Java-8

=> Lambdas

Functional Programming

those feature that define functional programming

streams

Executor (Future)

Concurrency Collection

Style :

Traditional : Imperative

(HOW)

#exposing the steps how to perform an operation

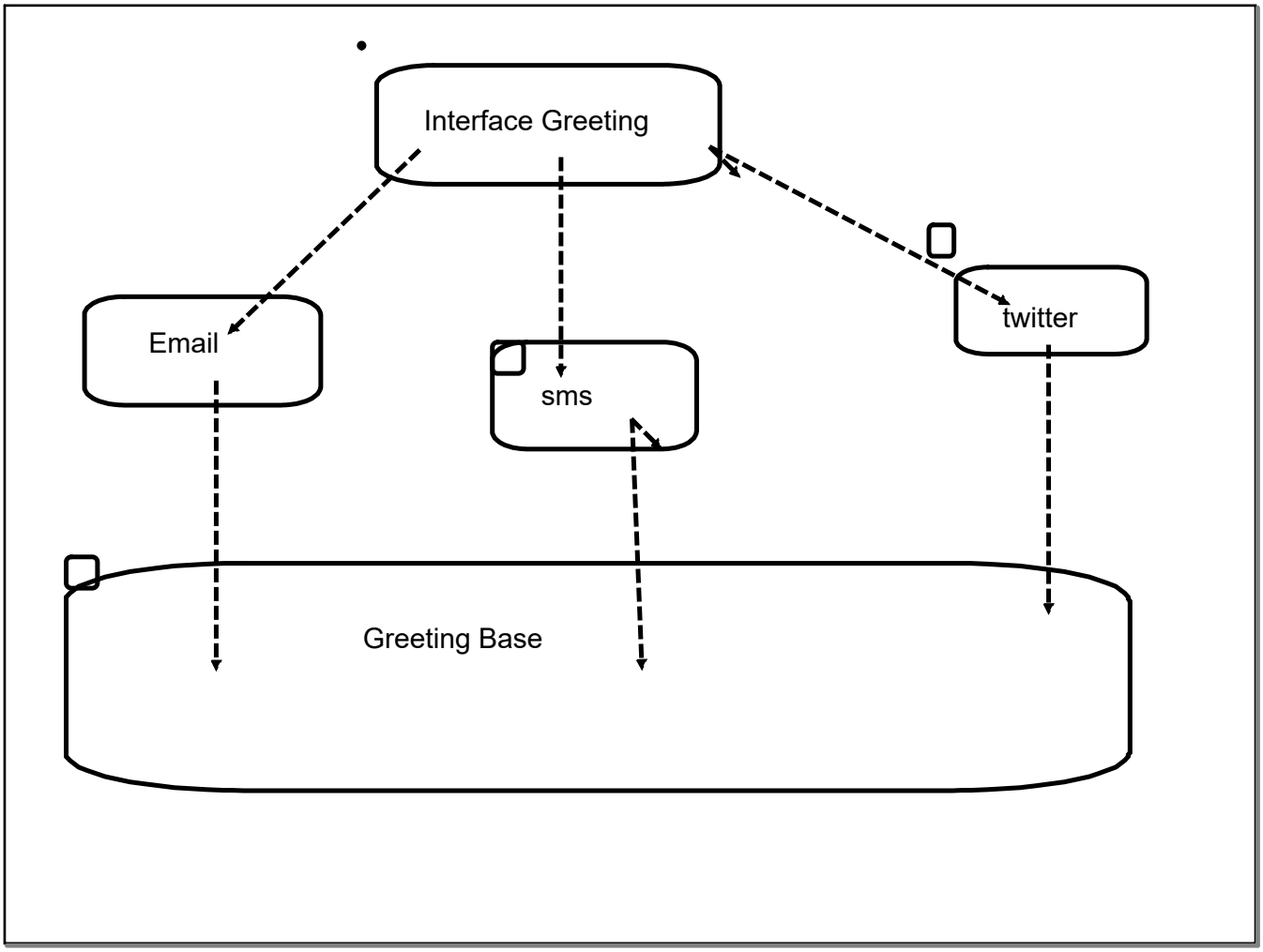
embrace object mutability (not in sync with concurrency)

Functional : Declarative

(What) : result

immutability

Analogous SQL

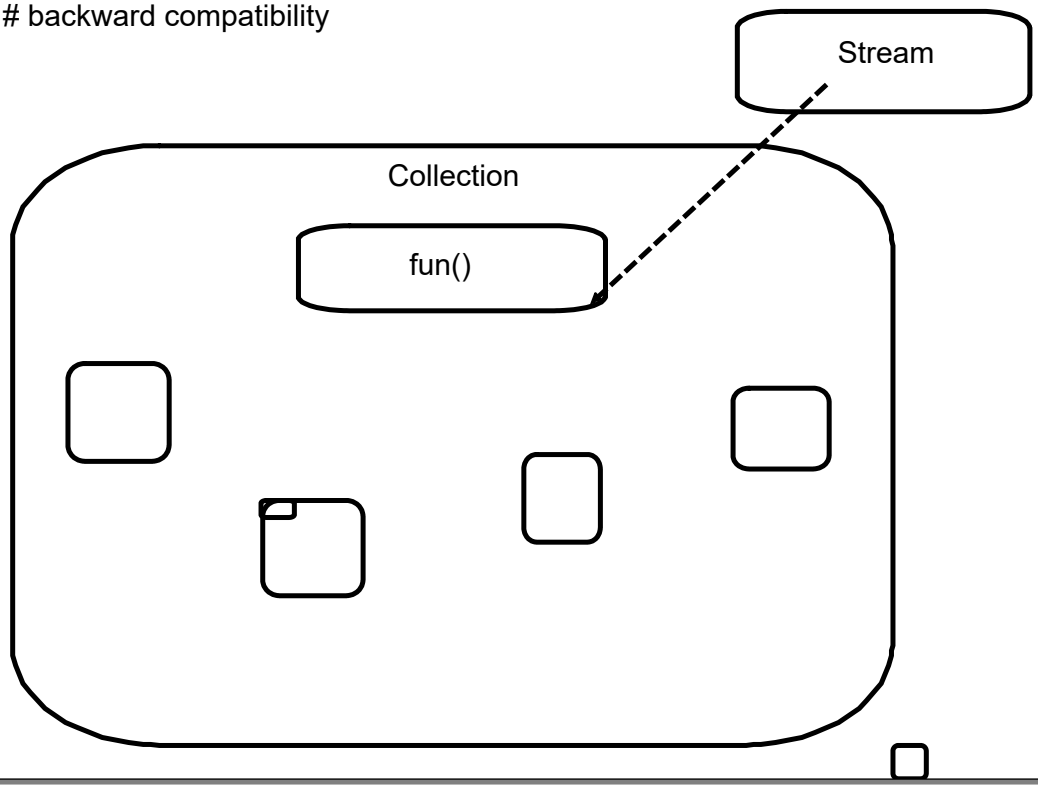

Interface Greeting
Email
sms
twitter
Greeting Base

Interface :

default method (definition)

multiple implementations can be inheritance

backward compatibility



Escape from OOPs

independent Functions (not wrapped inside an object)

Relationship between interface and function

1. interface must have only one abstract method (any number of default/static) :

Functional Interface : Annotation @FunctionalInterface

2. single method signature must match with function implementation

Lambda expression

(<arg1>,<arg2>) -> {

}

arg1 -> {

}

() -> {

}

(<arg1>) -> <return> <single instruction>

(a,b) -> <return>a+b;

(a,b) -> {

return a+b;

}

Pre defined functional interfaces

=> Runnable

=> Comparator

Explicit Functional Interface

Consumer

void accept(<>);

DoubleConsumer() // specialized implementations on primitive

BiConsumer

void accept(<>,<>);

Predicate (test)

boolean test(<>)

Supplier

<> get()

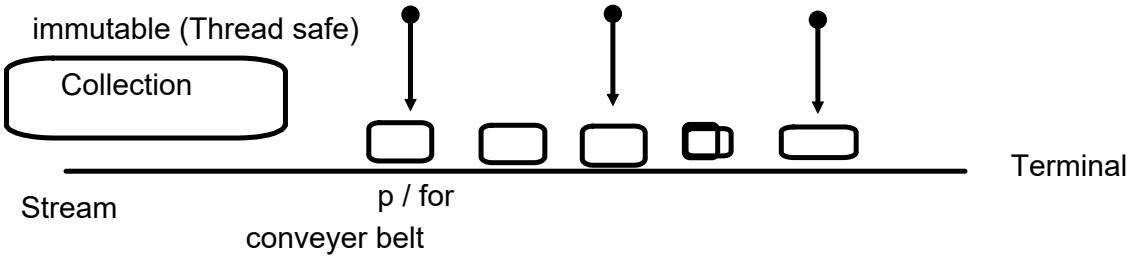
Function

<> apply(<>)

Stream :

not a data structure

immutable (Thread safe)



Stream : LAZY Processes

Stream can have 2 type of activities

intermidiate activities (filter/map/flat map)

Terminal activity : terminate/close the stream

forEach()

collect()

IF terminal activity is not present : stream will not initiate

groupingBy(<return> Function(student))

return value : would become a group

(Stream of) Multiple collection into (Stream of)single collection

Transforms

y map(x)

flatmap() : Collection into stream

map:

["",""]

["","",""]

["",""]

flat map

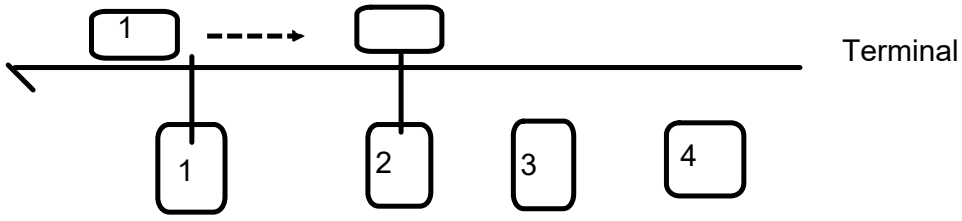
["","","","","","","","",""]

return type fixed : stream of data passed as argument

Stream :

Sequential Stream

Parallel Stream



Parallel Streaming not commended if working on external mutable data (not thread safe)

Activities that are inherently complex

Binary Operator : variant Function

y Function(x) : x and y can be of different type

z BinaryOperator(x,y) : x,y,z : must be of same type

Multithreading :

interleaved (Threaded Multitasking)

1. Multiple activities waiting for I/O : that time can be used by tasks
2. Multi-core architecture of micro-processor

Base Interface :

Runnable (run)

Implementation:

Core Functionality of Multithreading (Thread)

inheriting Runnable

inheriting Thread

Traditional approach:

create, execute, manage

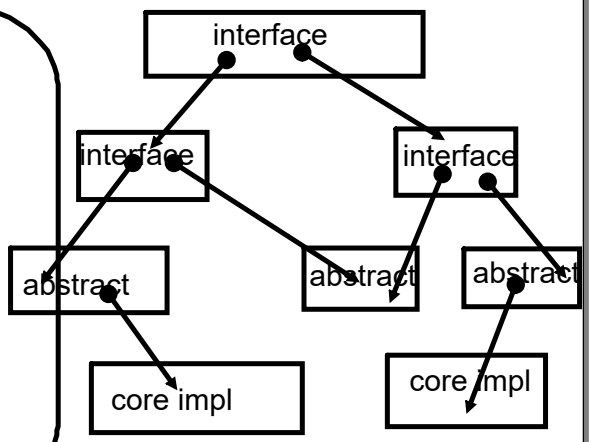
Thread per task

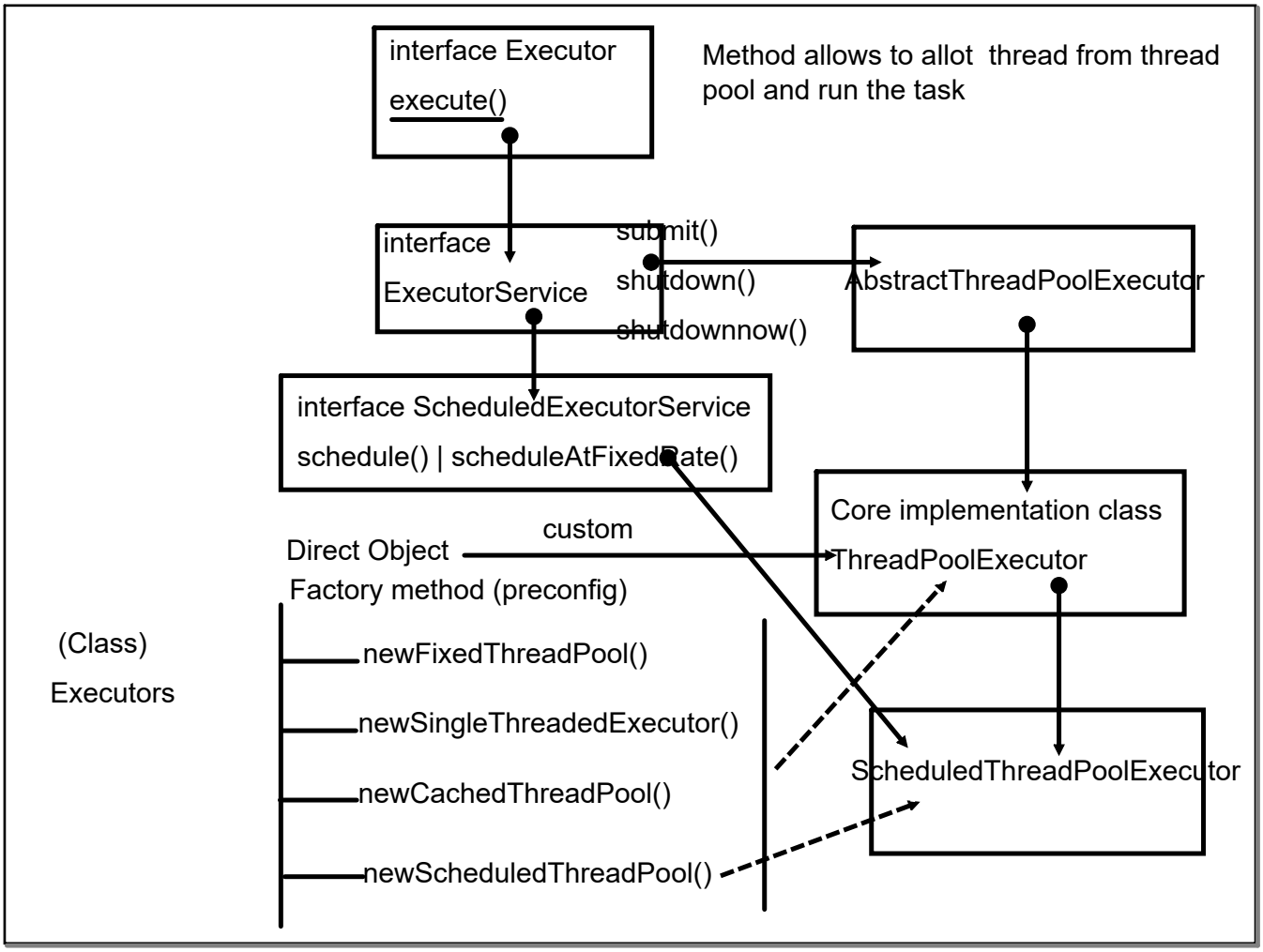
Not possible to keep a track of number of thread

Executor framework :

do all management + improve in performance

ThreadPool : create a predefined pool of thread




interface Executor
execute()
Method allows to allot thread from thread pool and run the task
interface
ExecutorService
submit()
shutdown()
shutdownnow()
AbstractThreadPoolExecutor
interface ScheduledExecutorService
schedule() | scheduleAtFixedRate()
Core implementation class
ThreadPoolExecutor
custom
Direct Object
Factory method (preconfig)
(Class)
Executors
newFixedThreadPool()
newSingleThreadedExecutor()
newCachedThreadPool()
newScheduledThreadPool()
ScheduledThreadPoolExecutor

Need to create instance of ThreadPoolExecutor

FixedThreadPool (number of thread are predefined(extra task alloted will added to queue)

CustomThreadPoolExecutor

<corePoolSize> : number of threads to always keep even if they are idle (2)

<maxPoolSize>: max no of thread (5)

<keepAliveTime> : time to wait before idle thread gets removed/released from thread pool

<TimeUnit> :

<queue capacity>: capacity of queue

<RejectedHAndler> : what to do if a task is rejected from queue

SingleThreadExecutor()

FixedThreadExecutor(1)

can change the thread capacity

CachedThreadPool() : Unbounded ThreadPool : Max Integer Val

if demand decreases : can tear down thread

default keep alive time : 1 min

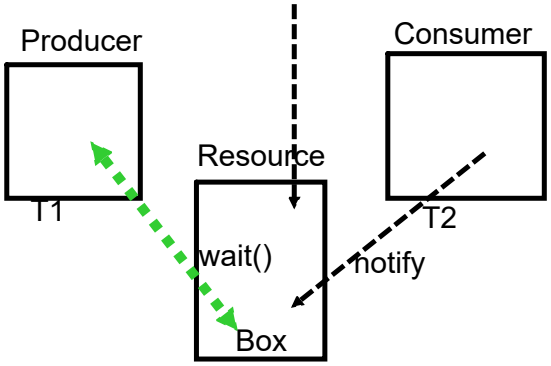
ScheduleThreadPool()

Thread returning some value

Special Interface Callable (call())

ThreadPool can work on Callable

wait/notify/notifyAll() : Object



ExecutorCompletionService

: will going to get results in order of completion of task

Future : blocking

CompletableFuture <callback : logic to follow when task is done>

Functional interfaces

Runnable

Callable

=> Supplier

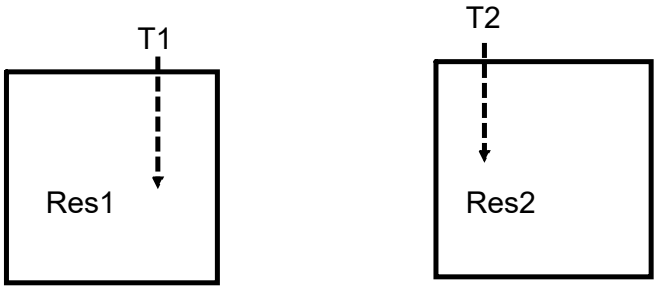
the method to associate a callback function

1. thenApply(Function); // transform
2. thenAccept(Consumer); // consuming and using

CompleatableFuture by default uses the inbuilt thread pool

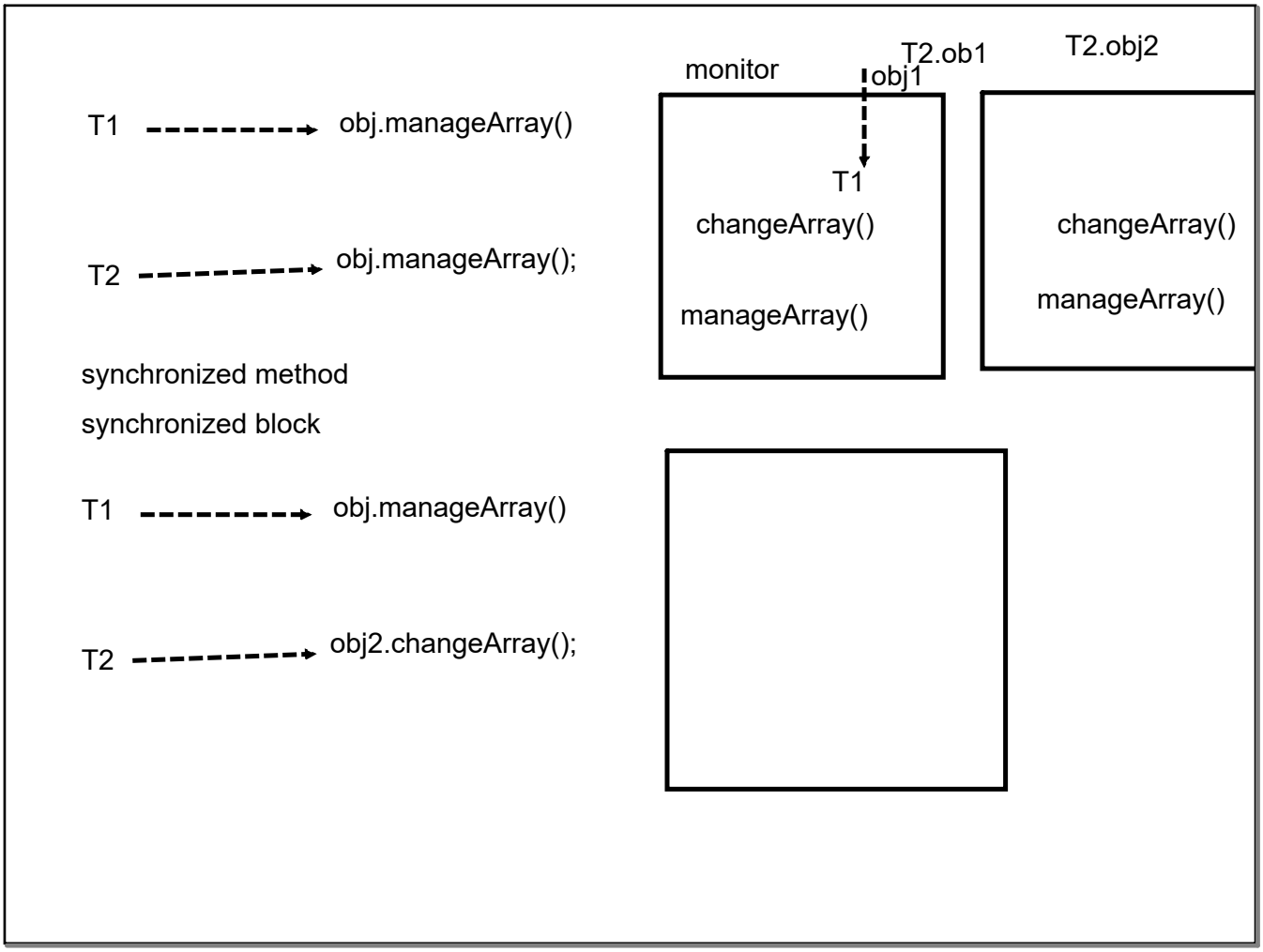
ForkJoinPool.commonPool();

Executor ThreadPool



Common Resource Shared among multiple threads (Thread safe)

Resolve Data inconsistency


T1
obj.manageArray()
T2
obj.manageArray();
synchronized method
synchronized block
T1
obj.manageArray()
T2
obj2.changeArray();
monitor
T2.ob1
obj1
T1
changeArray()
manageArray()
T2.obj2
changeArray()
manageArray()

locking :

=>wide spectrum locking : (synchronized...)

=>granular locking

java.util.concurrent.

API : Granular locking on resources

Collection API

1 .Traditional : 2

1. HashTable
2. Vector

2. To get a Thread safe variant of those class

Collections.concurrentList();

all methods are sync

Atomic operation : single CPU instruction

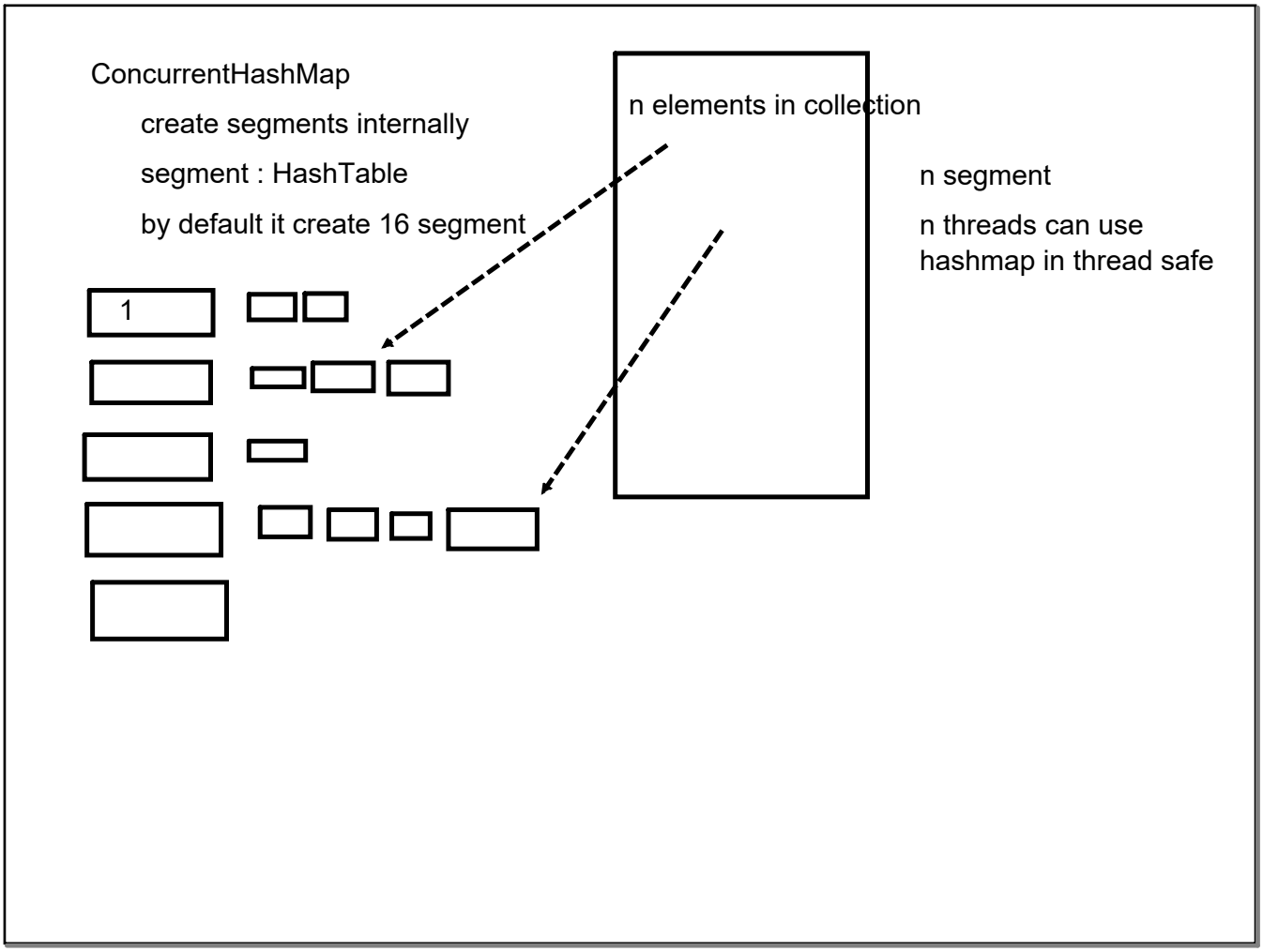
n=10; // Thread safe operations

assignment long/double are non-atomic

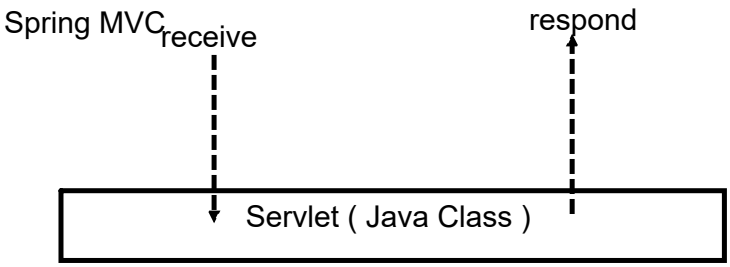
Concurent API : Focus on granular locking

Provides Atomic Variant of type : allow to convert non-atomic activities into atomic

multiple approach for ThreadSafety along with high level of concurrency


ConcurrentHashMap
create segments internally
segment : HashTable
by default it create 16 segment
1
n elements in collection
n segment
n threads can use
hashmap in thread safe

Servlet Technology



How to define java class as Servlet

Extends

HttpServlet/GenericServlet

GenericServlet : does not classifies between various HTTP Verbs

HttpServlet : can identify

GET/POST/PUT/DELETE/PATCH

Life Cycle

=> init

=> generic service() / http (doPost()/doGet()/doDelete()

=>destroy

Servlet Technology

(Controlled By web.xml)

lst request

Multiple request simultaneously

instance of servlet class is created

init phase gets activated

service() (Responsible for receive request/and respond)

Web Servers tends to cache the servlet object

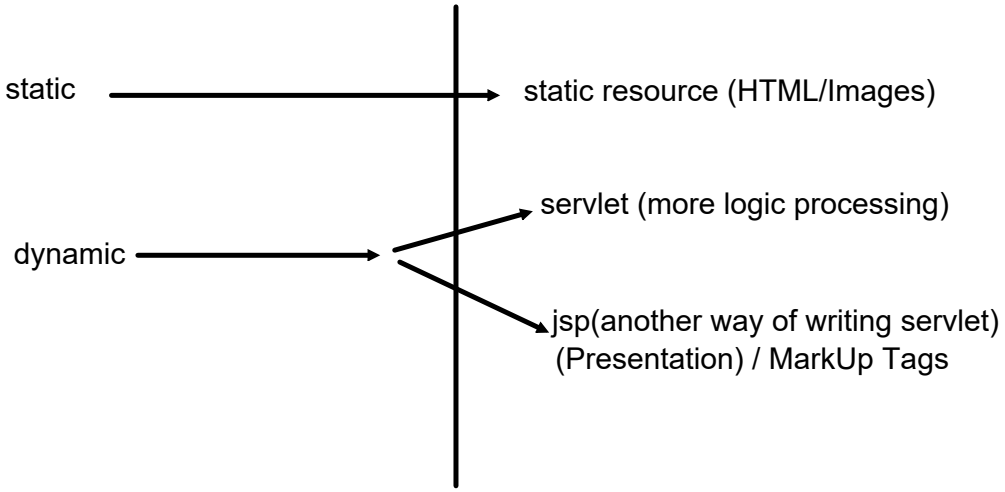
destroy

Object is released

service method and variant

have an access over HttpRequest/HttpResponse object

JSP : another way of writing servlet

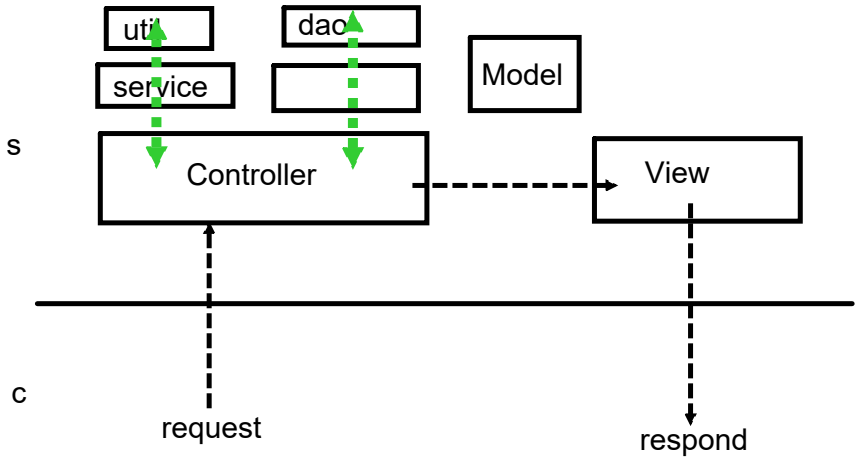


Spring uses Servlet Technology:

But provides a high level abstraction over complexities/ boilerplate req / config and enhances the seperation of concerns

MVC architecture

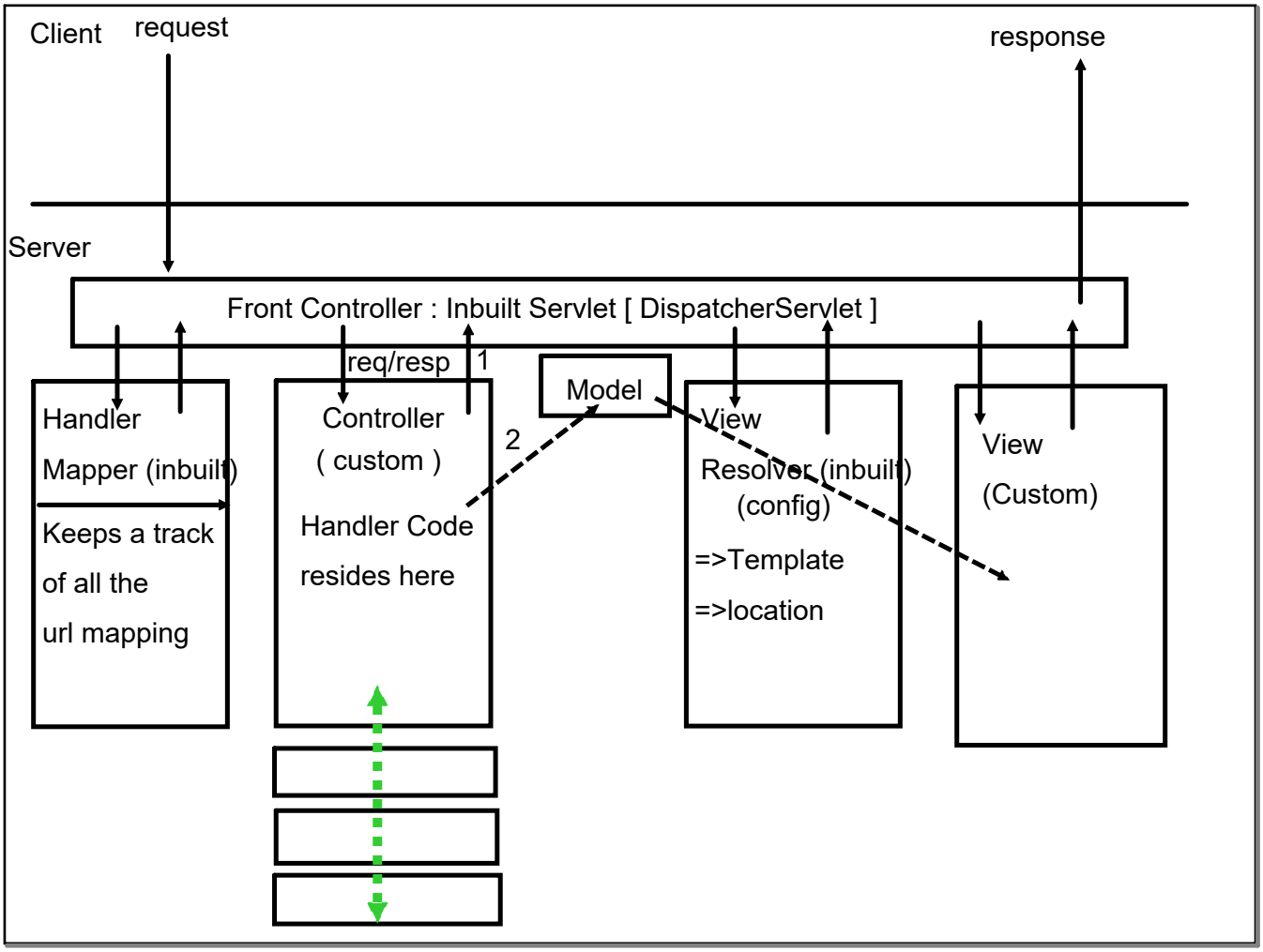
Controller : to receive request / process it



Servlet

service method as task :

assign it to thread


Client
request
response
Server
Front Controller : Inbuilt Servlet [DispatcherServlet]
req/resp
1
Handler
Mapper (inbuilt)
Keeps a track
of all the
url mapping
Controller
(custom)
Handler Code
resides here
2
Model
View
Resolver (inbuilt)
(config)
=>Template
=>location
View
(Custom)

we need to register your app resources (servlet spec)

Servlet :

need to register

registeration can also be done using annotation

Register DispatcherServlet

Config of Spring in place

xml file

java

Need Spring config to connect with DS

xml : <servlet-name>-servlet.xml

Controller : "index"

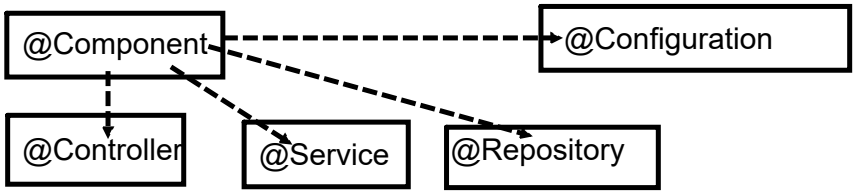
create a complete path

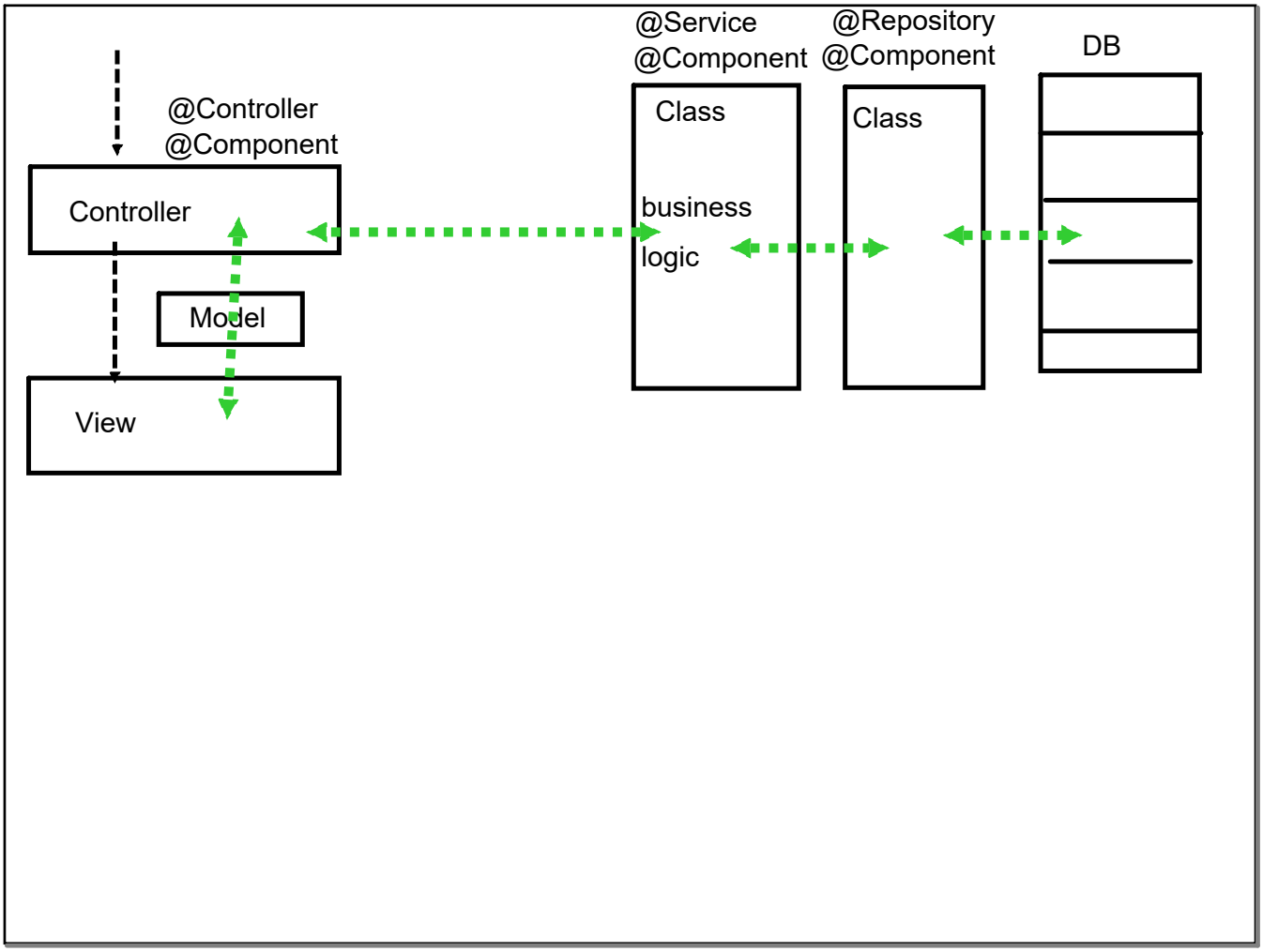
View Resolver : location + template (jsp+jstl) [extension]

```
<bean class="org.springframework.web.servlet.view.InternalResourceViewResolver">
        <property name="prefix" value="/WEB-INF/views/" />
        <property name="suffix" value=".jsp"/>
    </bean>
```

Controller return : "index"

/WEB-INF/views/index.jsp




@Service
@Component
@Repository
@Component
DB
@Controller
@Component
Class
Class
Controller
business
logic
Model
View

web.xml : ~ java config class

dispatcher-servlet.xml : ~ java config class

1. alternate for packaging : maven war plugin

Spring provides an inbuilt class to register DS

Spring-Boot

=> dependency management

=> configuration

dependency management

library + external other resources

spring-boot :

dependency

spring-starter-project

Spring - starter -(parent)project

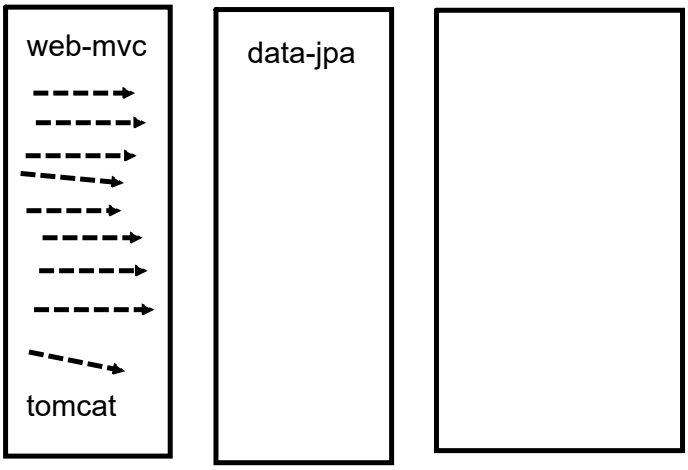
Support of Auto-configuration / less amount of code

manage Embedded Server

Created/Developed

and Launched

stand-alone way



start.spring.io

maven cli
maven command

Configuration
# Spring boot Annotation
# Dependency
# Customization : special file application.properties
key=value
key : predefined keys from different spring projects
: possible values
: custom keys/values
spring : yaml
: heirarchy
: application.yaml

Spring Boot Annotation

curated list of multiple annotation

EnableAutoConfiguration

tracking the dependencies

based on dependencies added:

add default config

expose the key

eg:

maven-web : Spring mvc:

DS servlet

spring-security

add default security

expose username/passed

tracking the properties files

looks for custom key-values pairs

defined in config-file

cli : key-values

mvc application

- controller
- view

pre-configured to use thymeleaf

View pages :

View Templates

Jsp-jstl

Thymeleaf

Mustache

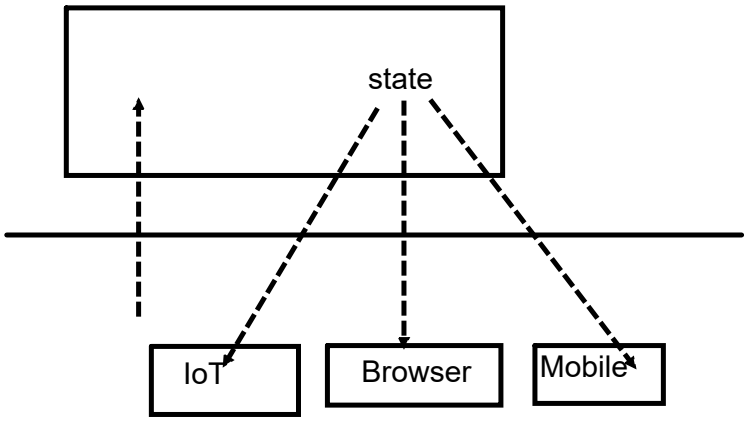
FreeMArker

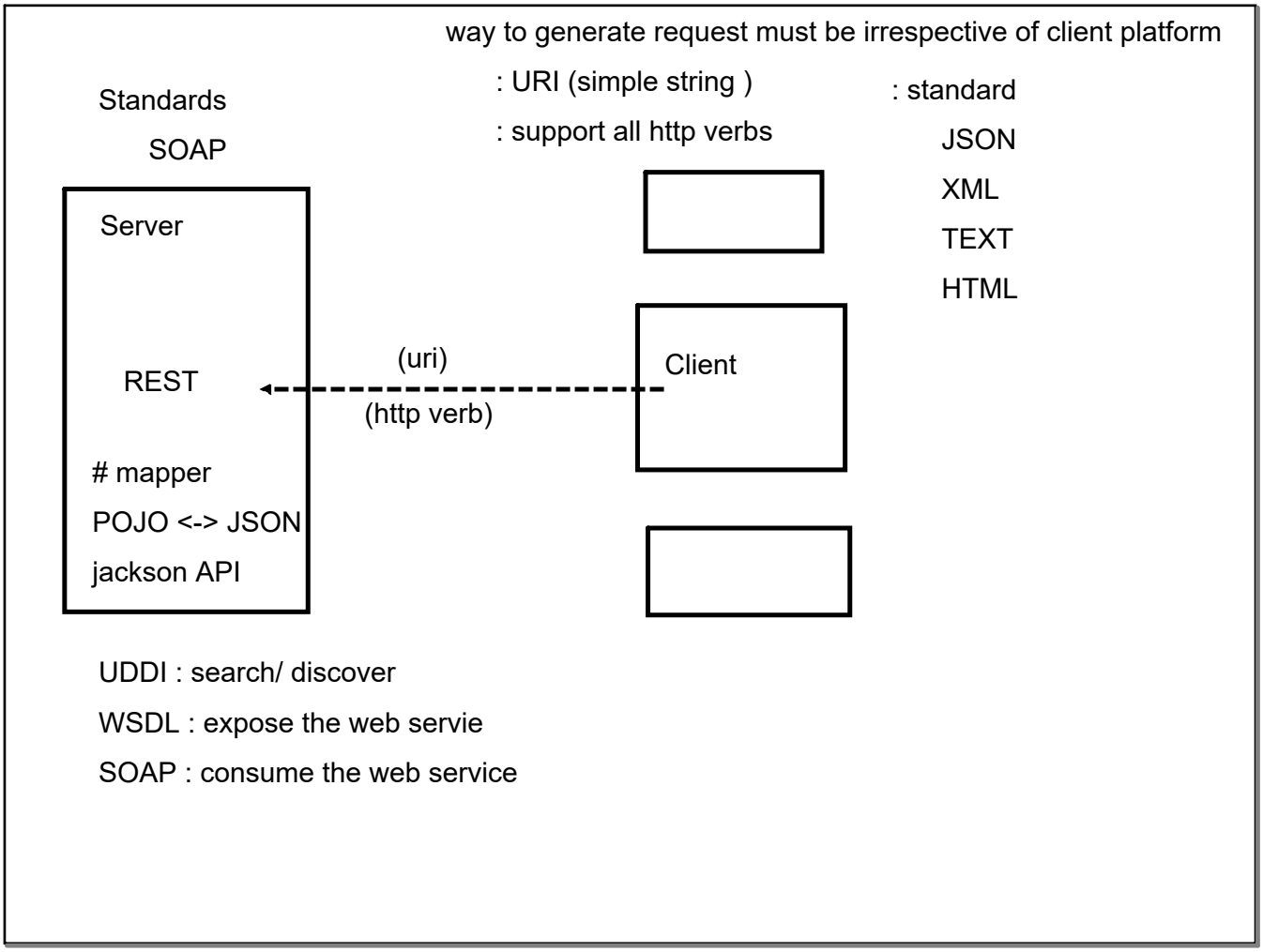
Tile

Velocity

Rest Based service

exposing server based info as Rest Service




way to generate request must be irrespective of client platform
: URI (simple string)
: support all http verbs
: standard
JSON
XML
TEXT
HTML
Standards
SOAP
Server
REST
# mapper
POJO <-> JSON
jackson API
(uri)
(http verb)
Client
UDDI : search/ discover
WSDL : expose the web servie
SOAP : consume the web service

@RestController : interconversion take care of

client intention

GET : data retrieval

POST : add new data

PUT : edition

DELETE : delete

Student /student

/getAll

Employee /employee

/getAll

@RestController : auto needs mapper in classpath (jackson)

@Controller

add mapper to classpath

produce

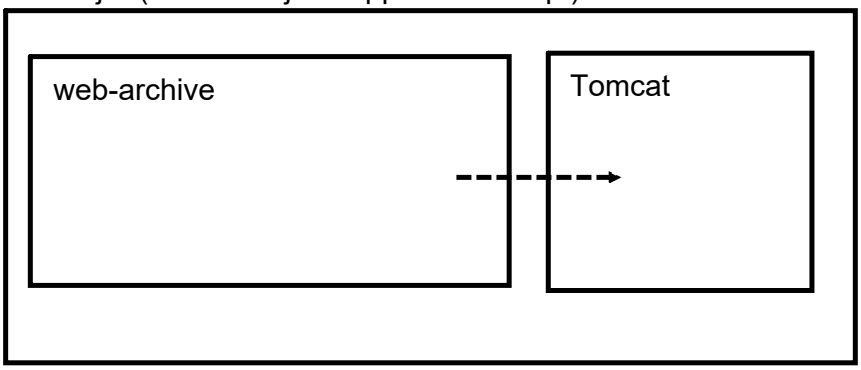
consume

@ResponseBody : content to be shared directly to client

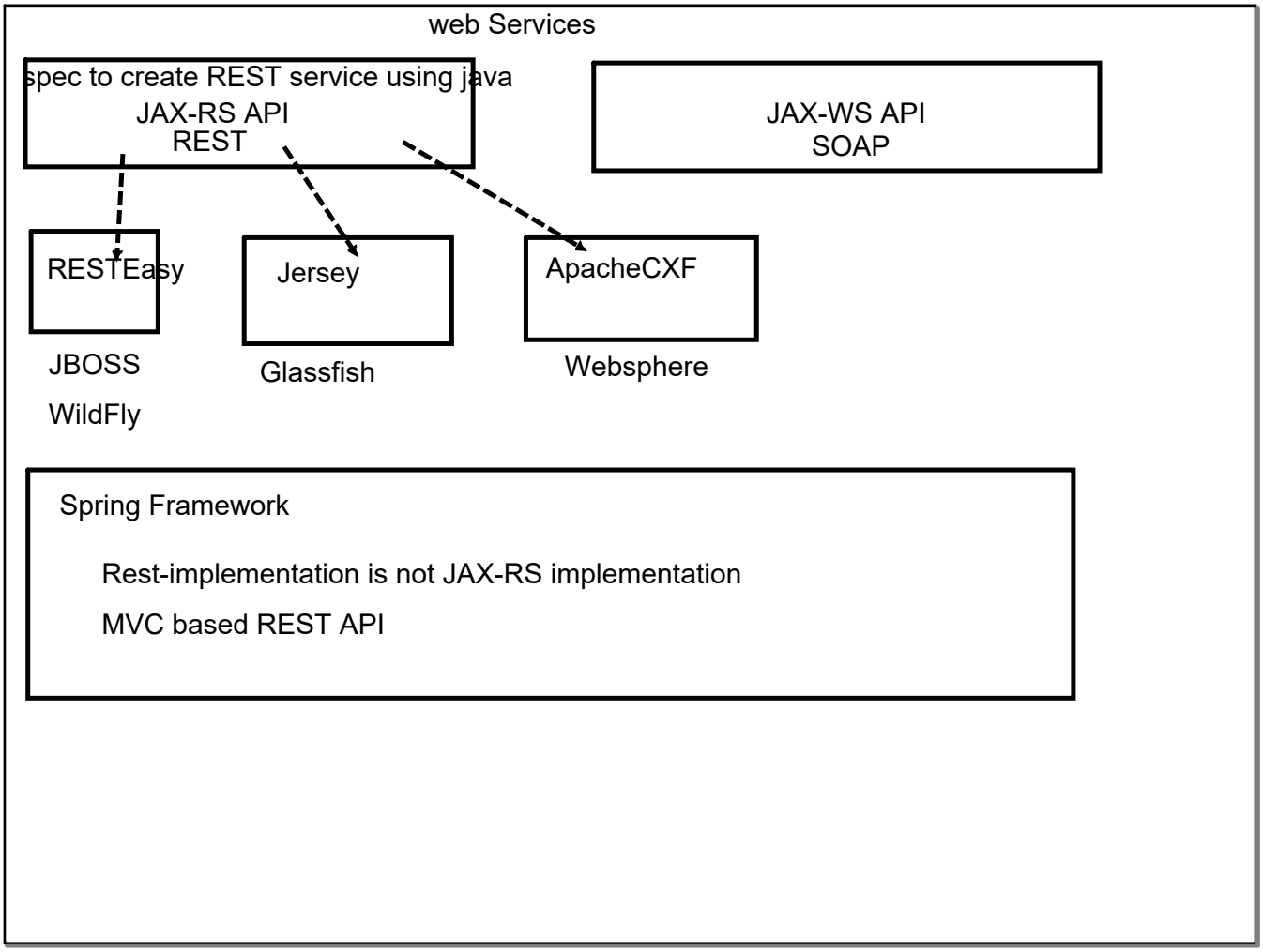
1.launch tomcat

2. deploy web-archive over tomcat

jar (additional java app build on top)



dev-tools: live reloading of server


web Services
spec to create REST service using java
JAX-RS API
REST
JAX-WS API
SOAP
RESTEasy
Jersey
ApacheCXF
JBOSS
WildFly
Glassfish
Websphere
Spring Framework
Rest-implementation is not JAX-RS implementation
MVC based REST API

actuator : exposes rest endpoint

Microservice architecture implements

Dividing a single large sized monolith application into
multiple smaller (independent) application

microservices : responsible to expose a particular service

DataDriven/Rest based

Stateless

Service Oriented Architecture : SOA :

Microservice : + technology/approach/design pattern

Monolith issues

involve light wight VS for deploying service components

Deployment :

Multi-Technology service component

DB : ideally must be using independent DB

Scaling : individual service comp

Robust in implementation

Design Guideline : MS (12 factor)

Design Pattern

Lightweight : concern/runtimes/data exchanging

Reactive : highly concurrent/longer processing

Stateless: scale better

Atomic : core design principle

Externalized config : config server

Consistent : style

Resilient : eliminate bottleneck

Good Citizens : expose usage statistics

well versioned :

Design Pattern:

Decompositions :

a) business capabilities

business-oriented rather than technical

b) sub-domain (technical)

domain class (parent/God classes)

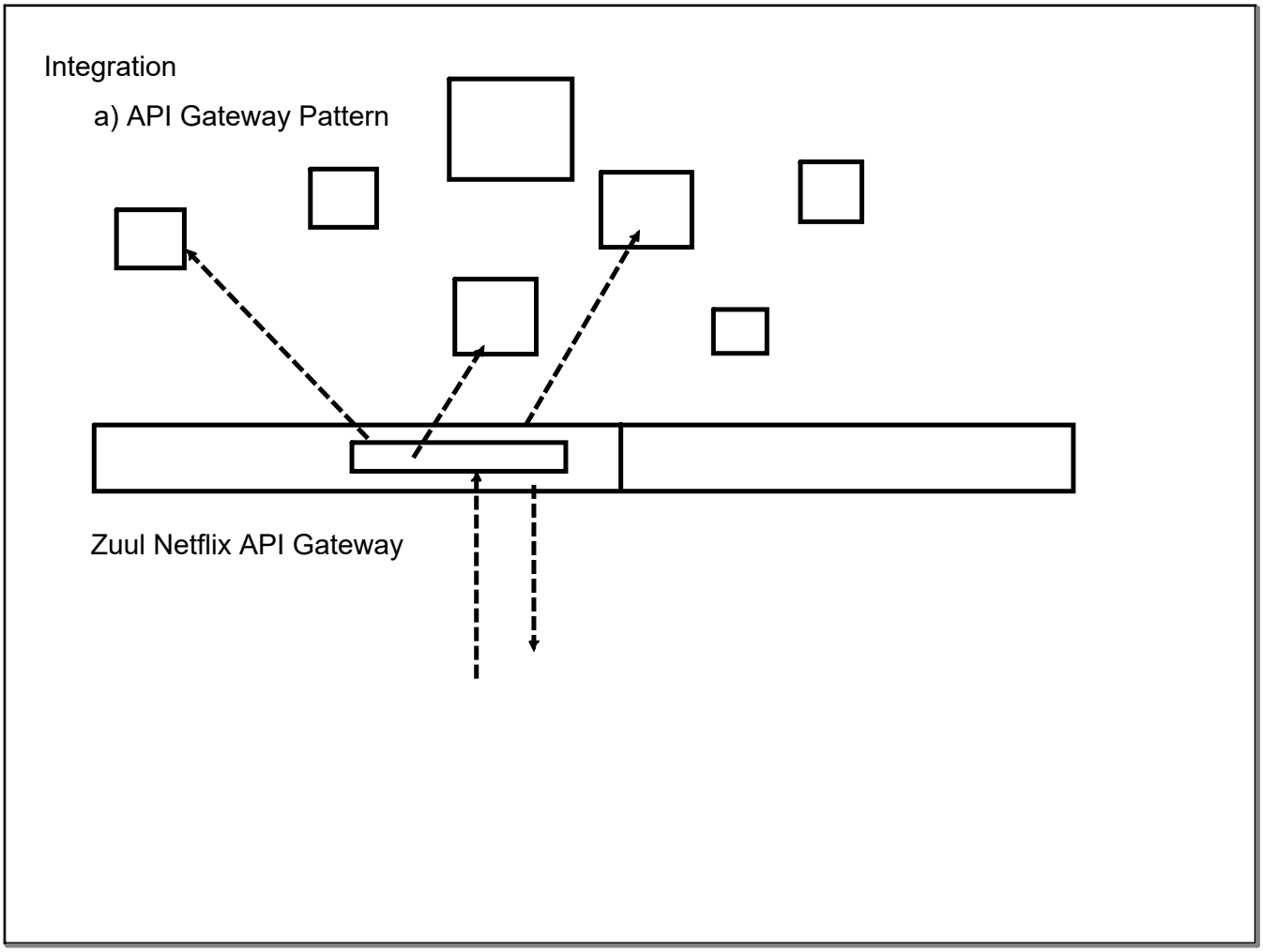
DDD : bounded context

sub-domains : BC with parent model

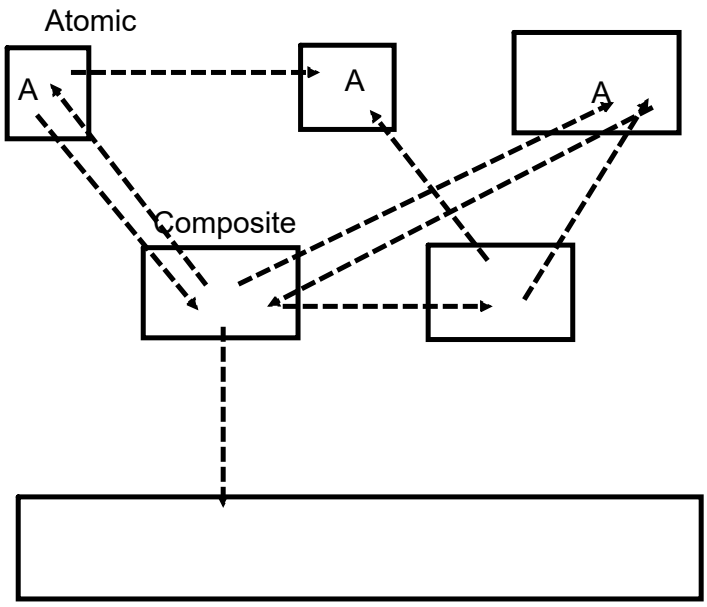
c) Strangler patterns

brownfield : converting monolith into MS

refactoring smaller req...

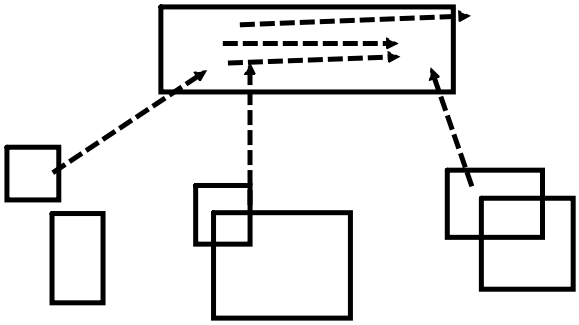

Integration
a) API Gateway Pattern
Zuul Netflix API Gateway

b) Aggregator Pattern



c) Client-side UI Composition

SPA : UI compositions



DataBase :

a) Ideal Pattern : Database Per Service

b) Shared Database :

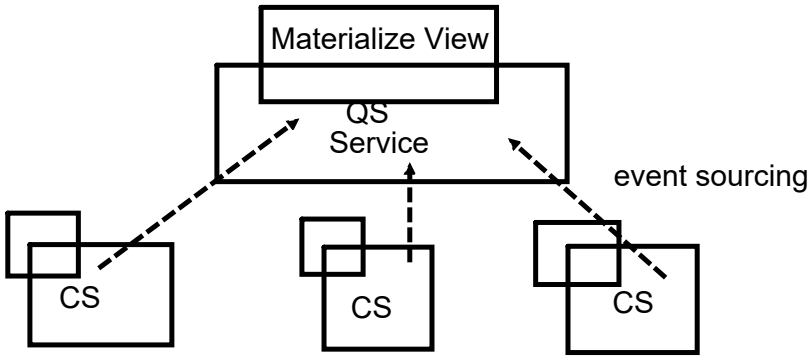
2-3 MS (DDD)

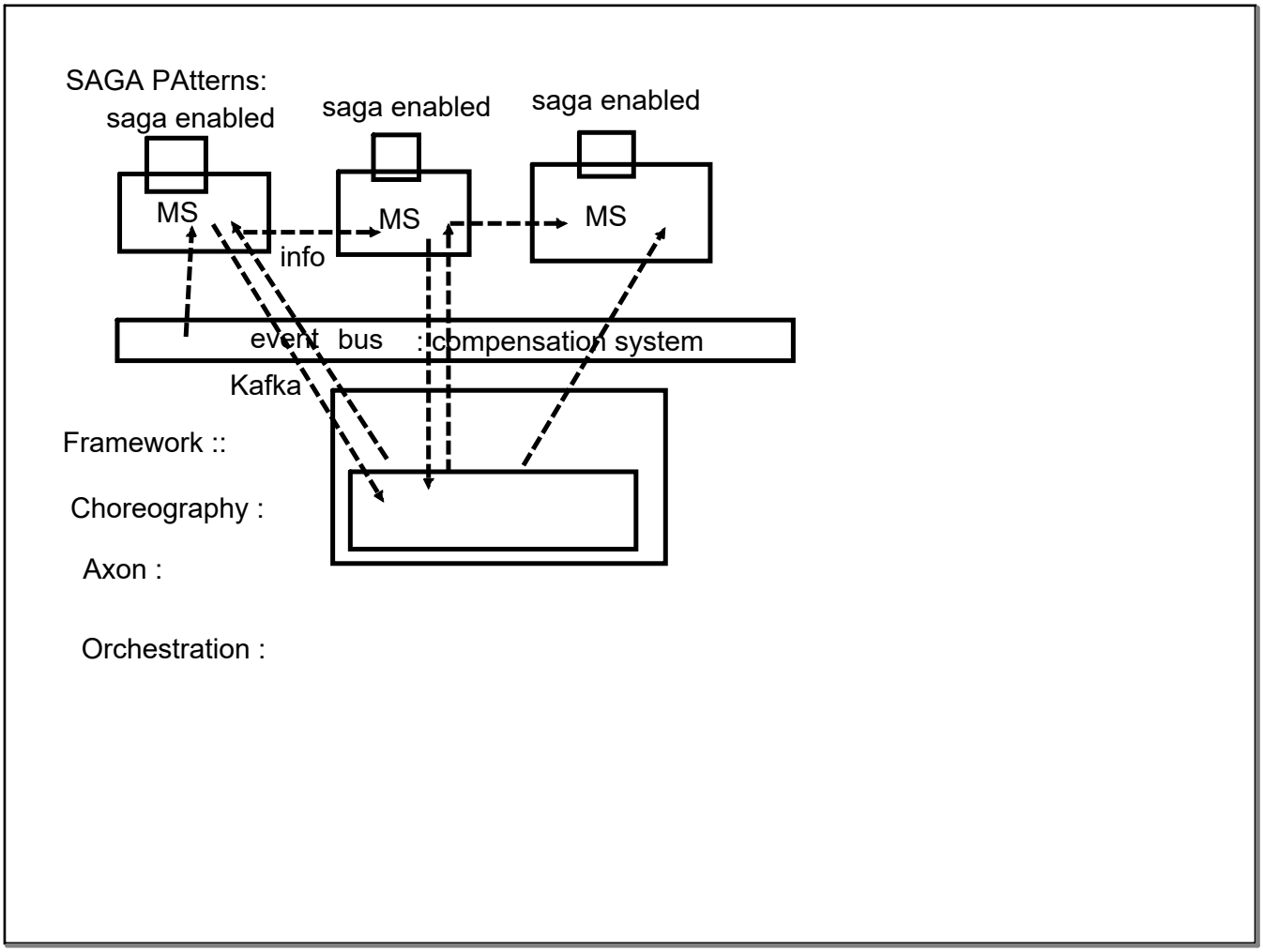
c) CQRS

Command Query Responsibility Segregation

Command Side (create/update/delete)

Query Side (retreive)




SAGA PAtterns:
saga enabled
saga enabled
saga enabled
MS
MS
MS
info
event bus
: compensation system
Kafka
Framework ::
Choreography :
Axon :
Orchestration :

Observability PAttern

a) Log Aggregation:

Centralized Logging pattern in place

track the log on request basis,

search

analysis

triggers alert

PCF : Pivotal Cloud Foundary

AWS Cloud Watch

b) Performance based

Centralized Metric service

push/pull model

=>NewRelics

=>Prometheus

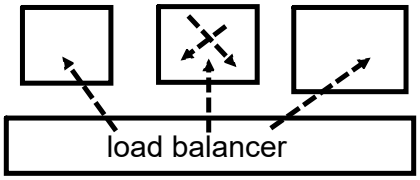
c) Distributed Tracing

system to track a request end-to-end

request id

Zipkin Server

Spring Cloud Slueth



d) Health Check

actuators /health :

Ribbon

Cross-Cutting Concerns

a) External Configuration

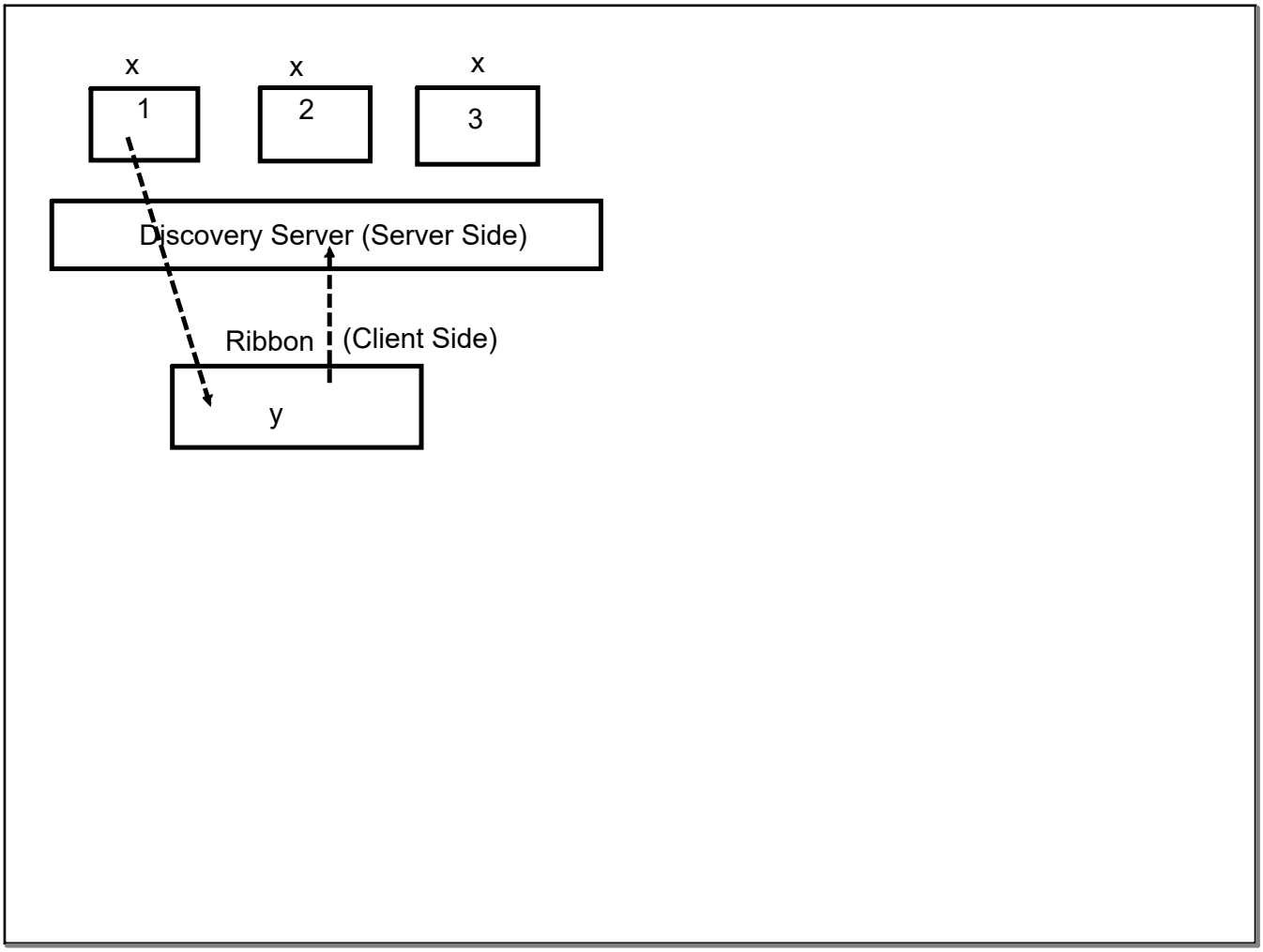
Spring Cloud Config Server

b) Service Discovery Pattern

all service shall register with registry system

Netflix Eureka Server

AWS ALB

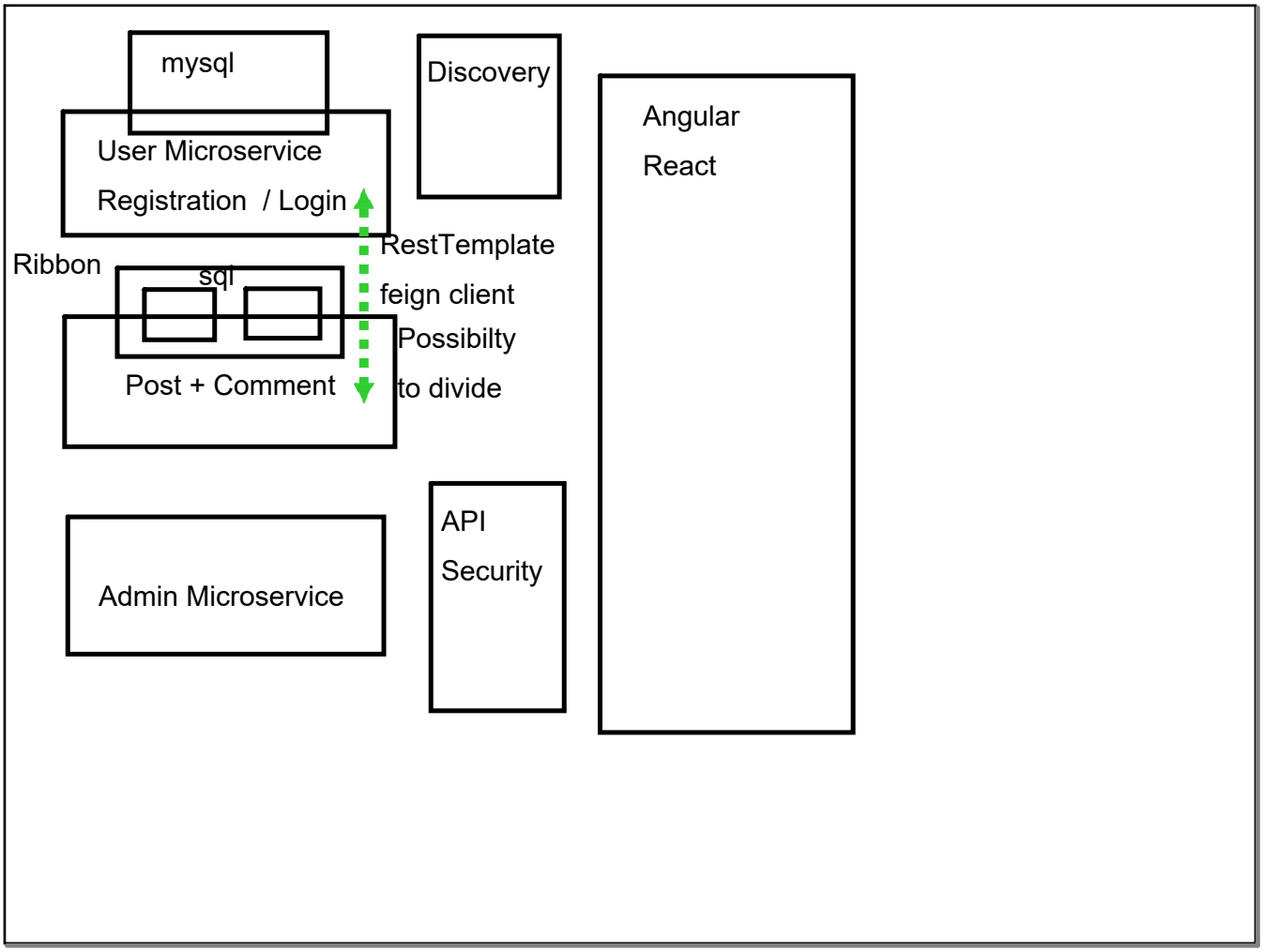

x
x
x
1
2
3
Discovery Server (Server Side)
Ribbon
(Client Side)
y

c) Circuit Breaker Pattern

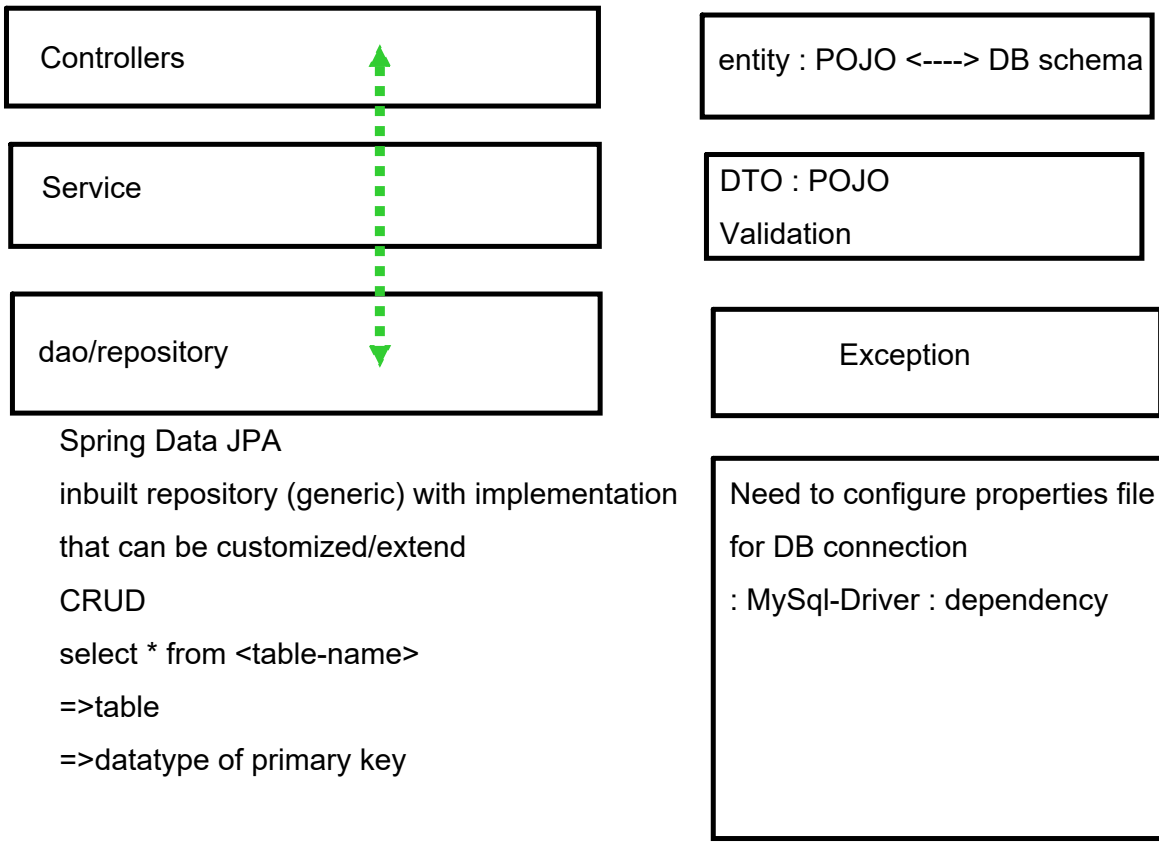
- threshold
- default response
- keep on trying

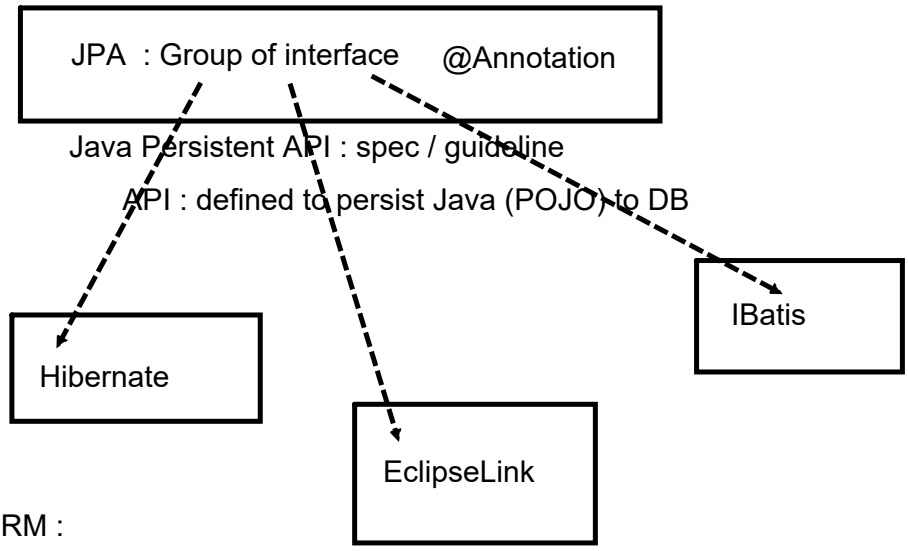
Netflix Hystrix

- 10 sec
- 5
- fallback


mysql
Discovery
Angular
React
User Microservice
Registration / Login
RestTemplate
Ribbon
sql
feign client
Possibilty
Post + Comment
to divide
API
Security
Admin Microservice

Spring Cloud :

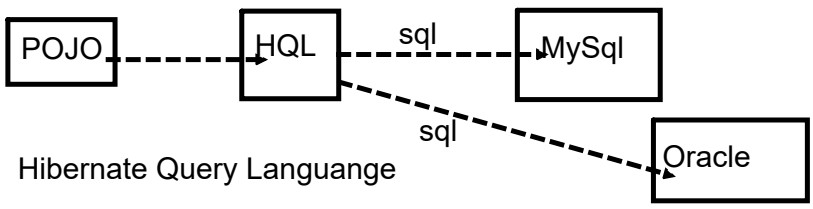




ORM :

behind the scene

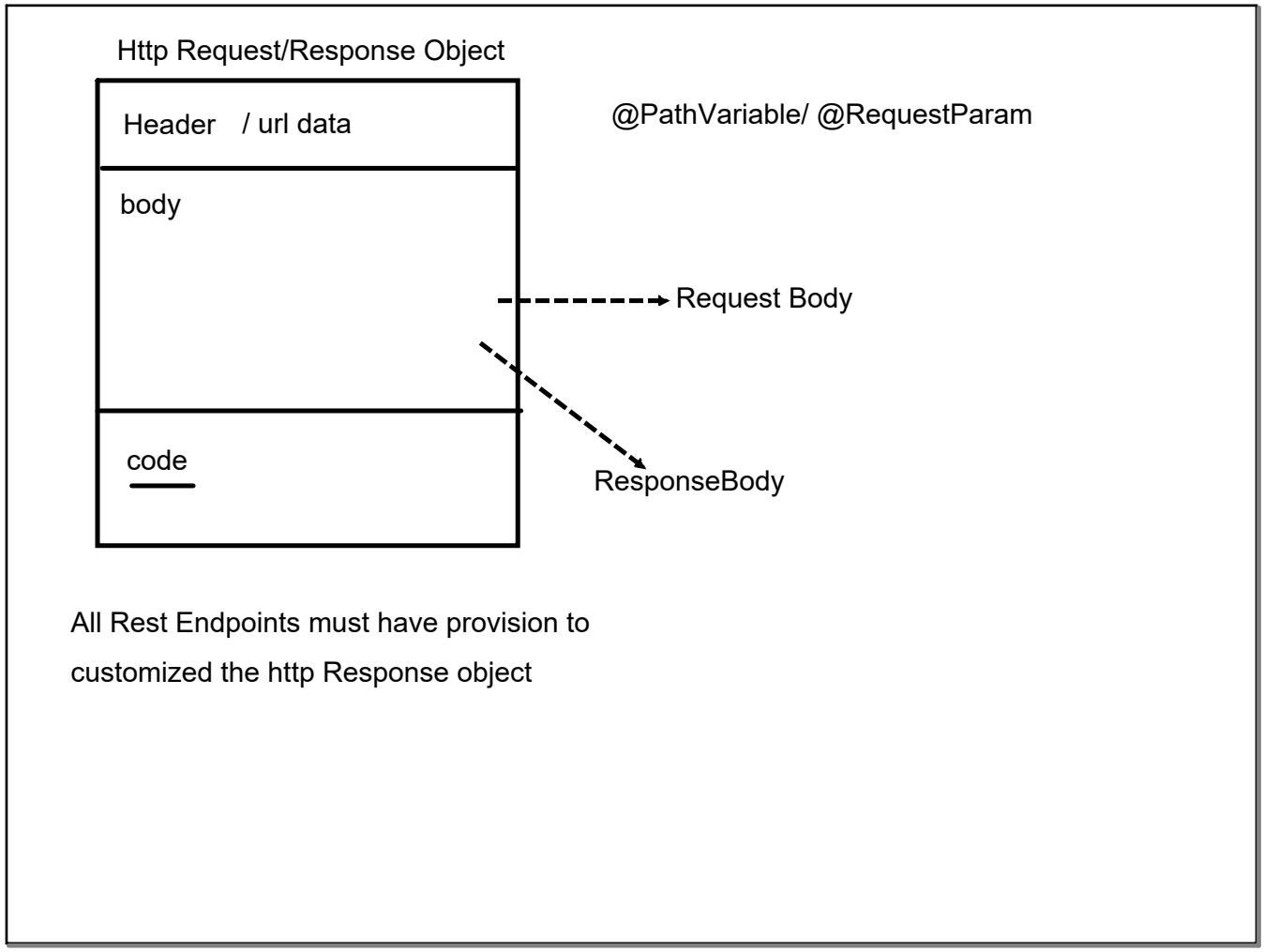
SQL queries would be created by Hibernate



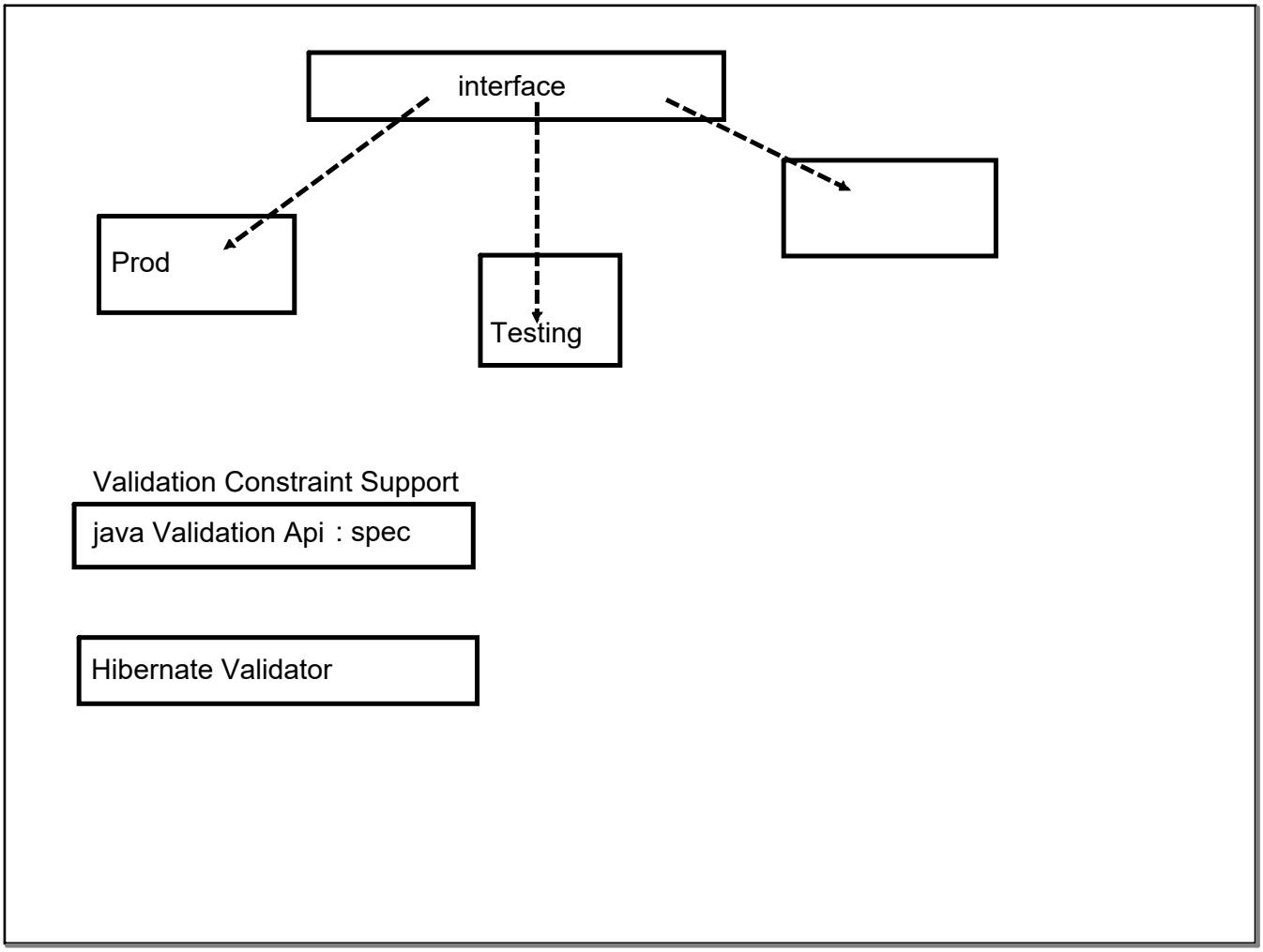
Hibernate Query Languange

Need to specify the dialect

c3p0 : dependency



All Rest Endpoints must have provision to customized the http Response object


interface
Prod
Testing
Validation Constraint Support
java Validation Api : spec
Hibernate Validator

Client Expecting : UserDetailDto (Success status)

Exception : UserExceptionDto (Failed status): throw an exception on client end of type mismatch

Server shall respond with appropriate status code

REst Client have provisions to check the status code