

## 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:

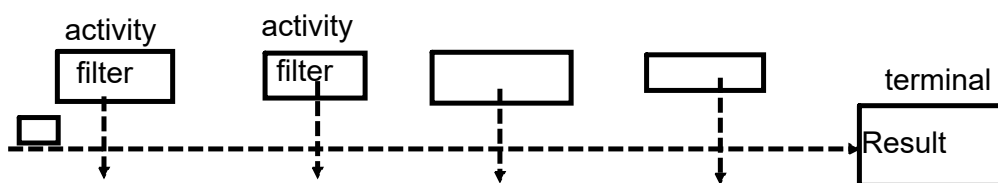
BiPredicate, Primitive type implementation

### Function : BiFunction

Functional programming remove overhead of creating objects and loading class files

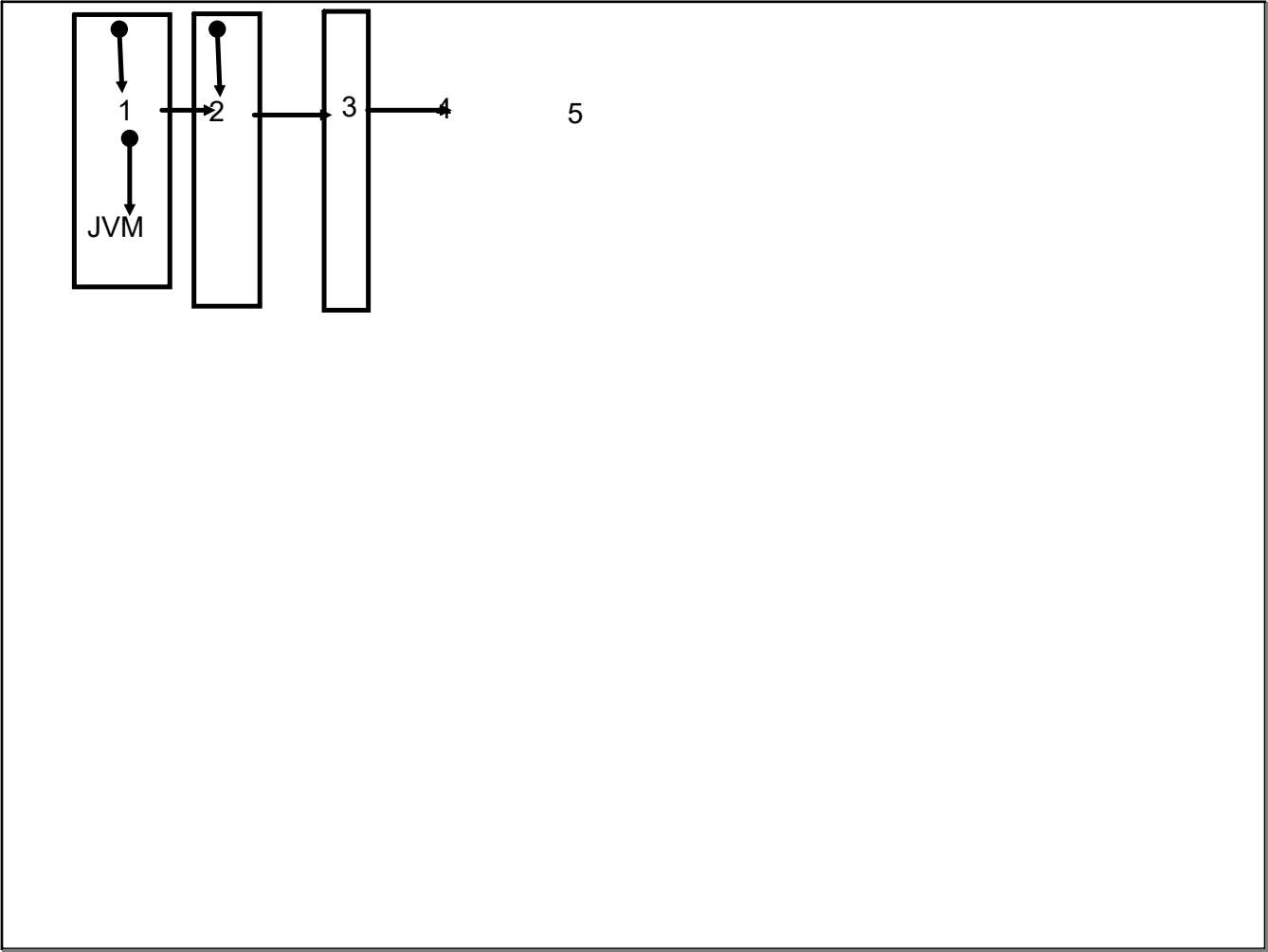


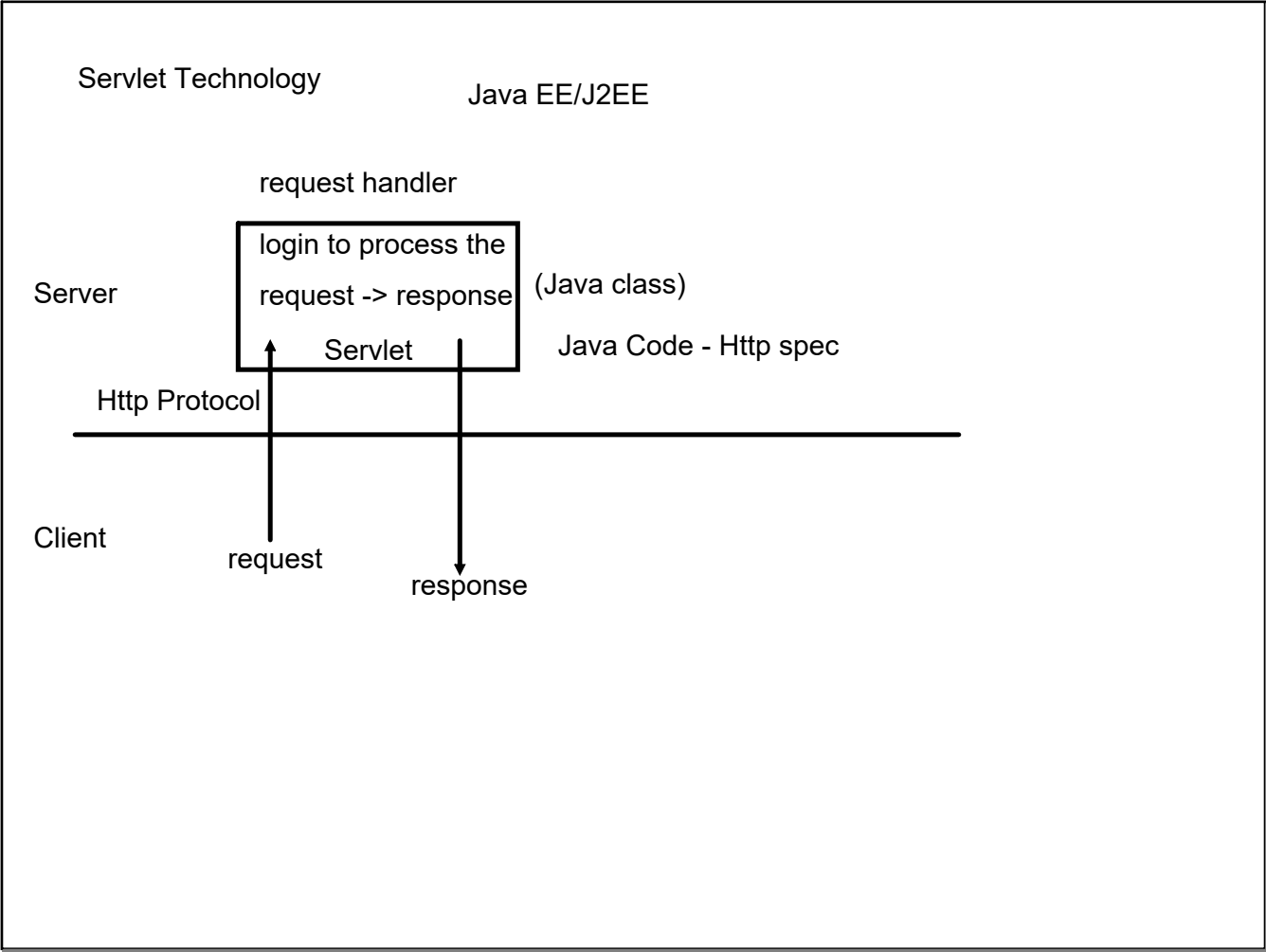
Conveyer belt



Parallel Processing not to be preferred

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







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

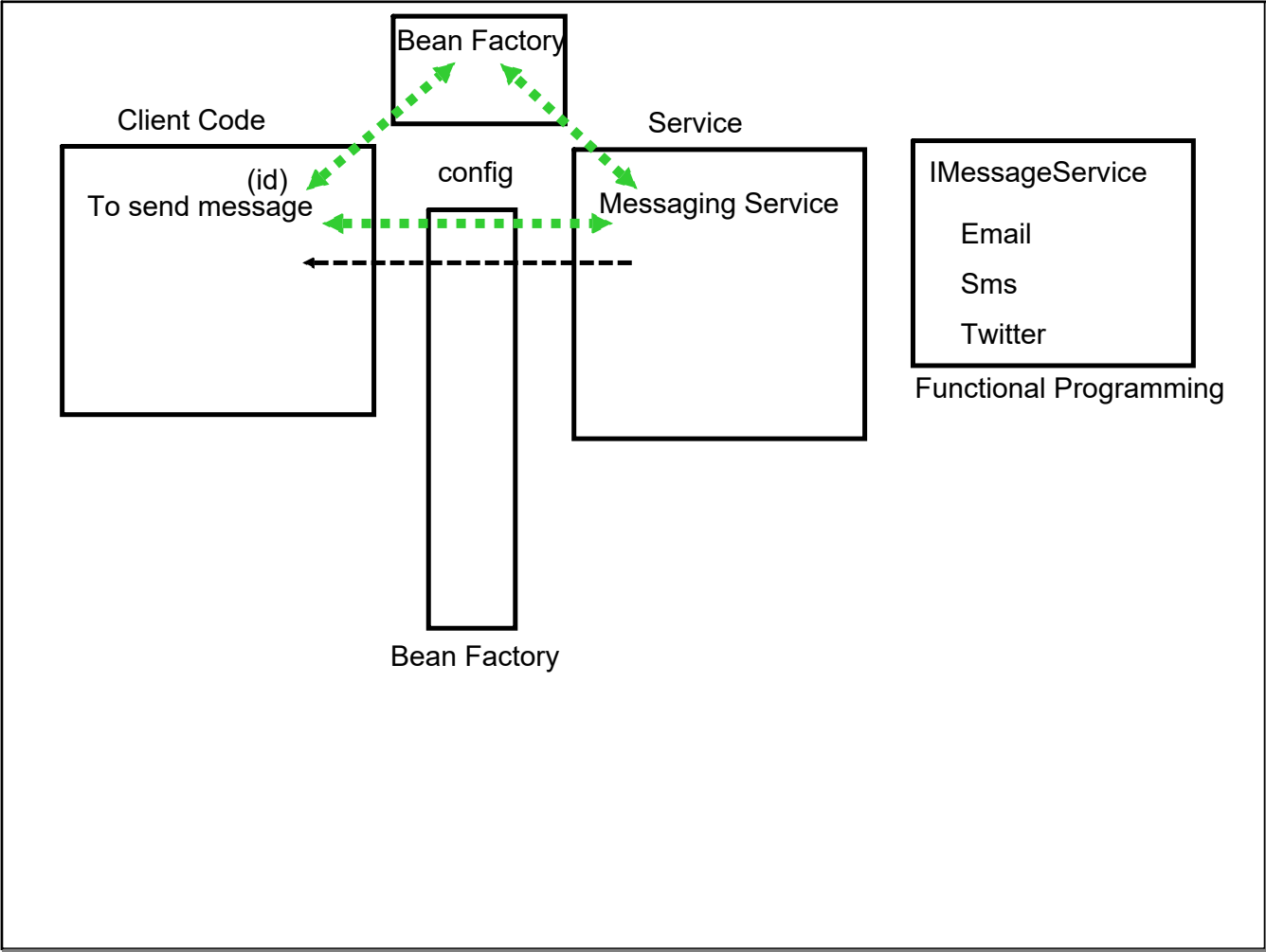
AOP

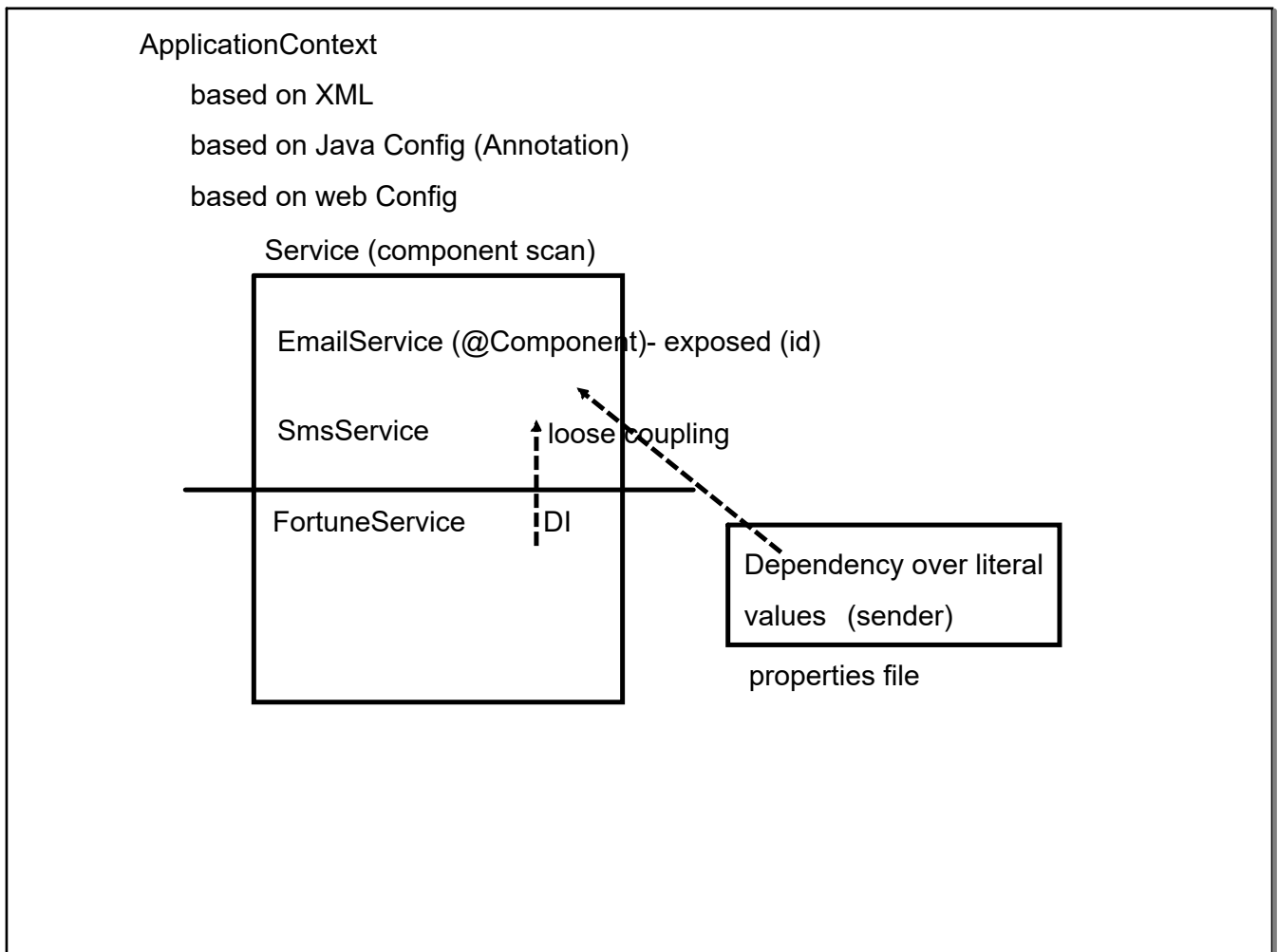
IoC : Outsourcing the creation and management of object

Bean : Java Object managed by container

AOP : Aspect Oriented Programming (Proxy)

Clean, Loosly Coupled, reusable JAva Code



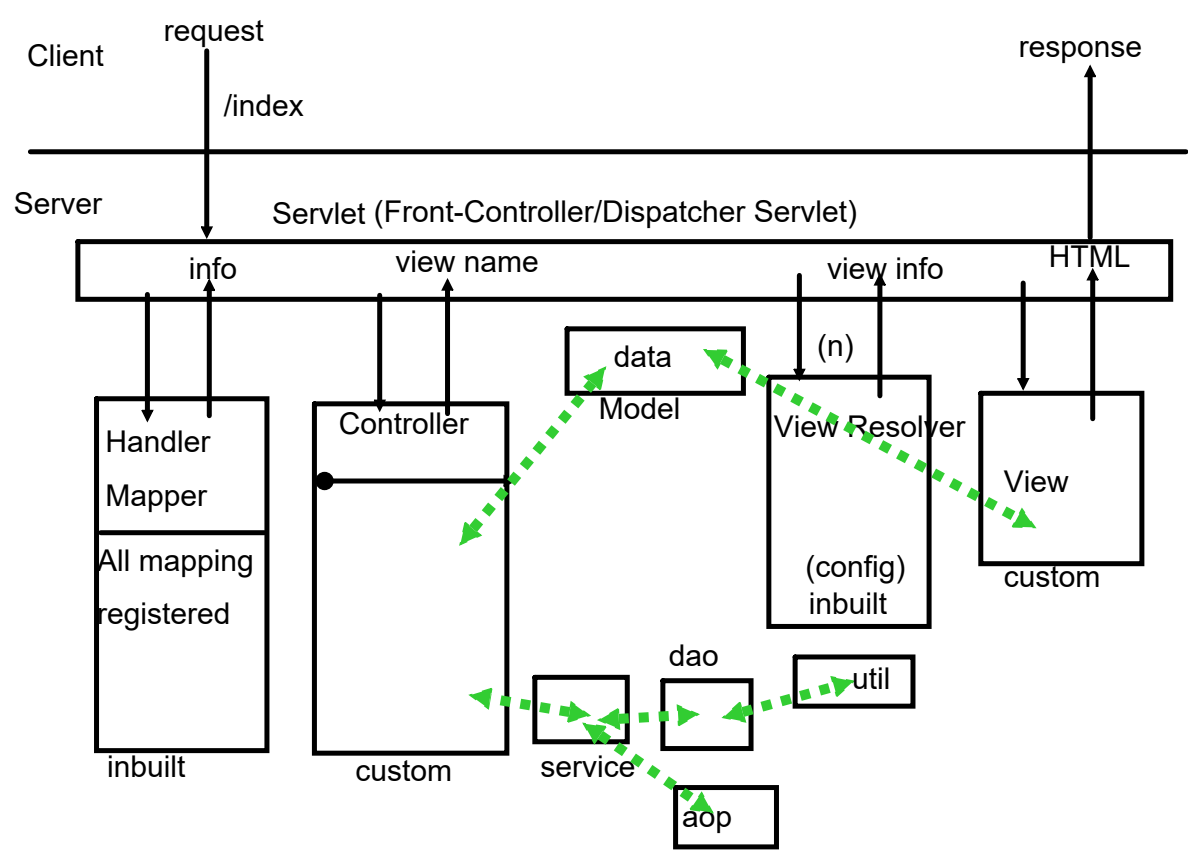


Scope : Singleton (Default)  
Prototype

request : single request-response cycle  
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



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):

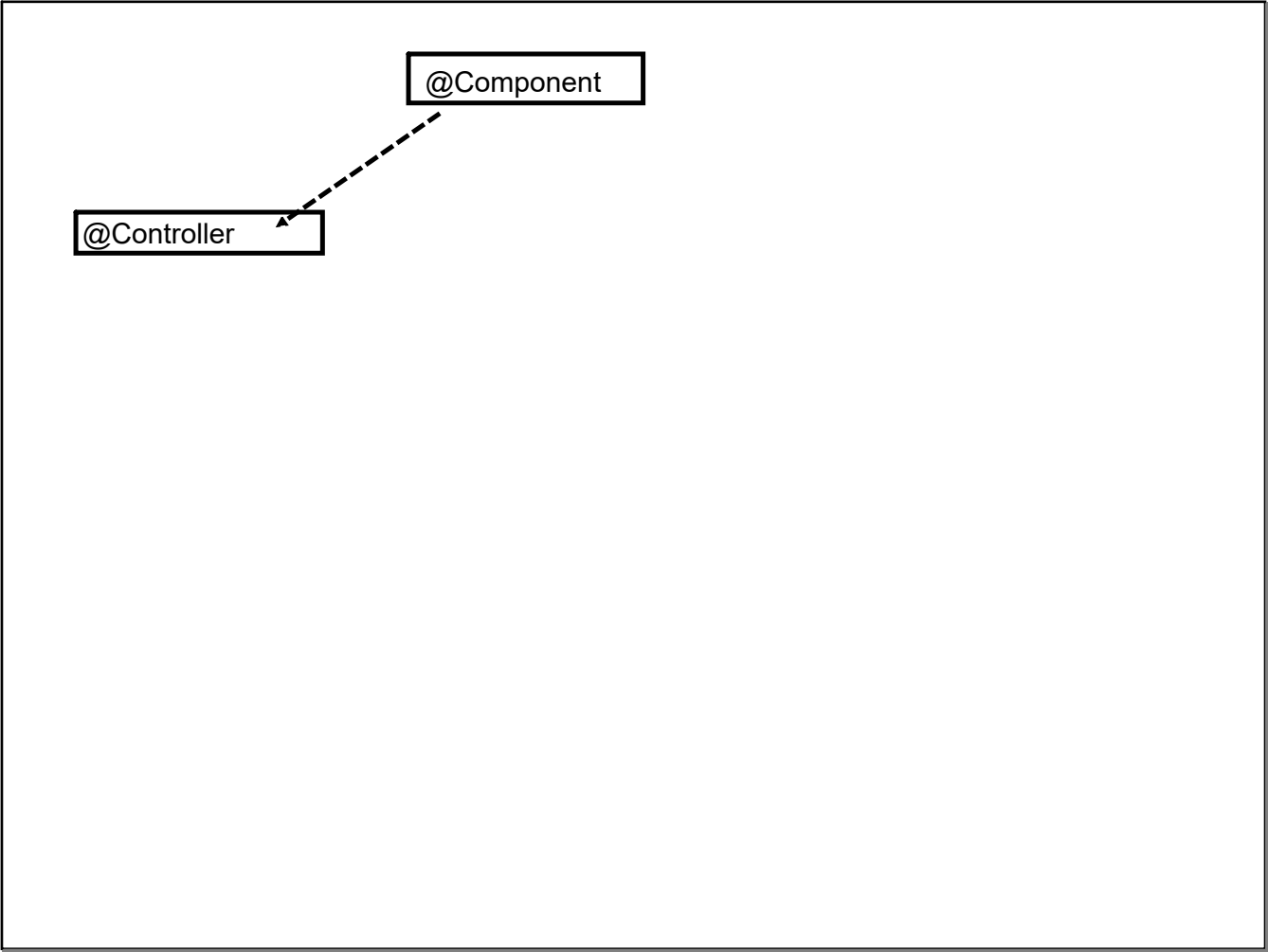
inbuilt servlet : register that servlet (DispatcherServlet : config)

config code: as per more requirement

controller code ( multiple support layers)

model : data structure

views : presentation purpose



**View Resolver:**

Single View File (jsp)

Modular View File (Tiles)

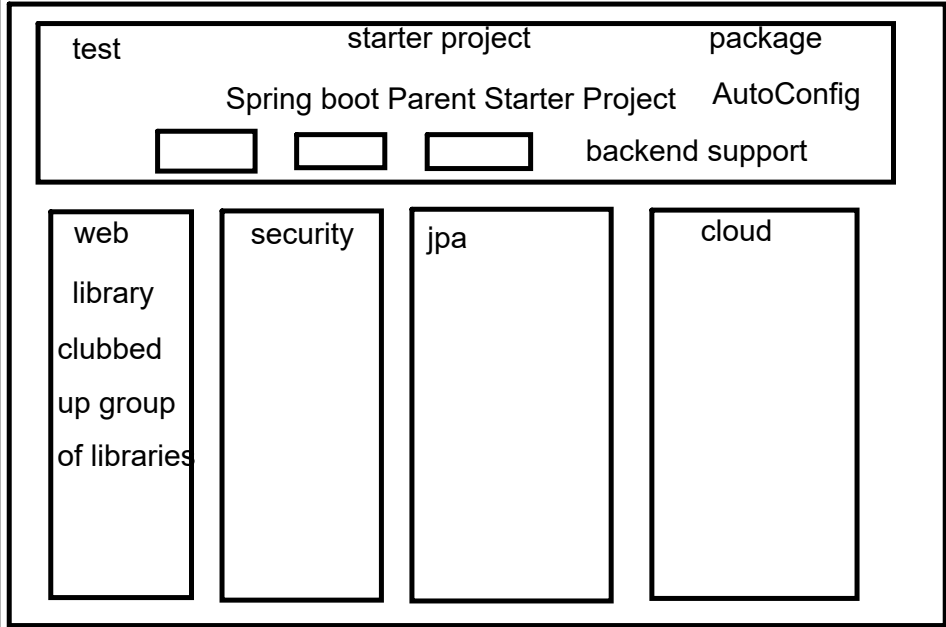
Multipart response (as downloadable file)

=>What type of responses you want

=>What type of responses your view templates

Spring Boot:

1. Dependency management 2.2



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

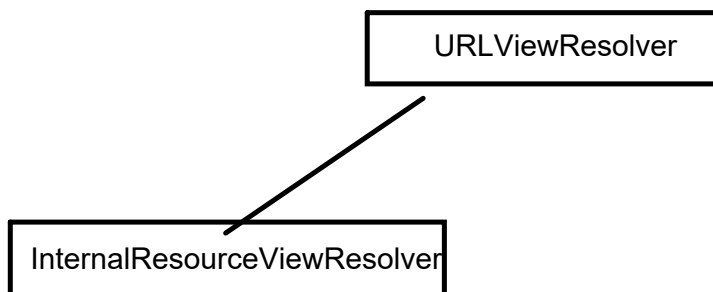
not configured to use jsp-jstl view templates

Spring boot is by default configured to use :

thymeleaf

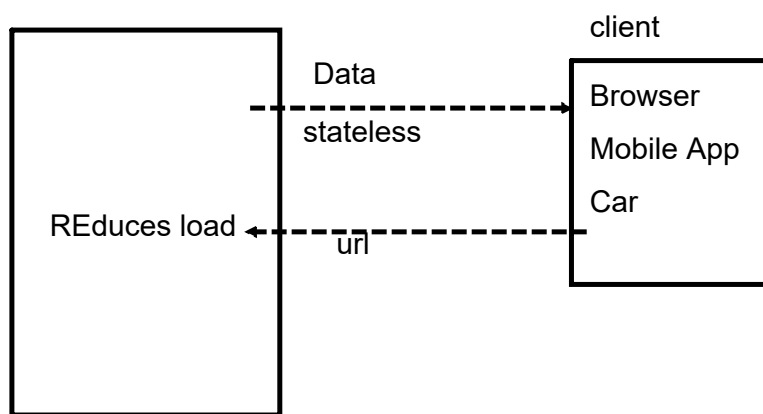
```
> java -jar bootapp.jar server.port=9091
```

InternalResourceViewResolver (



## REST-WS

REpresentational State Transfer



1. how to generate a request ?
2. what is format in which data will arrive?



Request, REST :

=>purely the URL

=>Conventions for URL

=>ALL HTTP VERBS (intention)

eg :POST : add some

PUT : edit

SOAP / WSDL : programmatic request

REceiving data:

standard, simple as possible

JSON,XML,HTML,TEXT

Allowed to explore the concept of micro-service architecture

JAX-RS (specification)

Jersey

Restlet

RESTEasy

Apache CXF

Spring :

not a JAX-RS implementation

### @RestController

1. does auto : interconversion of JSON<->JAVA (jackson-databind project)
2. DEALS with Request /Response

Request Object



Response Object

header
content
status code

Jackson - databind project :

uses the getter/setter method for interconversion

lombok project

Convention:

Employee

/api/employees	GET	: asking for all employee records (/api/get-all-records)
/api/employees/{id}	GET	: asking for a single emp record with id : {id}
/api/employees	POST	: a record is submitted (add)
/api/employees	PUT	: a record is submitted (update)
/api/employees/{id}	DELETE	: delete a record with id : {id}
/api/employess	DELETE	: a array of id is submitted
/api/employees/{id1}/{id2}	DELETE	

spring-data-rest

## Actuators

### Microservice architecture

monolith :

Interdependency

Fragile in nature

deployment :

usage of resources

bound to specific technology

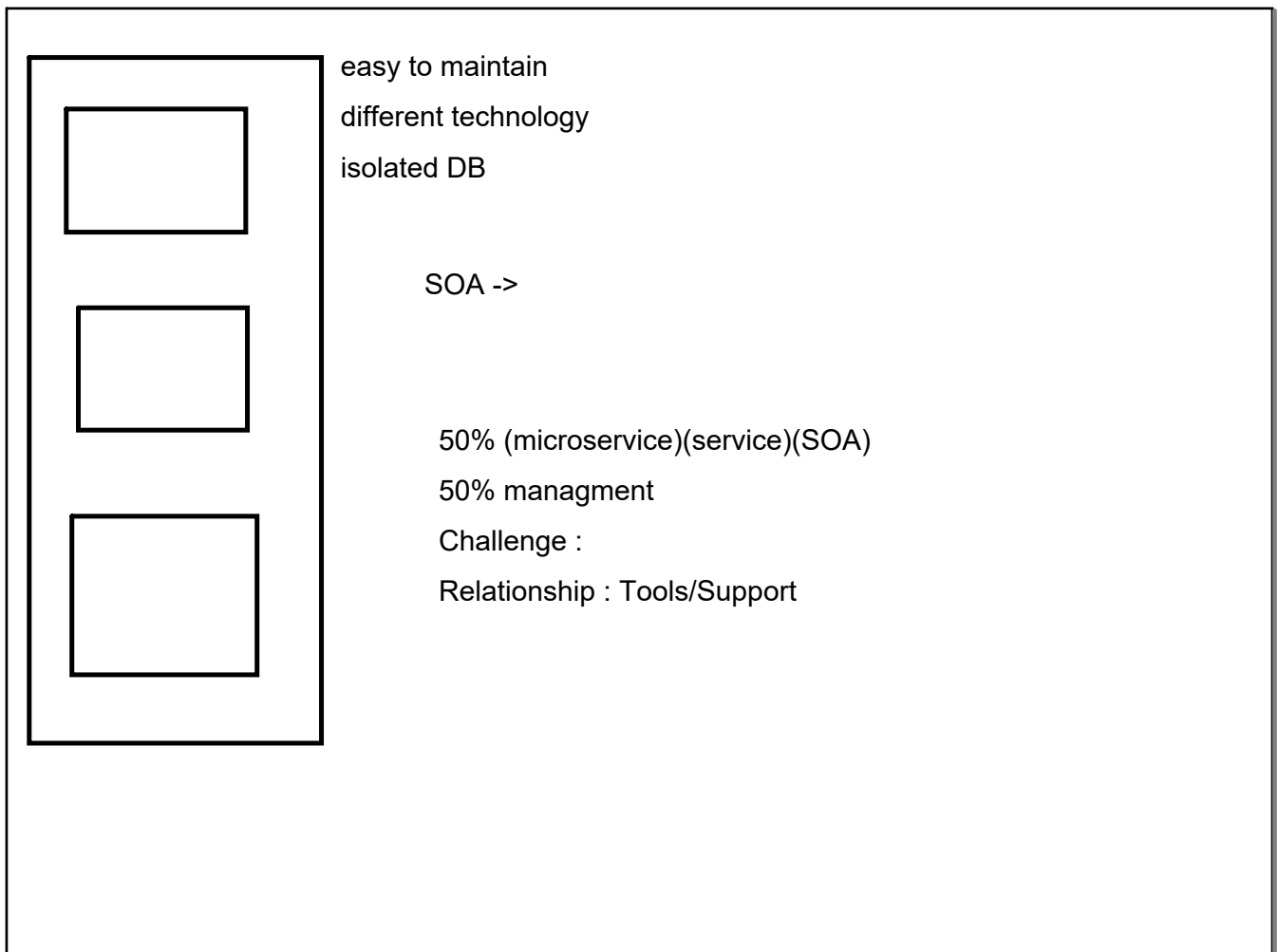
team division / management

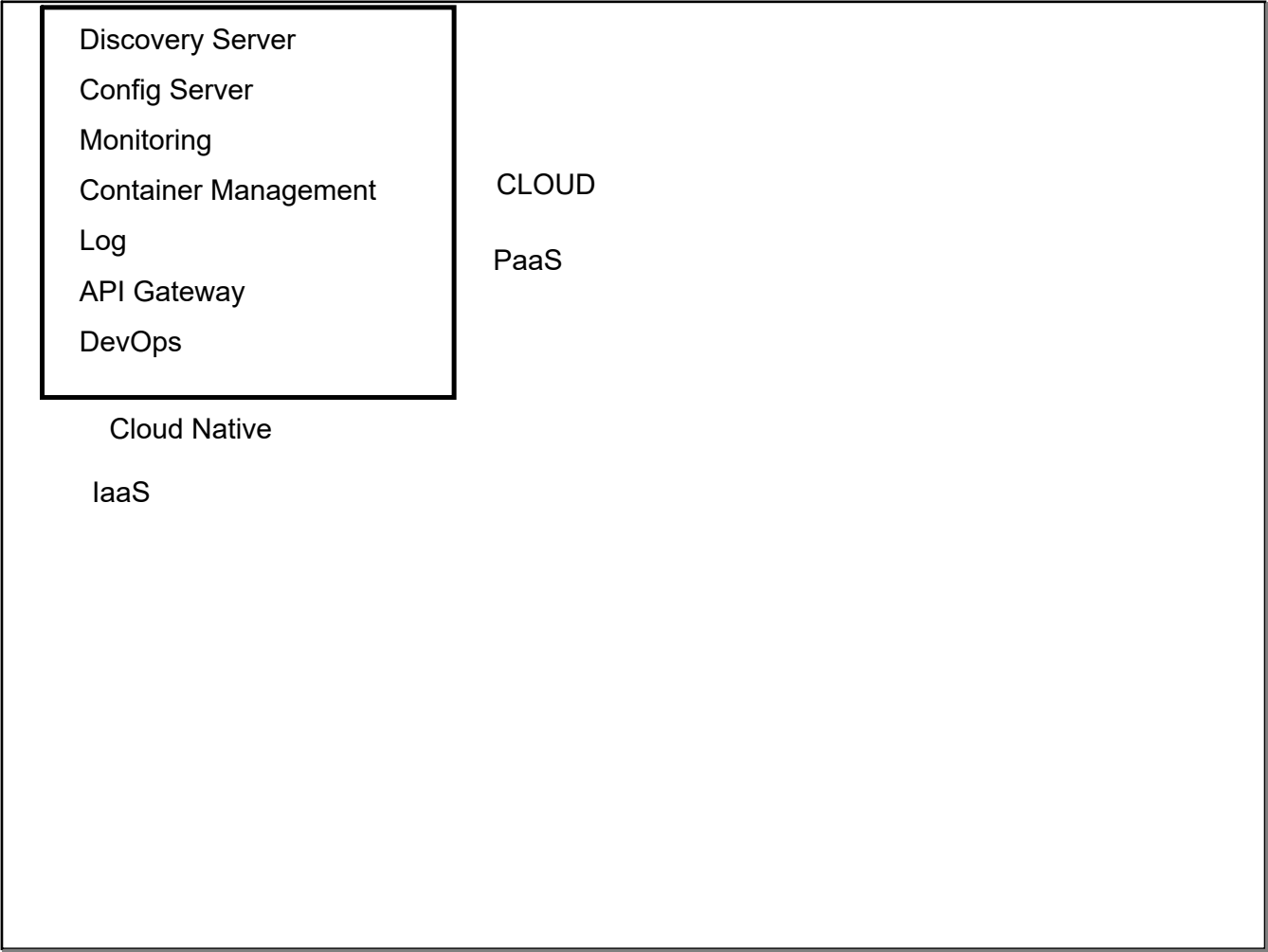
new team member inclusion

---

1. does not easily integrate/comply agile

2. CI/CD implementation is a challenge





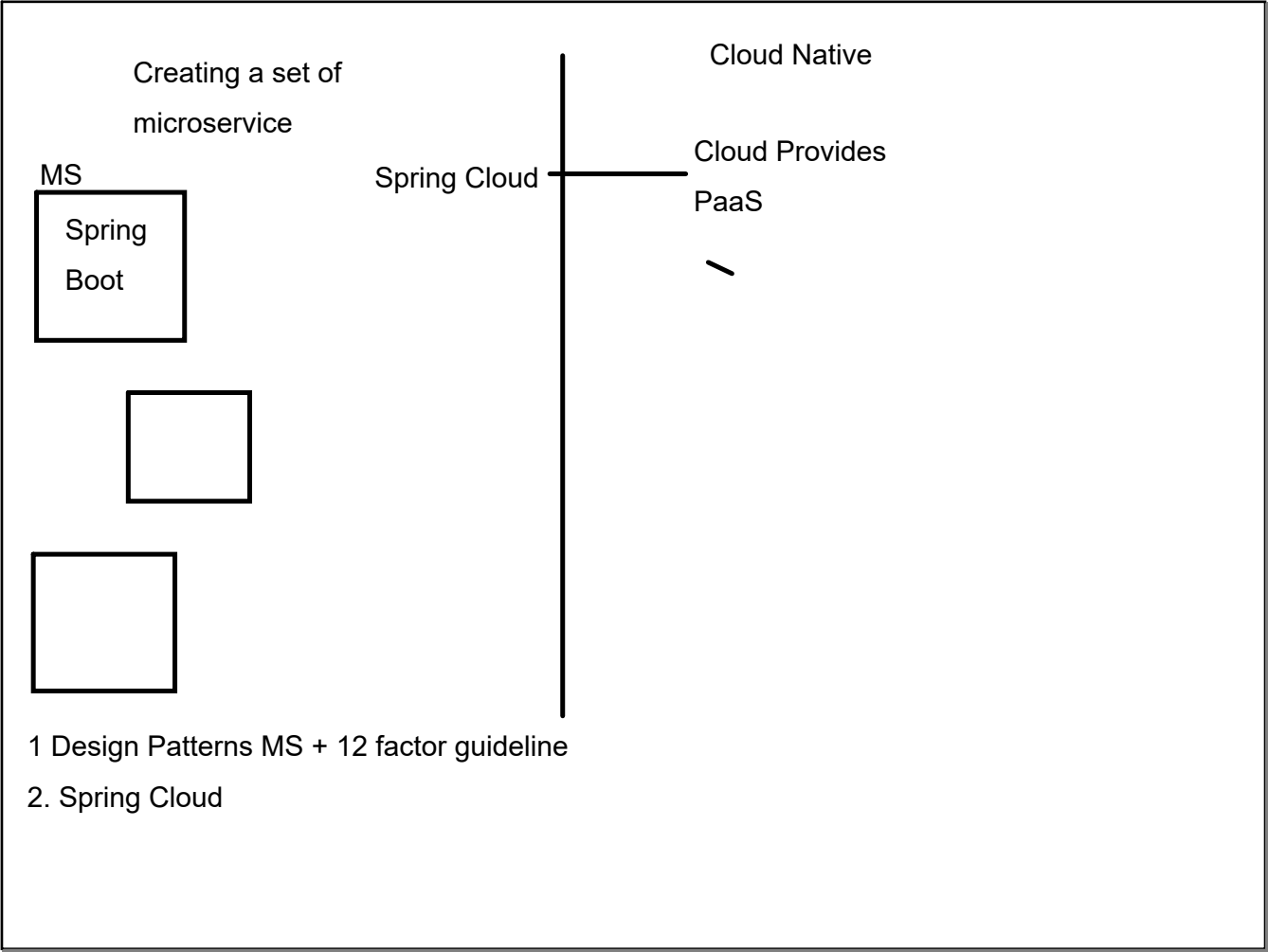
Spring boot :

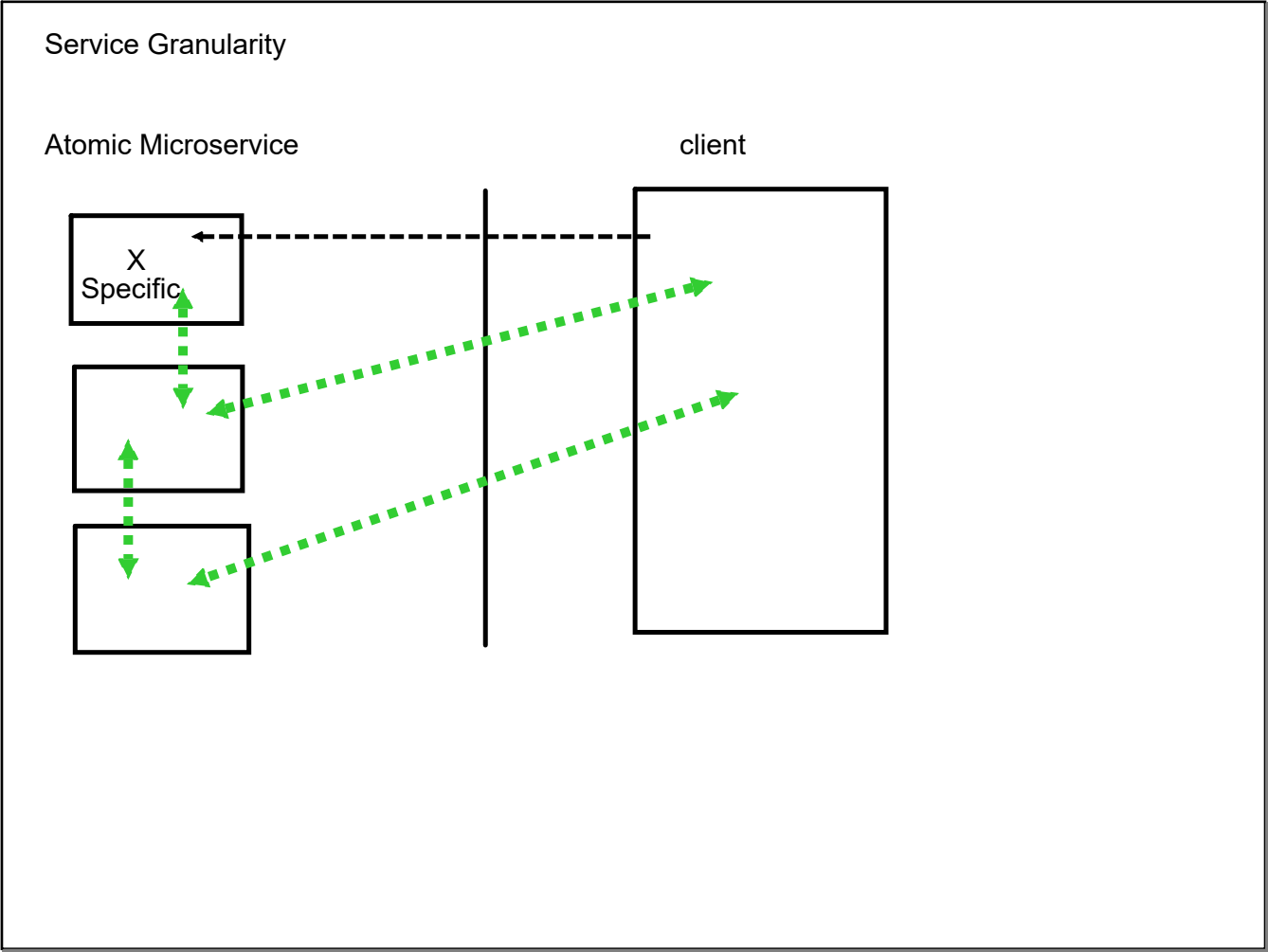
Spring / Spring MVC

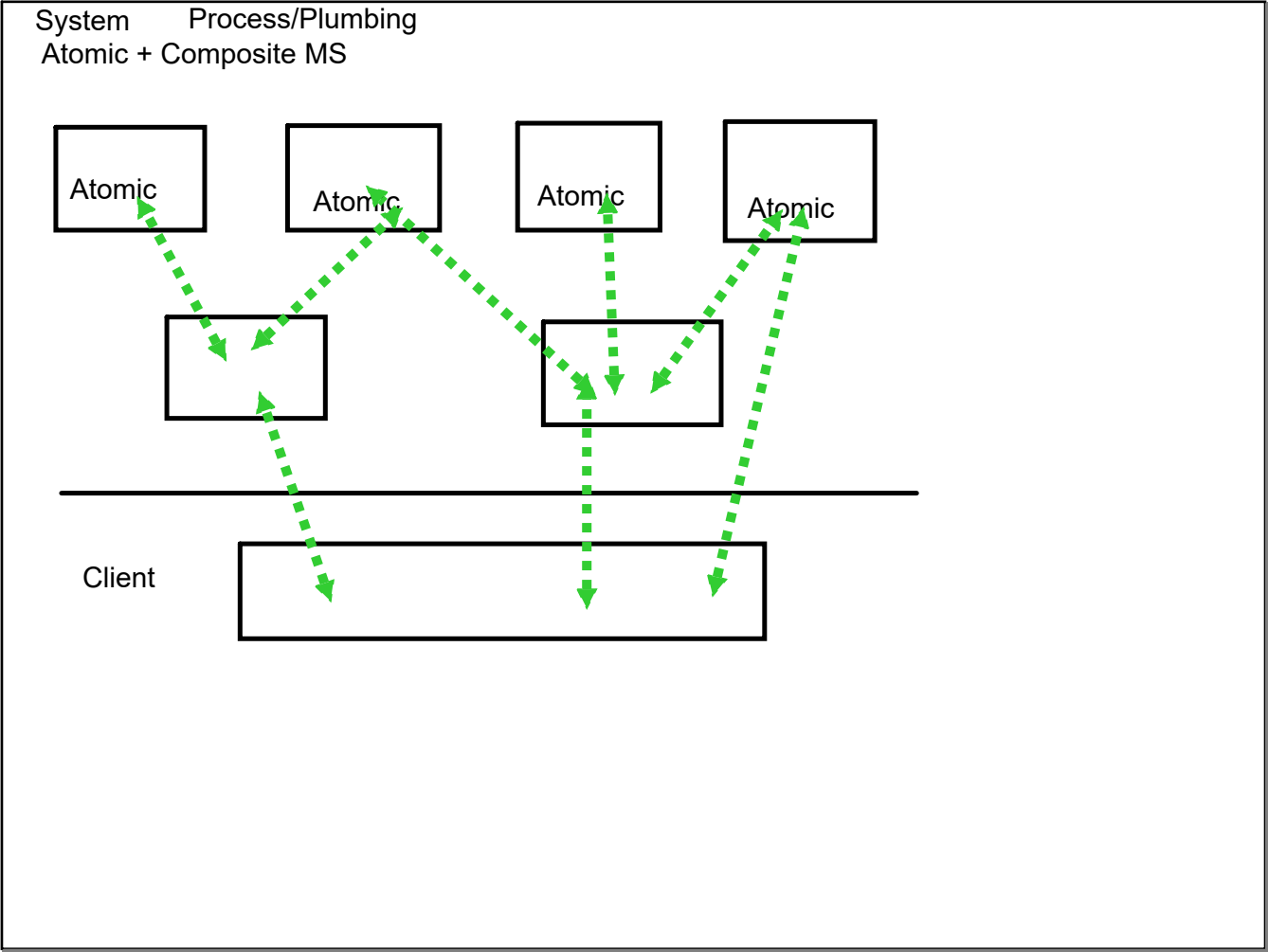
Most resonable default

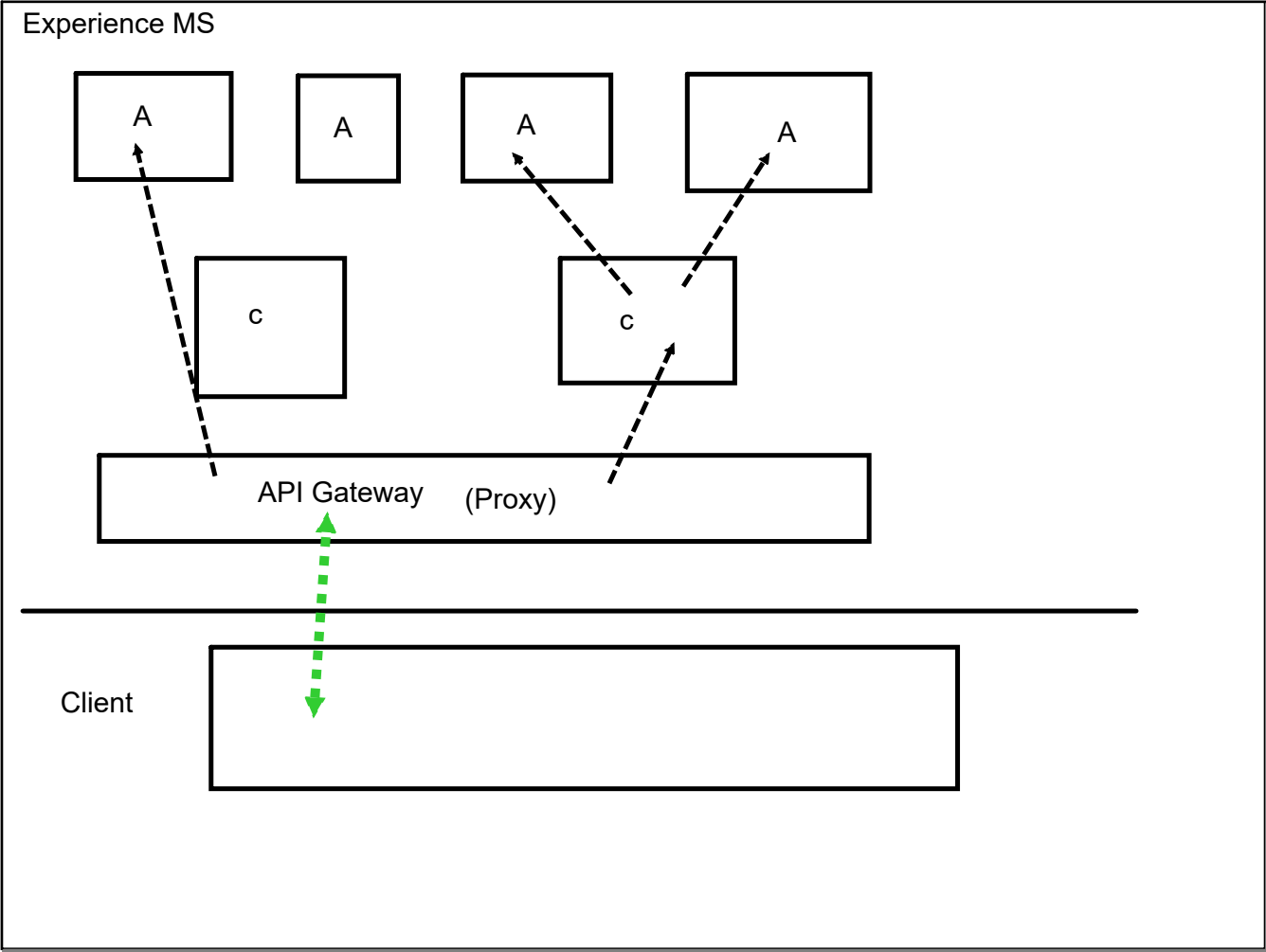
Integrates Spring cloud out of the box

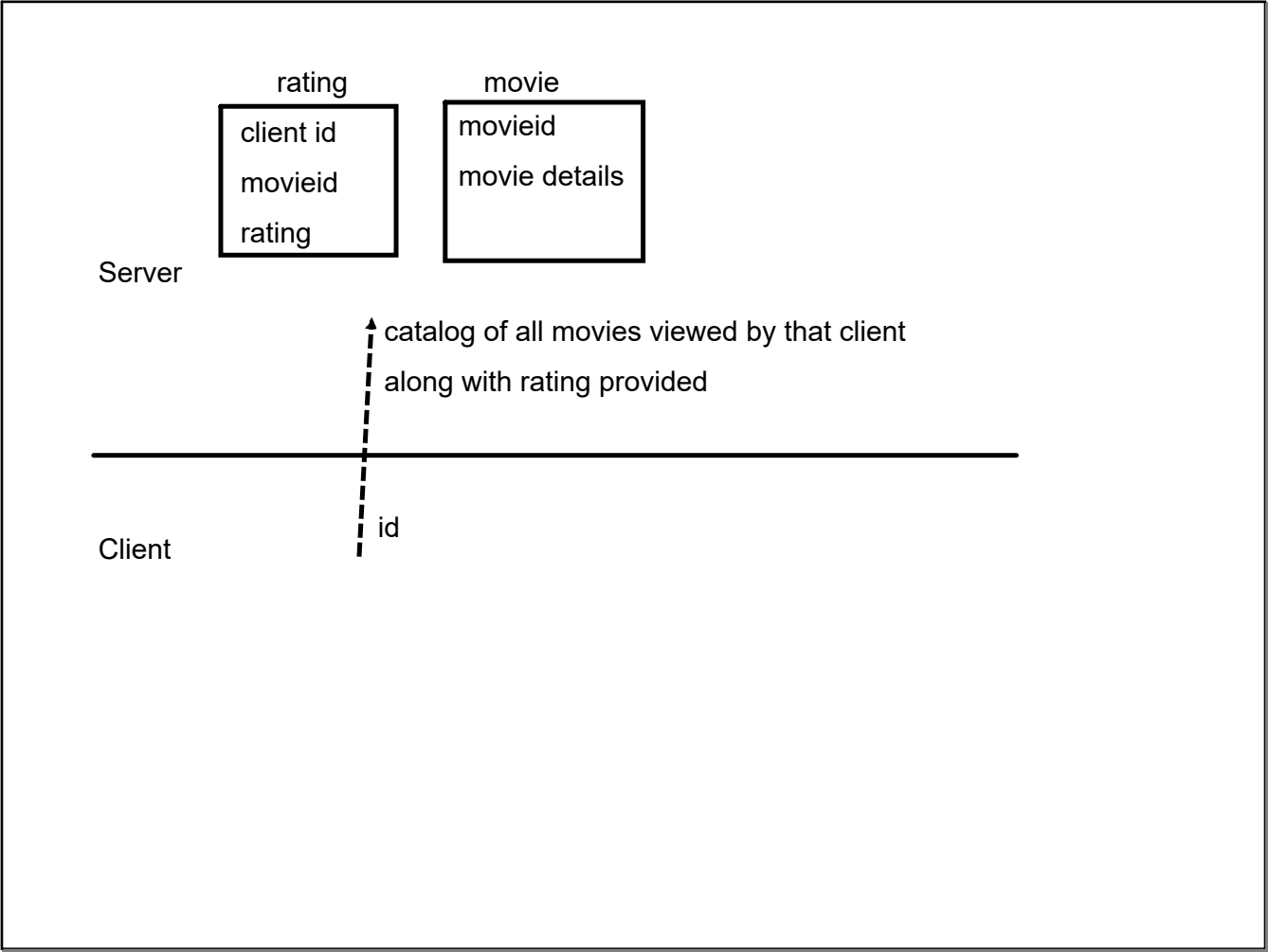


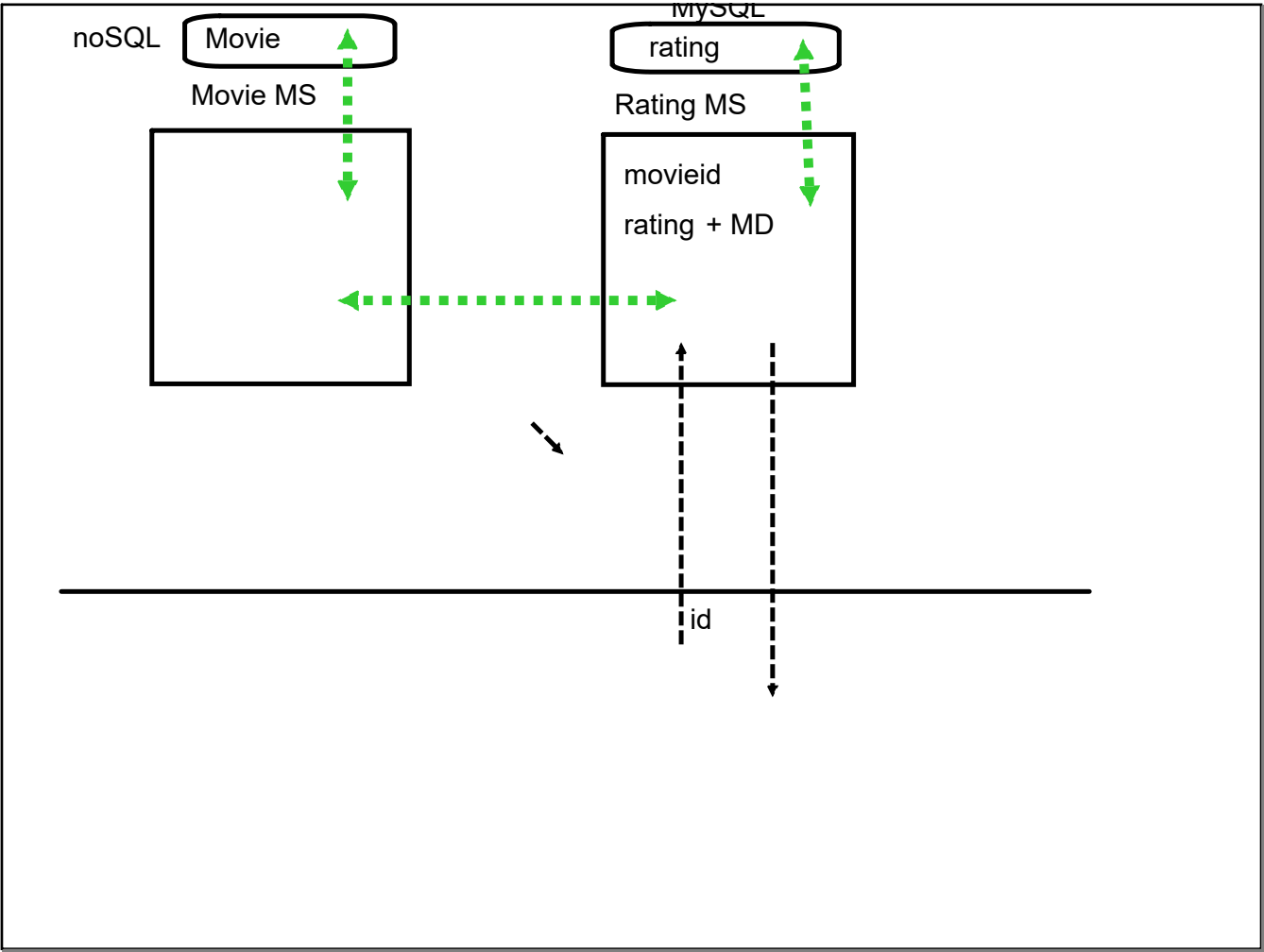


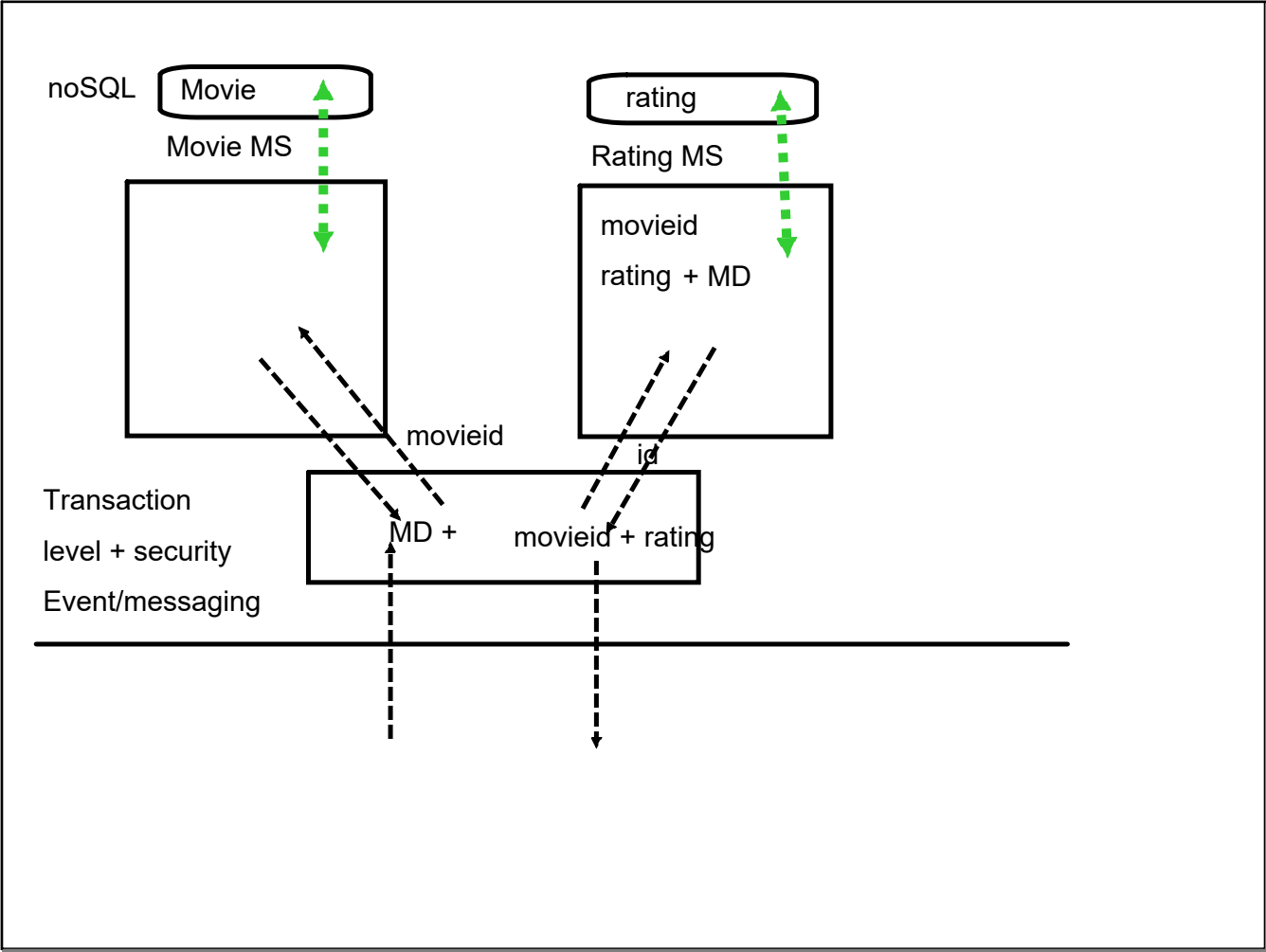












12-factor guideline

Lightweight

Reactive

Stateless

Atomic

Externalized

Consistent

Resilient

Good Citizens

Versioned     x.x.x



