Java 8

Classical : Imperative

How

Object mutability

Java 8 : Declarative Style

What we want

Object immutability

Interface:

default method (definition)

static method (definition)

collection api

interface (10 functionalities + 2)

stream

Object Oriented approach : interface

Functional Interface :

Contain only one abstract method

might have static , default method in any count

Lambda:

anonymous function

no method param type, return type

not be encapsulated in any class

can be assigned to a variable of functional interface

```
() -> <single line>
()->{
        // multi line
      }
```

```
(msg)->{}
msg -> {}
(a,b) -> {}
(a,b)->{
            int sum = a+b;
           return sum;
           }
(a,b)-> a+b; // by default associated with return statement
```

the method signature of the only abstract method of Functional interface must match with method signature of lambda expression

java.util.function

functional interface containing some very common prototype method

4 categories

- Consumer
- Predicate
- Function
- Supplier

Consumer :

```
void accept(<T>)
```

Predicate

```
boolean test(<T>)
```

Function

```
<R> apply(<T>);
```

Supplier

```
<T> get()
```

Variants

Consumer : BiConsumer (Generic)

void accept(<T>,<M>)

Primitive type implementation

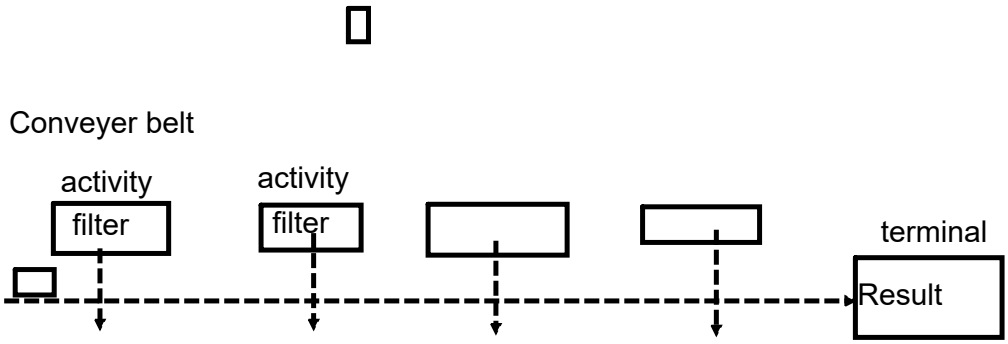
IntConsumer()

Predicate:

BiPredicate, Primitive type implementation

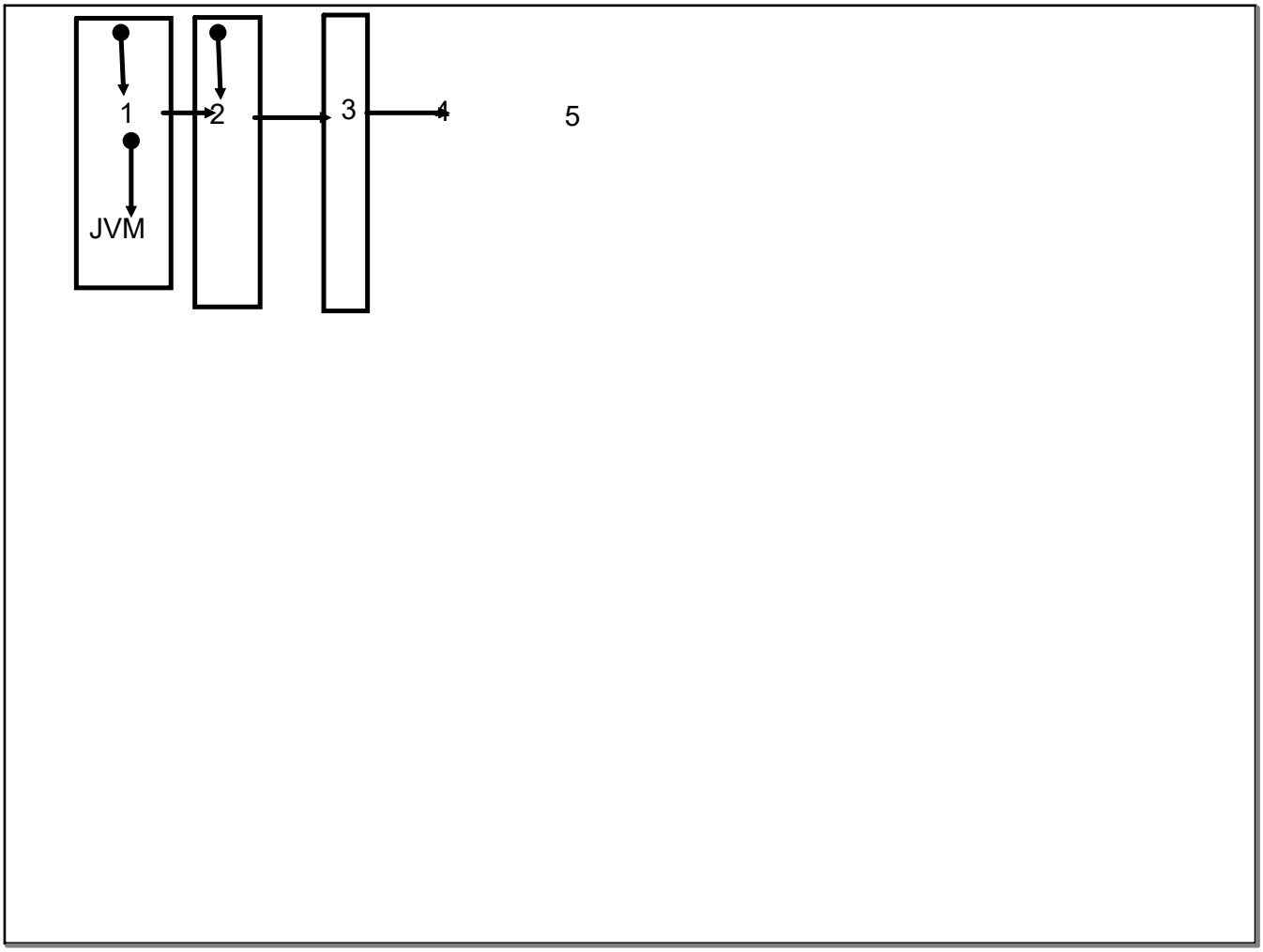
Function : BiFunction

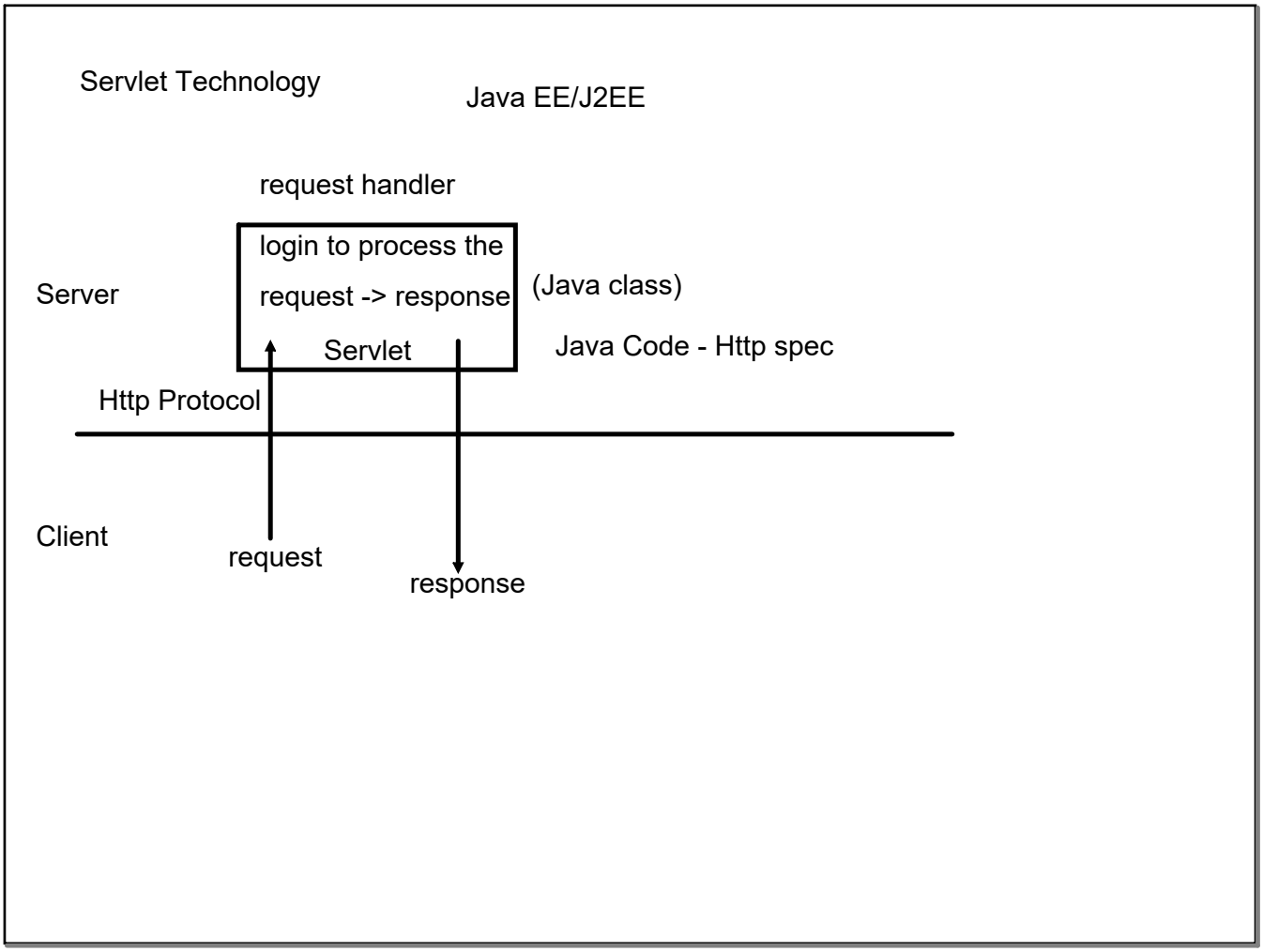
Functional programming remove overhead of creating objects and loading class files



Parallel Processing not to be preferred

1. when an external mutable object is involved
2. when the stream activities involve some inherent complexity


1
2
3
4
5
JVM


Servlet Technology
Java EE/J2EE
request handler
login to process the
request -> response
(Java class)
Server
Servlet
Java Code - Http spec
Http Protocol
Client
request
response

Java EE

Servlet

Java Class

class MyServ extends GenericServlet/HttpServlet{

}

GenericServlet : Support only generic Http Verb (Form verbs) get/post

HttpServlet : identifies HTTP Verbs (get,post,put,delete)

identifies intention of http verb

```
class MyServ extends GenericServlet/HttpServlet{
      <---- lifecycle ------>
      init (load)
      service(GenericServlet) (doPost,doGet,doPut,doDelete: HttpServlet) : request/response
      destroy (unloaded)
}

  service/(doPost,doGet,doPut,doDelete) (HttpServletRequest request,
                                                HttpServletResponse response){

  }
```

Spring

- Spring Core
- Spring MVC (maven)
- Spring Boot

Spring Framework : Servlet technology

CORE

Bean Factory

IoC

DI

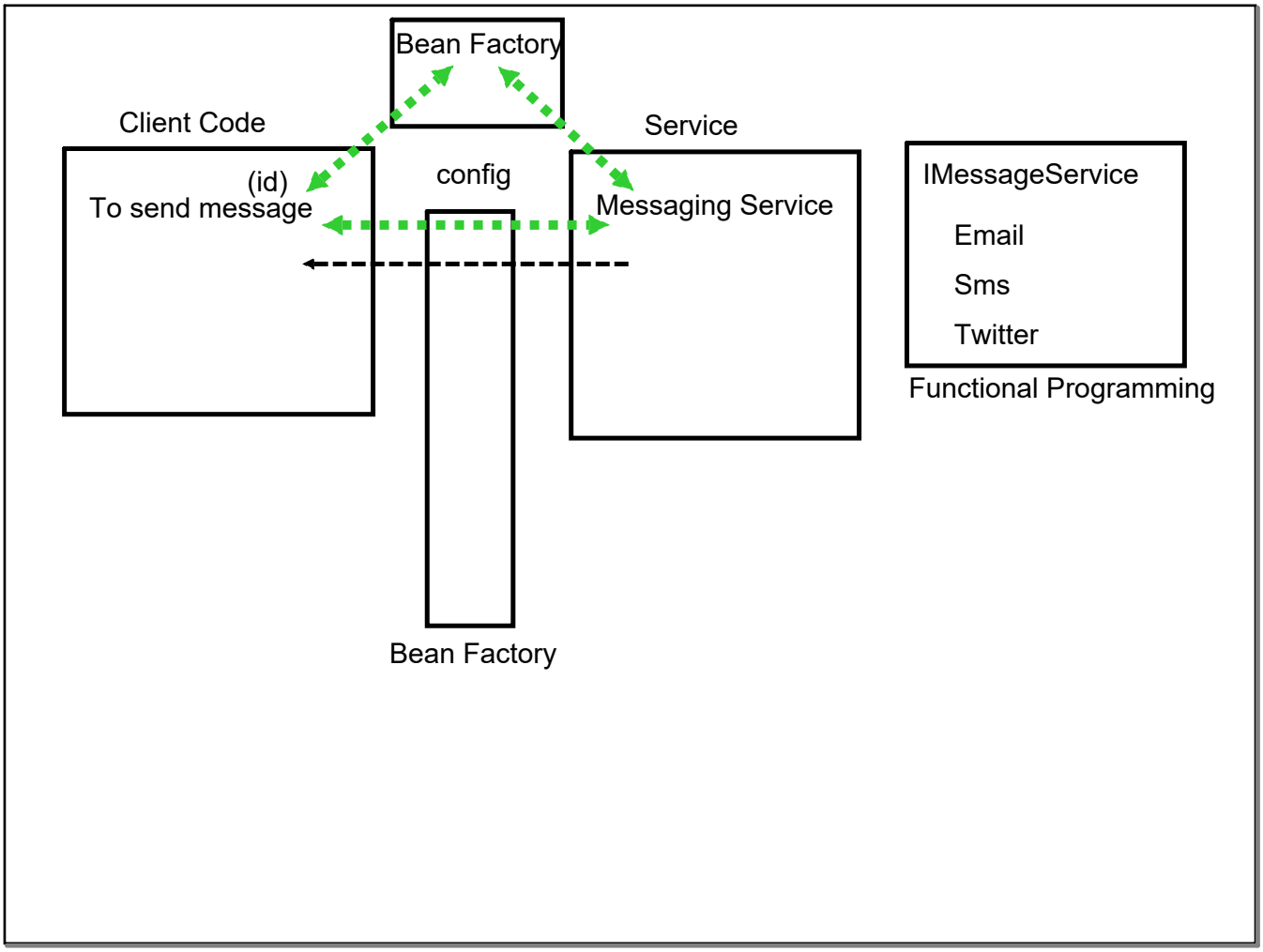
AOP

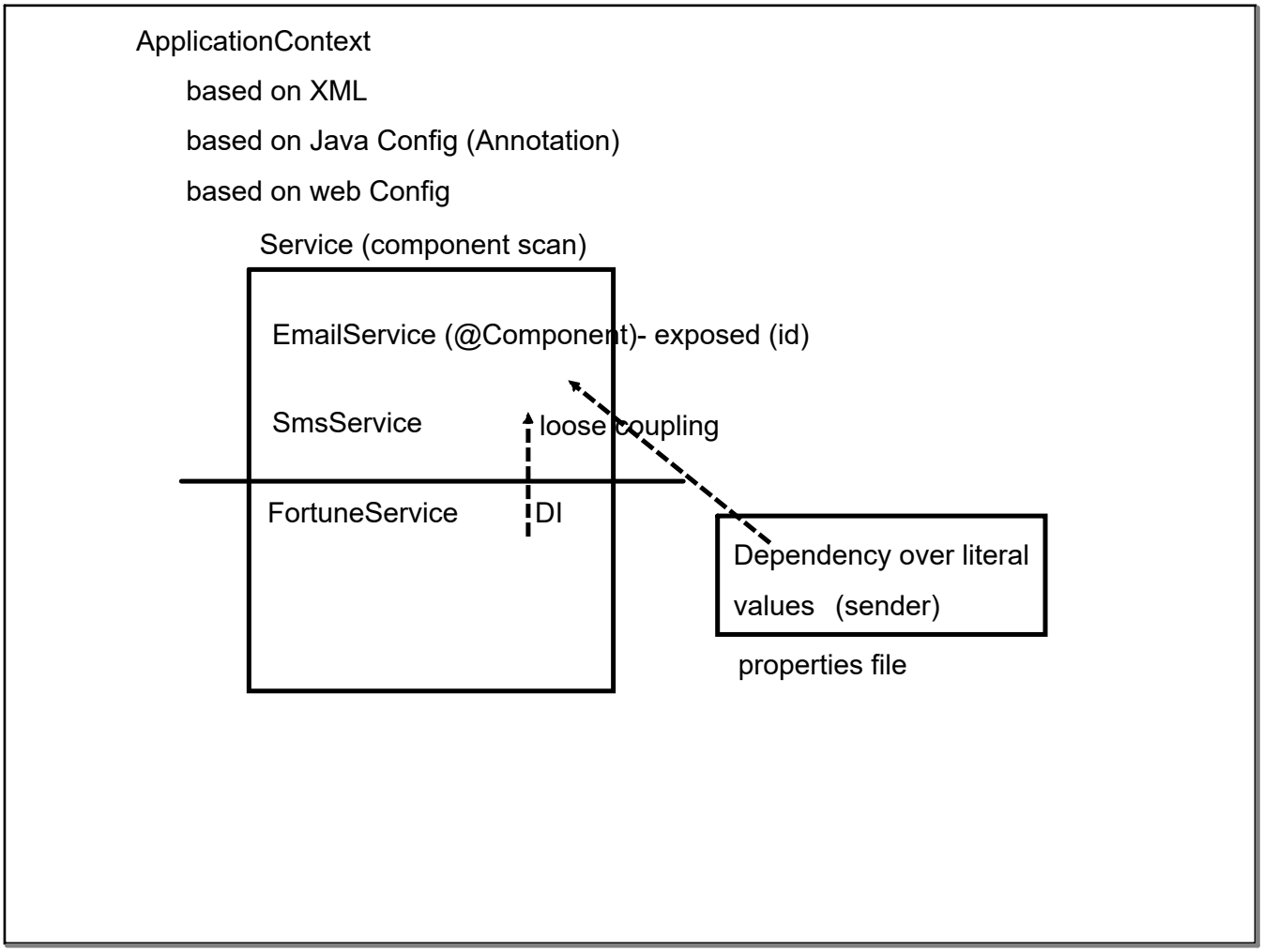
IoC : Outsourcing the creation and management of object

Bean : Java Object managed by container

AOP : Aspect Oriented Programming (Proxy)

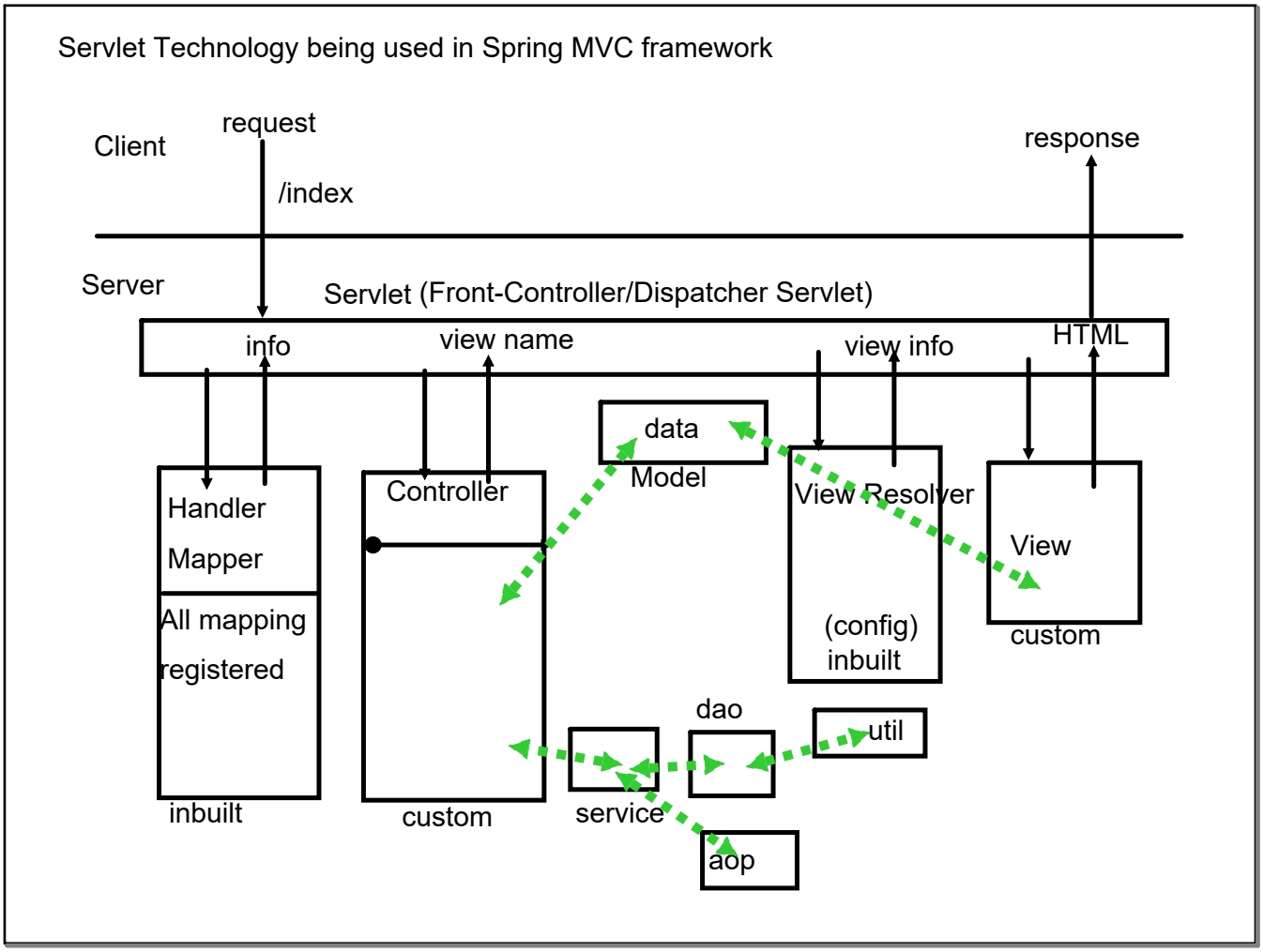
Clean, Loosly Coupled, reusable JAva Code


Bean Factory
Client Code
Service
(id)
To send message
config
Messaging Service
IMessageService
Email
Sms
Twitter
Functional Programming
Bean Factory


ApplicationContext
based on XML
based on Java Config (Annotation)
based on web Config
Service (component scan)
EmailService (@Component)- exposed (id)
SmsService
loose coupling
FortuneService
DI
Dependency over literal values (sender)
properties file

Scope : Singleton (Default)	request : single request-response cycle
Prototype	session : all request-response cycle for a particular user
	global : all request-response cycle for all user(web context)

Spring Context does not maintains complete lifecycle of Prototype bean


Servlet Technology being used in Spring MVC framework
Client
request
/index
response
Server
Servlet (Front-Controller/Dispatcher Servlet)
info
view name
view info
HTML
data
Model
Handler
Mapper
All mapping
registered
inbuilt
Controller
custom
View Resolver
(config)
inbuilt
View
custom
dao
util
service
aop

Maven:

Dependency Management

Standard folder/file system

build

test

documentation

pom : project object model

all config related to maven activity

web.xml : a must file for servlet config

web.xml : Servlet config

Custom spring servlet config (java):

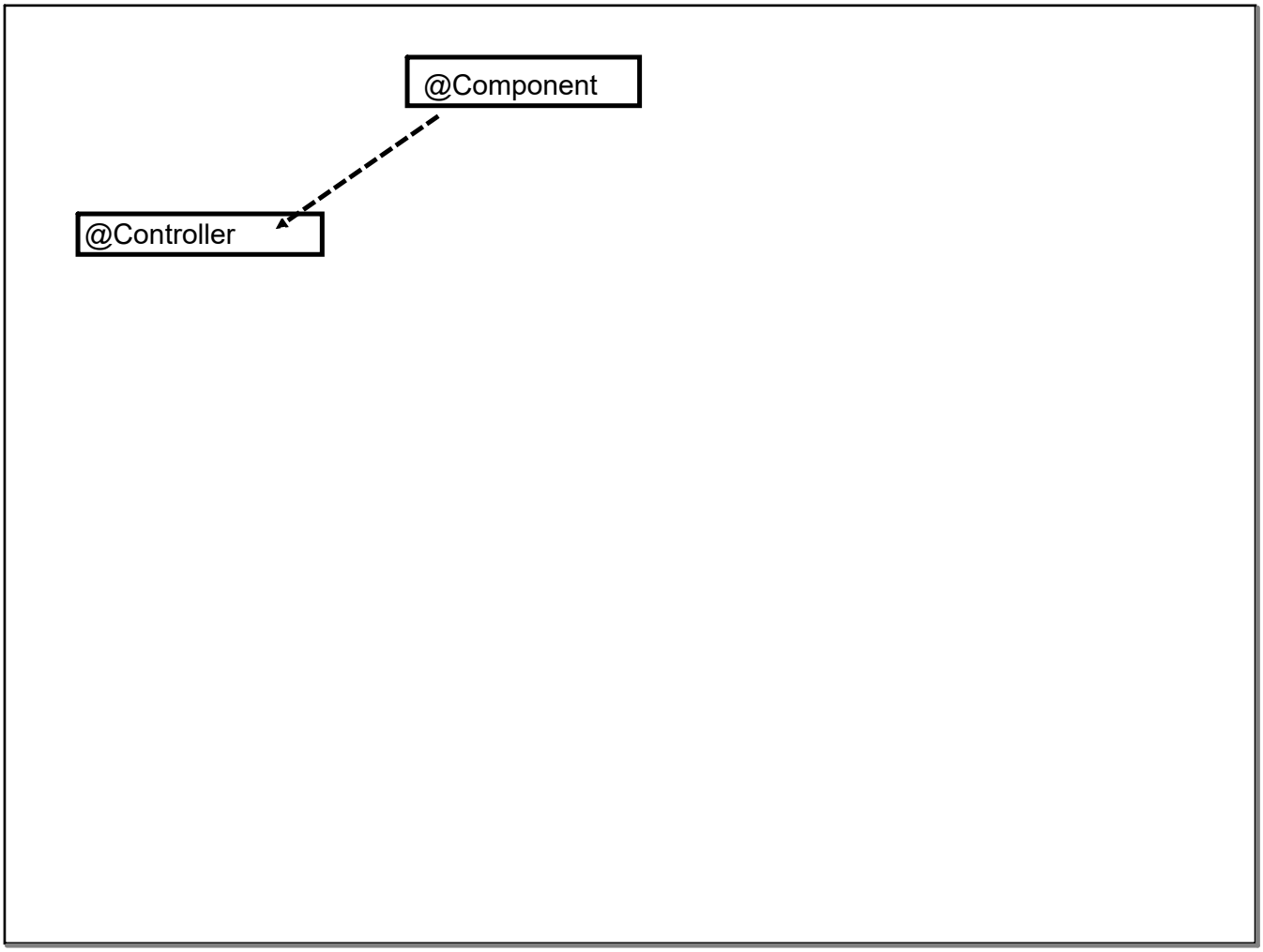
inbuilt servlet : register that servlet (DispatcherServlet : config)

config code: as per more requirement

controller code (multiple support layers)

model : data structure

views : presentation purpose


@Component
@Controller

View Resolver:

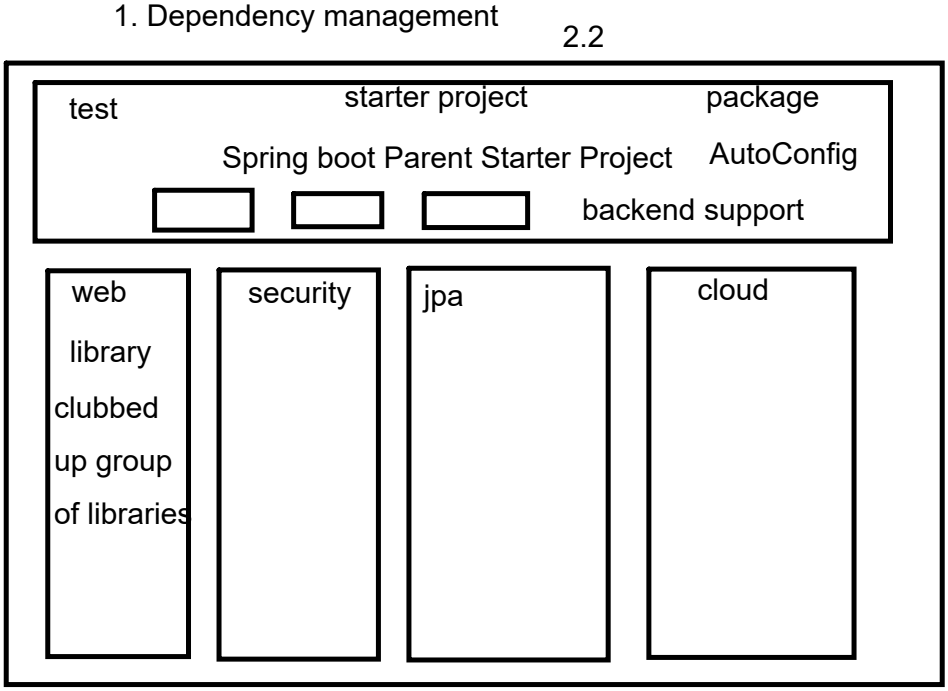
- Single View File (jsp)
- Modular View File (Tiles)
- Multipart response (as downloadable file)

=>What type of responses you want

=>What type of responses your view templates

Spring Boot:

1. Dependency management



Configuration:

Auto/Easy

=> Curated clubbed up Annotation

=>Added new annotation for custom config

=> property files : add correct key-values pair

=> adding dependency : will activate that feature and auto configure

some default behavior

spring-security (

spring-actuator

spring-devtool

web application

spring boot web application packaged as jar

standalone : executed like a simple java

Tomcat is embedded

Spring boot is self-sufficient for maven tool

eg : mvn package/test/clean/install

spring boot tool:

eg: mvnw package/test/clean/install

Spring boot application are by default

not configured to use jsp-jstl view templates

Spring boot is by default configured to use :

thymeleaf

```
> java -jar bootapp.jar server.port=9091
```