Agenda

* Microservices
* What are frameworks & why they're needed
* Intro to spring ; Dependency injection
* Spring Boot.

Microservices

Functionalities of scaler - - -

1. live classes
2. dashboard
3. authentication
4. Career platform
5. Assignments/homeworks
6. Mentor fun
7. notifications
8. Support centre
9. Instructor fun

Can we have a single codebase for all of the above features?

- Monolithic.

Problems with monolithic services

1. Your app^n takes a lot of time
   - compile
   - start
   - deployment
2. It'll be difficult to understand the codebase
3. The collaboration needs to happen, it's difficult.
   People can't work independently
4. Single point failure.
5. Not possible to have diff stacks
6. Selective scaling
   The load on each functionality is different, hence the no. of

machines required will also depend on functionality.

But we cannot achieve this.

Microservices

Each function if it has independent meaning it can be a separate service [codebase. These codebases will interact with each other to achieve the overall functionality.

Frameworks

Consider the codebase of

Google
FB
Amazon

} All of them will have some common use case-cases.

For ex;

1. Authentication
2. Notifications
3. search
4. DB interaction

Frameworks will have ready to use functionalities which can be further configured based on your requirement.

Two common things.

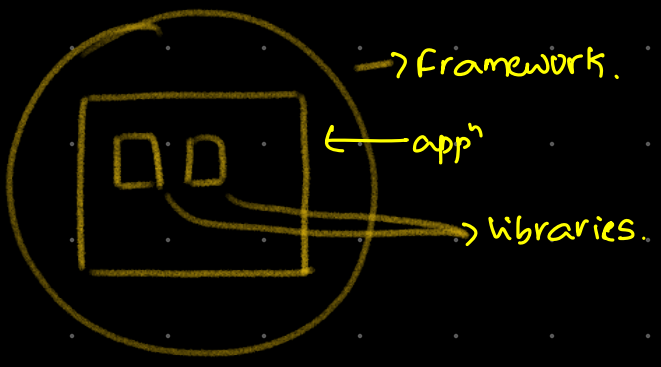
① Stand on the shoulder of giants

② Do not re invent the wheel

Diff between libraries & framework.

Framework is an ecosystem where you build the app'.

library is a piece of code that you can use to build your app'.



diff frameworks exist for diff languages.

(i) Java → Spring

(ii) Python → Django

(iii) JS → ExpressJS etc.

Spring framework.

Dependency injection

```
class Crow implements Flyable

    FlyBehaviour fl;

    public Crow(FlyBehaviours fl)
        this.fl = fl;

    @Override
    public void fly()
```

fl.makefly();

FlyOne implements FlyBehaviour.

fly()
| fly@ 10kms/hour.

etc --

Class App

main()

Flyable crow = new Crow(new FlyTwo())

crow.fly()

Advantages

(i) Loose coupling.

(ii) Easy to test.

Who's is creating the objects of controllers / services / repositories..

Spring IoC container

Inversion of control.

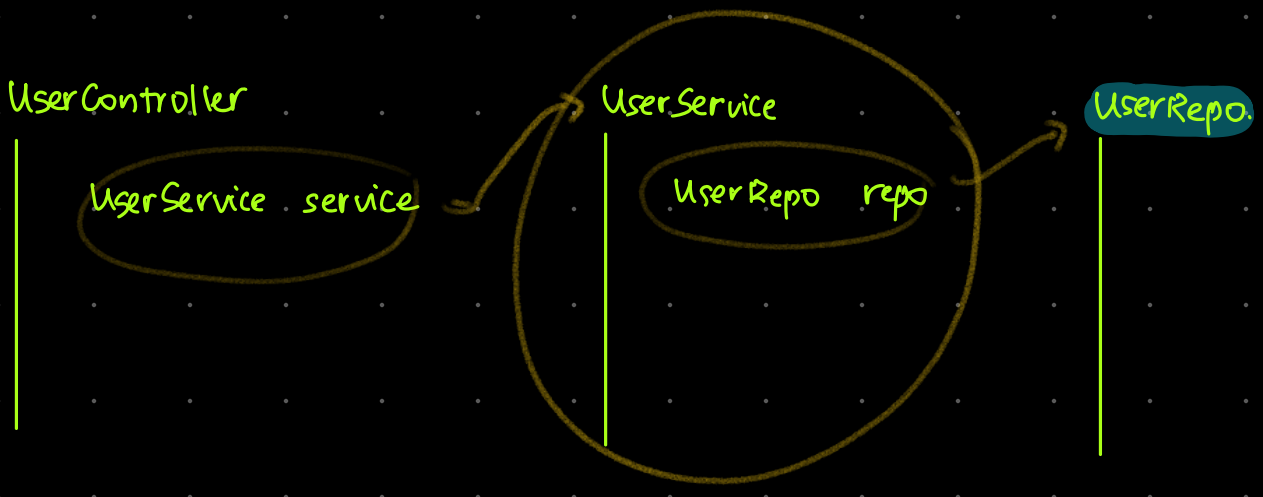
Once a class in marked as a bean.

DBConnection

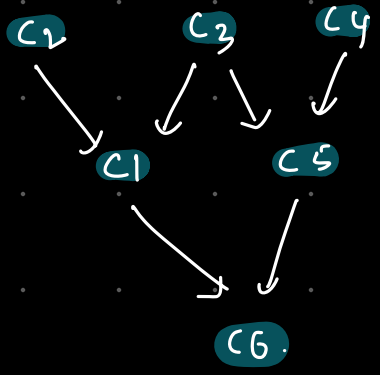
user
product
auth

new DBConnection().

Object life cycle management will be done by Spring IOC Container.



Repo
↳ Service
↳ controller



@RestController
@Service
@Repository

} will mark the classes as beans along with some more features.

Spring provides DI, but a lot of other common features will be added via add-ons.
[plugins for spring project].

Before Spring boot.

For each add-on,

- create xml file
- Do the configurations
- write a lot of boiler plate code to use it

After spring boot

1. Just add the dependencies.
2. Default configurations will be added.

spring.jpa.show-sql = false

Spring Boot

It's a Spring project, that makes it easy to work with other add-ons, without having to write any configurations / boiler plate code.

- other spring projects.

These days every spring project is a spring boot project.

	Spring	Spring Boot
1.	comprehensive framework	opinioted framework.
2.	Developers responsibity to configure	default configurations exist
3.	You need to deploy in a Server like Tomcat, JBoss,	Embedded server exist
4.	For every change, you need to redeploy.	Click on restart, everything will be taken care.