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

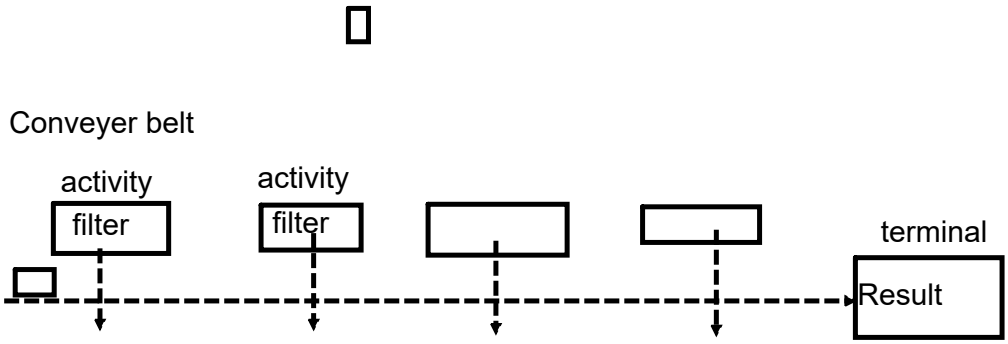
IntConsumer()

Predicate:

BiPredicate, Primitive type implementation

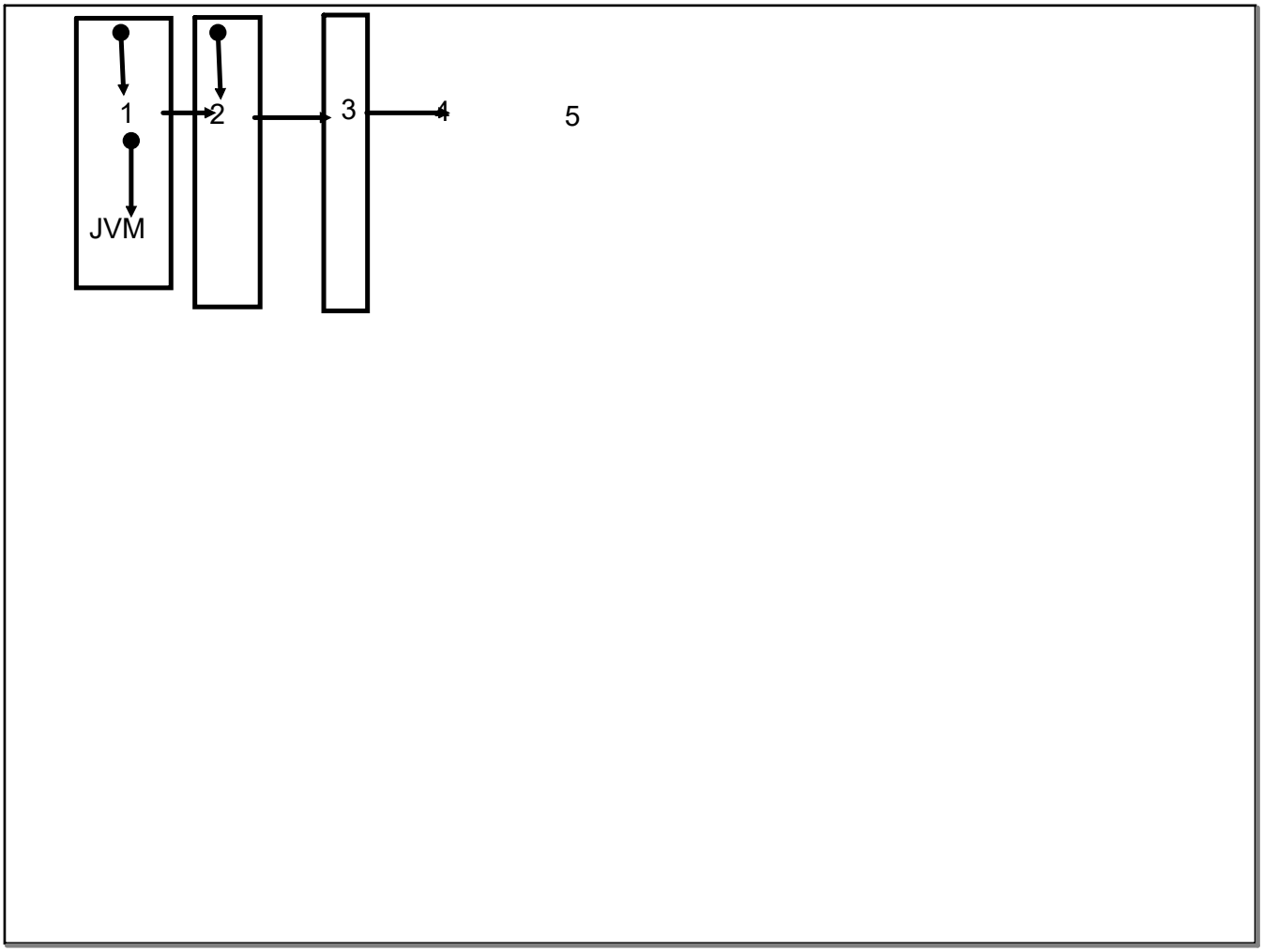
Function : BiFunction

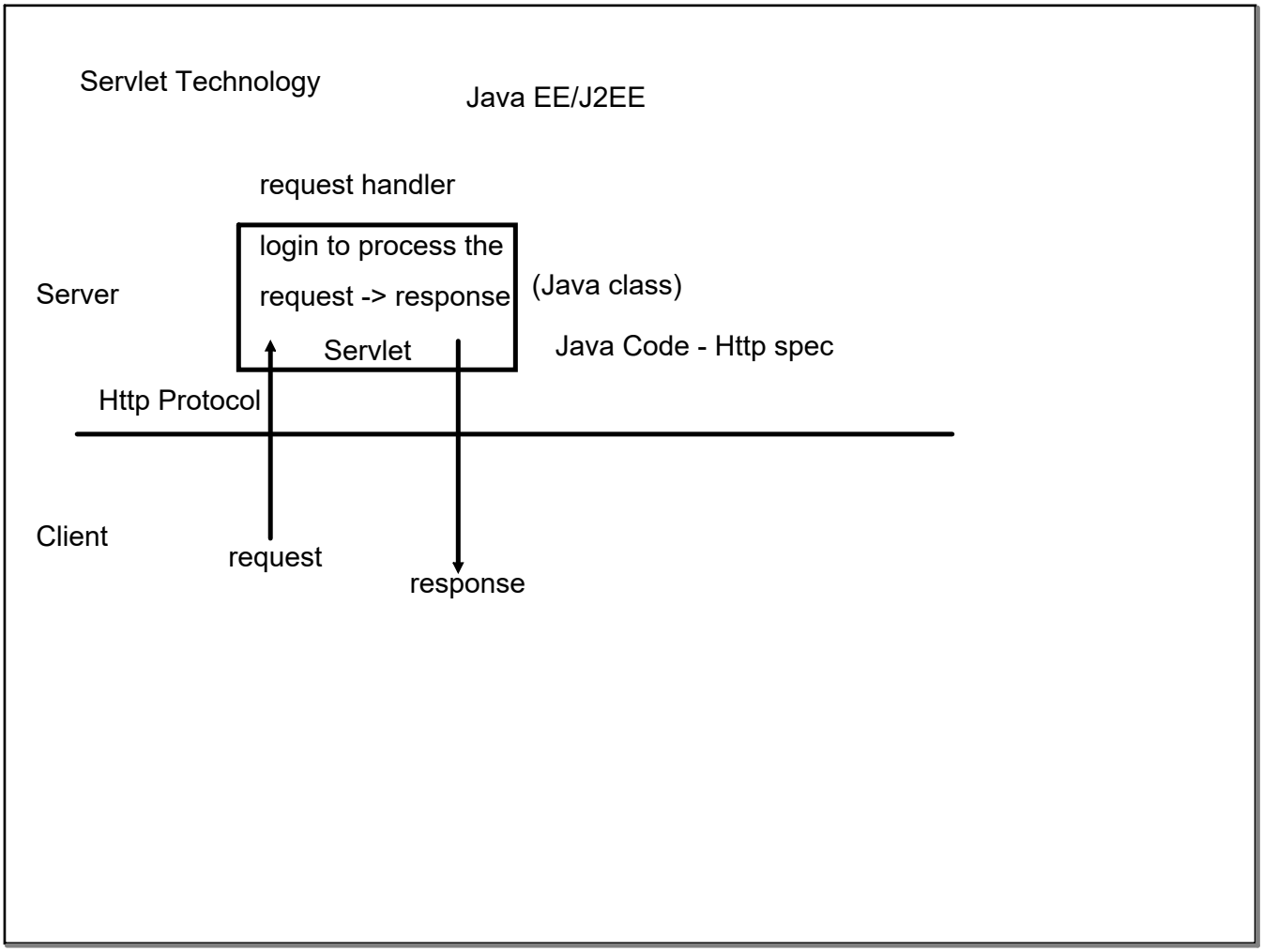
Functional programming remove overhead of creating objects and loading class files



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){

  }
```

Named Core Datatypes of TS

number	-1/5.3/200 5~5.0
string	'Hello',"Hello",`Hello`
boolean	true/false

Spring

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

Spring Framework : Servlet technology

CORE

Bean Factory

IoC

DI

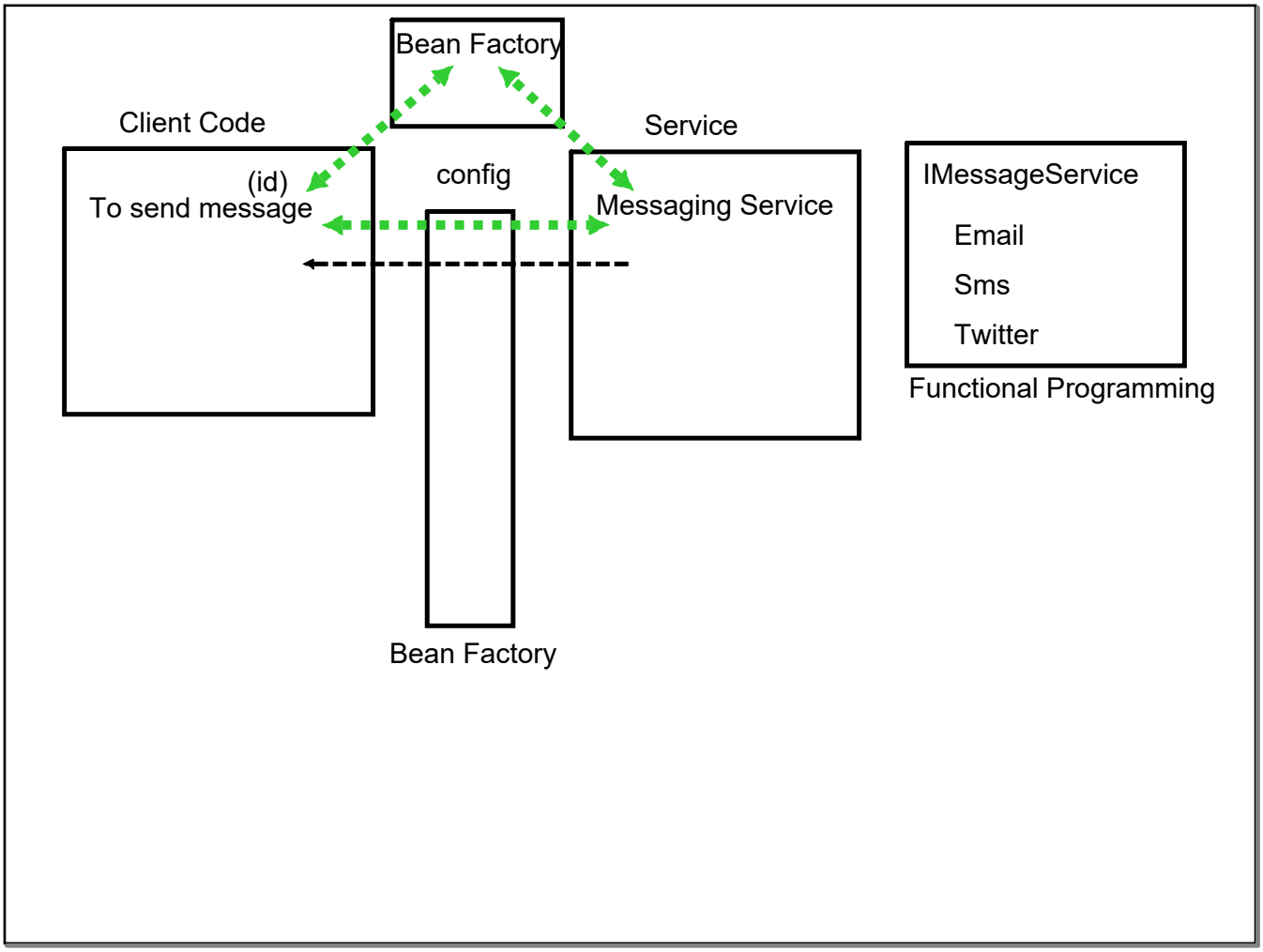
AOP

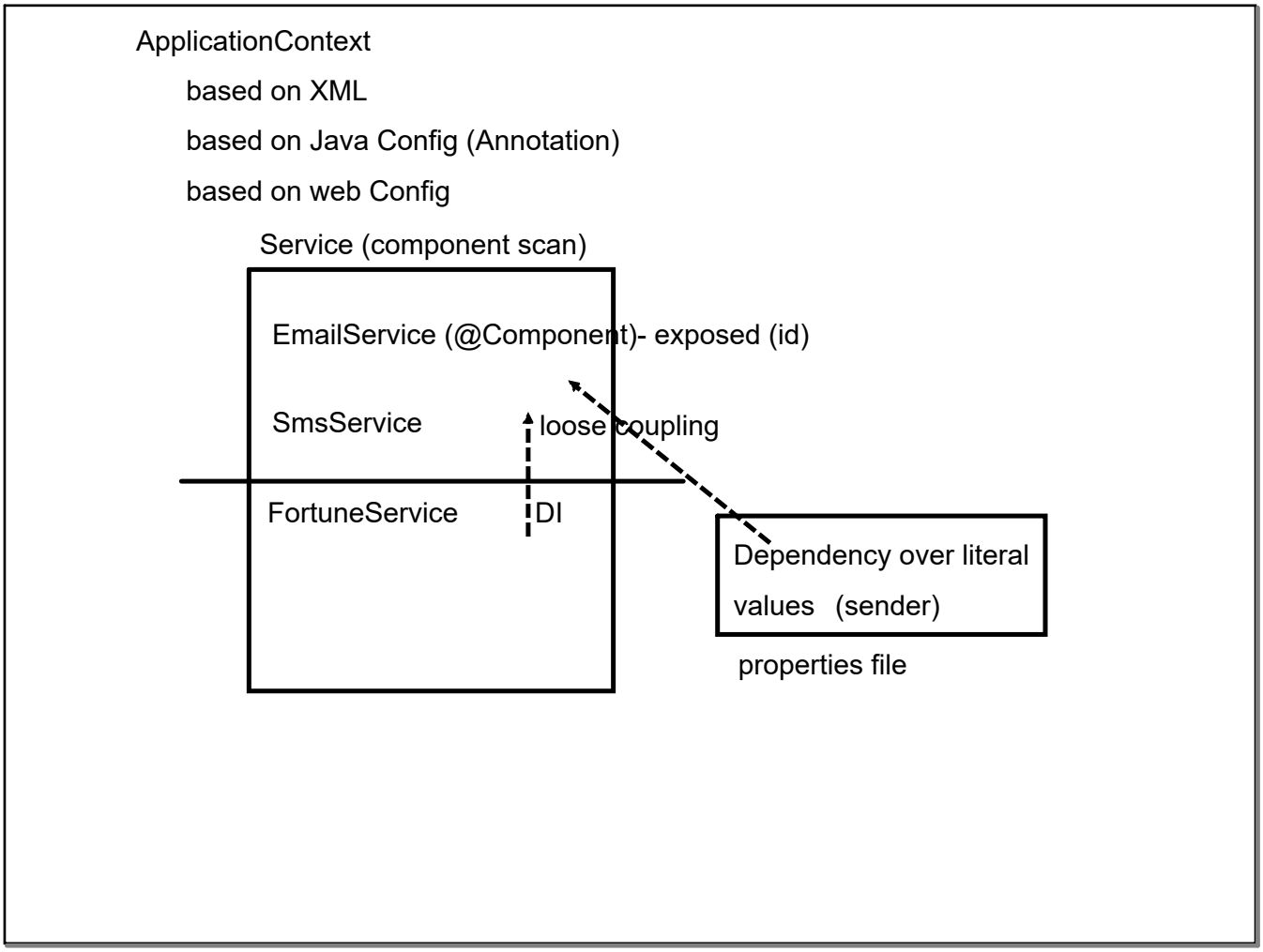
IoC : Outsourcing the creation and management of object

Bean : Java Object managed by container

AOP : Aspect Oriented Programming (Proxy)

Clean, Loosly Coupled, reusable JAva Code

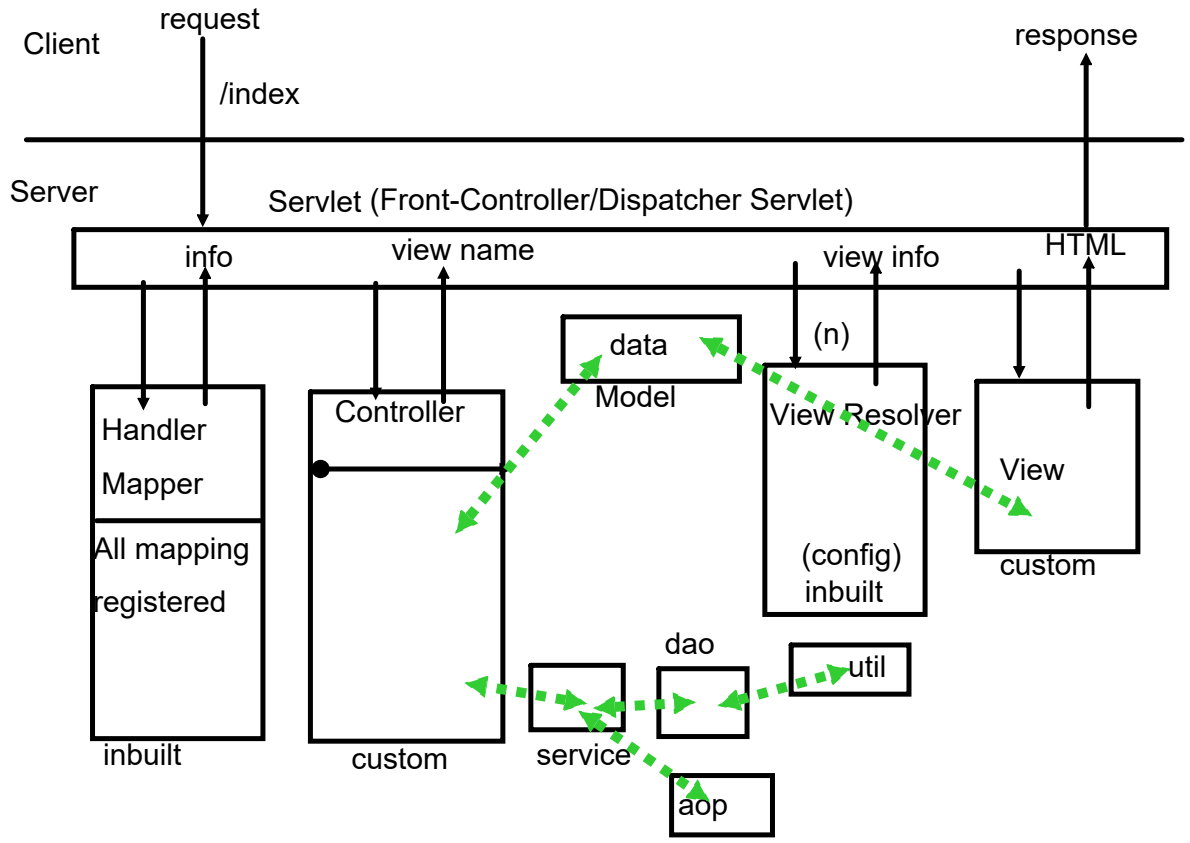

Bean Factory
Client Code
Service
(id)
To send message
config
Messaging Service
IMessageService
Email
Sms
Twitter
Functional Programming
Bean Factory


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) 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):

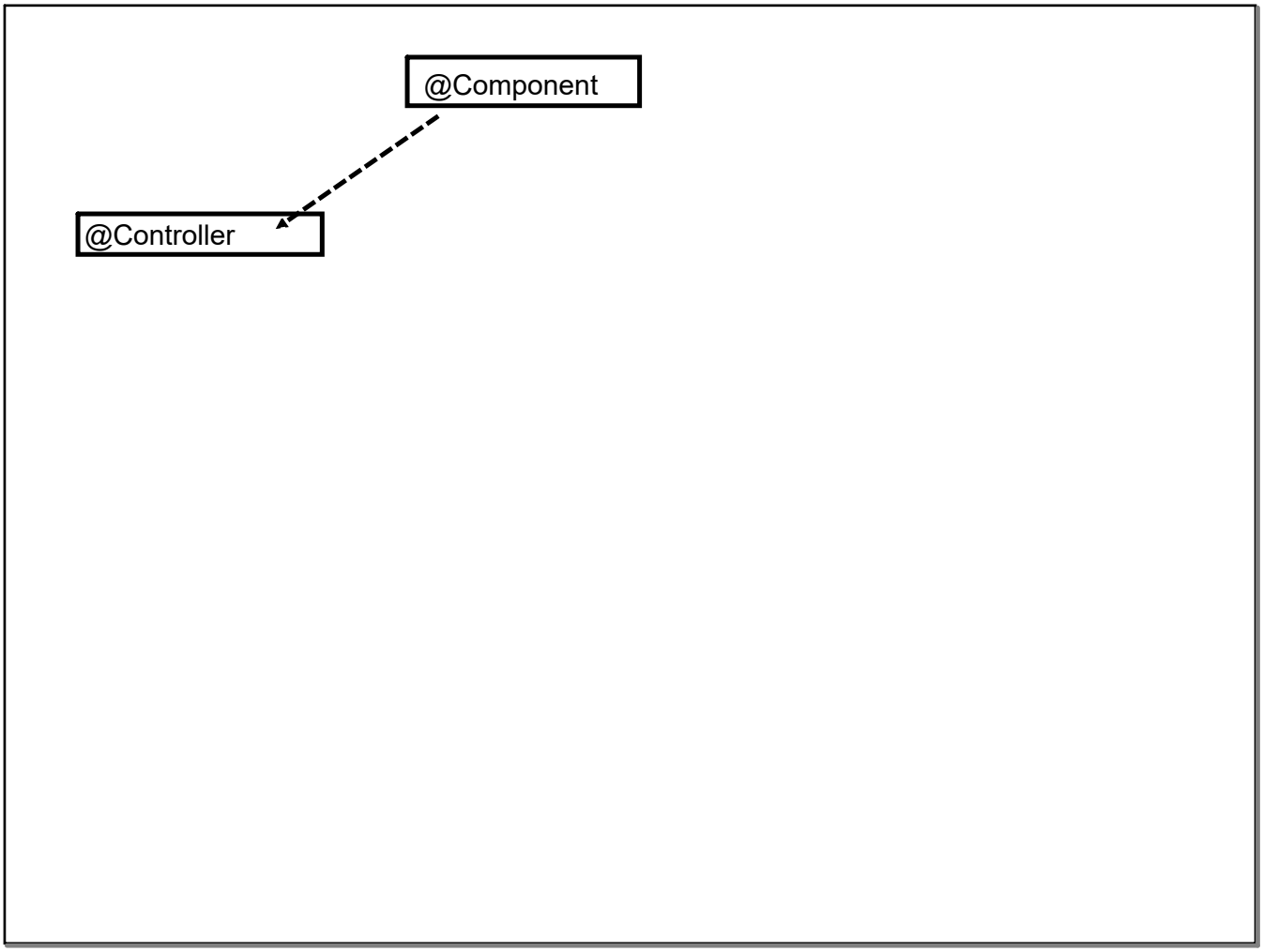
inbuilt servlet : register that servlet (DispatcherServlet : config)

config code: as per more requirement

controller code (multiple support layers)

model : data structure

views : presentation purpose


@Component
@Controller

View Resolver:

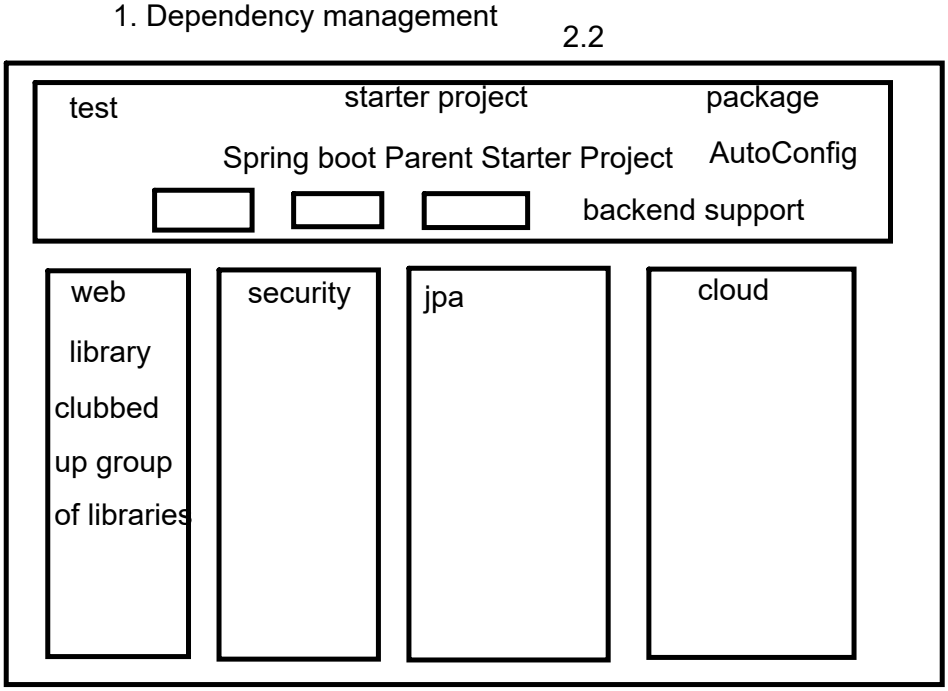
- Single View File (jsp)
- Modular View File (Tiles)
- Multipart response (as downloadable file)

=>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

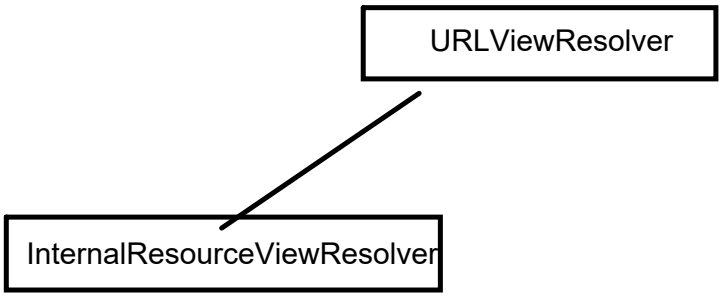
not configured to use jsp-jstl view templates

Spring boot is by default configured to use :

thymeleaf

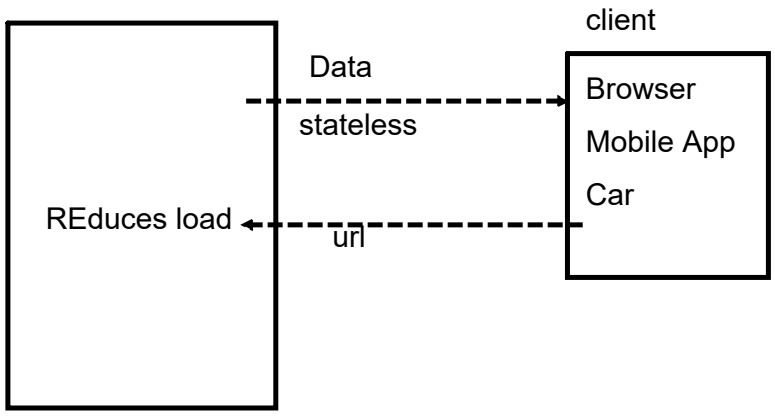
> java -jar bootapp.jar server.port=9091

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)

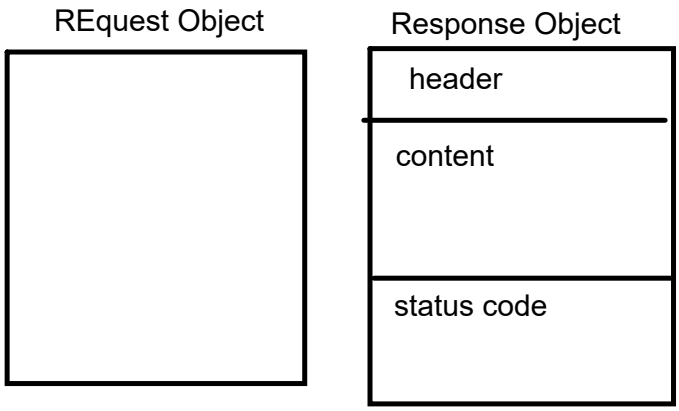
- Jersey
- Restlet
- RESTEasy
- Apache CXF

Spring :

- not a JAX-RS impelementation

@RestController

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



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

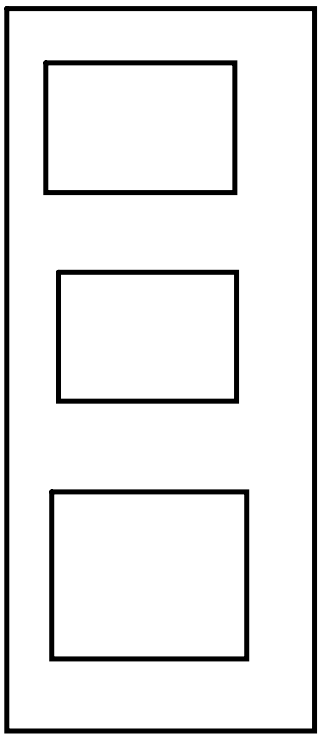
Microservice architecture

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



easy to maintain

different technology

isolated DB

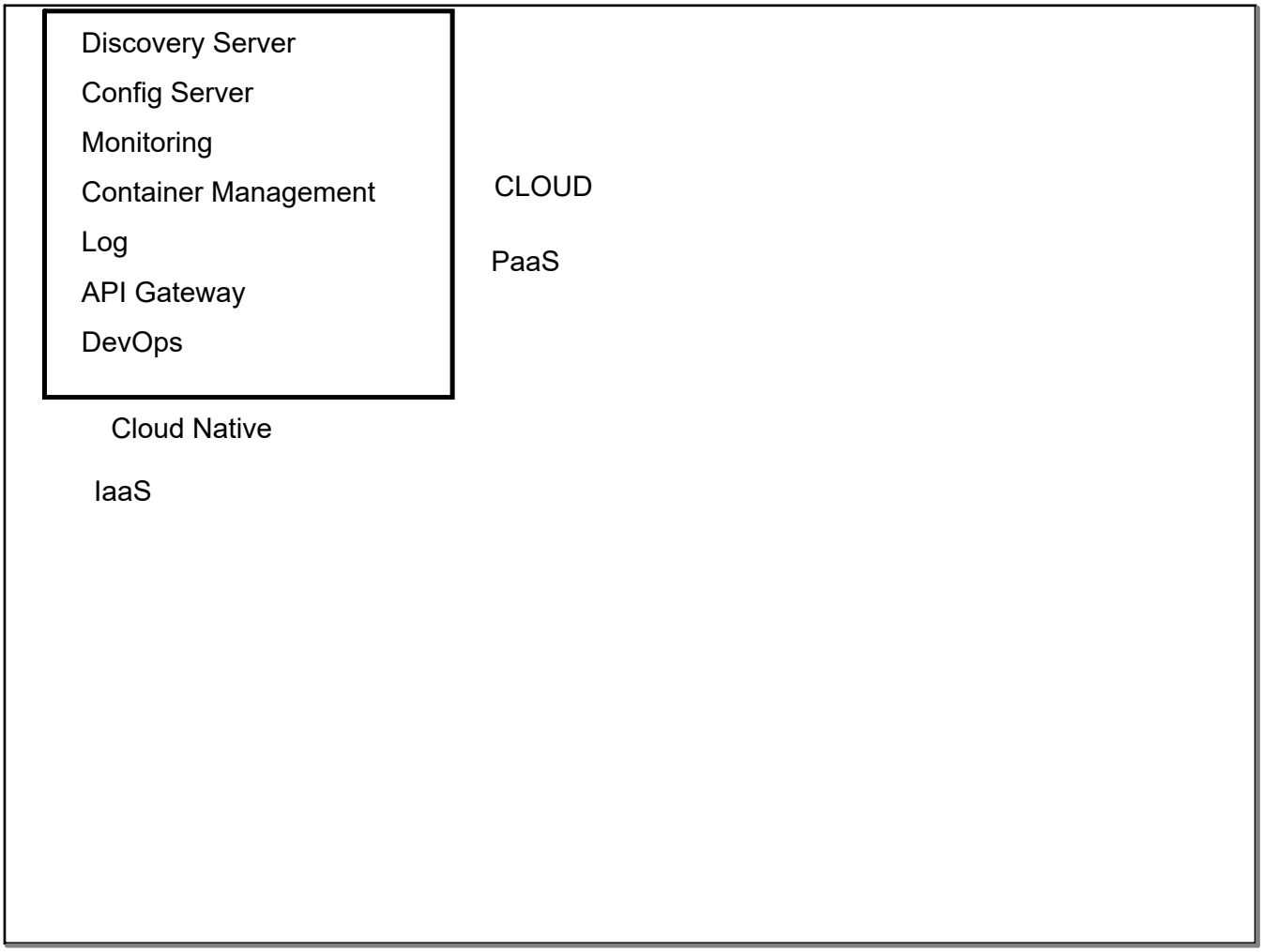
SOA ->

50% (microservice)(service)(SOA)

50% managment

Challenge :

Relationship : Tools/Support

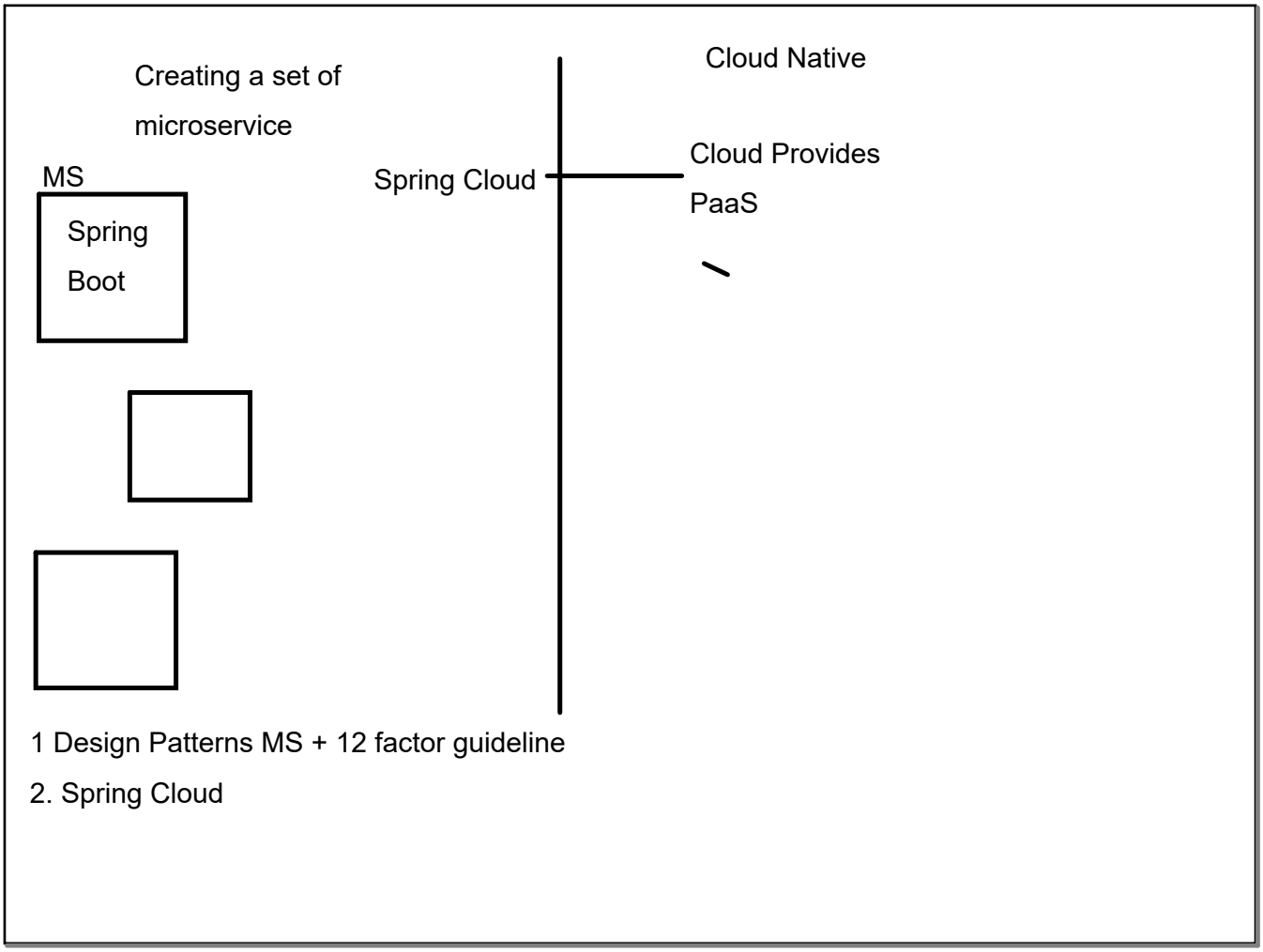

Discovery Server
Config Server
Monitoring
Container Management
Log
API Gateway
DevOps
Cloud Native
IaaS
CLOUD
PaaS

Spring boot :

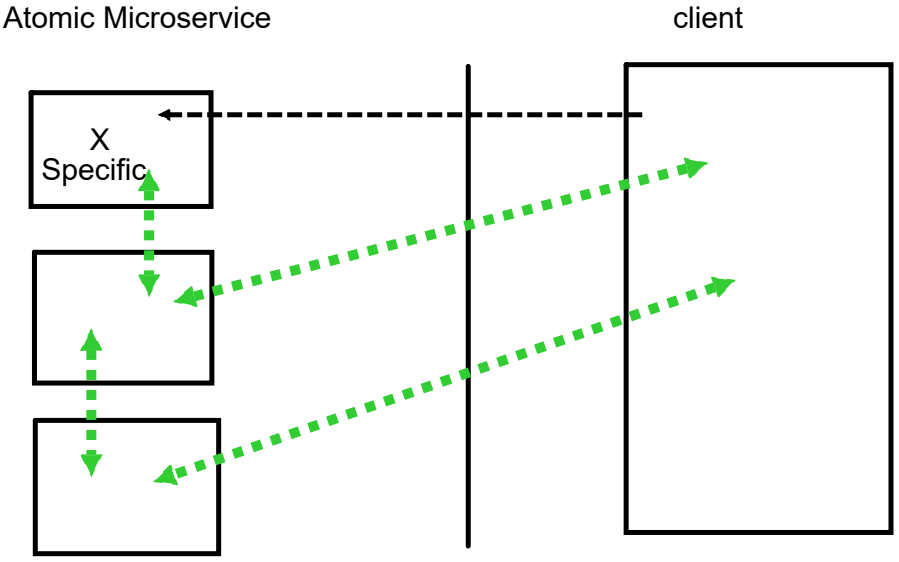
Spring / Spring MVC

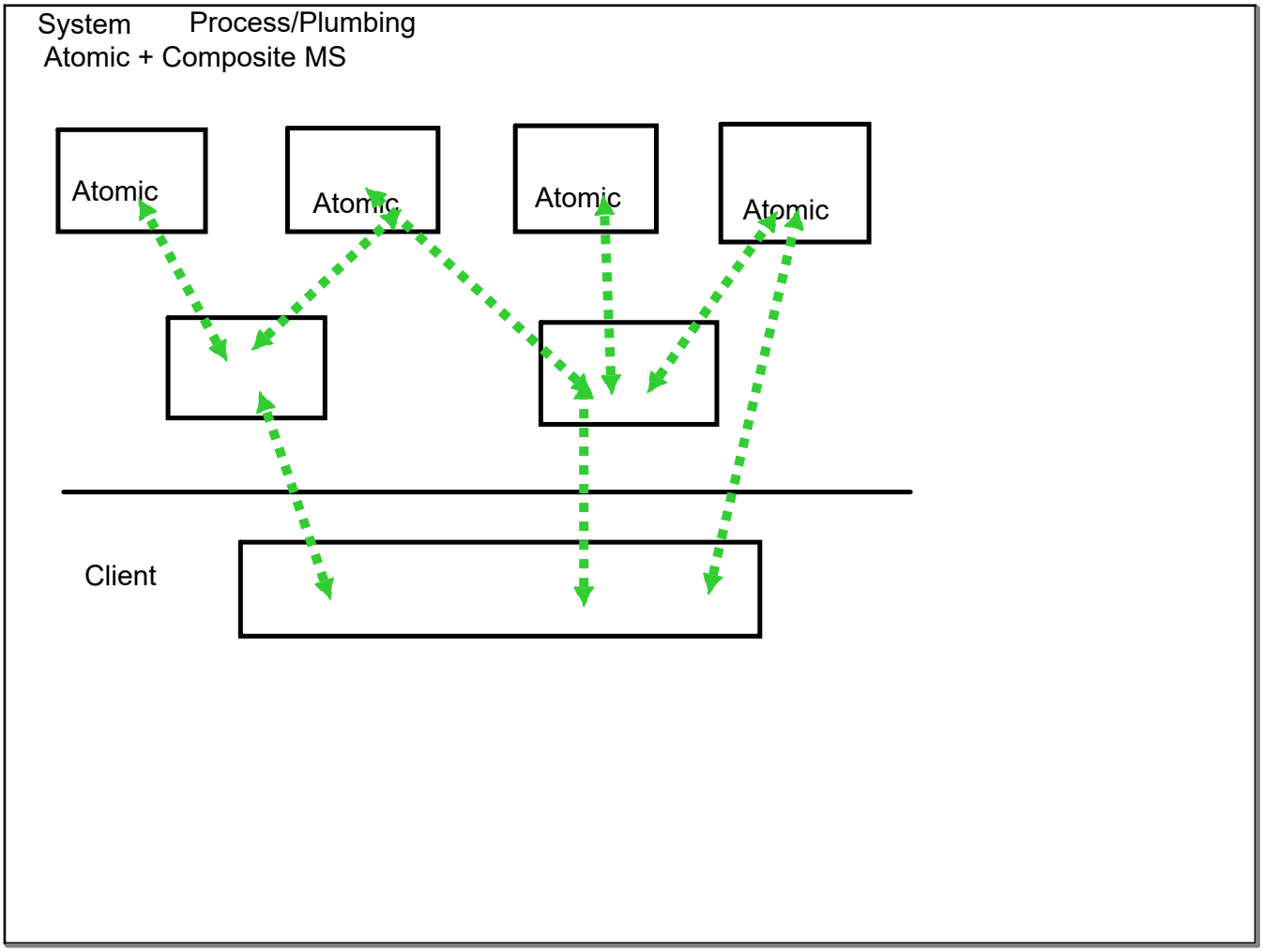
Most resonable default

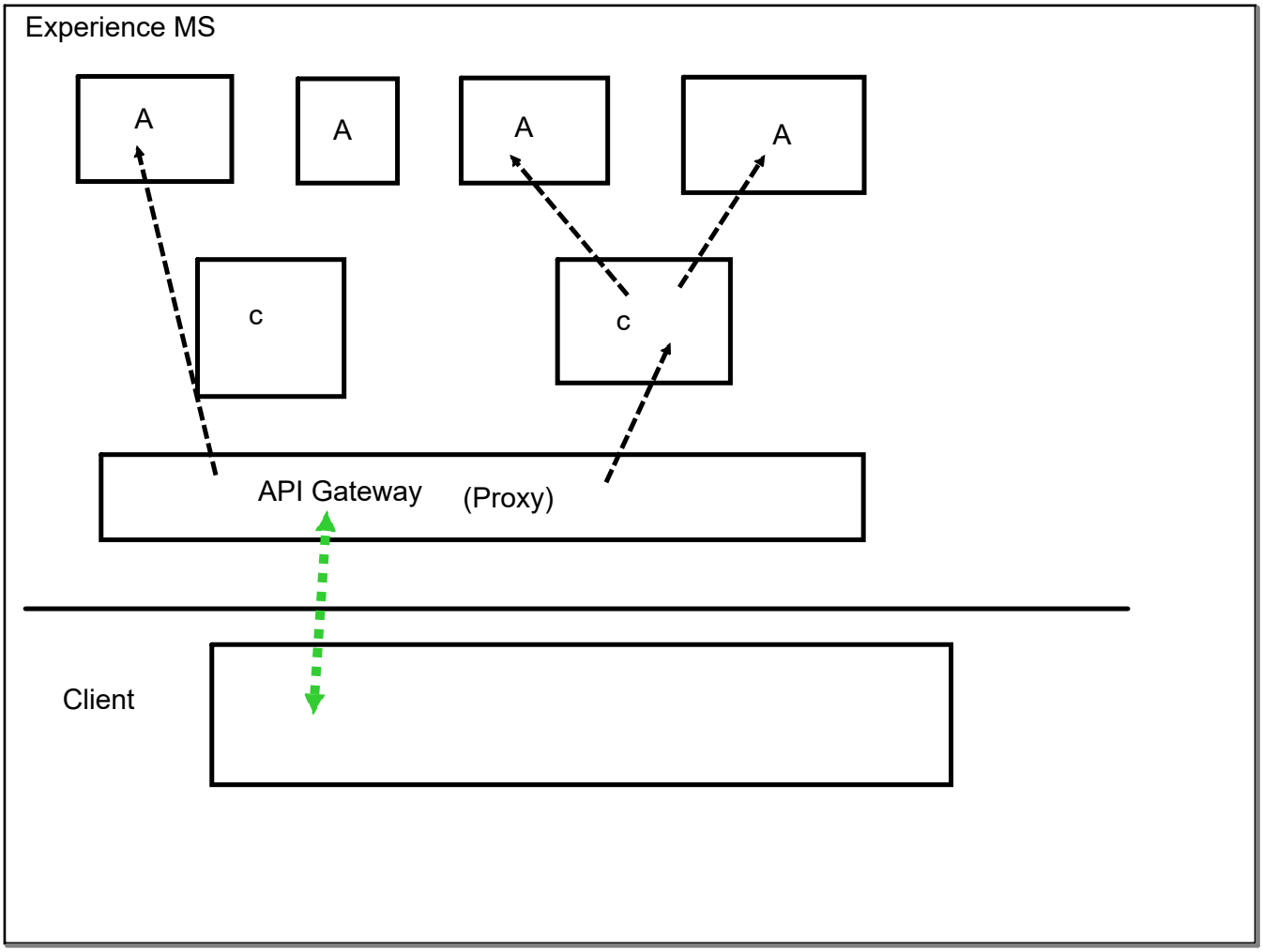
Integrates Spring cloud out of the box

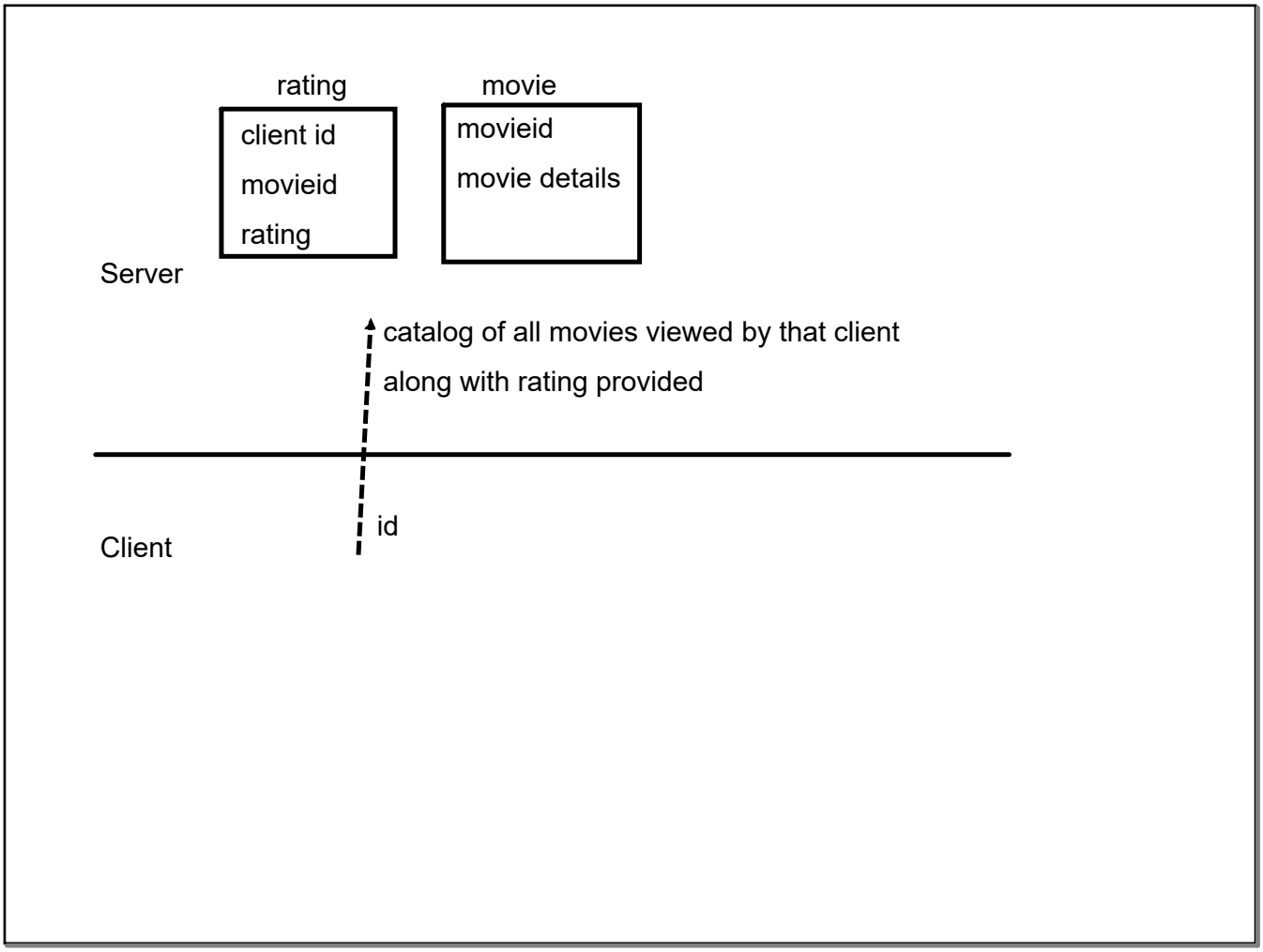

Creating a set of
microservice
MS
Spring
Boot
Spring Cloud
Cloud Native
Cloud Provides
PaaS
1 Design Patterns MS + 12 factor guideline
2. Spring Cloud

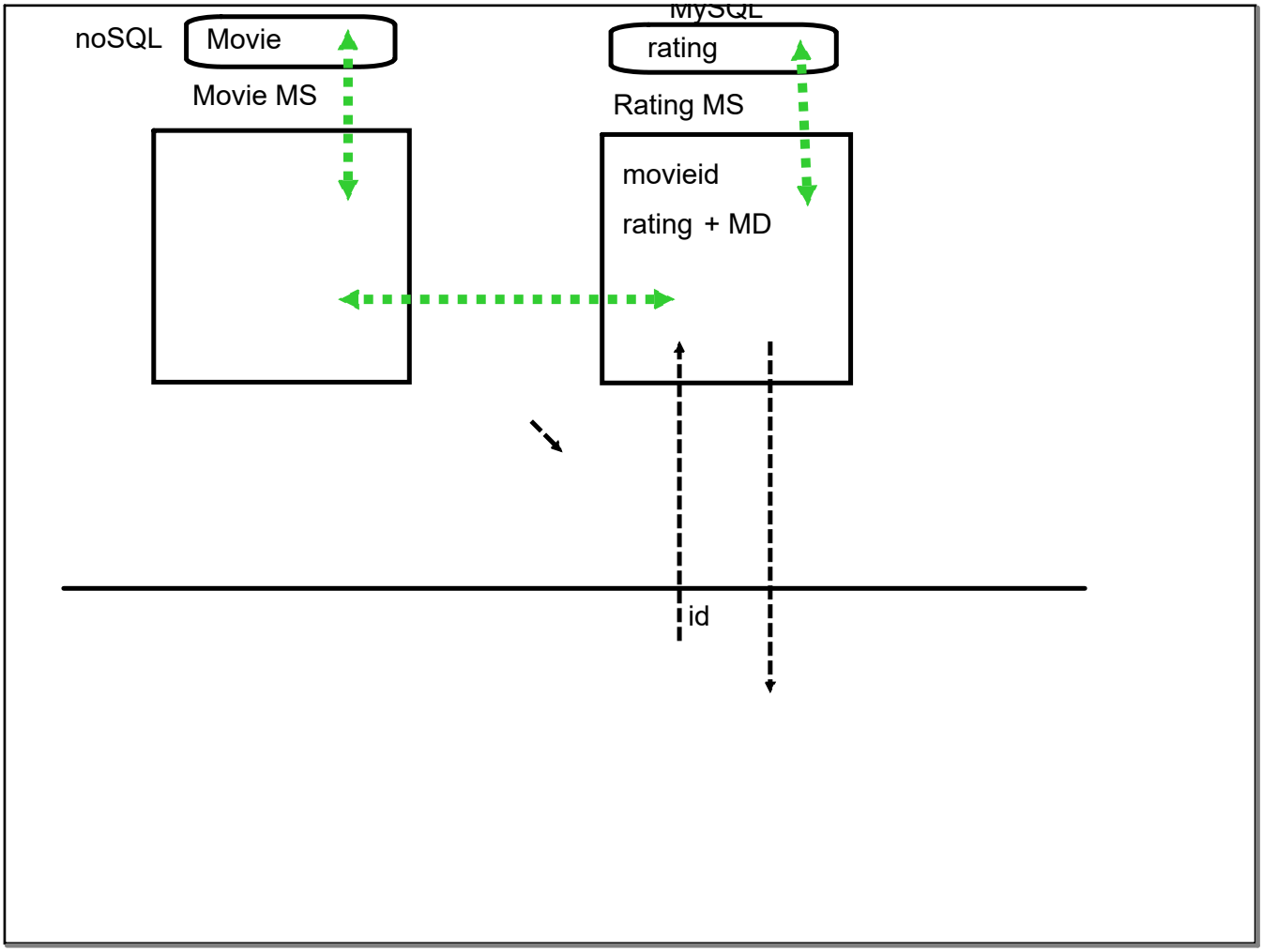
Service Granularity

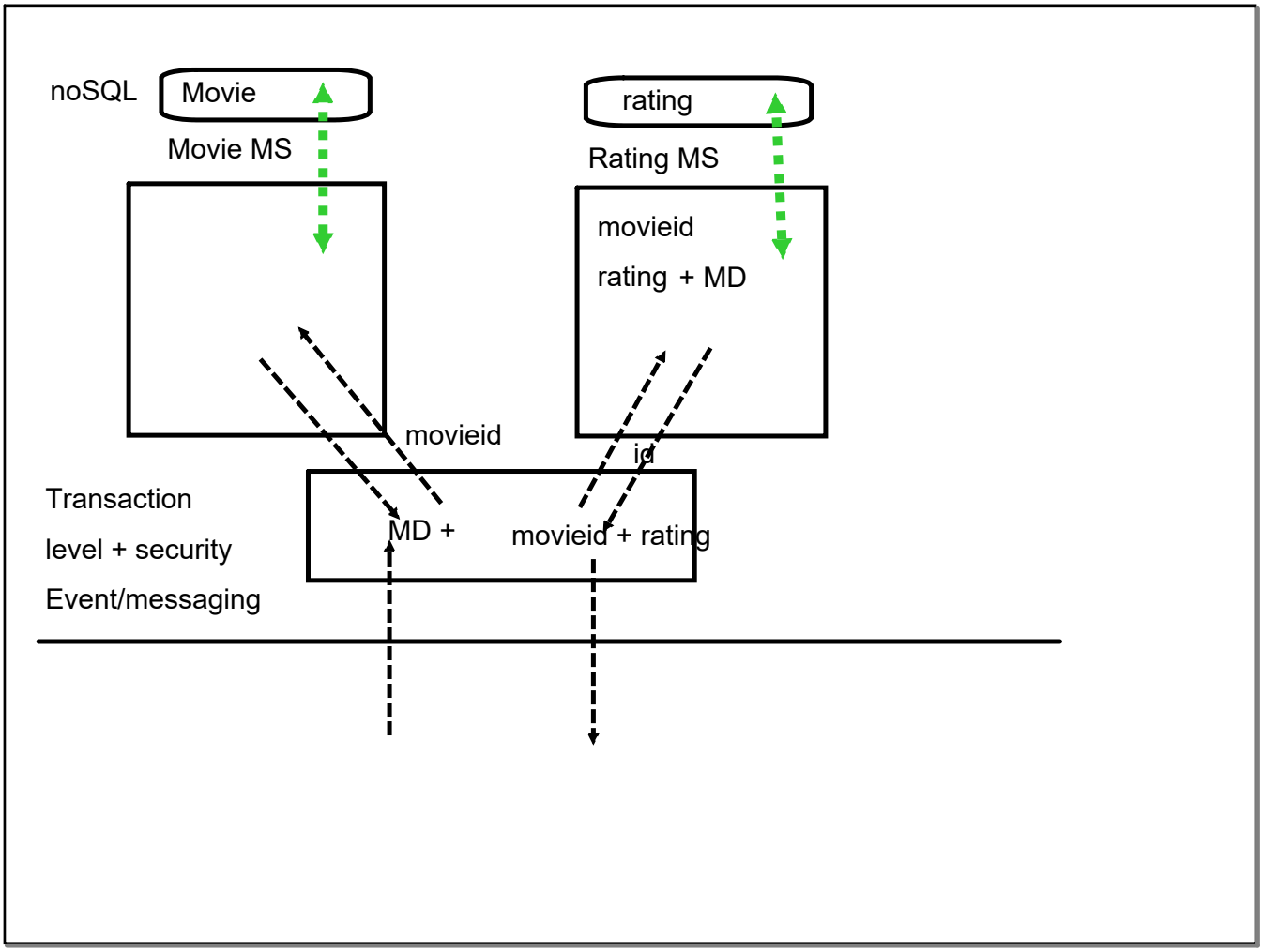



System
Process/Plumbing
Atomic + Composite MS
Atomic
Atomic
Atomic
Atomic
Client


Experience MS
A
A
A
A
c
c
API Gateway
(Proxy)
Client


rating
client id
movieid
rating
movie
movieid
movie details
Server
catalog of all movies viewed by that client
along with rating provided
id
Client


noSQL
Movie
Movie MS
MySQL
rating
Rating MS
movieid
rating + MD
id


noSQL
Movie
Movie MS
rating
Rating MS
movieid
rating + MD
movieid
id
Transaction
level + security
Event/messaging
MD +
movieid + rating

12-factor guildeline

Lightweight

Reactive

Stateless

Atomic

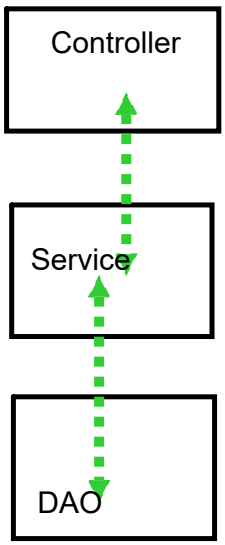
Externalized

Consistent

Resilient

Good Citizens

Versioned x.x.x



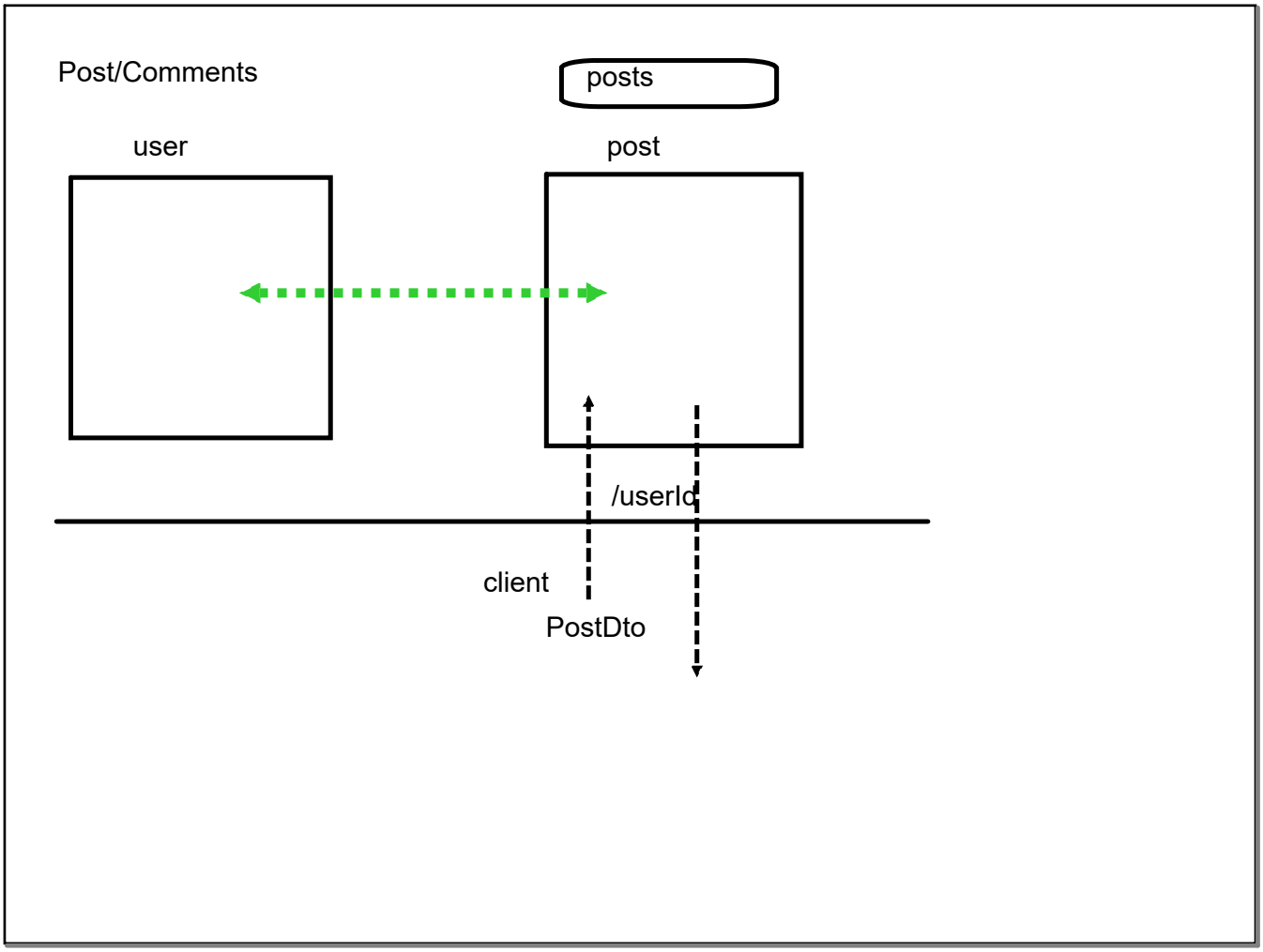
Spring-Data (persistent API)

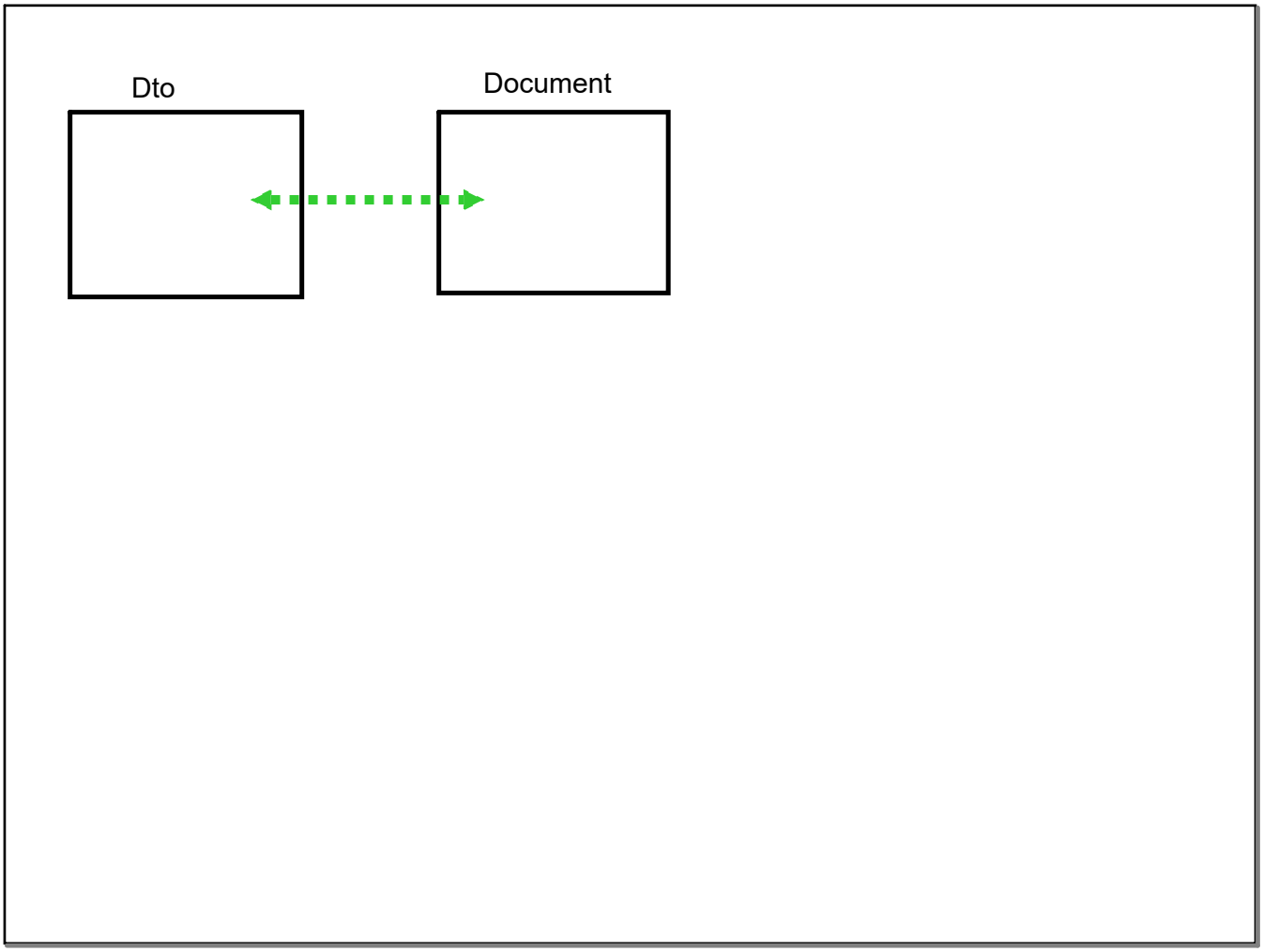
: Mysql : JPA

: Mongo-DB impl

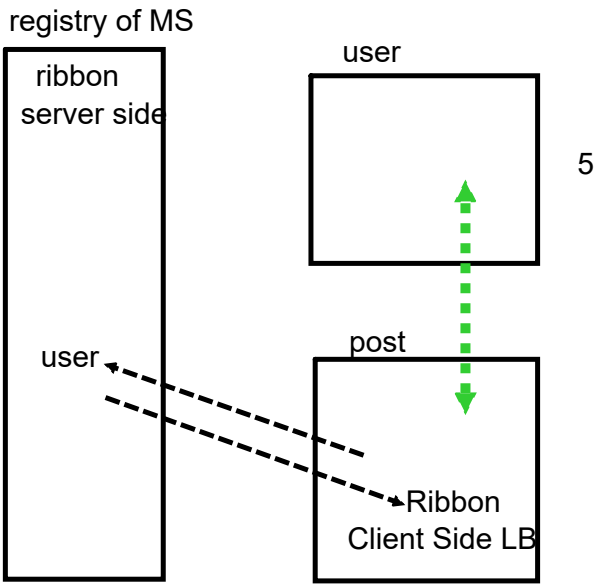
=> Lots of pre-built DB interaction

=> Add custom method : implementation provided on the fly
proper naming convention


Post/Comments
posts
user
post
/userId
client
PostDto


Dto
Document

Discovery Server (Eureka naming : netflix)



HTML : STRUCTURE

CSS : PRESENTATION

Javascript : BEHAVIOR

HTML-5

backward compatibility

Standardized the error handing

New Semantics Elements

Built-in API

audio/video

Semantically correct Structural elements

traditional : <p>,,

article,

section

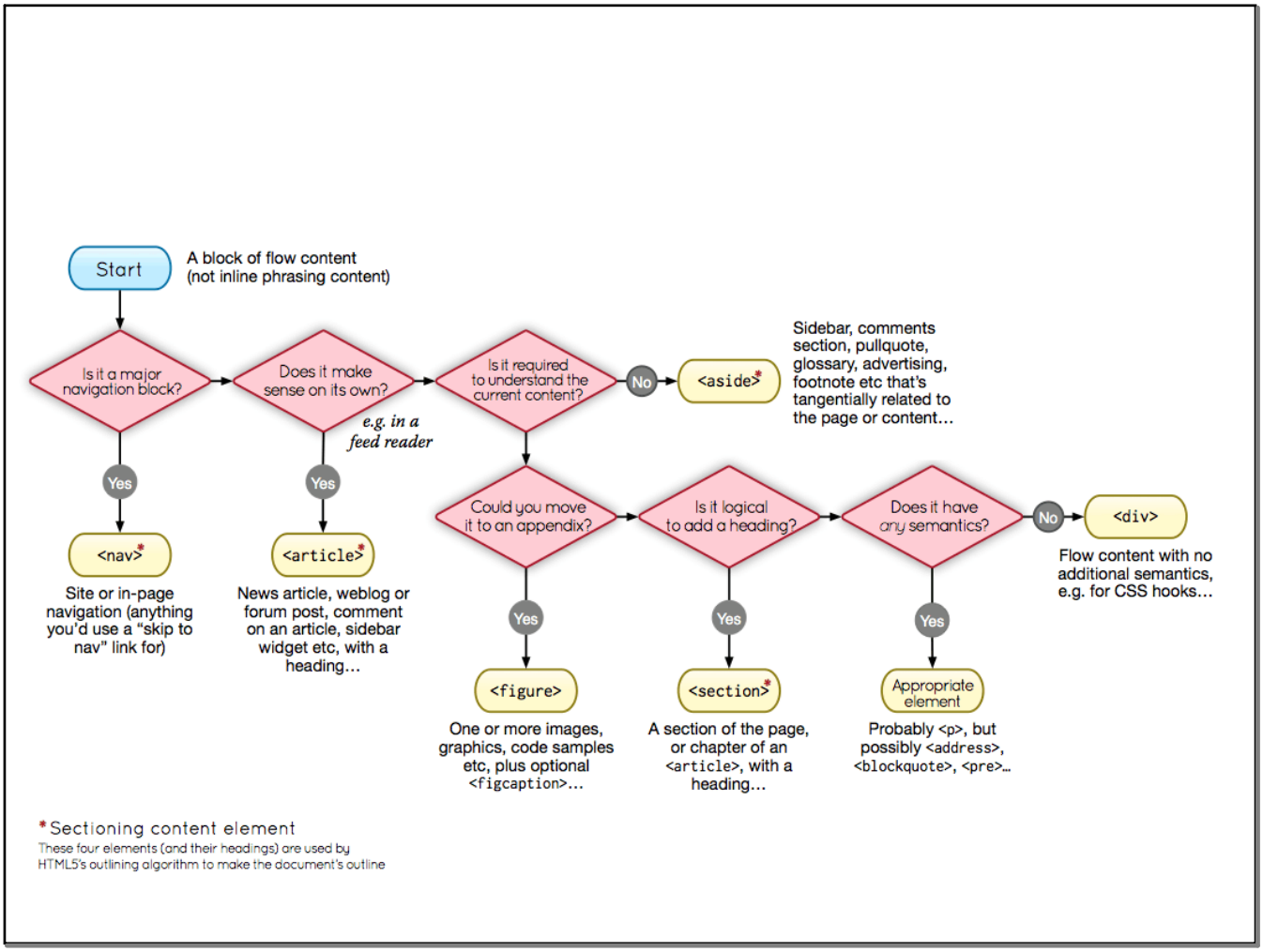
aside,

header,

footer

OUTLINE ALGORITHM :

- Smooth renderring
- Assisstive technology
- Search EO


Start
A block of flow content (not inline phrasing content)
Is it a major navigation block?
Yes
<nav>*
Site or in-page navigation (anything you'd use a "skip to nav" link for)
Does it make sense on its own?
e.g. in a feed reader
Yes
<article>*
News article, weblog or forum post, comment on an article, sidebar widget etc, with a heading…
Is it required to understand the current content?
No
<aside>*
Sidebar, comments section, pullquote, glossary, advertising, footnote etc that's tangentially related to the page or content…
Could you move it to an appendix?
Yes
<figure>
One or more images, graphics, code samples etc, plus optional <figcaption>…
Is it logical to add a heading?
Yes
<section>*
A section of the page, or chapter of an <article>, with a heading…
Does it have any semantics?
Yes
Appropriate element
Probably <p>, but possibly <address>, <blockquote>, <pre>…
No

Flow content with no additional semantics, e.g. for CSS hooks…
*Sectioning content element
These four elements (and their headings) are used by HTML5's outlining algorithm to make the document's outline

Categorization

- Metadata
- Metadata
- Interactive
- HEading
- Phrasing
- Sectioning

API

- DOM spec is part of HTML5

Built in APIs (internal activities)

A/V Api

Offline Application API

History API

WEb Protocol API

Drag n Drop API

Geolocation

2D Canvas

Local Storage

Session Storage

Messaging API

Local DB API

CSS

Cascade Style Sheet

SS : presentation rul

Cascading : rule for resolving conflicts with multiple SS applied on same elements

Browser SS	Location of style
external	Heirarchy of HTML
internal	Assembly of properties
inline	

selector { property : value}

p{color: #FFFFF}

Cascading rule

Specificity

element

(ID)

. Class

Bootstrap + JS

BootStrap : mobile first CSS library (device independent)

Grid System : use 5 grid breakpoint

extra small : <567 px : col (auto layout)

small 567-768 col-sm

medium 768-992 col-md

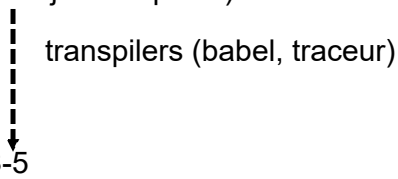
large 992-1200 col-lg

xl > 1200 col-xl

VAnilla JS (Ecma Script)

ES - 5

ES-5 > (external javascript file)



JAvascript (rich set of library allowed to integrate)

Objects

=> HTML/CSS code (DOM Tree)

=> Browser window

=> history

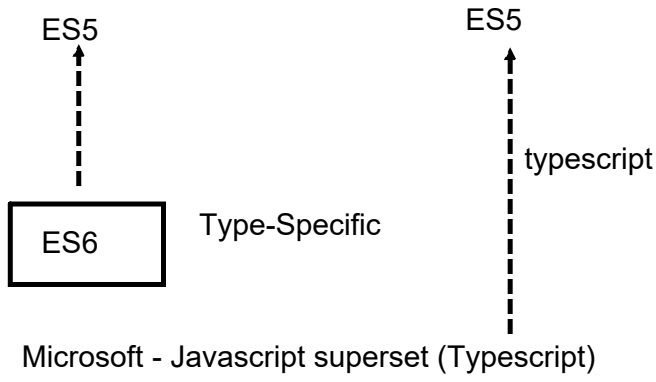
=> navigation

ES6

JS : making HTML page more dynamic/responsive/interactive

Server based logic --> Client (Outsource)

New Programming constructs



Microsoft - Javascript superset (Typescript)

```
function add(num1, num2){
    return num1 + num2;
}

add(20,30);  50
add('20','30'); 2030
```

named Type

interface/generic

npm

npm install -g typescript

```
var button = document.querySelector("button");
var input1 = document.getElementById("num1");
var input2 = document.getElementById("num2");



function add(num1, num2){
    return num1 + num2;
}

button.addEventListener("click", function(){
    console.log(add(+input1.value, +input2.value));
});
```