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

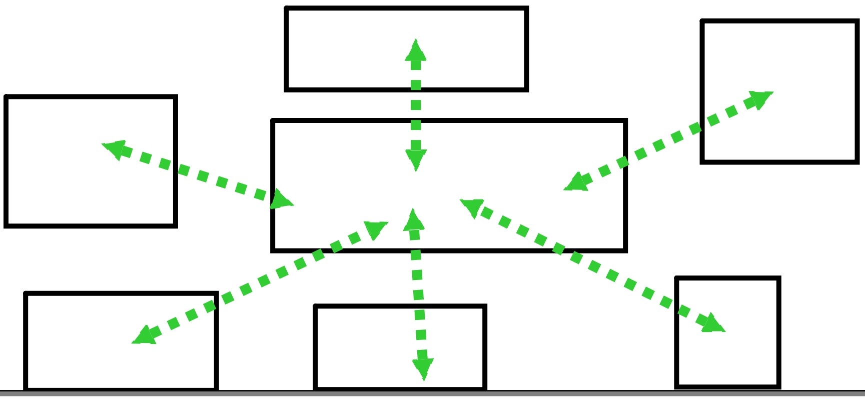
    Uses lots of deployment descriptors : xml files

Rod Johnson

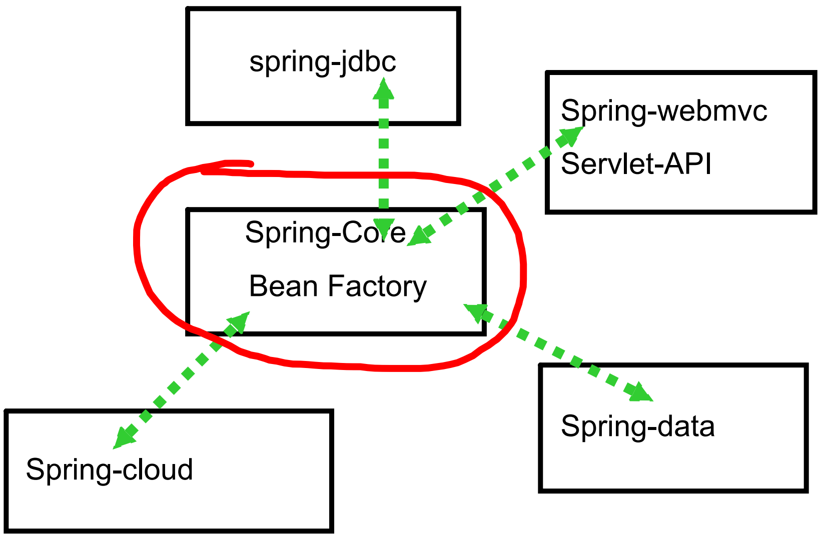
    Create a tool/resources : Object Factory/Bean Factory

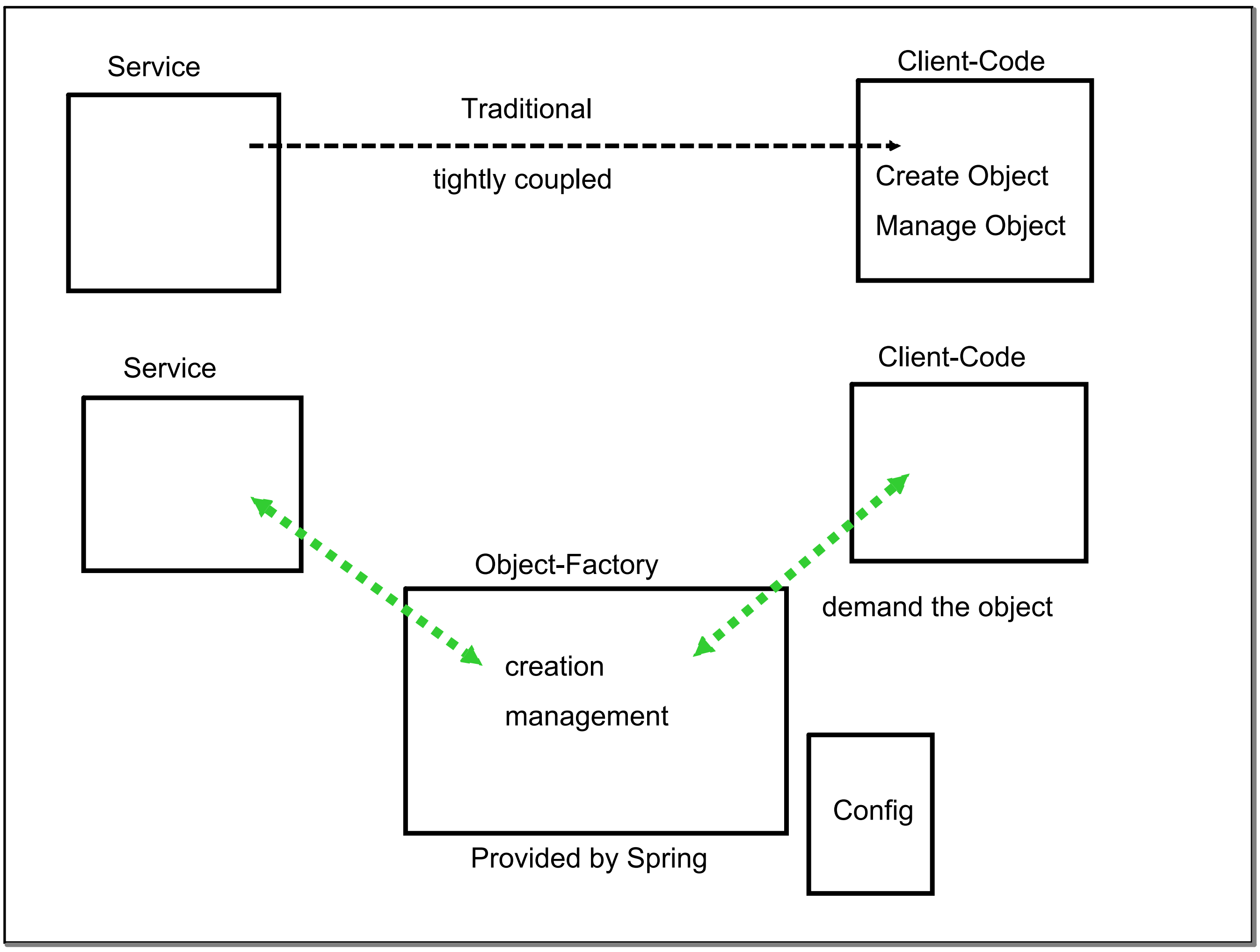
    create and manage object

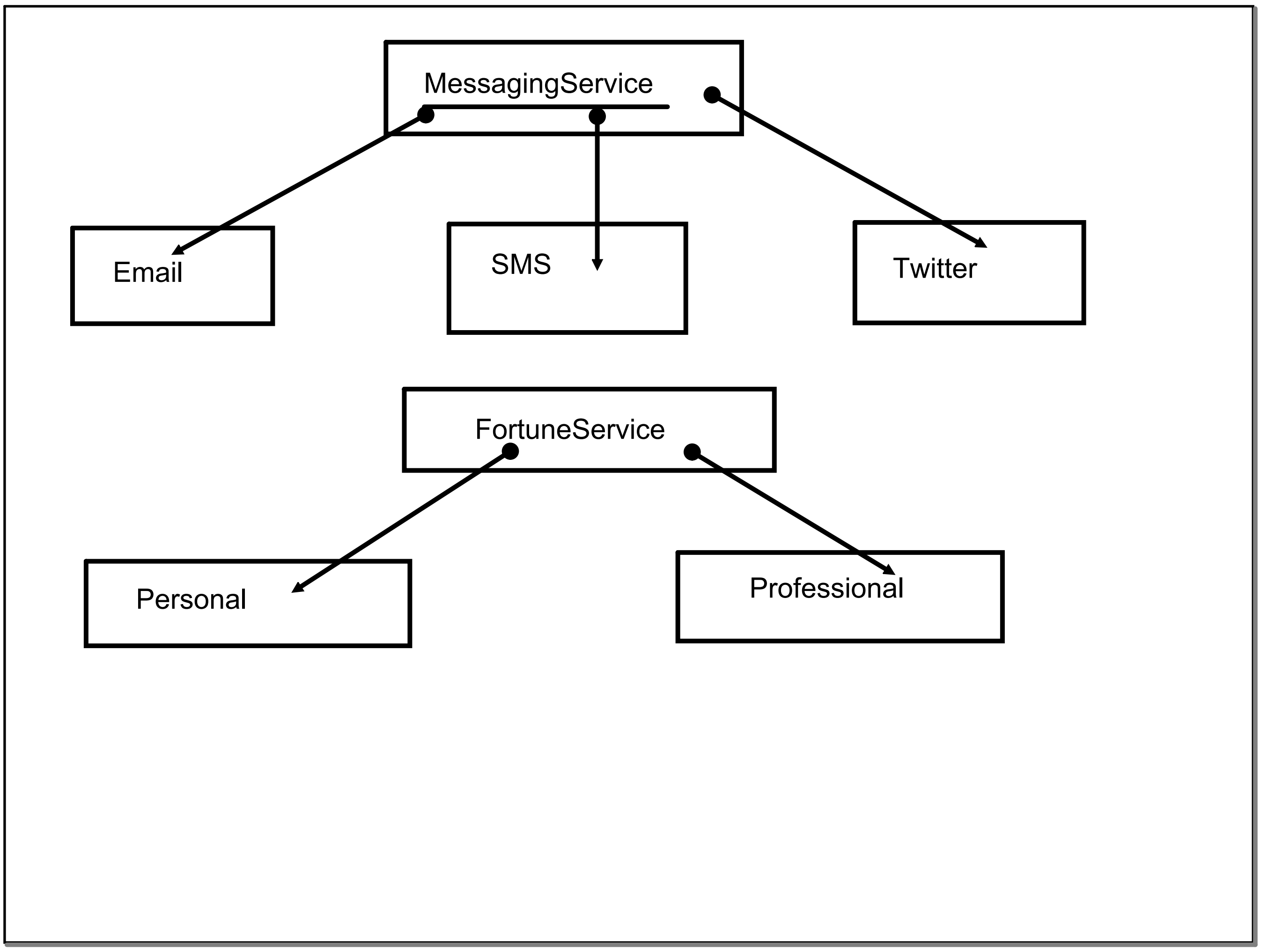
Spring : Lightweight



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




Service
Traditional
tightly coupled
Client-Code
Create Object
Manage Object
Service
Client-Code
Object-Factory
creation
management
demand the object
Config
Provided by Spring


MessagingService
Email
SMS
Twitter
FortuneService
Personal
Professional

Need to add spring framework

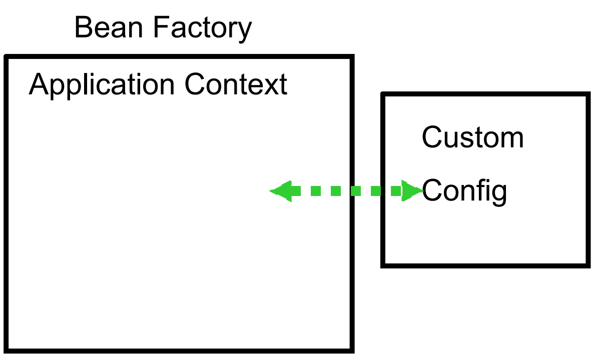
Std framework:

Few key modules

#Core - Module

# Web-Module

# AOP Modules (proxy)



Bean Factory Config

1. XML Based Config (legacy)
2. Annotation Based Config (Modern)
3. Pure Java Config (Modern)

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

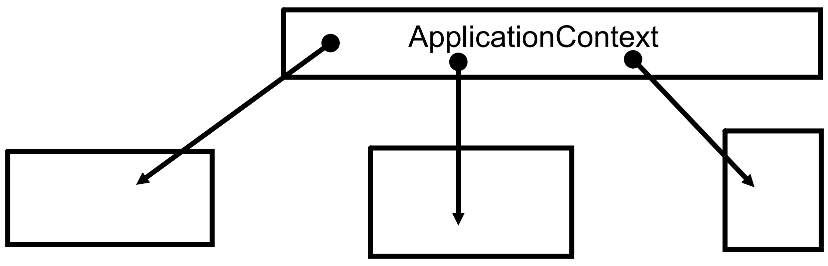
IoC : Outsourcing the (control of)creation and management of Object

Bean Factory :

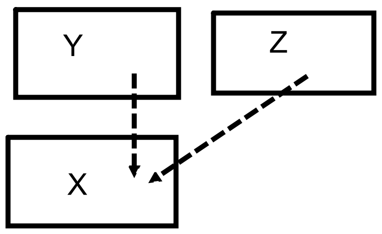
to represent multiple classes

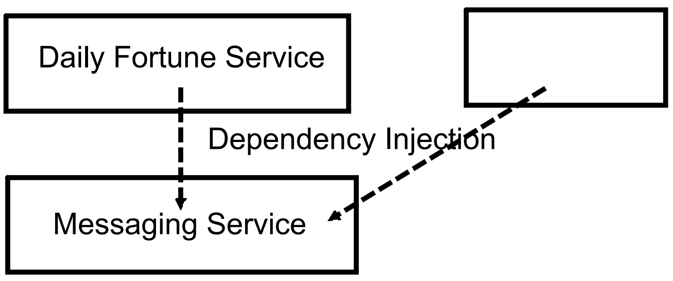
# type of config (xml, java...)

# env (java, web ...)



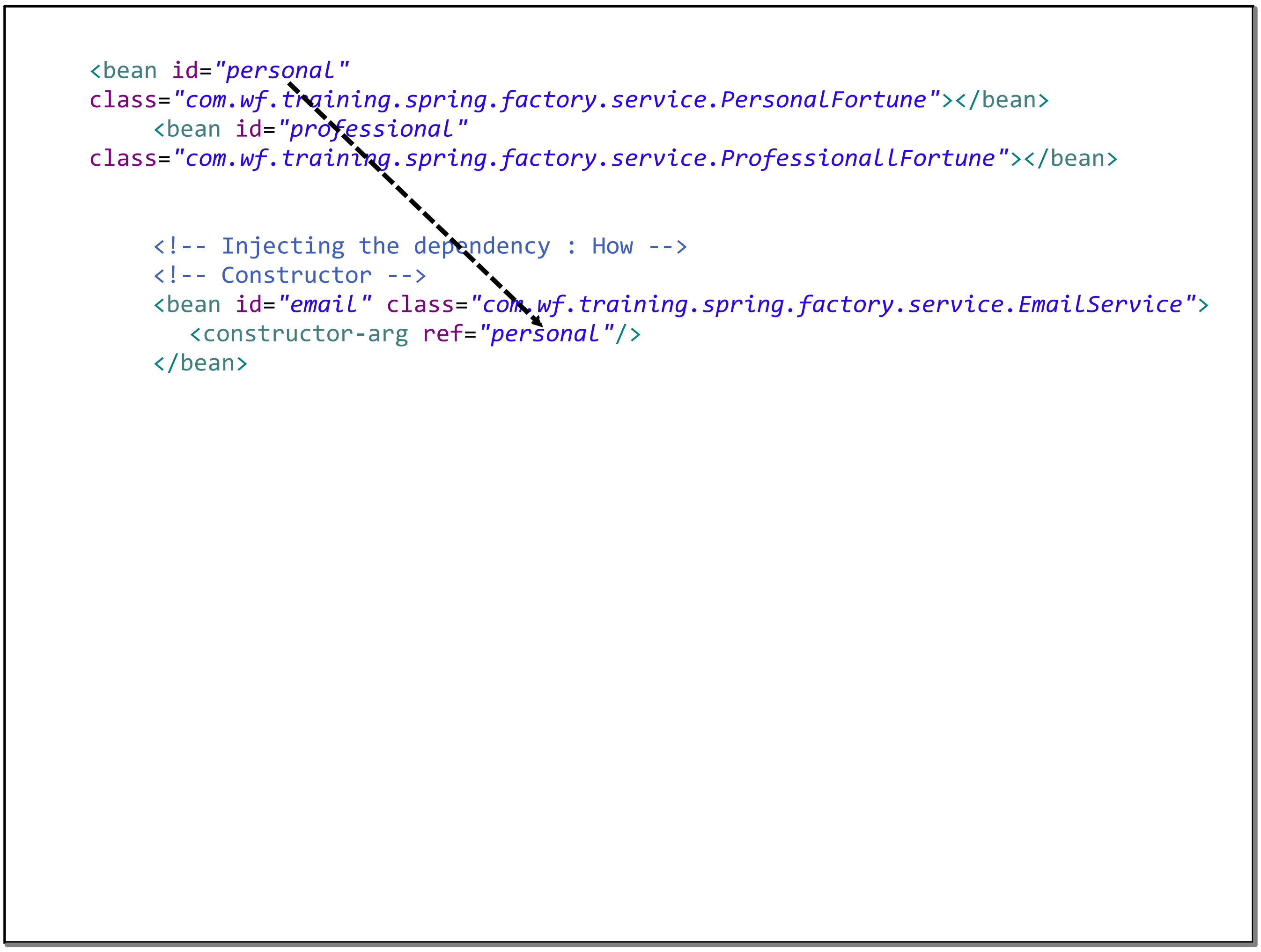
Dependency





Two types of Dependency injection

1. Constructor based
2. Setter based



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


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