

Spring Framework : Popular frameworks to develop java application

Highly Modular in nature

Framework :

1. Strict and disciplined implementation of an architecture
2. Reduce the boiler-plate code
3. Abstract implementations of API
4. Focus more on Business logic

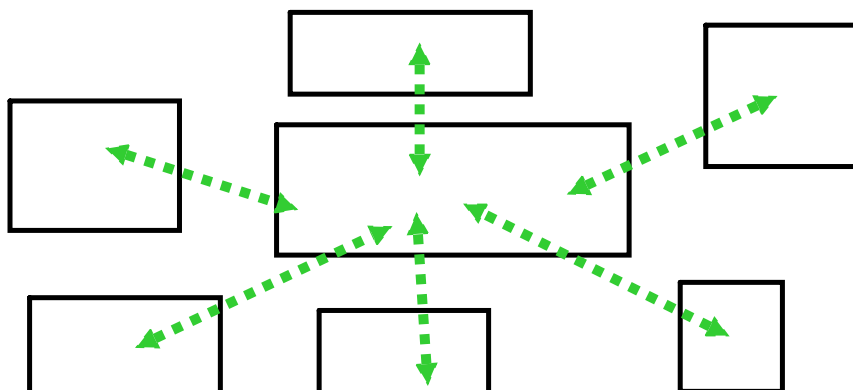
J2EE Framework :**1. Complex in nature :**

Service : need to create lots of interface, abstract classes, inherit class and interface
reduces the productivity of developer
reduces the efficiency
Uses lots of deployment descriptors : xml files

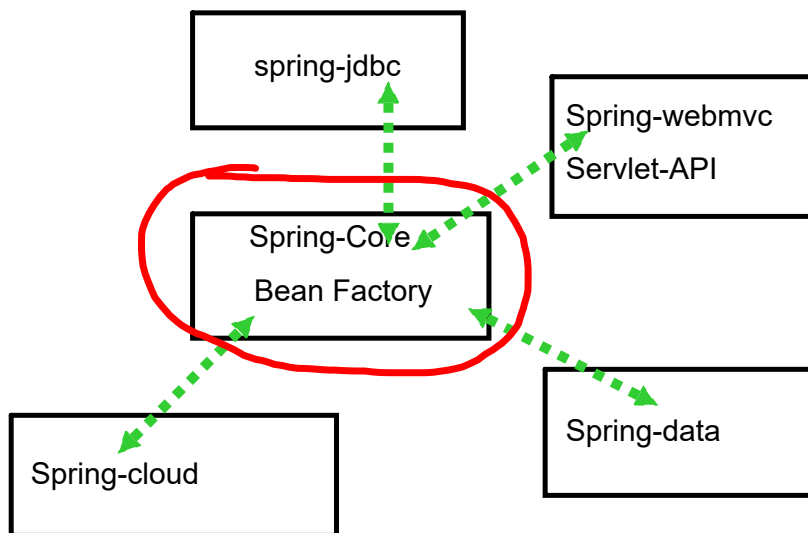
Rod Johnson

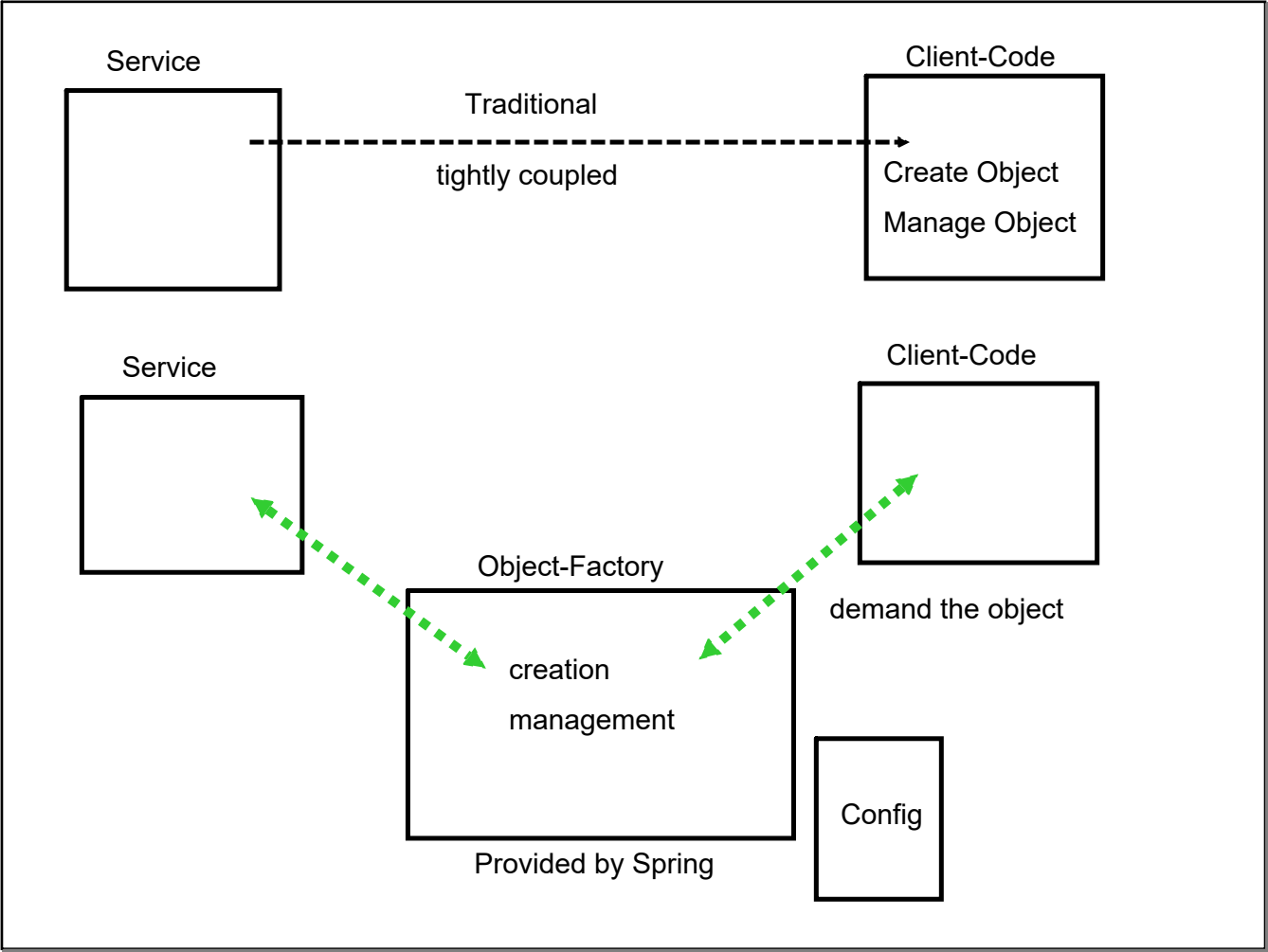
Create a tool/resources : Object Factory/Bean Factory
create and manage object

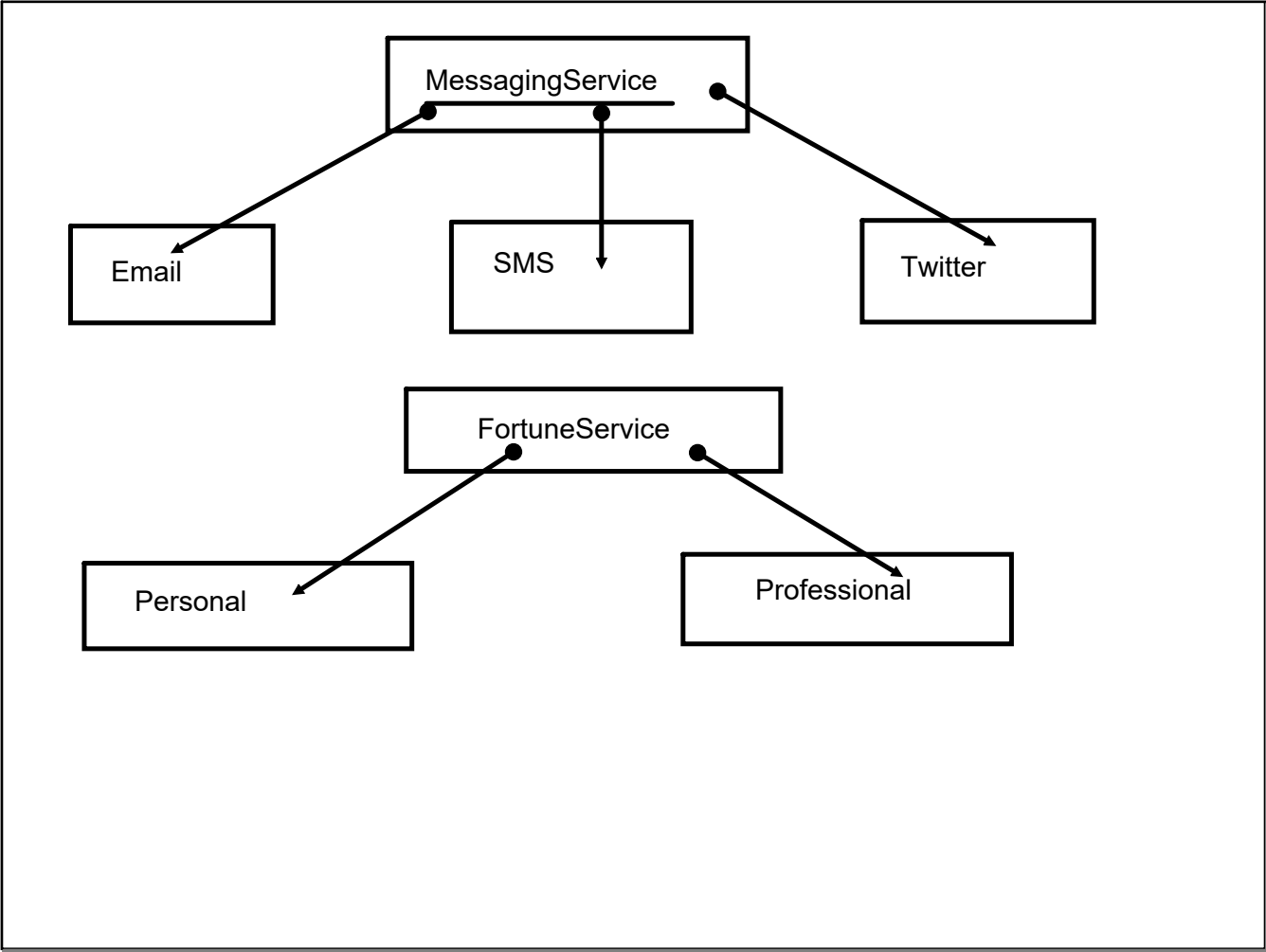
Spring : Lightweight

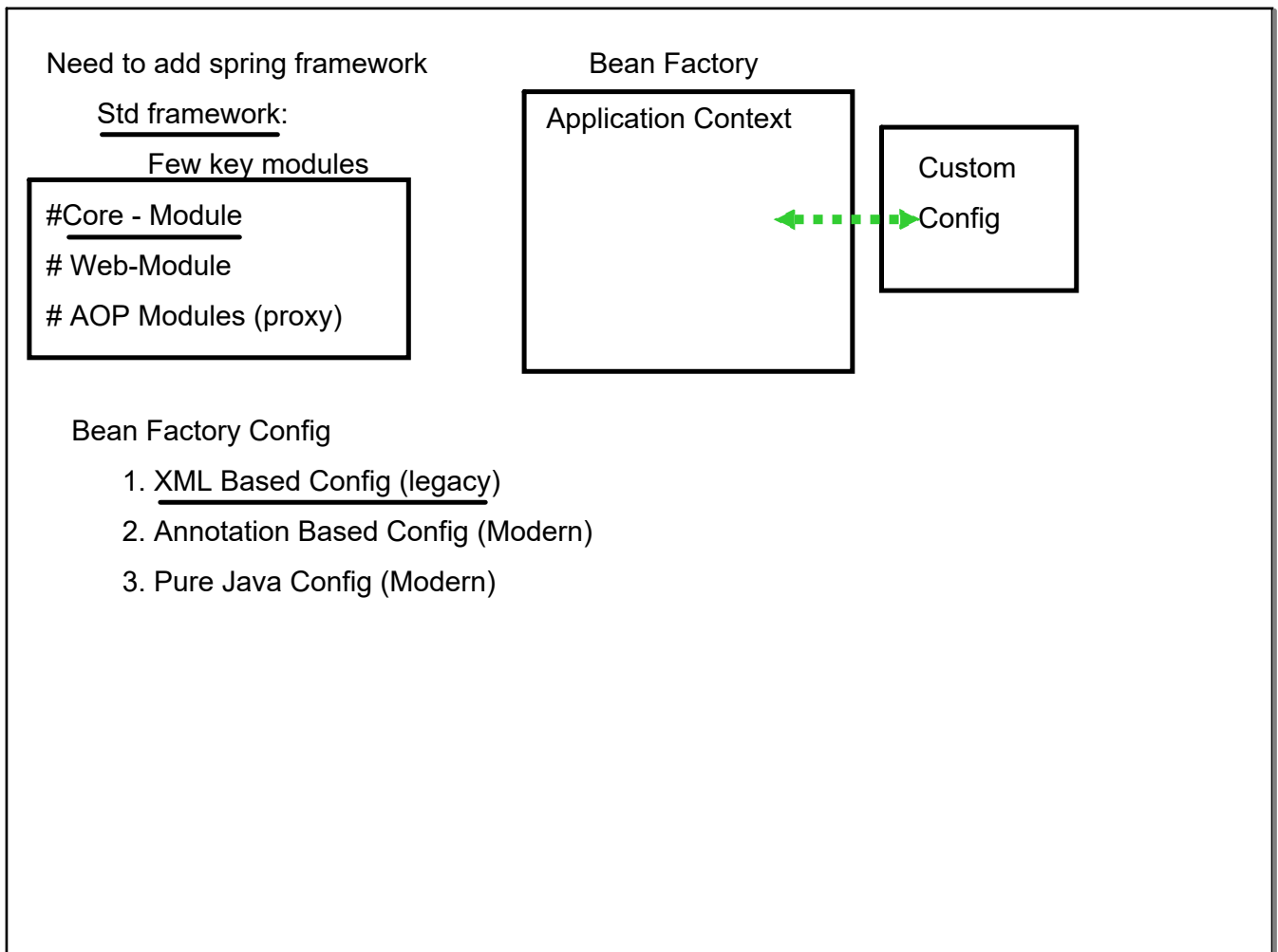


1. All implementation will be based on Object/Bean Factory
2. highly modular in nature
3. All implementation will be based on POJOs









XML Based config : XML file + with spring dependency add (additional tags)
xml config file

BEANS : Container(Object/Bean Factory) Managed Object

Two key principals of Bean Factory

1. IoC : Inversion of Control
2. DI : Dependency Injection

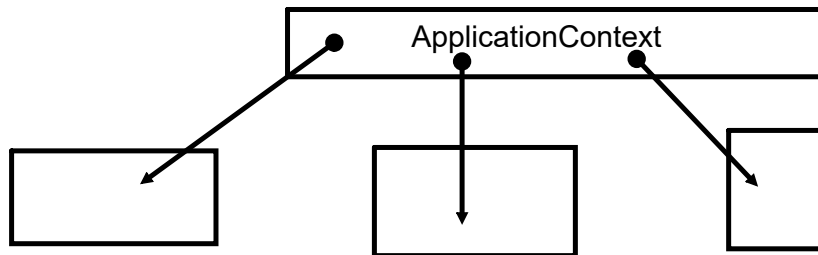
IoC : Outsourcing the (control of)creation and management of Object

Bean Factory :

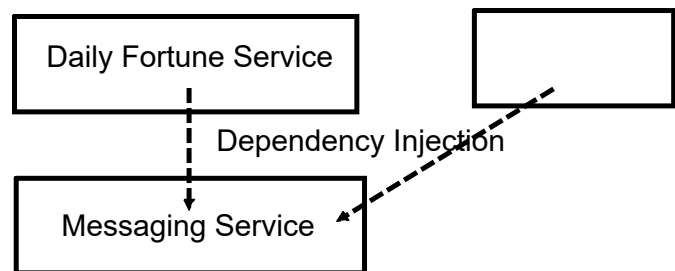
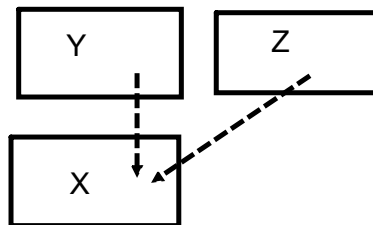
to represent multiple classes

type of config (xml, java...)

env (java, web ...)



Dependency

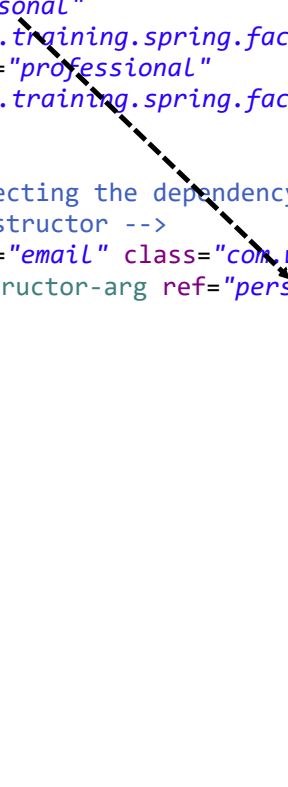


Two types of Dependency injection

1. Constructor based
2. Setter based

```
<bean id="personal"
class="com.wf.training.spring.factory.service.PersonalFortune"></bean>
  <bean id="professional"
class="com.wf.training.spring.factory.service.ProfessionallFortune"></bean>

  <!-- Injecting the dependency : How -->
  <!-- Constructor -->
  <bean id="email" class="com.wf.training.spring.factory.service.EmailService">
    <constructor-arg ref="personal"/>
  </bean>
```



Injecting Literal Values :

Delegate values to a text file : properties file
key-value pair

IoC | DI : Create a Bean

Managing the Bean

1. Scope
2. Life cycle

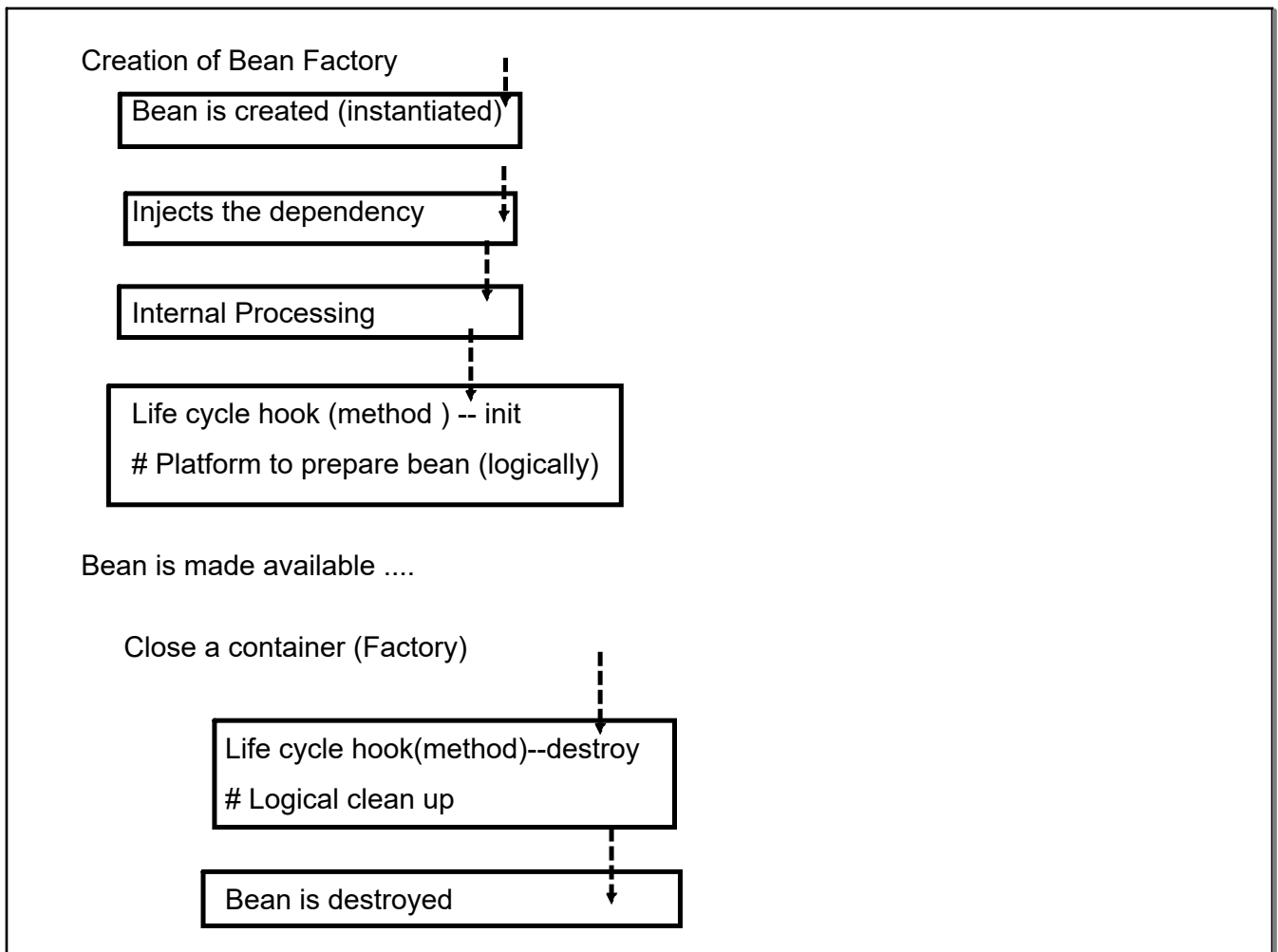
Scope : Accessibility of Bean

by-default : singleton
explicit : prototype

request

session : Web App

application



Prototype beans : Bean Factory does not manage the complete life-cycle

Annotation based configuration

still xml file : referencing the path/location

want to create and expose a bean of Messaging service

@Component : Any class decorated this ann. will instantiated by Bean Factory

XML : need to mention package(s) to scan for @Component ann.

id : Class Name is by-default considered as id : first character in lower case

DI ways:

1. Constructor
2. Setter Based
3. Field Based

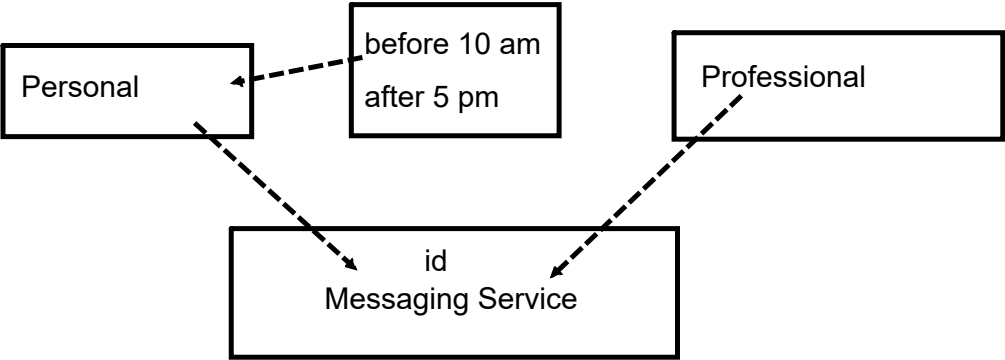
@Autowired : searches for bean of param type, if found , inject it

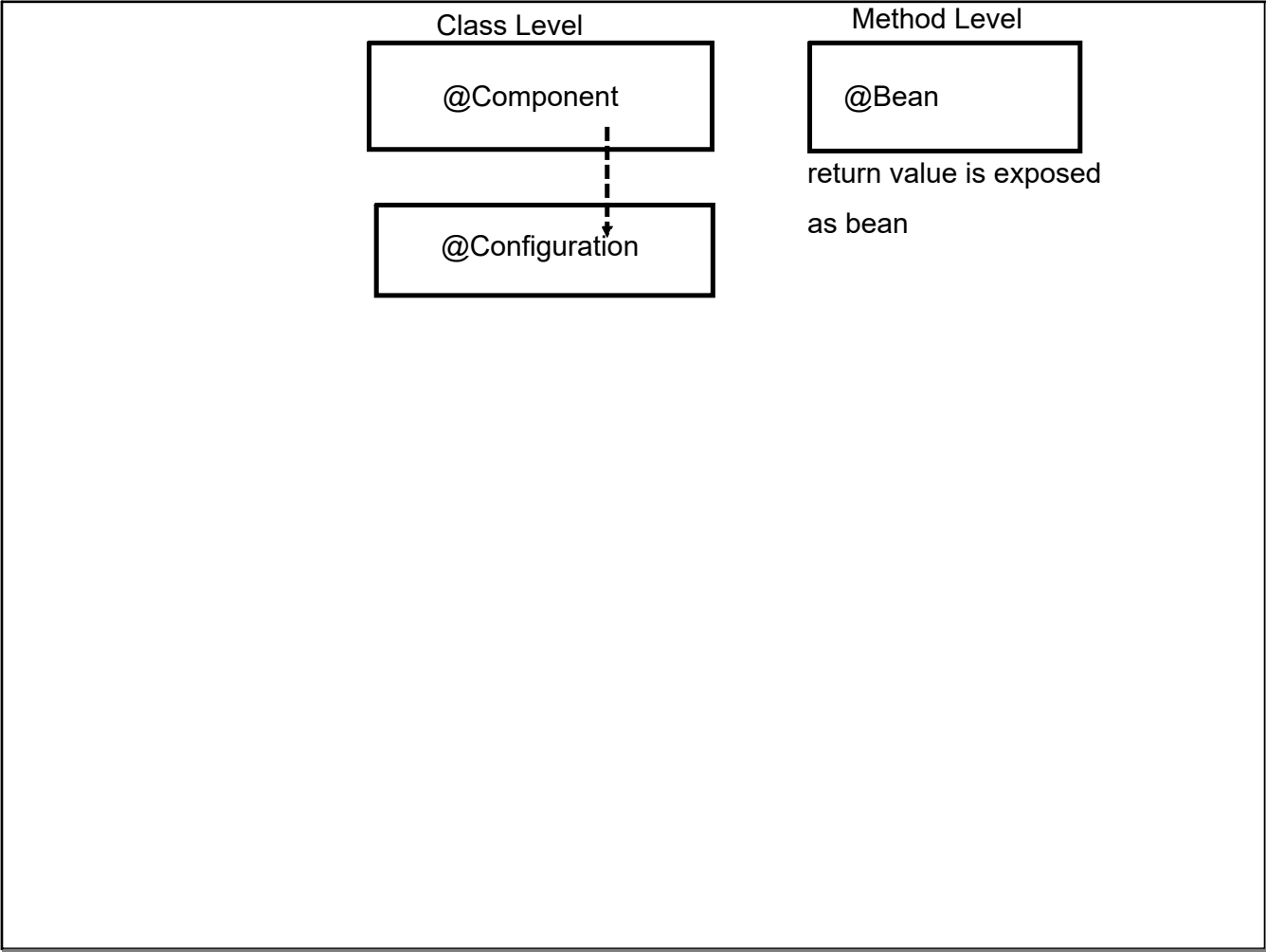
@Qualifier : differentiate the bean

Literal Value : @Value

@Scope : for defining the scope of the bean

Pure Java Config
XML file will be replace with a Java class

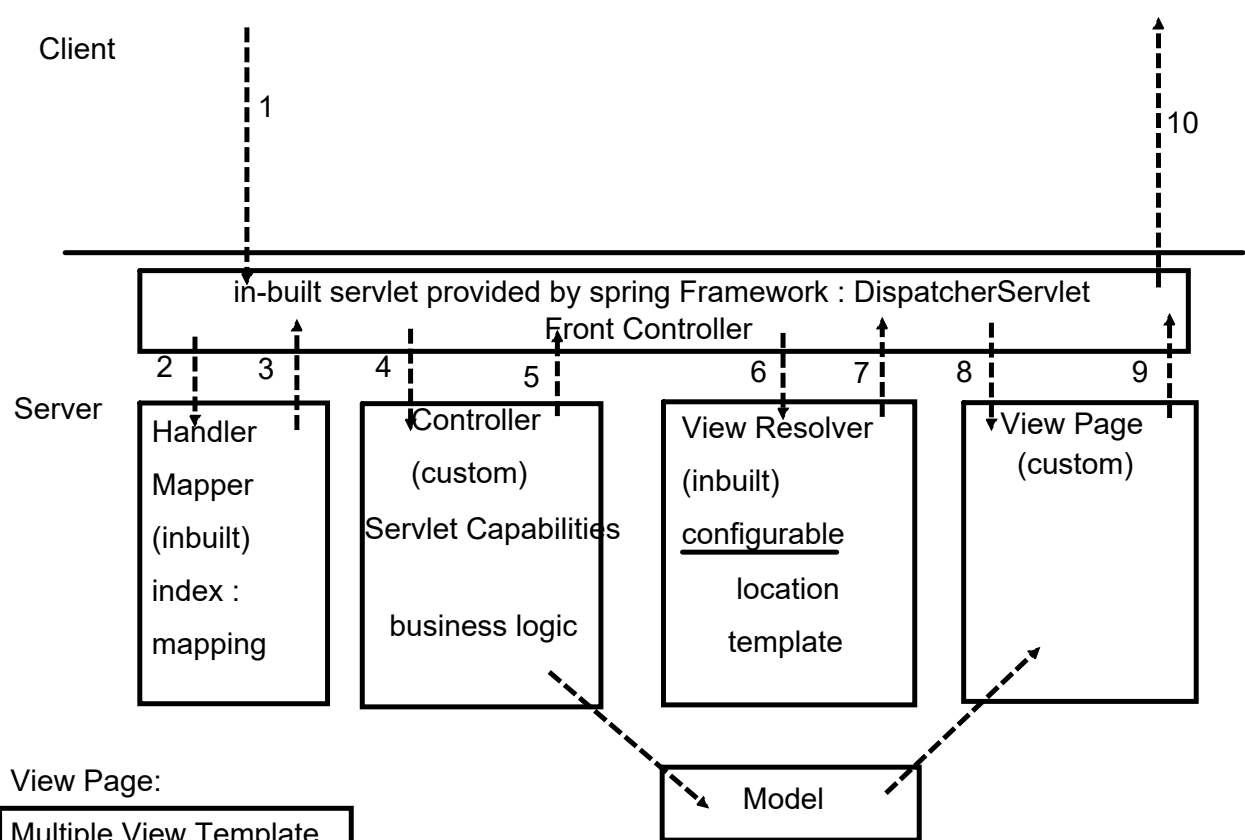




Spring Web-MVC Module : implements MVC architecture strictly

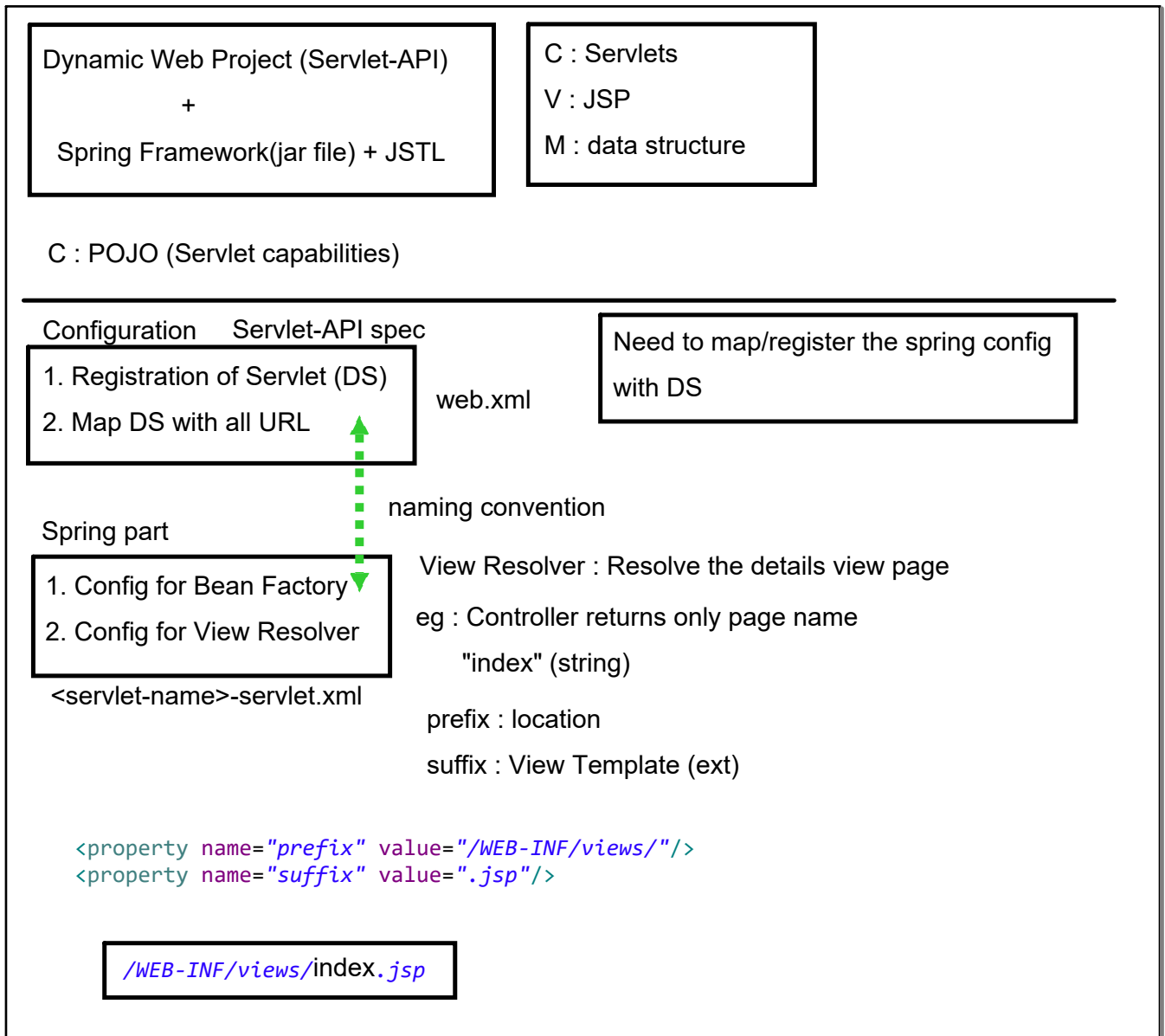
integrate the Servlet API and implement it in MVC

Front Controller Design Pattern

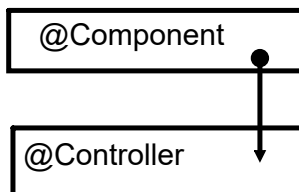


View Page:

Multiple View Template
default : JSP + JSTL
Thymeleaf
Mustache
Velocity
Tiles
FreeMarker



Controller : POJO, registered with HAndler Mapper



All handler methods must have unique URL mappings