=> Java => ✓

{ Java Technologies => JDBC
JSP
Servlet }

=> Framework :-

{ Helps to avoid / reduce boiler plate code

{ Framework :- solution to Develop Application in easier, faster, & effient way.
↳ Semi Developed app
→ Abstraction build on top tech

Web Application => Handling Request & response
{ Struts }

DAO → DataBase => ORM Band => Hibernate

Spring => Application Development Framework
⇓
End to End Application

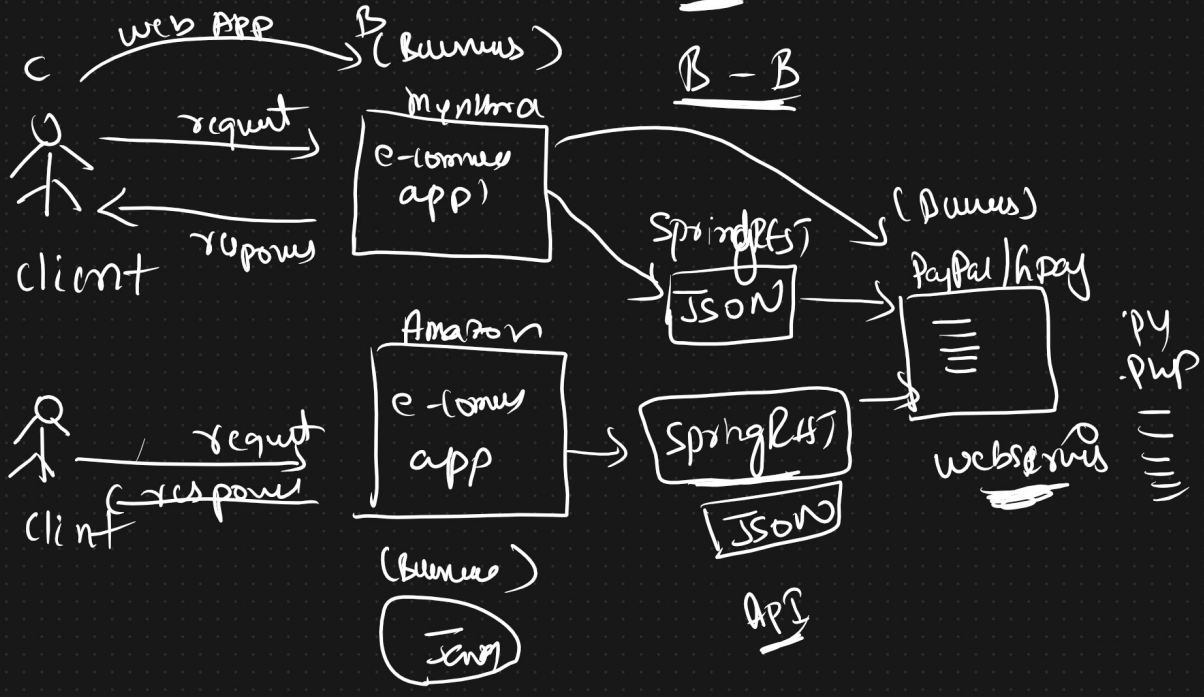
⇒ Distributed Applications.

Web Application ⇒ Internet

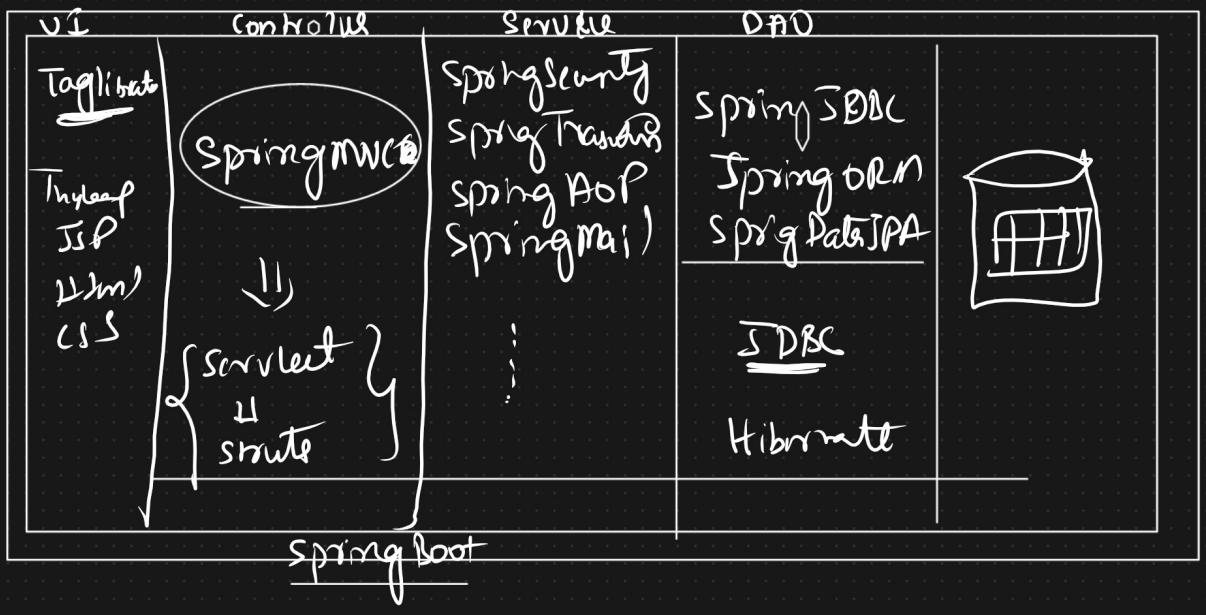
{ client – server }

{ C–B }

Amazon, Myntra, ---- Distributed



App



=> spring core => spring container

Base of All spring modules

=> How to make our spring work for us?-

- => XML Driven configuration
- => Annotation Driven configuration
- => Pure Java configuration

=> { Auto Driven Config => Rapid App Development RAD }

SpringBoot

Develop App => Spring Framework

{ { Configuration } { XML / Annotation } }

SpringBoot

=> Spring Framework

End to End Application => ___

=> Spring => Module Based Designed Framework

Spring core → Base of All modules

Spring JDBC
Spring ORM → DataJPA
Spring AOP
Spring security
Spring WebMVC → Spring REST

⇒ Spring Framework ⇒ versatile

struts ⇐ Spring ⇒ Hibernate

JDBC, Servlet

⇒ Non-Invasive Framework. → It will not force us to extend(or) import any classes(or) interface.

Invasive

class Servlet extends HttpServlet
{
}

eg:- (Non Invasive) ⇒ eg:- Hibernate, Spring Frame

Invasive ⇒ Servlets & struts

Spring Framework => 2003 => 0.9

=> 2004 => 1.0
!!
:::
3.0

<= 6.0

=> All kinds of App can be developed
=> Application Development => Non-Invasive
=> Easy to learn & easy to code =>
=> Versatile => Enabled => light weight App
Development Development

=> Spring Core :- { xml Approach
Annotation Approach

{ -> IOC => Invasion of control }
-> Depindency Injection

Container

{
class A
{
C c=new C();
c.displ();
}

class B
{
C c=new C();
c.displ();
}

class C
{
displ()
{
}
}
}

=> interface & Polymorphism

=> loose coupling

=> Creational DP
=> Structural DP
=> Behavioral DP
   (Runtime Algo change)

Strategy DP

- => Close for modification open for extension
- => code to interface rather implementation.

=> of Spring Framework =>

- IOC
- DI

=> Invasion of control
DI -> Dependency Injection
Spring container.

<=> Basic App => core java <=> Spring Framework

Spring => IOC =>
- create object
- Target
- Dependent

=> which ever Obj is created & managed by Spring container

=> Spring <= Bean

class B {
@Bean
void disp()
{
A a = new A();
}
}

class A
{
}

}

Big Picture Springframework.

Spring core
↳ IoC
↳ DI

without spring
loose coupling