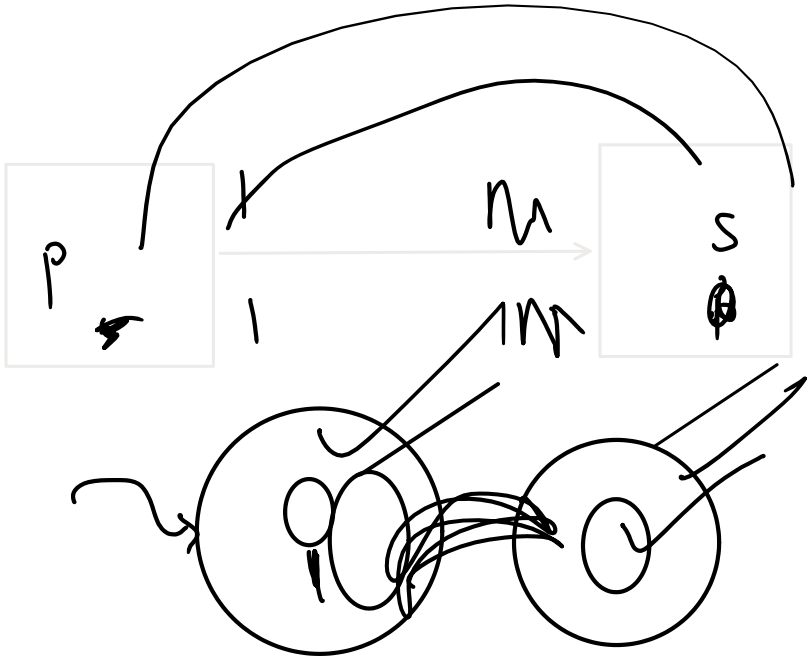
WebSocket---AI

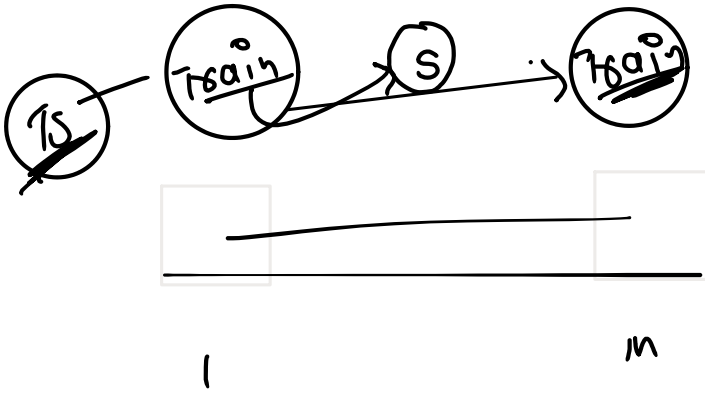
Find Trains Between stations

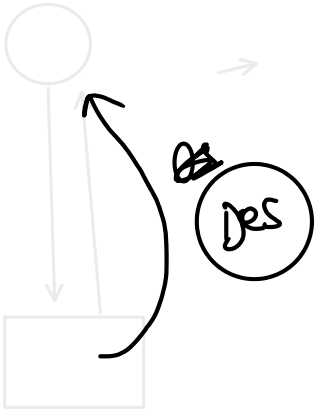
Actuator


LB
req
req
req
req
10.12.13.14:8080
/health
/beans
/info
/metrics


TS
FK
FK







one to many

many to one