

Java Based Framework :

Web Based Application Dev using Java : J2EE

Spring : improvisation over J2EE

=> Multiple Deployment Descriptor (EJB)

=> Multiple Interface

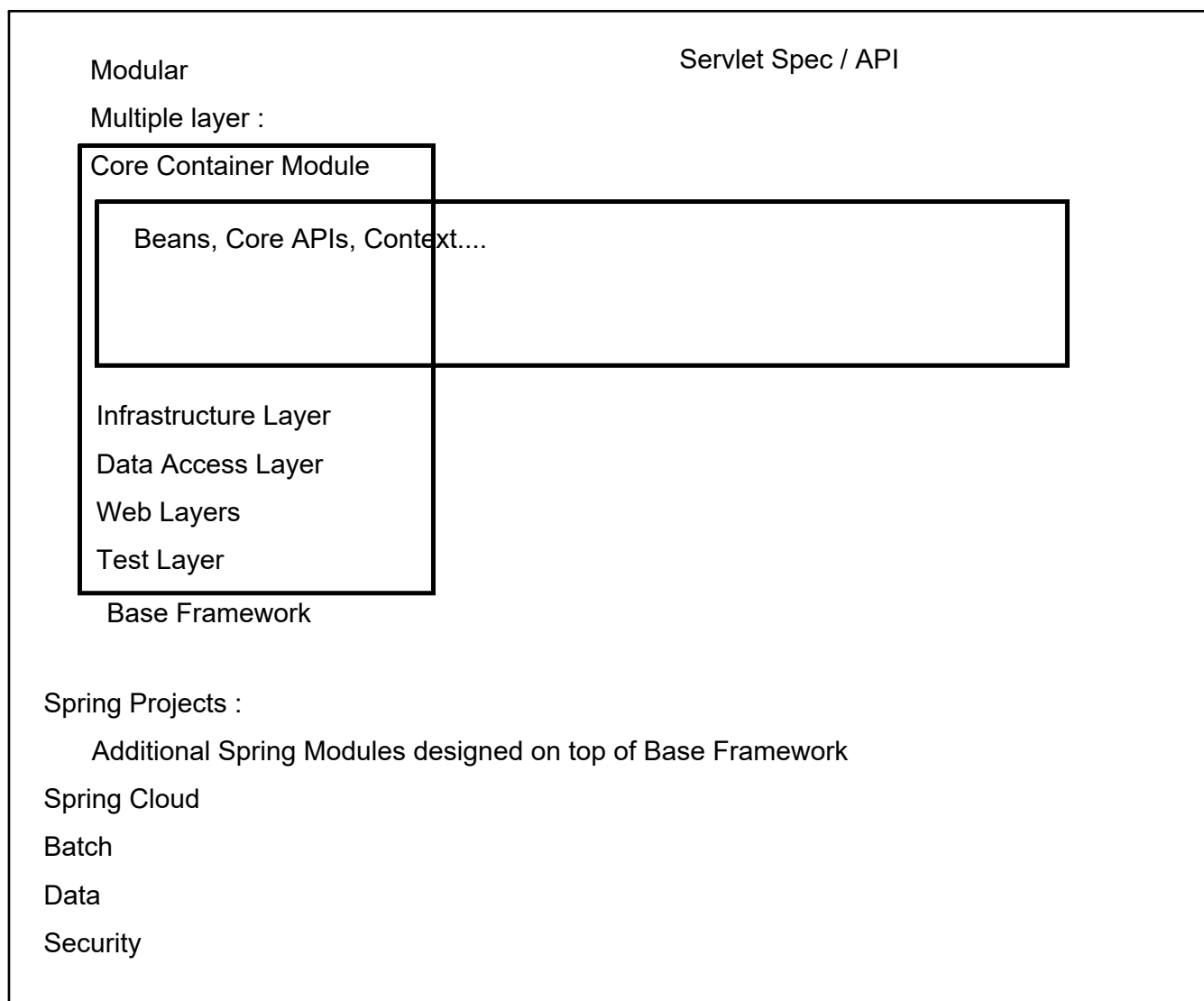
=> Poor in performance in production

Rod Johnson : lightweight variant of J2EE (without EJB) :

Spring

Spring Framework : goals

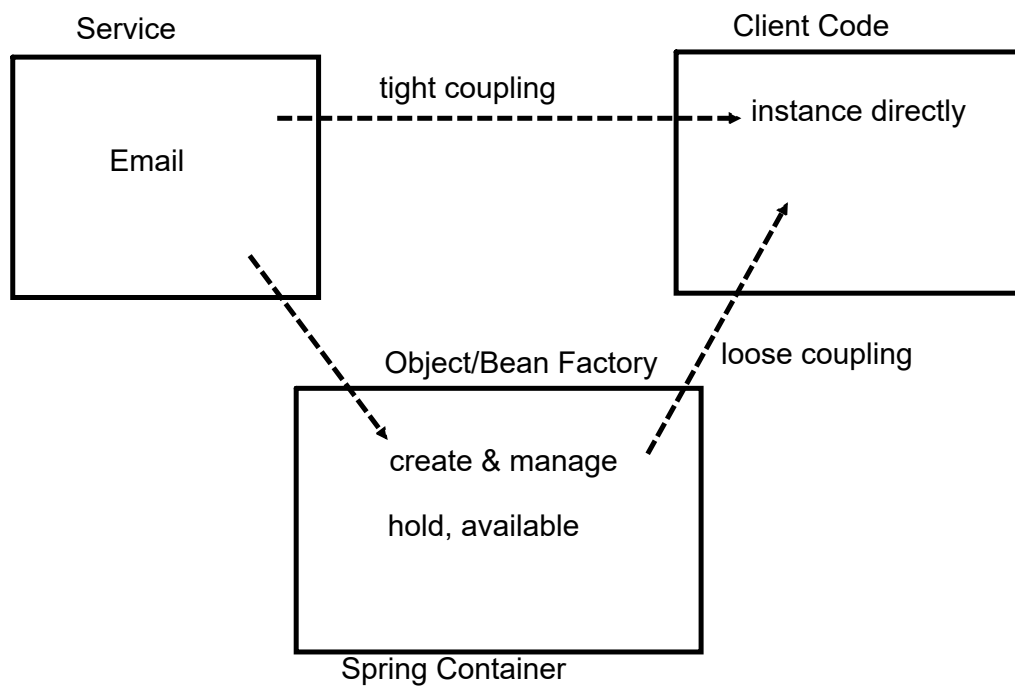
1. Any Java Based Application
2. Highly modular development
3. Lightweight : POJO (Plain Old Java Object)
4. Minimize the boiler plate code
- 5 Three pillar
 - IoC
 - DI
 - AOP

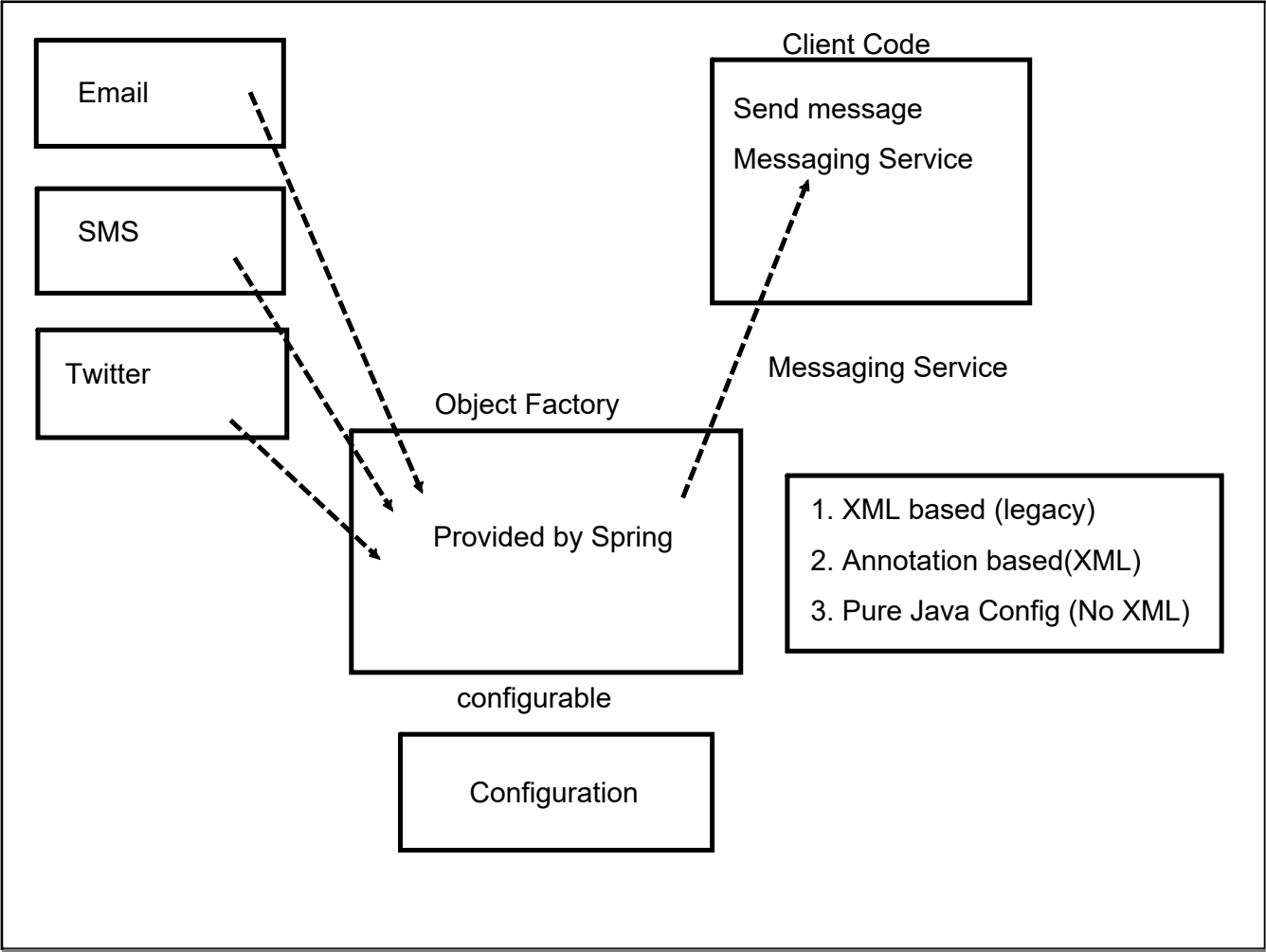


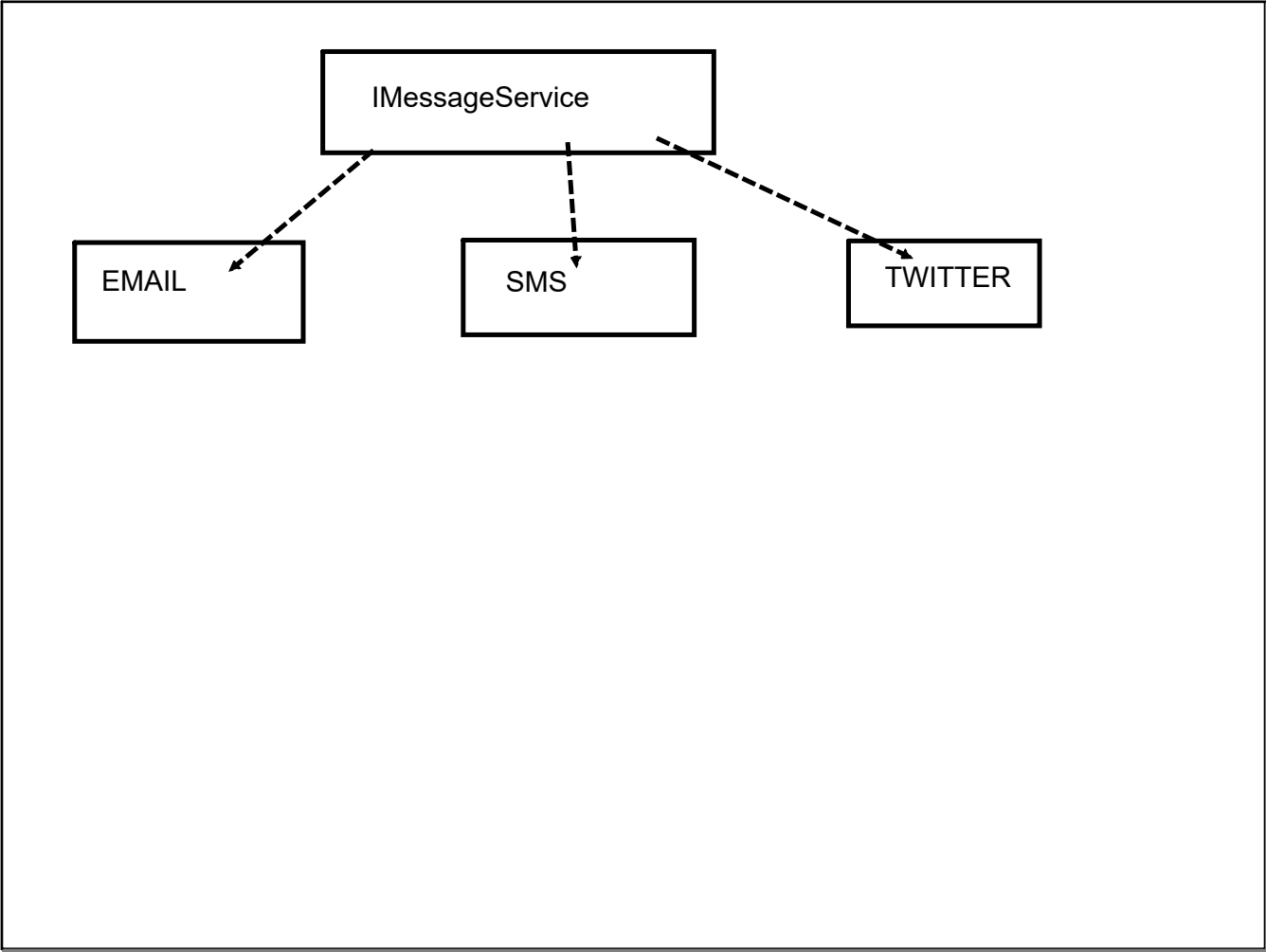
Base Framework : library jar

IoC : Inversion Of Control :

Outsourcing the construction and management of Object : IoC





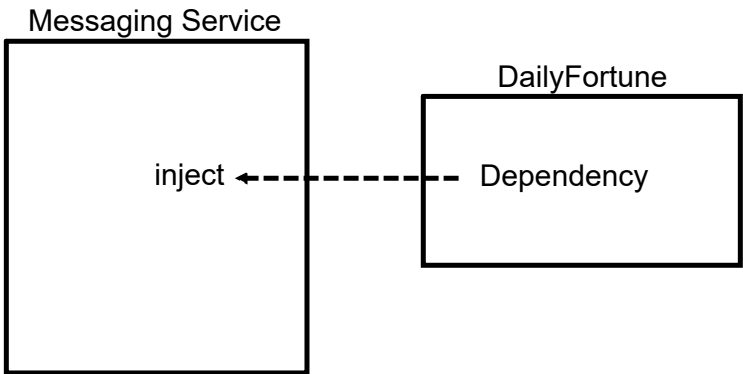


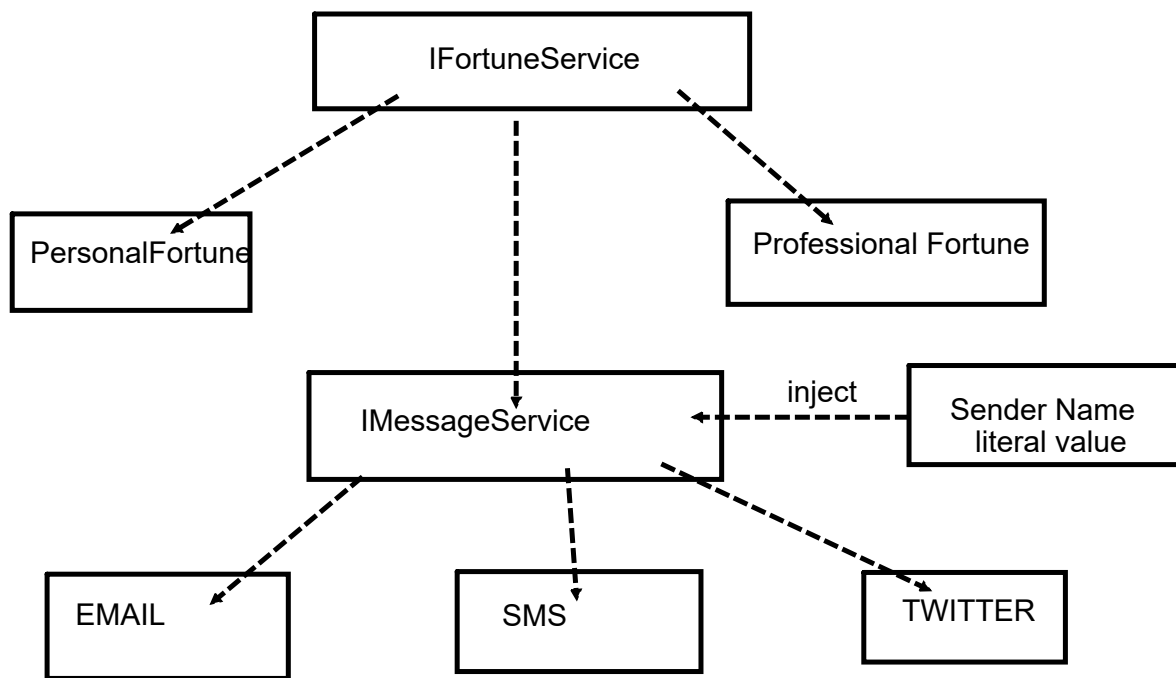
Dependency : Service may be dependent of multiple service

Car :

- parts of the car
- color
- engine power

DI : Dependency Injection





Literal values must be saved in a repo/properties files

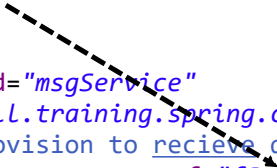
Types of DI

Constructor based DI

Setter Based DI

```
<bean id="fService" class="com.dell.training.spring.core.service.PersonalFortune"></bean>

    <bean id="msgService"
class="com.dell.training.spring.core.service.EmailService">
    <!-- Provision to recieve dependency : Cosntructor based -->
    <constructor-arg ref="fService"/>
</bean>
```



Bean creation

Manage the bean (life cycle of bean)

1. How long bean will live???
2. How many instance are created ???
3. How beans are shared ???

Scope :

Default Scope is : Singleton

Create only single instance, cached in memory
and shared the same instance

prototype :

new object on every demand

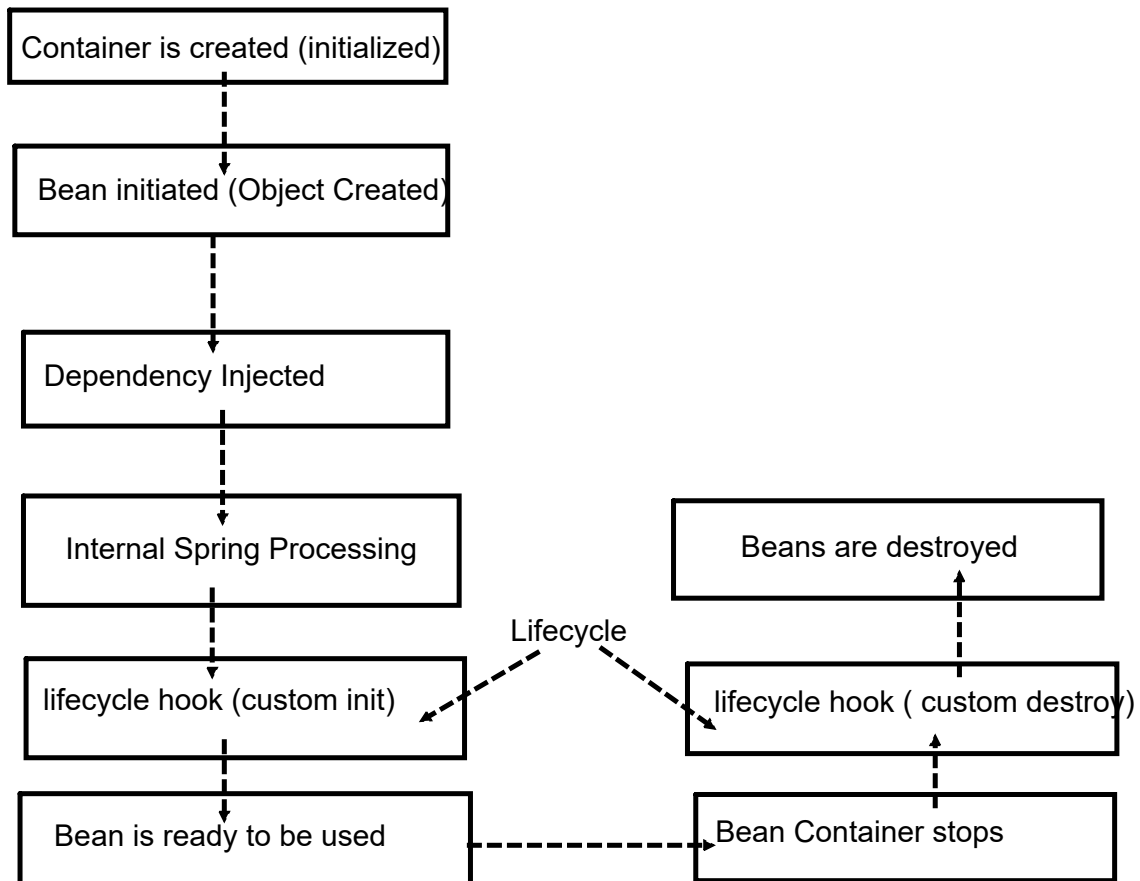
Web App

request

session

global-session

Lifecycle stages:



Lifecycle hook

1. any name
2. any access modifier
3. must not be static
4. may have a return value, but can't be collected
5. Must not have any args

Prototype : Spring Bean Factory does not maintain the complete lifecycle of
prototype beans !

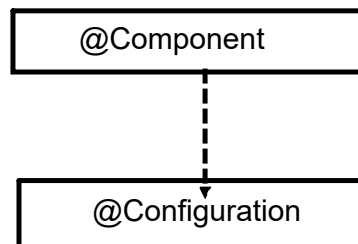
Creates , shares , forgets...

Annotation based config : XML file would be refer the resources (path)

Three approaches of DI

1. Constructor
2. Setter
3. Field based

Pure JAVa Config
XML ~ JAVa class



@Component : applicable on class

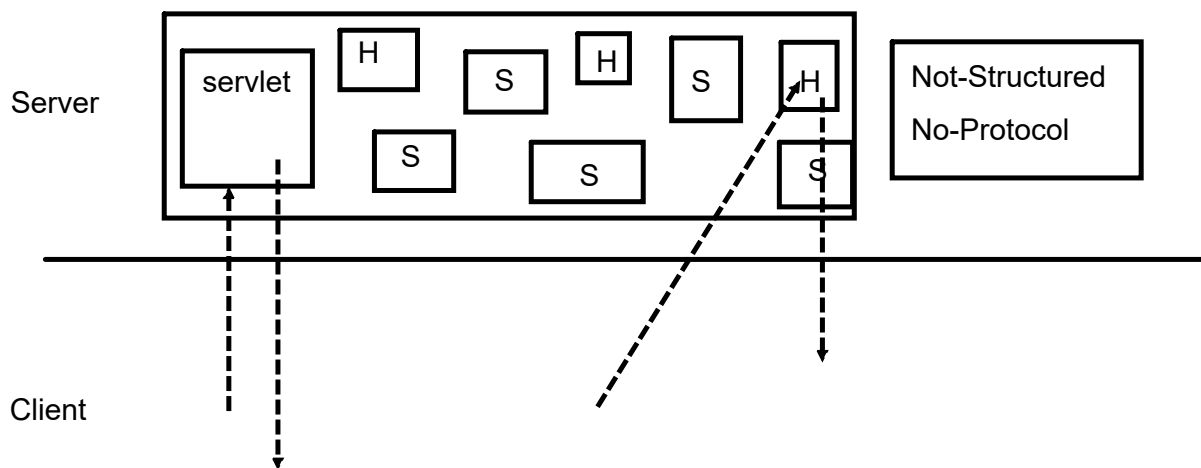
@Bean : applicable on method

Spring Web Module : APIs for developing Web Applications

Designed on top of Servlet API

static : HTML page

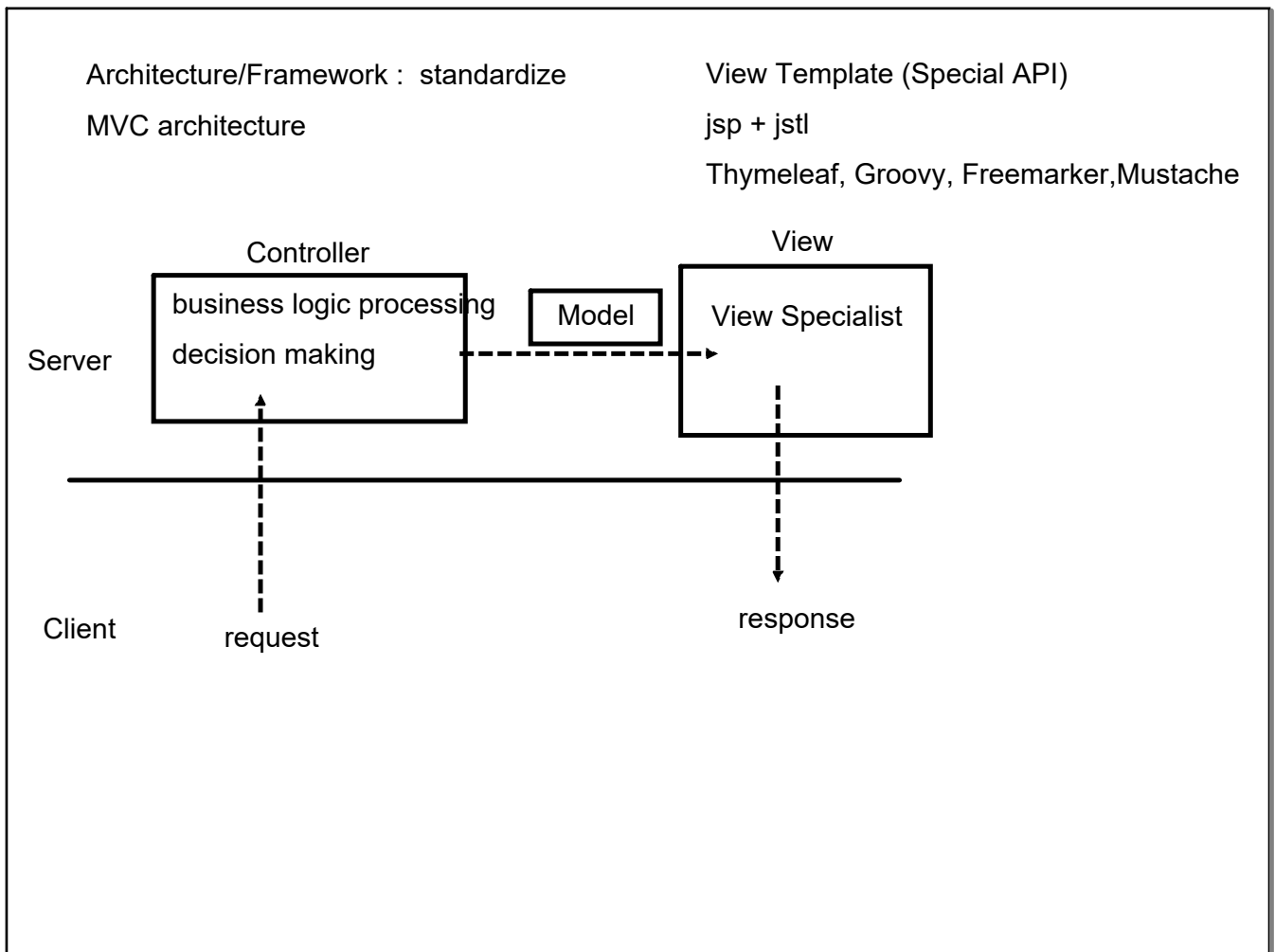
dynamic : Java class (Servlets)

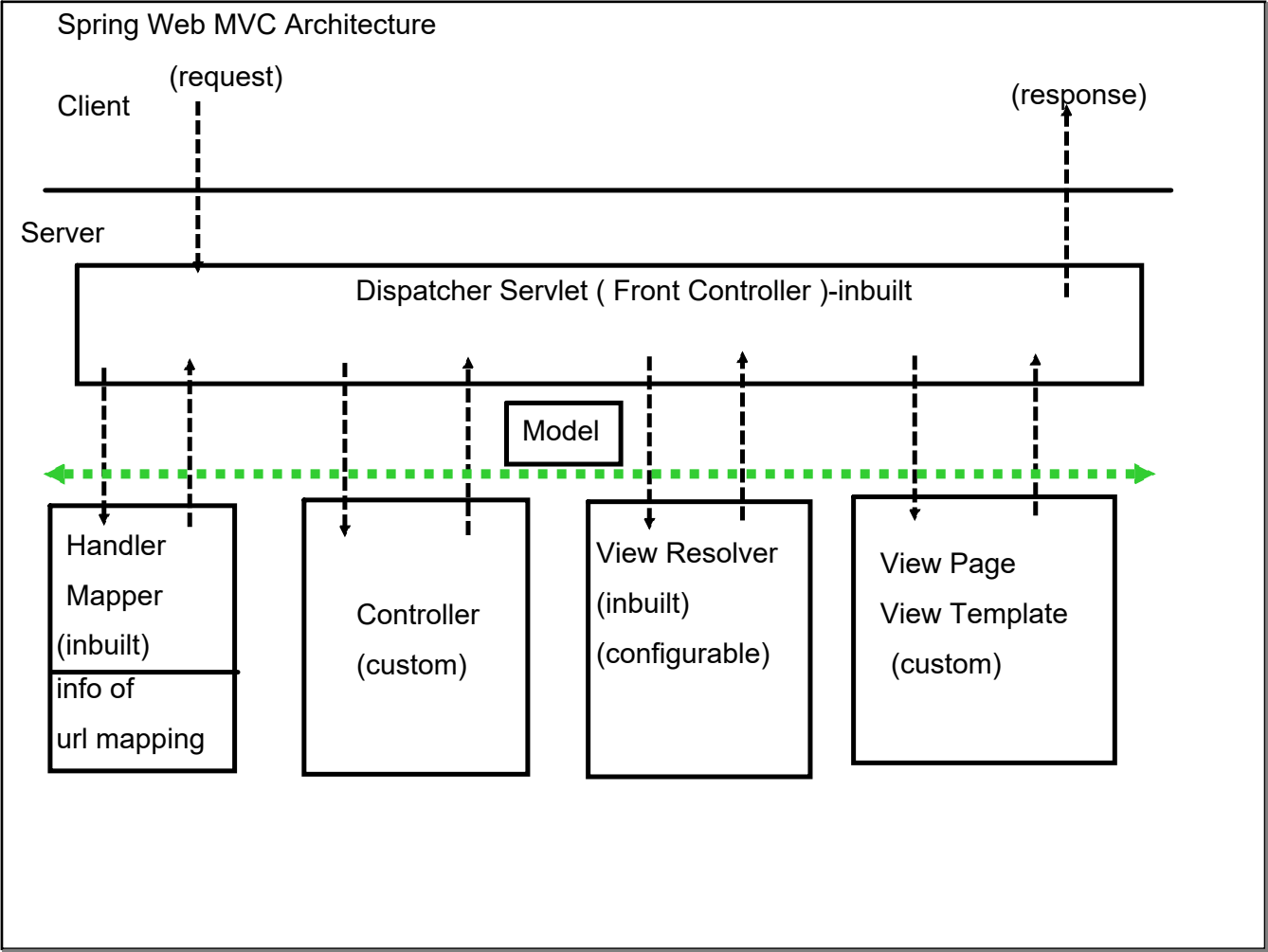


JSP : Another way of writing servlet --->
compiled into a servlet

More presentation : JSP --> Servlet

More Logic : Servlet





Resources:

1. Spring framework
2. external api - Servlet API/JSP-JSTL
3. Development Server (Java based Dynamic Web App) : Tomcat, Glassfish, JBoss

Tomcat : integrate with Eclipse IDE (launch & deployment is automatic)