

Spring Framework:

Framework for building Java Applications

Simpler and lightweight alt to J2EE

Early version of J2EE complex object management

Multiple deployment descriptors

Multiple interface

Poor Performance

Rod Johnson

Lightweight Object management tool : Object Factory/Bean Factory/Application Context

Spring :

=> highly Modular

=> Independent to be used : loose coupling among modules

=> Java POJOs : Lightweight dev process

Relationship among resources of Spring application

1. IoC : Inversion of control
2. DI : Dependency Injection
3. AOP : Aspect Oriented Programming (proxy)

Modules

Core Container : Bean Factory

Web Layer : MVC Framework

Data Access Module : JDBC/ORM/Transaction

Infra : AOP/Messaging

Testing : JUnit/Mocking

Spring Projects:

Spring Data

Spring Cloud

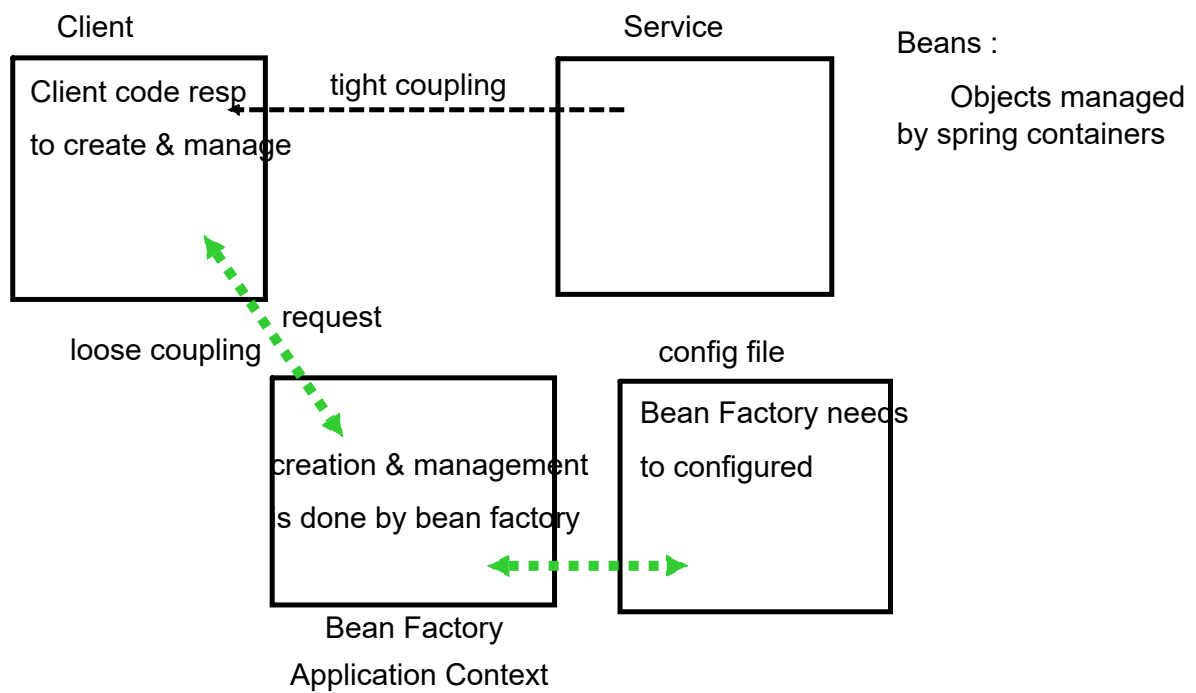
Spring security

WebFlux

1. IoC : Inversion of Control

Outsourcing the Creation and Management of Object :

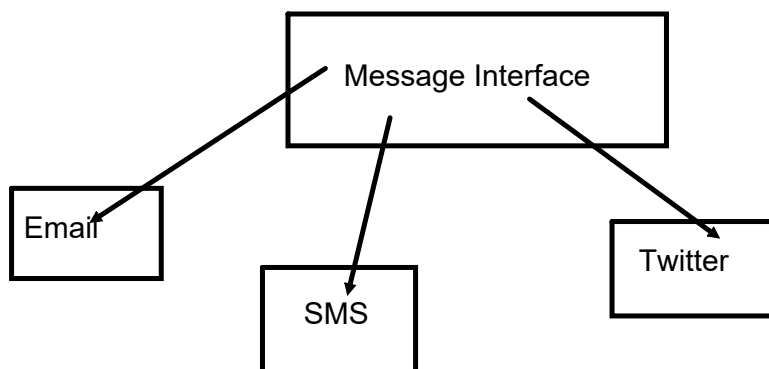
Object Factory/Bean Factory



Spring Jar files : <https://repo.spring.io/release/org/springframework/spring/>

Configuration of Bean Factory

1. XML based config (legacy)
2. Annotations
3. Pure Java Code



```
<bean id="mbservice" class="com.wf.training.spring.service.EmailService">
```

key: by which bean is exposed

class: whose bean we want to create

DI : Dependency Injection

Creation of complex object might have dependency on other object :

injecting those dependency with the help of Spring Container

XML based config:

1. Constructor based
2. Setter based



literal values should not be integrated in config file

==> property file to keep the literal values

KEY - VALUE PAIRS

Referred by SpEL

Scope of Beans

Default scope : Singleton

Only one instance which shared among all calls

Scope possibility :

singleton

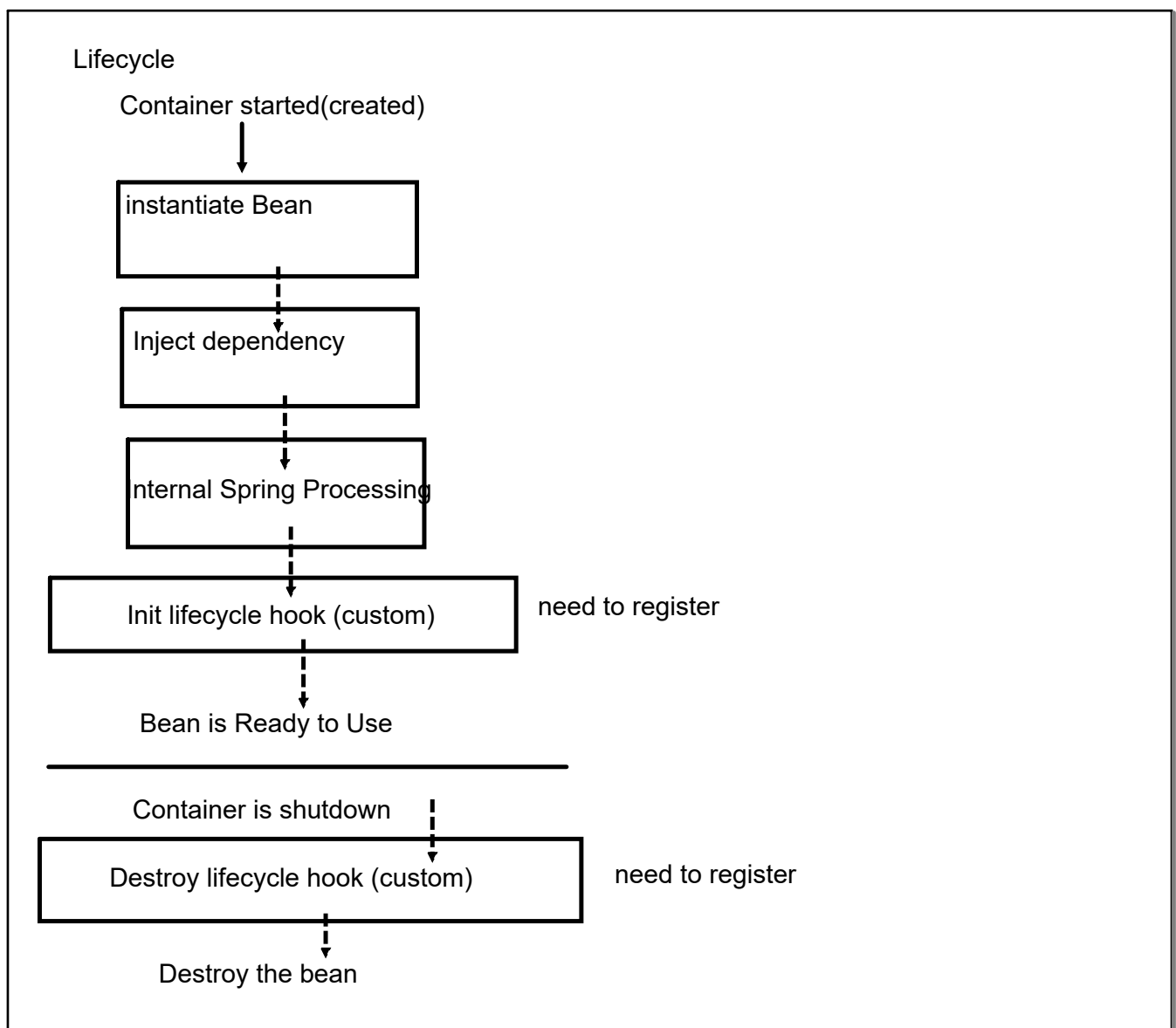
prototype : a new bean would be created

Web Context :

request

session

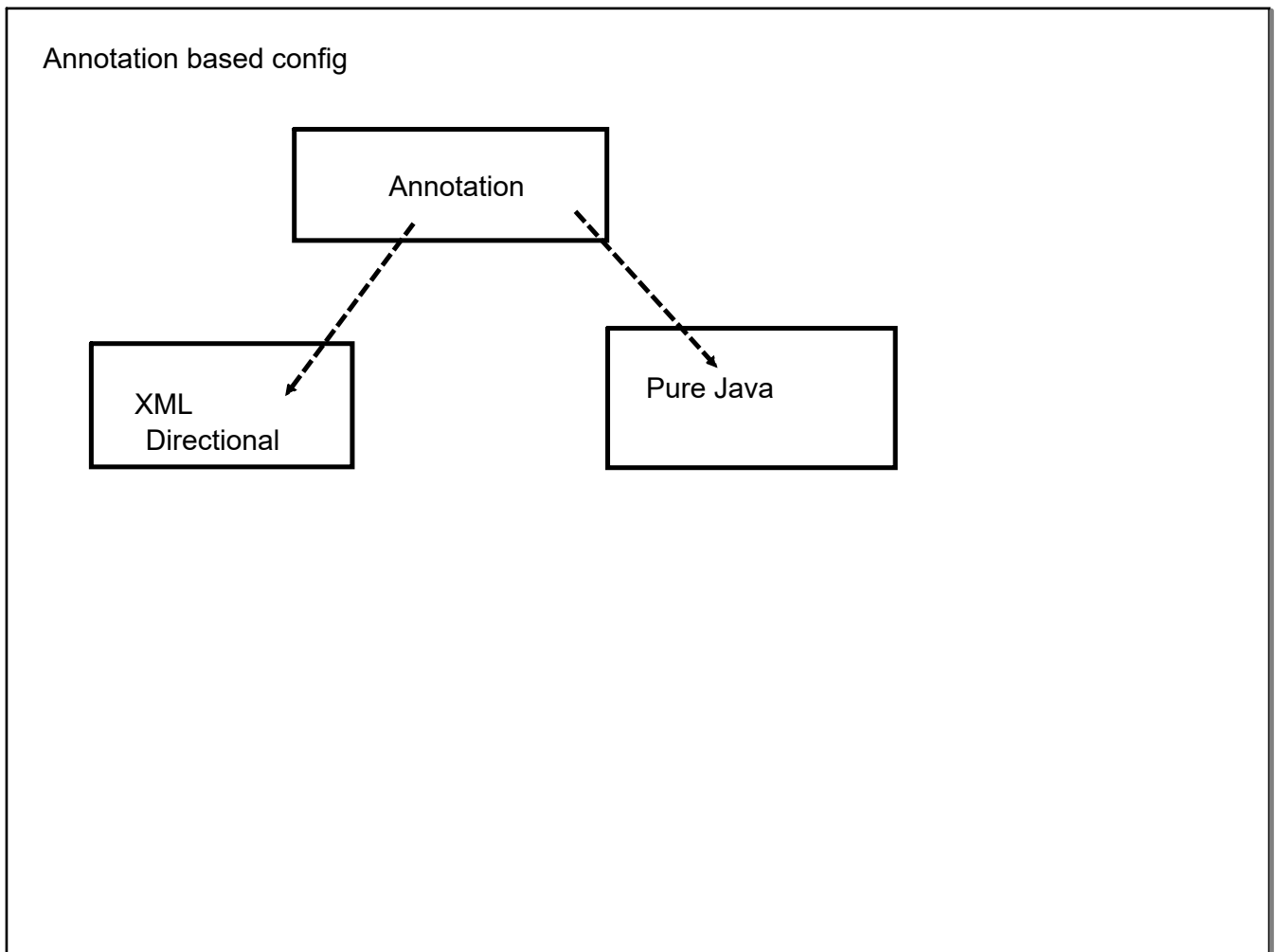
global-session



Life cycle hook method

1. any name
2. any access modifier
3. not static
4. they may return values but can't capture
5. No parameter

Prototype : Spring container does not maintain lifecycle



@Component : informing spring to create and manage bean of that class

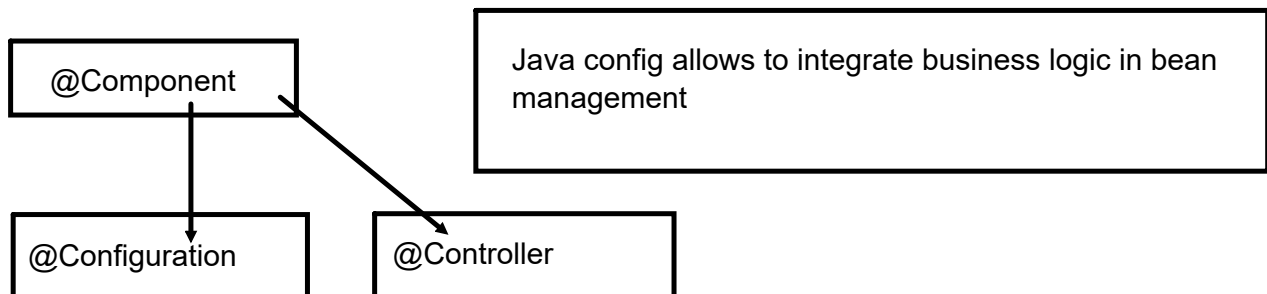
Every bean must be exposed by an id (Class name (with first char small) is default id)

Annotation based DI

1. Constructor
2. Setter
3. Field

Java Based Configuration

XML file is replaced by a Java Config



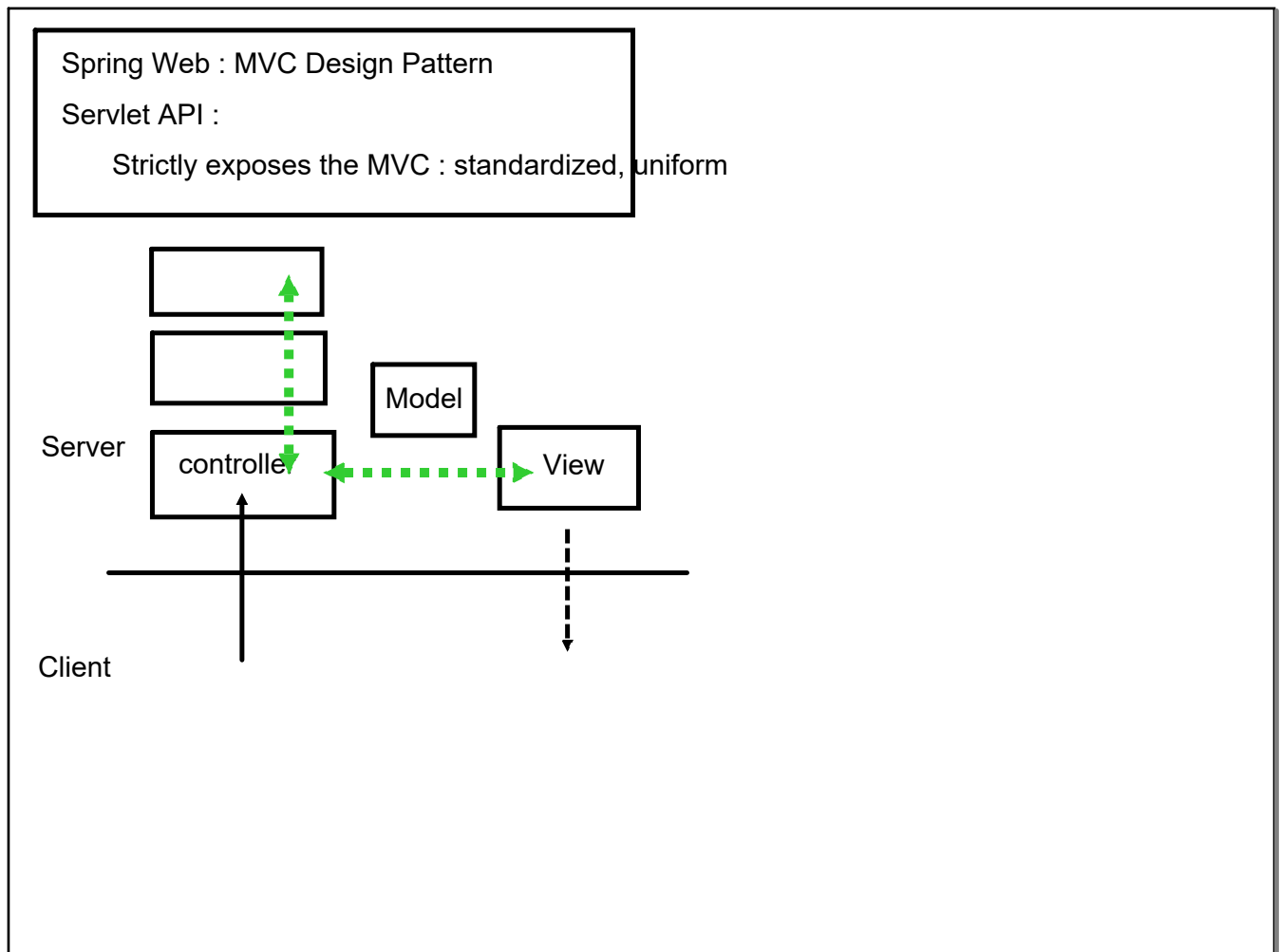
Developing Web Based Application using Spring Framework

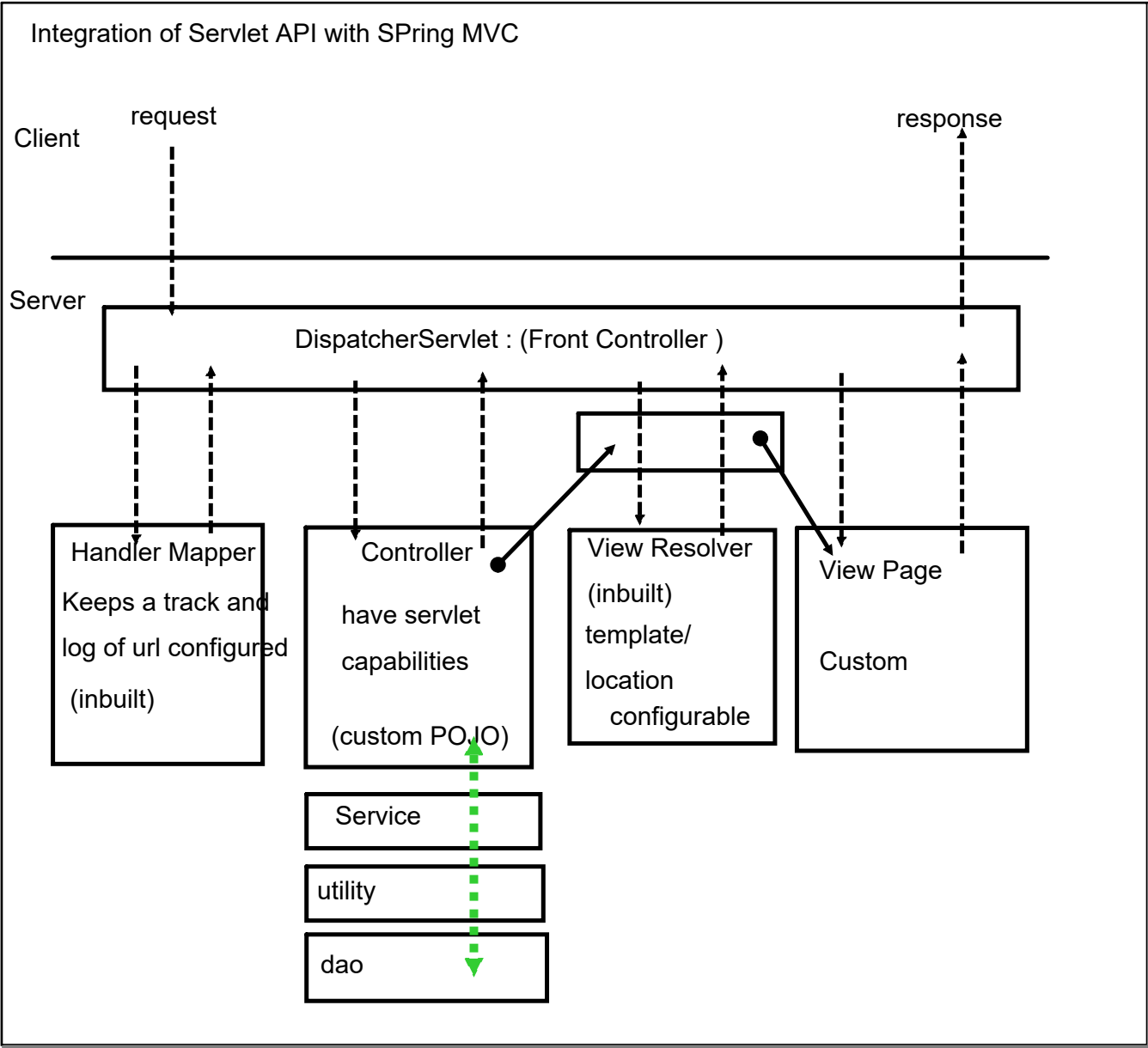
spring web-mvc
spring-core

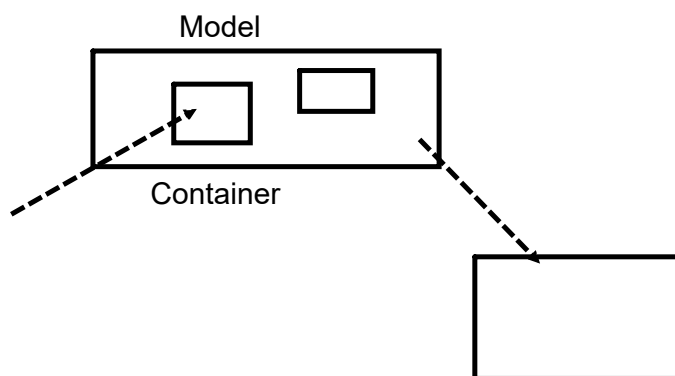
Spring also uses Servlet-API (Servlet Spec)

spring container

highly abstraction : POJOs







Spring MVC:

Multiple View Templates :

Traditional Default : JSP + JSTL (lib added for jsp+jstl)

Thymeleaf

FreeMarker

Tiles

Velocity

Mustache

Two configurations

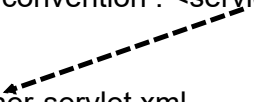
1. Servlet API based : Need register and configure the Dispatcher Servlet
2. Spring config for that servlet, helpers for servlet

xml file needs to tightly bind with servlet

naming convention : <servlet-name>-servlet.xml

eg:

dispatcher-servlet.xml



View Resolver : Creating and exposing a bean of View Resolver

Controller revert back : Name of view page

eg: "my-view"

[/WEB-INF/views/my-view.jsp](#)

