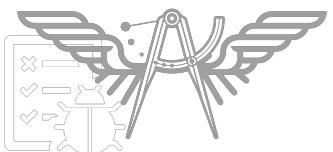


# Workshop with Spring Boot





Somkiat Puisungnoen

Somkiat Puisungnoen

Update Info 1 View Activity Log 10+ ...

Timeline About Friends 3,138 Photos More

When did you work at Opendream? X

... 22 Pending Items

Intro

Software Craftsmanship

Software Practitioner at สยามชัมนาณกิจ พ.ศ. 2556

Agile Practitioner and Technical at SPRINT3r

Post Photo/Video Live Video Life Event

What's on your mind?

Public Post

Somkiat Puisungnoen 15 mins · Bangkok · ⚙️

Java and Bigdata



Facebook somkiat.cc

Somkiat | Home | [Profile](#) [Messenger](#) [Pages](#) | ? ▾

Page Messages Notifications 3 Insights Publishing Tools Settings Help ▾

somkiat.cc  
@somkiat.cc

Home Posts Videos Photos

Like Liked Following Share ...

+ Add a Button



# Agenda

- RESTful API
- Java frameworks
- Building RESTful API with Spring Boot
- Spring Boot with Spring Data (JPA)
- Testing with Spring Boot



# REST



# **REST**

**RE**presentation **S**tate **T**ransfer

The style of software architecture behind  
**RESTful** services

Defined in 2000 by Roy Fielding

[https://www.ics.uci.edu/~fielding/pubs/dissertation/rest\\_arch\\_style.htm](https://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm)



# Goals

Scalability  
Generality of interfaces  
Independent deployment of components



# **RESTful service**



# REST Request Messages

RESTful request is typically in form of  
**Uniform Resource Identifiers (URI)**



# REST Request Messages

RESTful request is typically in form of  
**Uniform Resource Identifiers (URI)**

Structure of URI depend on specific service



# REST Request Messages

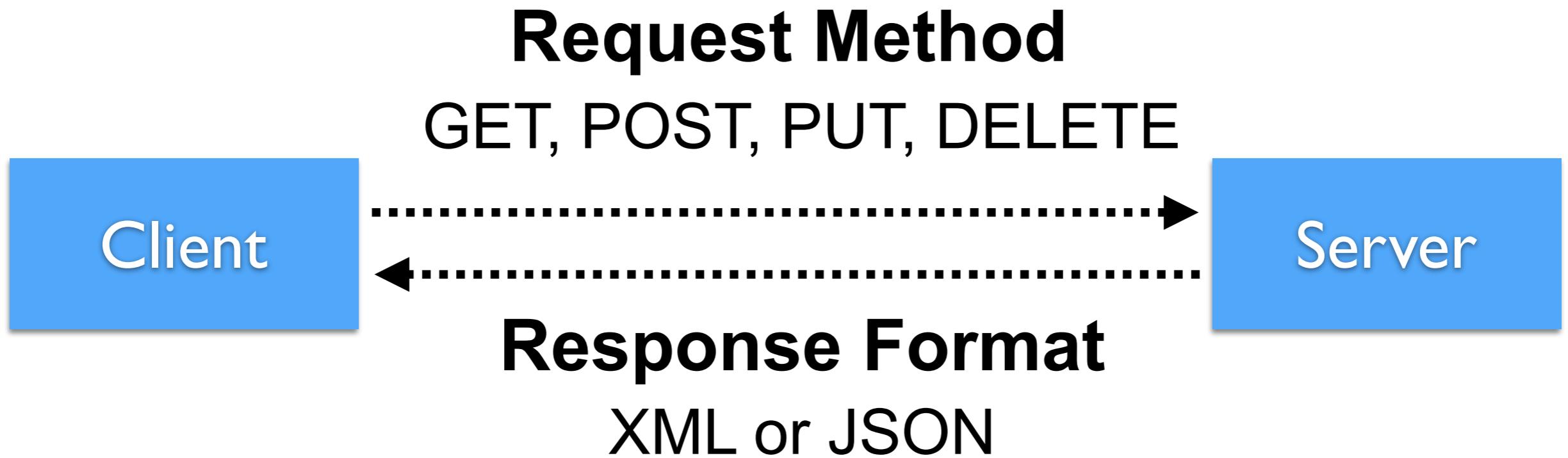
RESTful request is typically in form of  
**Uniform Resource Identifiers (URI)**

Structure of URI depend on specific service

Request can include parameter and data in  
body of request as XML, JSON etc.



# REST Request & Response



# HTTP Methods meaning

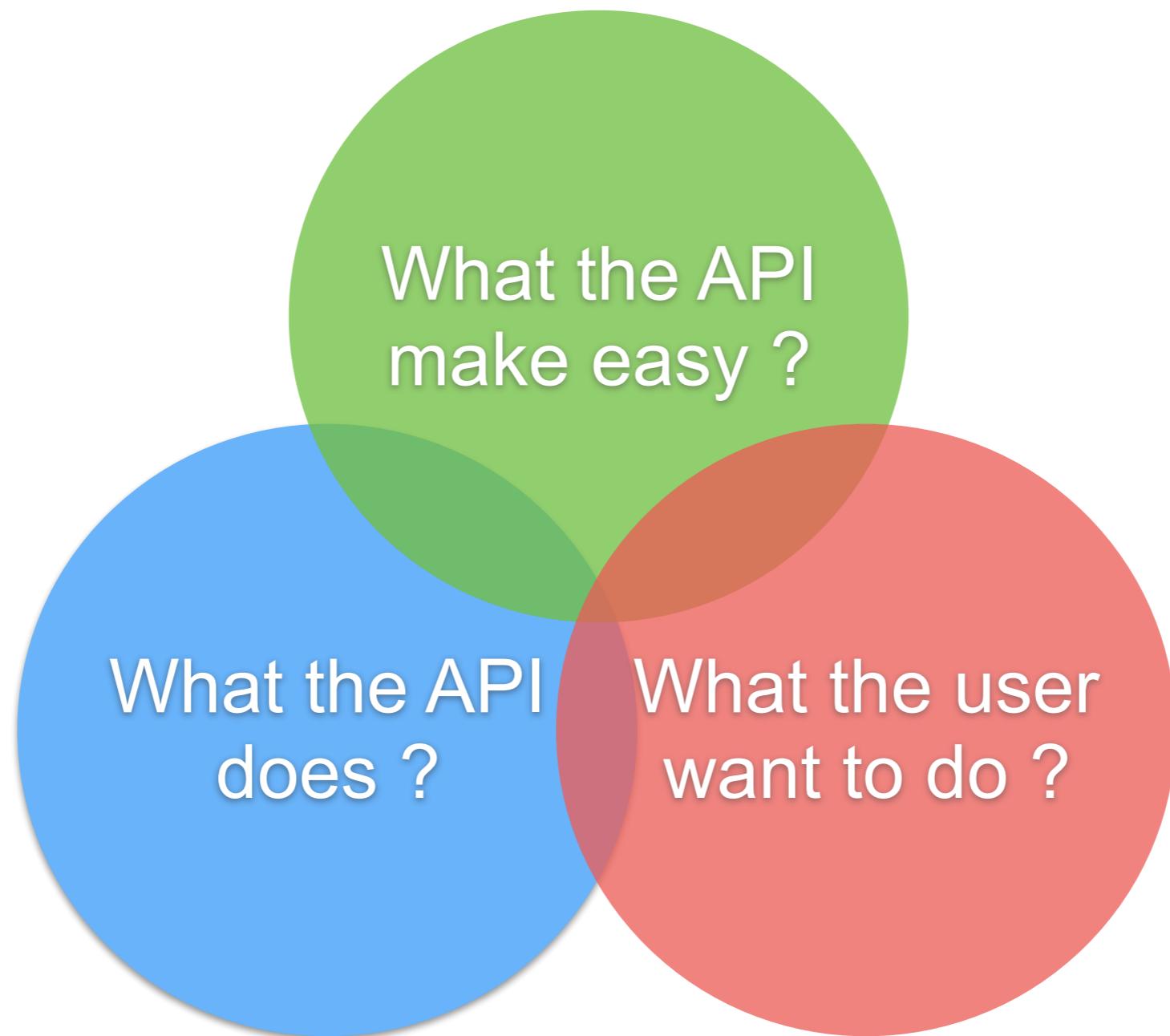
Method	Meaning
GET	Read data
POST	Create/Insert new data
PUT/PATCH	Update data or insert if a new id
DELETE	Delete data



# Response format ?



# Good APIs ?

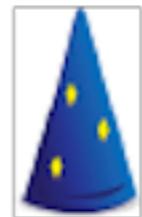


# Java Framework for RESTful



## unirest

Lightweight HTTP Request Client Libraries



## Dropwizard



## Restlet



## ACT.framework



Spring Boot

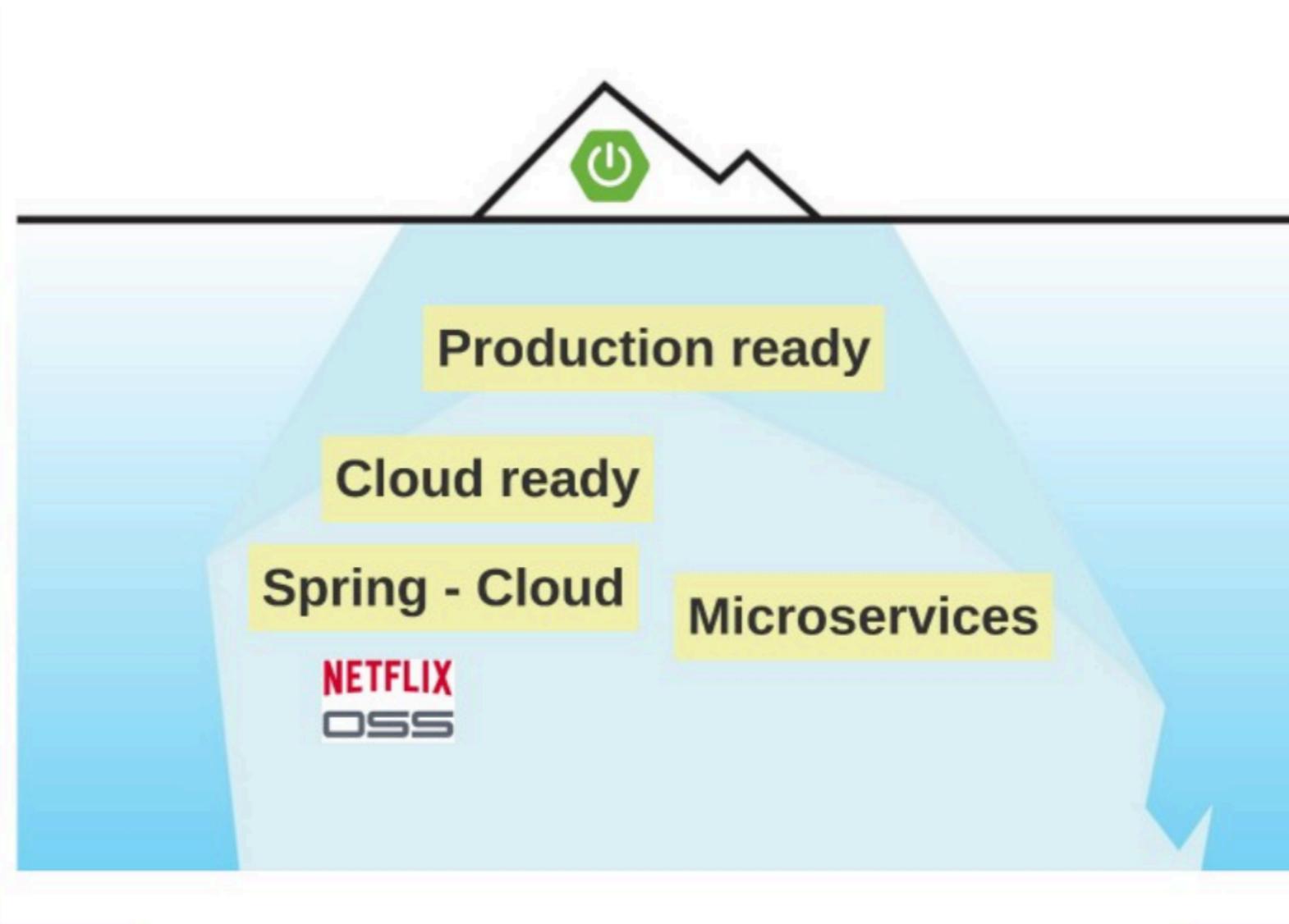
© 2017 - 2018 Siam Chamnankit Company Limited. All rights reserved.

# Hello Spring Boot 2.x



# Why ?

Application skeleton generator  
Reduce effort to add new technologies



# What ?

Embedded application server

Integration with tools/technologies (starter)

Production tools (monitoring, health check)

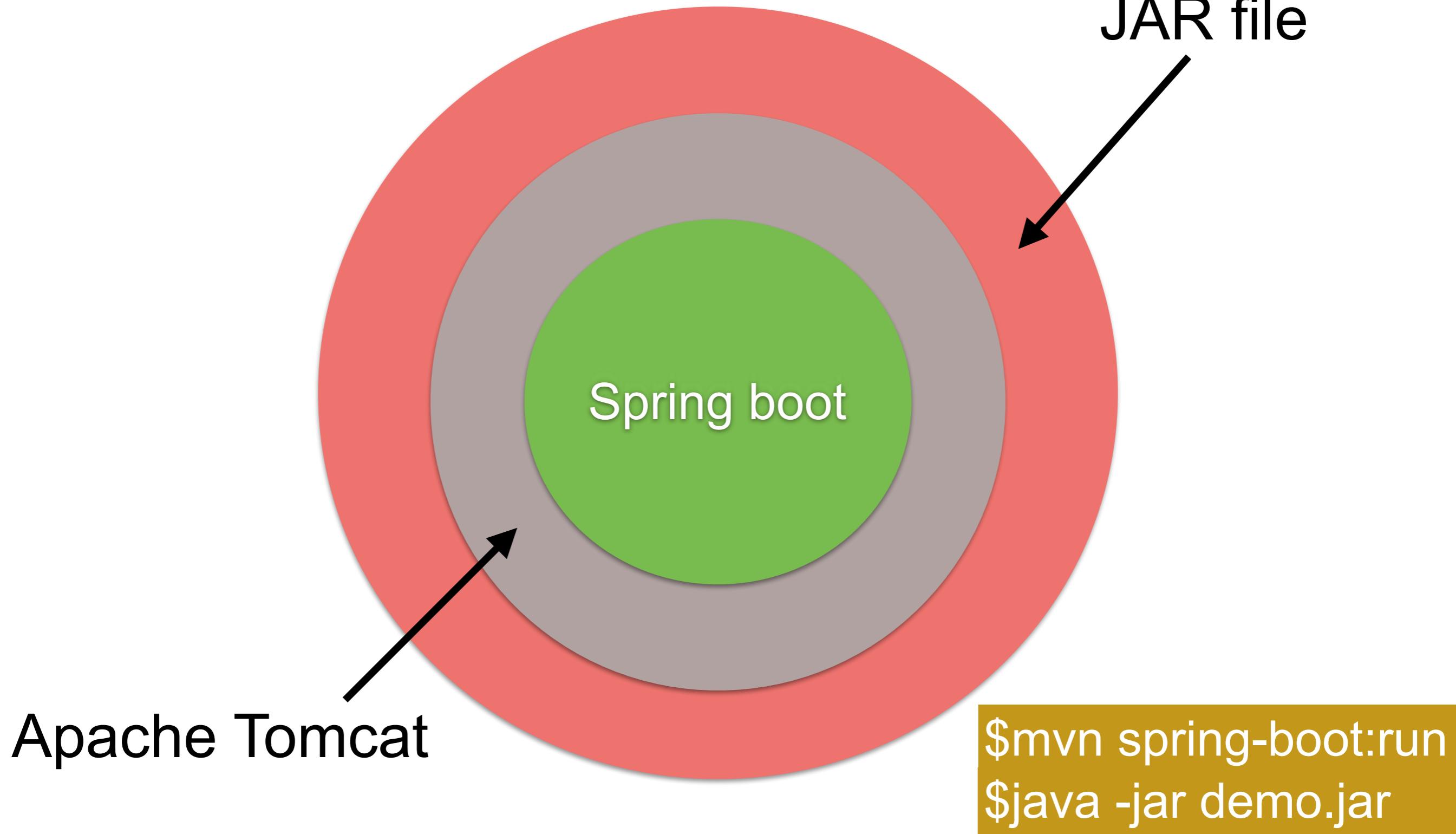
Configuration management

Dev tools

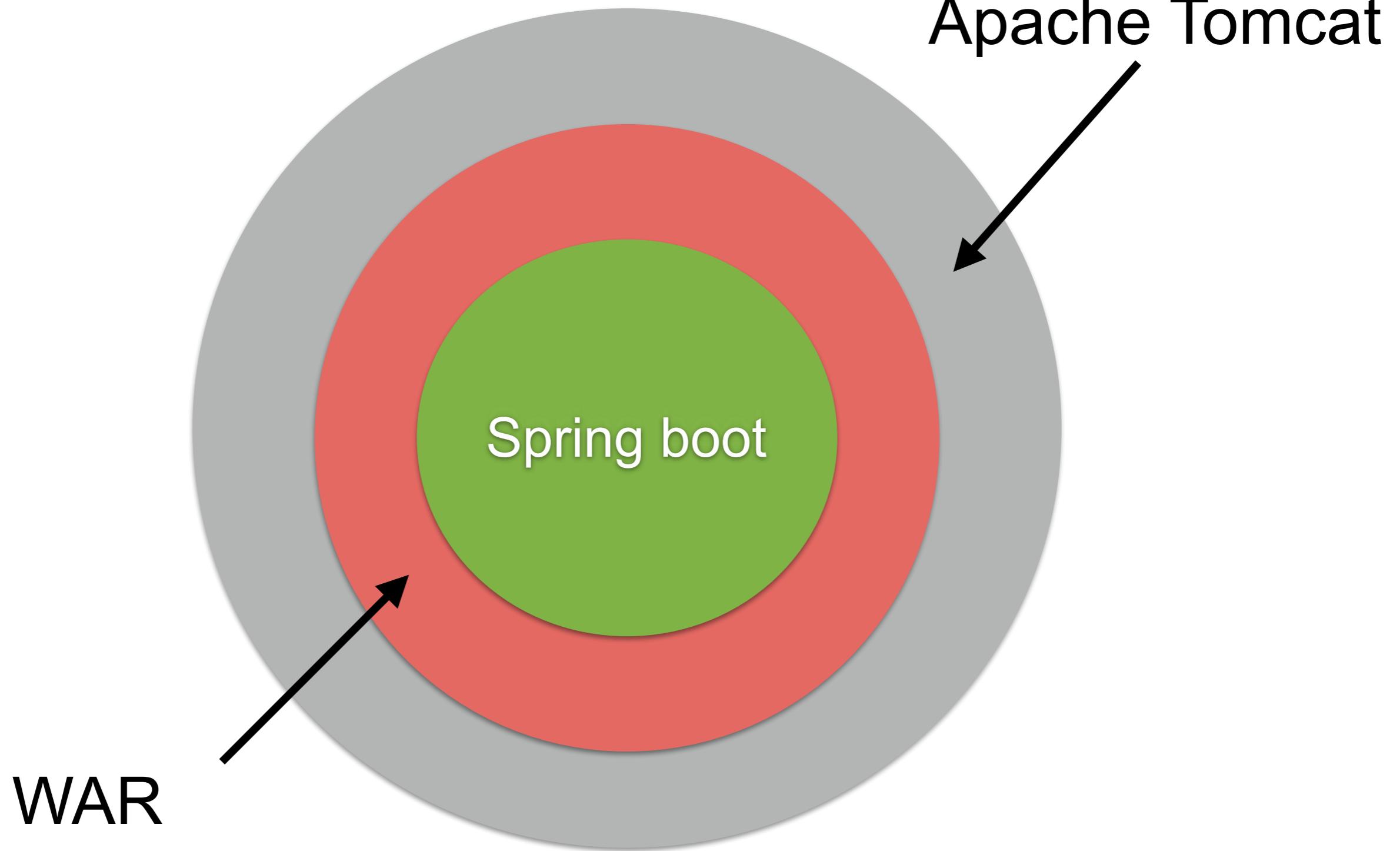
No source code generation, no XML



# How ?



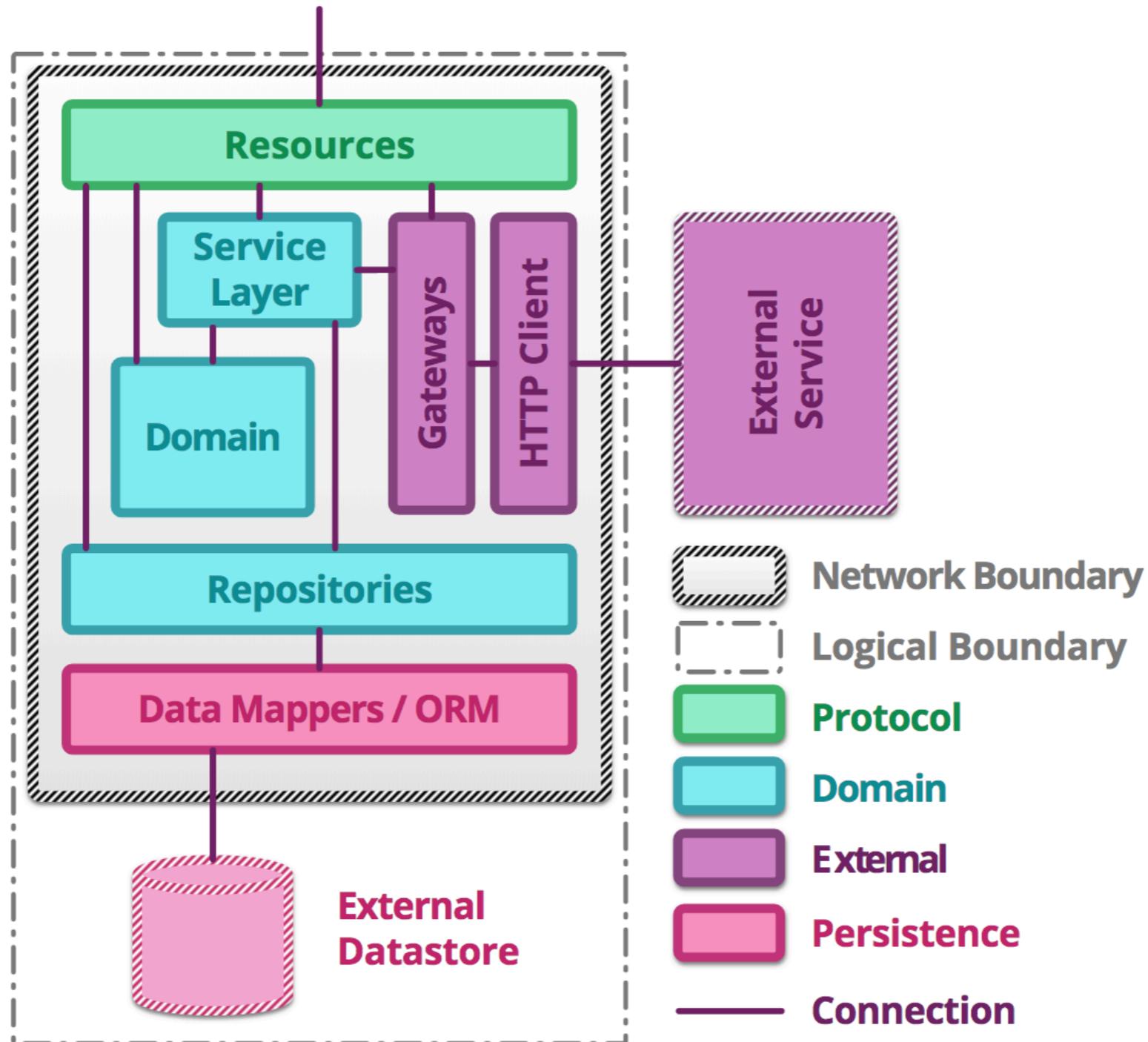
# How ?



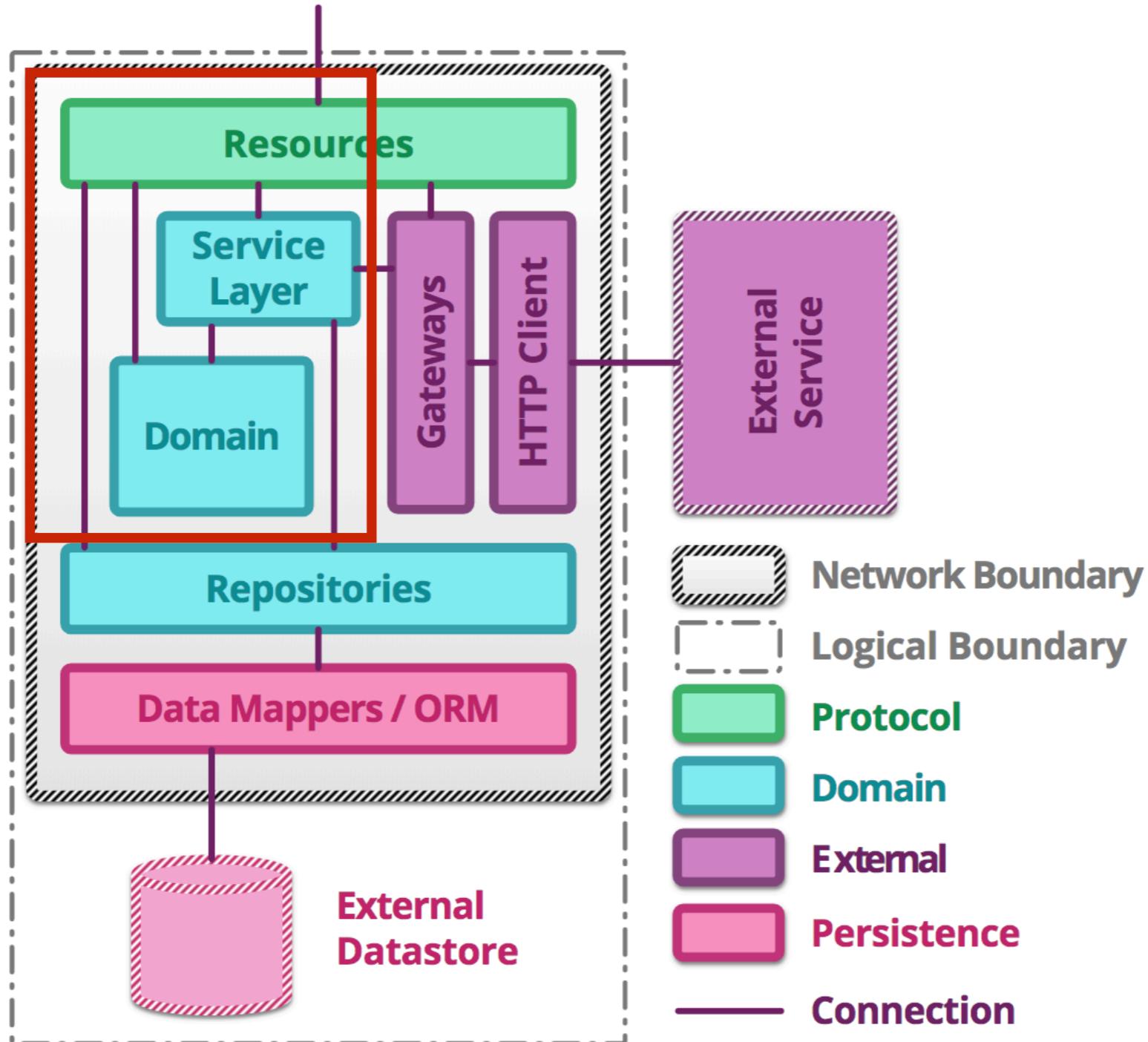
# **Building RESTful API with Spring Boot**



# Service Structure



# Service Structure



# Create project with Spring Initializr



# Spring Initializr

<https://start.spring.io/>

SPRING INITIALIZR bootstrap your application now

Generate a  with  and Spring Boot

**Project Metadata**

Artifact coordinates

Group

Artifact

**Dependencies**

Add Spring Boot Starters and dependencies to your application

Search for dependencies

Selected Dependencies

**Generate Project**

Don't know what to look for? Want more options? [Switch to the full version.](#)

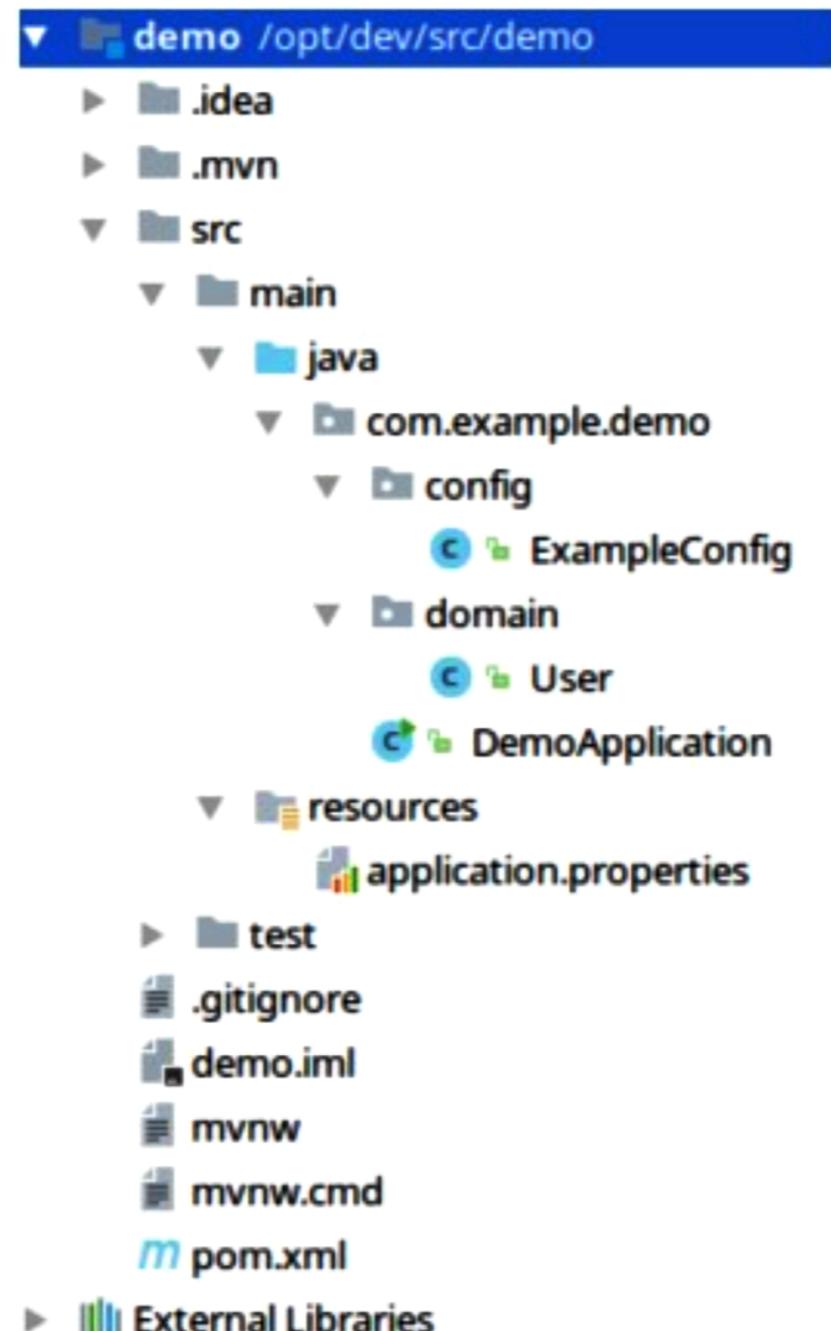
start.spring.io is powered by [Spring Initializr](#) and [Pivotal Web Services](#)



Spring Boot

© 2017 - 2018 Siam Chamnankit Company Limited. All rights reserved.

# Structure of Spring Boot



# Spring Boot main class

```
package com.example.demo;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

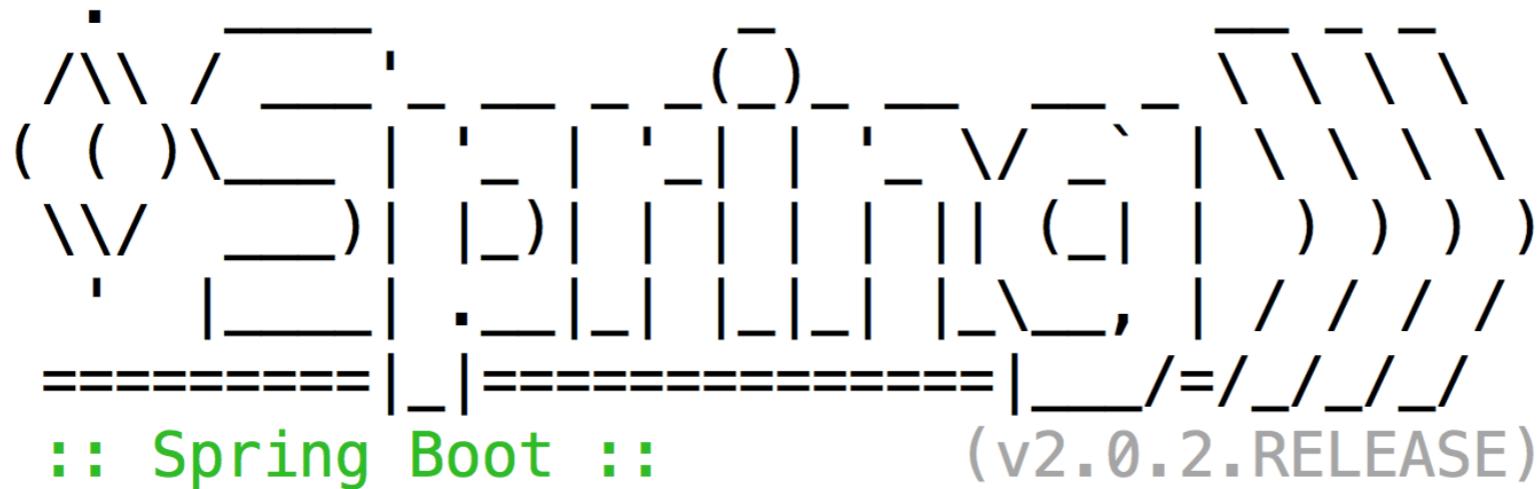
@SpringBootApplication
public class DemoApplication {

    public static void main(String[] args) {
        SpringApplication.run(DemoApplication.class, args);
    }
}
```



# Run project (Dev mode)

```
./mvnw spring-boot:run
```

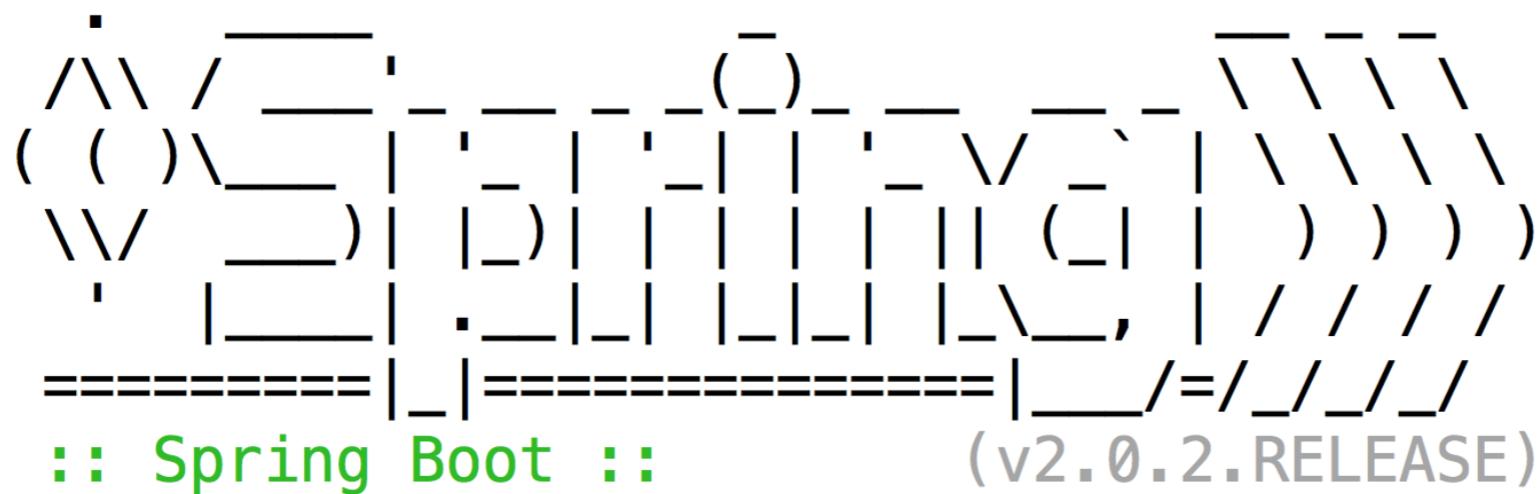


```
2018-06-07 13:03:30.412  INFO 12828 --- [  
  oApplication           : Starting DemoApplication on  
D 12828 (started by somkiat in /Users/somkiat/Down  
2018-06-07 13:03:30.418  INFO 12828 --- [  
  oApplication           : No active profile set, fall
```



# Run project (production mode)

```
$./mvnw package  
$java -jar target/<file name>.jar
```



```
2018-06-07 13:03:30.412  INFO 12828 --- [  
oApplication           : Starting DemoApplication on  
D 12828 (started by somkiat in /Users/somkiat/Down  
2018-06-07 13:03:30.418  INFO 12828 --- [  
oApplication           : No active profile set, fall
```



# Custom Banner



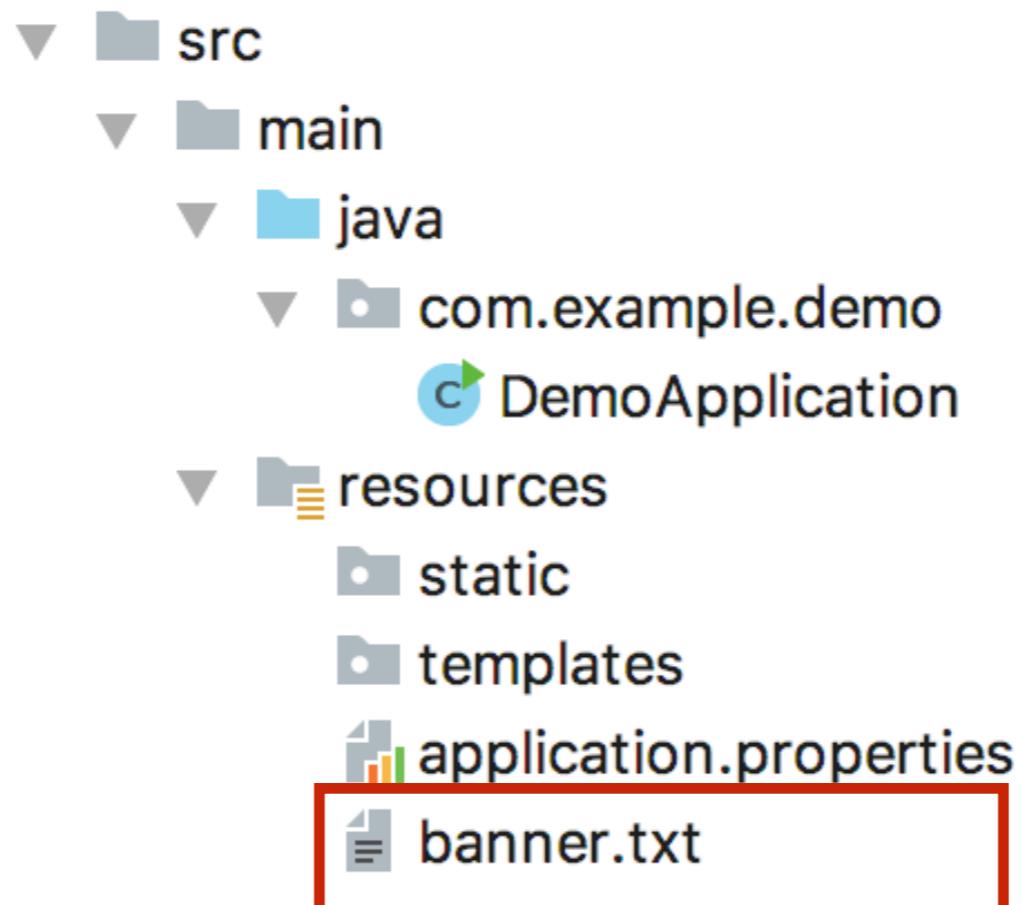
# Custom banner

```
2018-06-07 13:03:30.412  INFO 12828 --- [  
  oApplication : Starting DemoApplication on  
D 12828 (started by somkiat in /Users/somkiat/Down  
2018-06-07 13:03:30.418  INFO 12828 --- [  
  oApplication : No active profile set, fall
```



# Custom banner (1)

Create file banner.txt or banner.png in resources folder



# Custom banner (2)

```
@SpringBootApplication
public class DemoApplication {

    public static void main(String[] args) {
        ImageBanner banner = new ImageBanner(
            new ClassPathResource("try.png"));

        new SpringApplicationBuilder()
            .sources(DemoApplication.class)
            .banner(banner)
            .run();
    }
}
```



# Disable banner

```
@SpringBootApplication
public class DemoApplication {

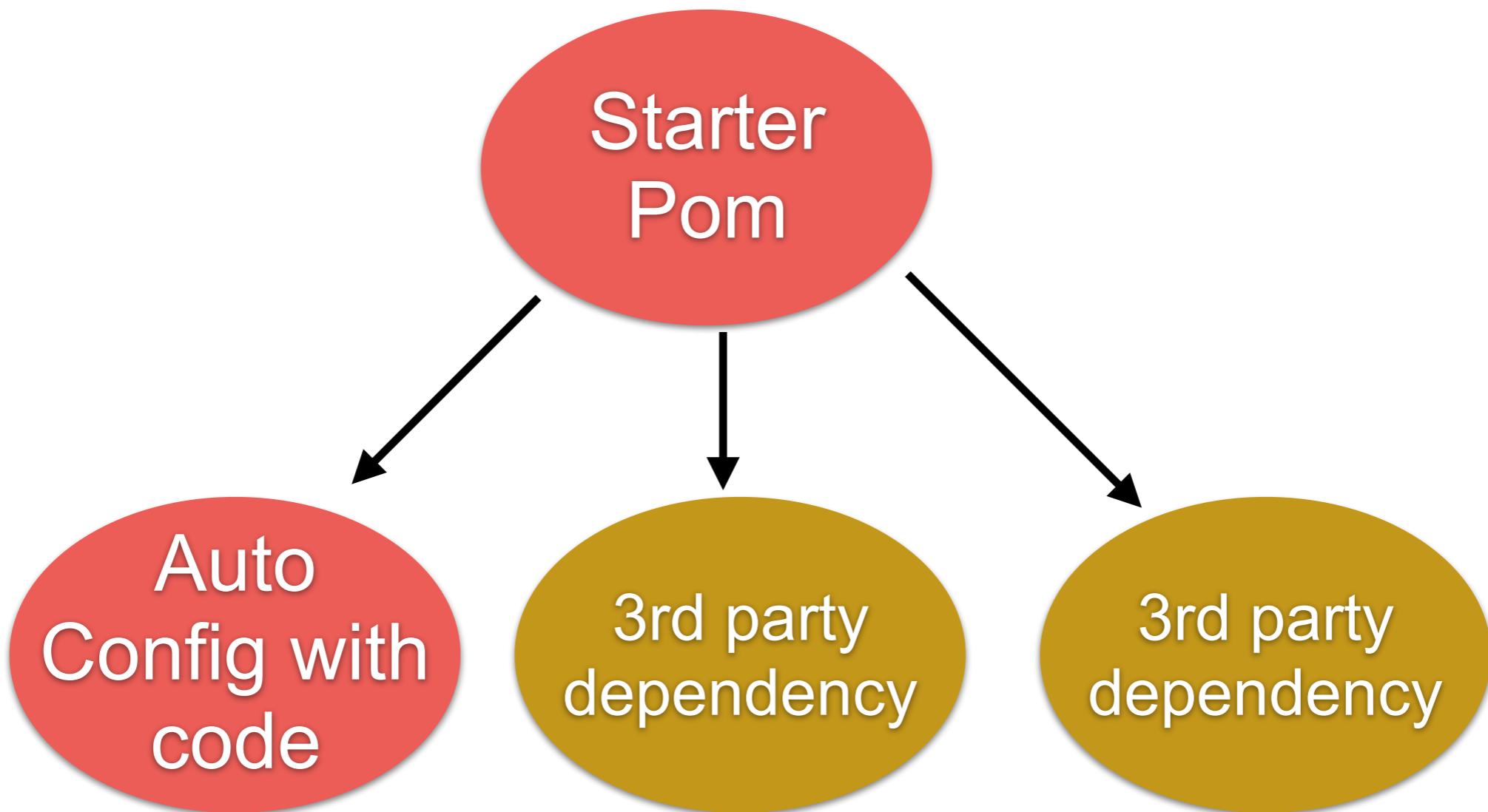
    public static void main(String[] args) {
        new SpringApplicationBuilder()
            .sources(DemoApplication.class)
            .bannerMode(Banner.Mode.OFF)
            .run(args);
    }
}
```



# Anatomy of Starter



# Anatomy of Starter



# Configuration management

Properties/XML/YAML

Config server (App, Spring cloud config, git)

17 ways !!!

<https://docs.spring.io/spring-boot/docs/current/reference/html/boot-features-external-config.html>



# Look at POM.xml



# POM.xml (1)

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
<modelVersion>4.0.0</modelVersion>

<groupId>hello</groupId>
<artifactId>hello</artifactId>
<version>1.0-SNAPSHOT</version>
<packaging>jar</packaging>

<parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>2.0.0.RELEASE</version>
    <relativePath/> 
</parent>

<properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    <project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>
    <java.version>1.8</java.version>
</properties>
```



# POM.xml (2)

```
<dependencies>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
    </dependency>

    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-test</artifactId>
        <scope>test</scope>
    </dependency>
</dependencies>

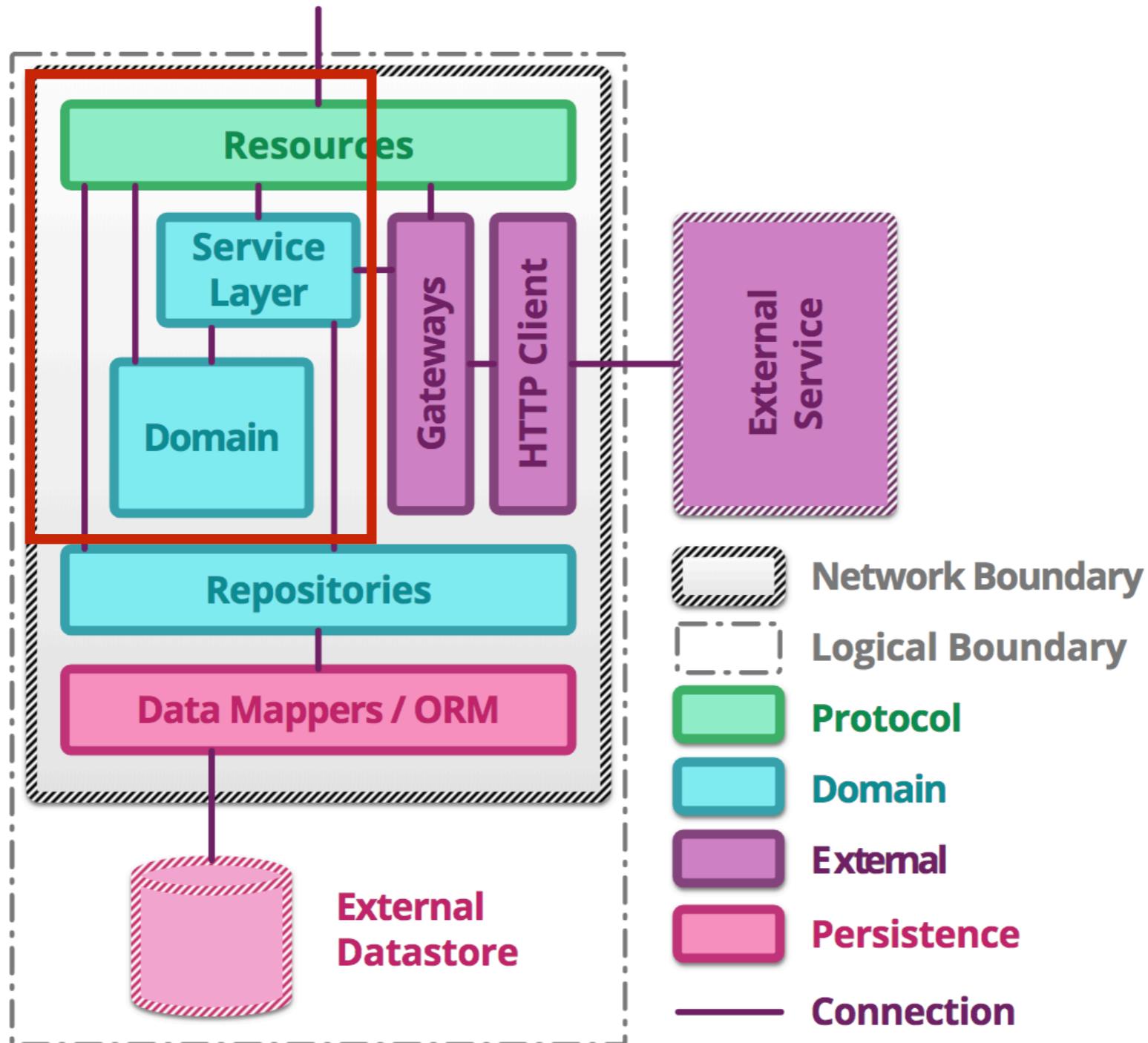
<build>
    <finalName>hello</finalName>
    <plugins>
        <plugin>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-maven-plugin</artifactId>
        </plugin>
    </plugins>
</build>
```



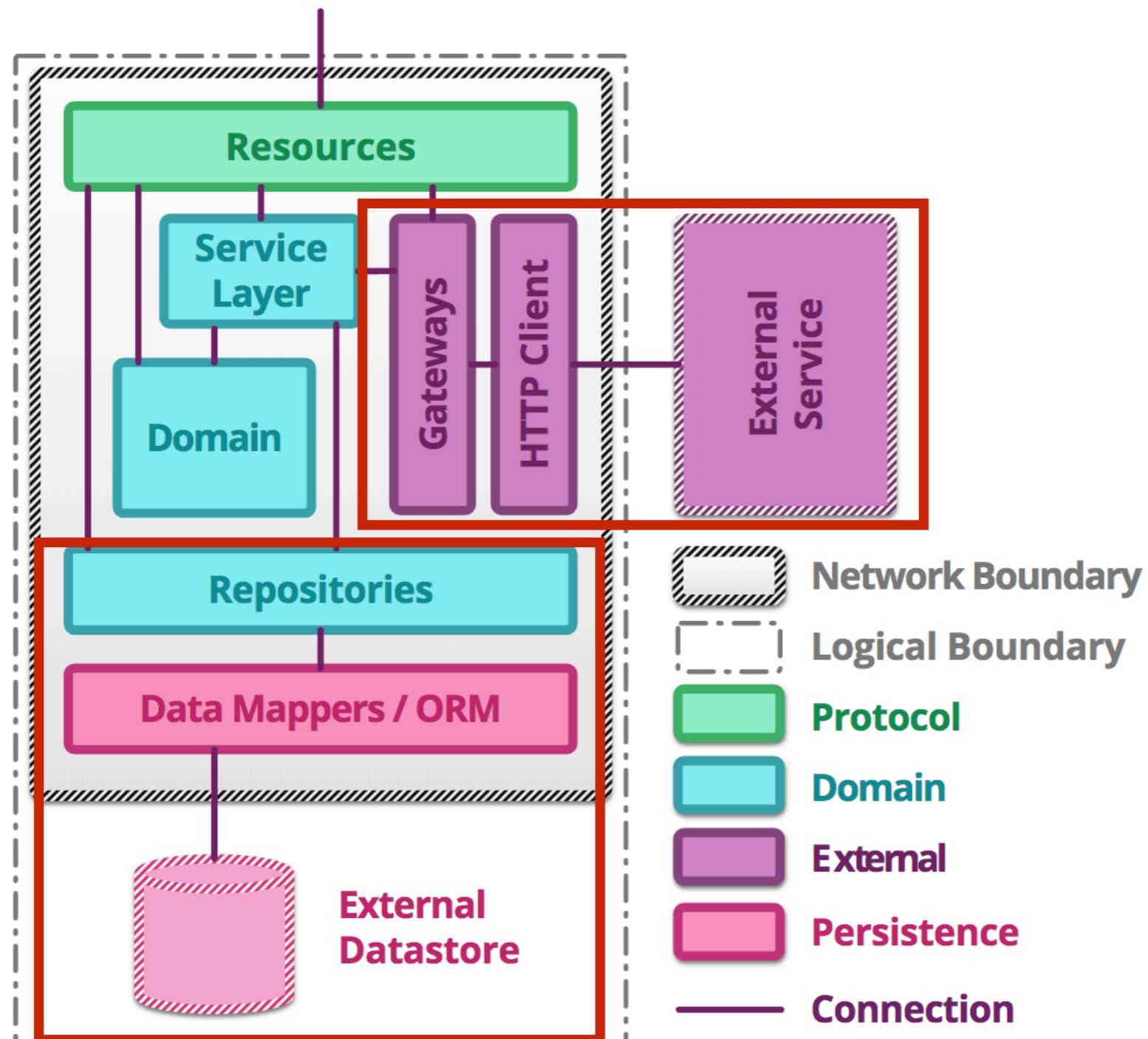
# Basic structure of Spring Boot



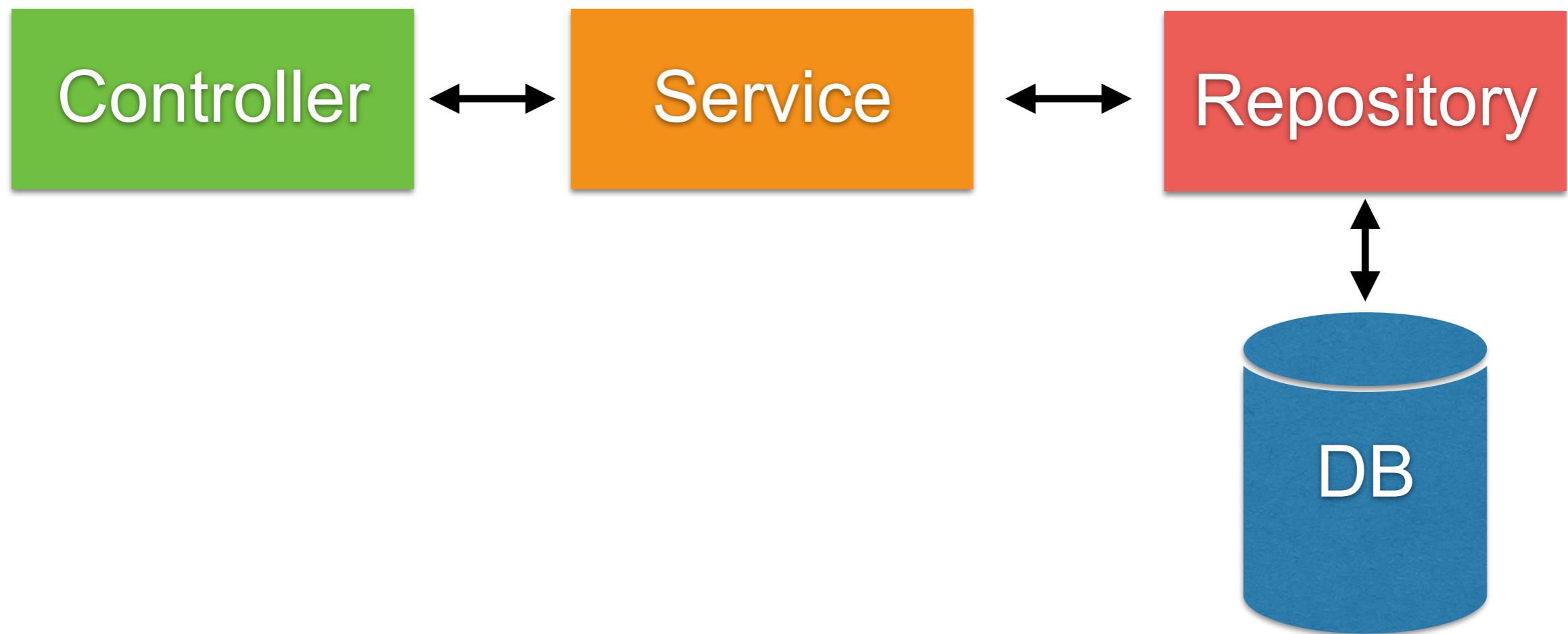
# Service Structure



# Service Structure



# Basic structure of Spring Boot



# Controller

Request and Response  
Validation input

Delegate to others classes such as service and repository



# Service

Control the flow of main business logic  
Manage database transaction  
Don't reuse service



# Repository

Manage data in data store such as RDBMS and  
NoSQL



# Gateway

Call external service via network such as  
WebServices and RESTful APIs



# Spring Boot Structure (1)

Separate by function/domain/feature

**feature1**

- controller
- service
- repository

**feature2**

- controller
- service
- repository

<https://docs.spring.io/spring-boot/docs/current/reference/html/using-boot-structuring-your-code.html>

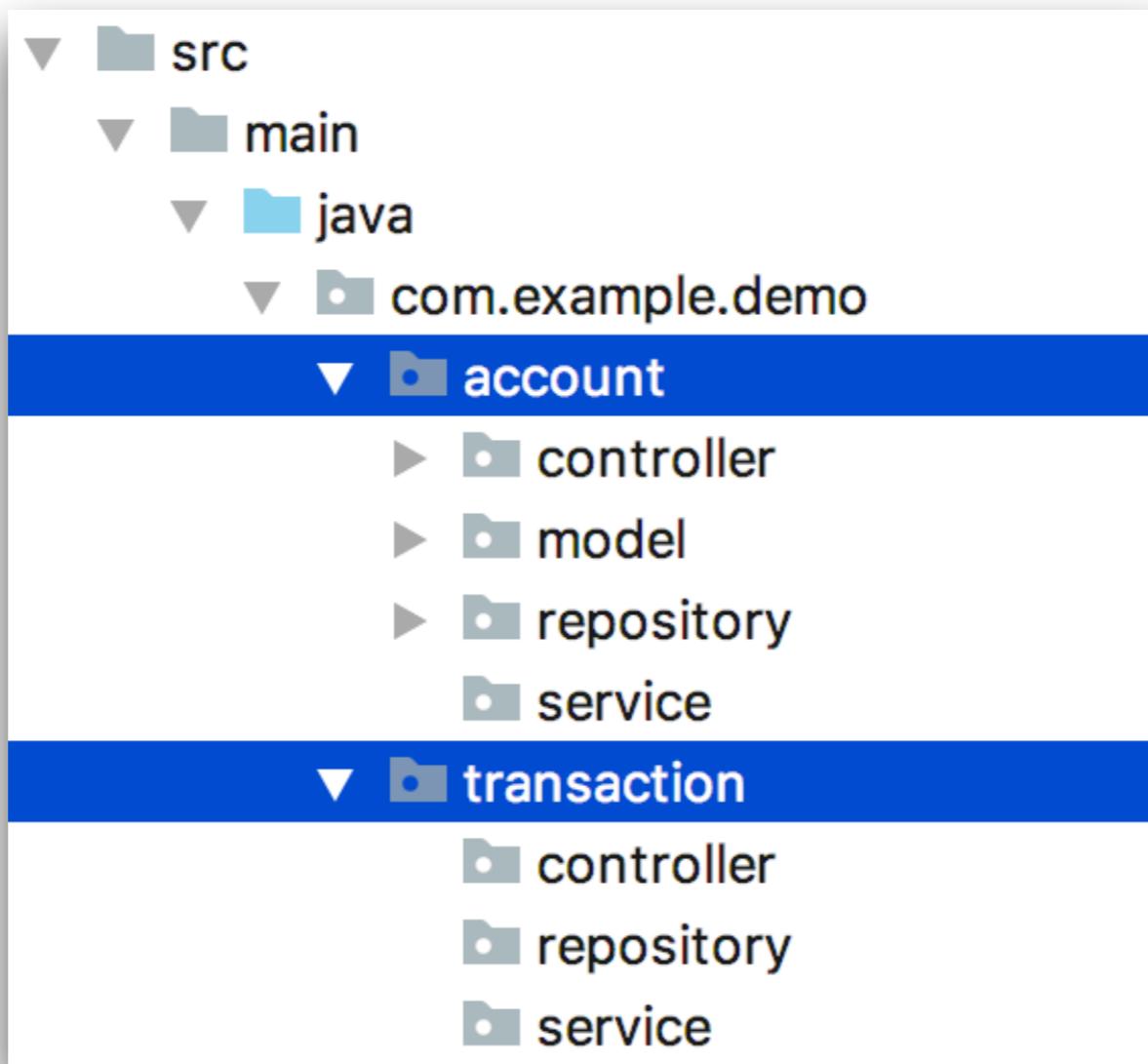


Spring Boot

© 2017 - 2018 Siam Chamnankit Company Limited. All rights reserved.

# Spring Boot Structure (2)

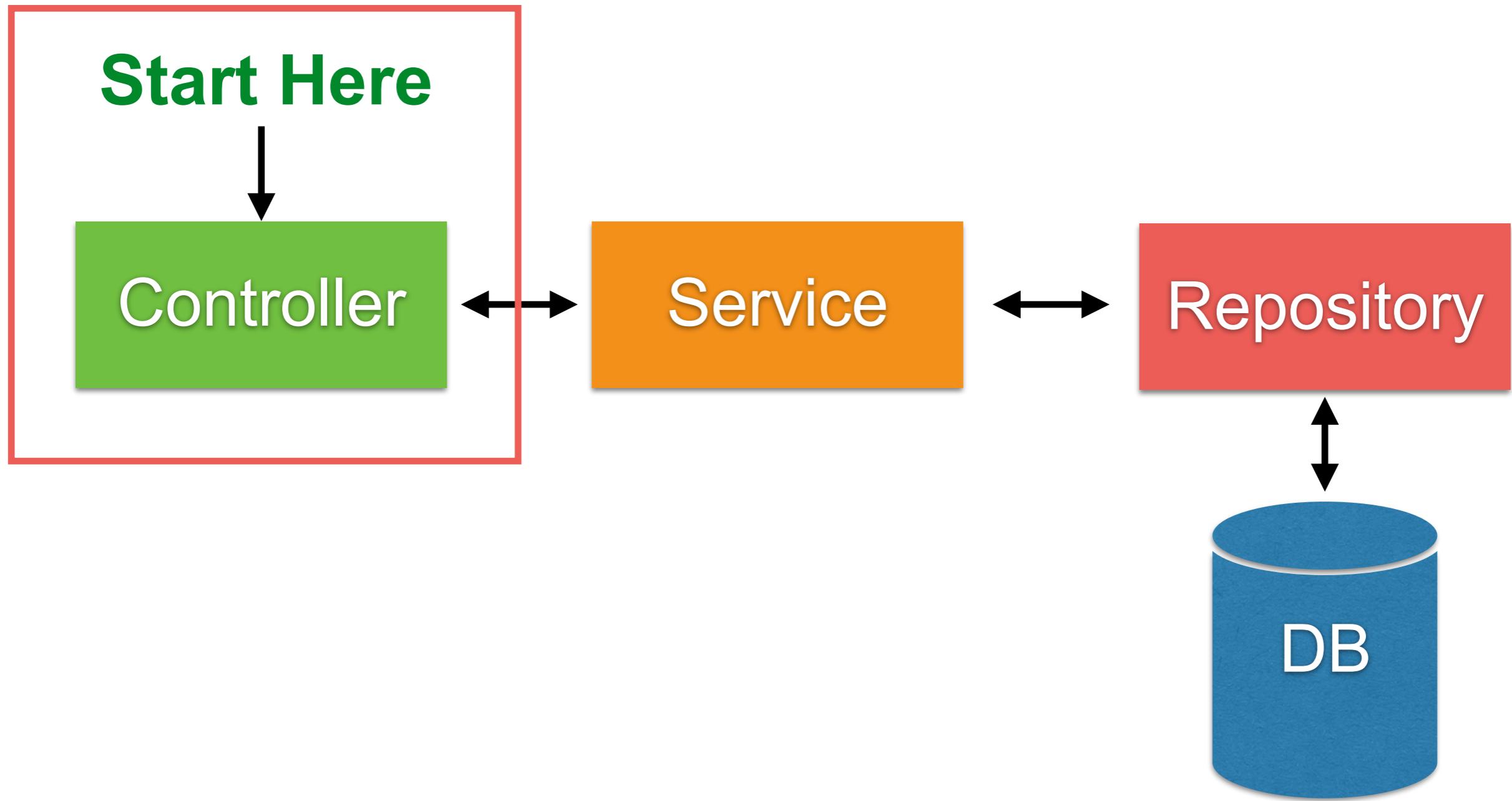
Separate by function/domain/feature



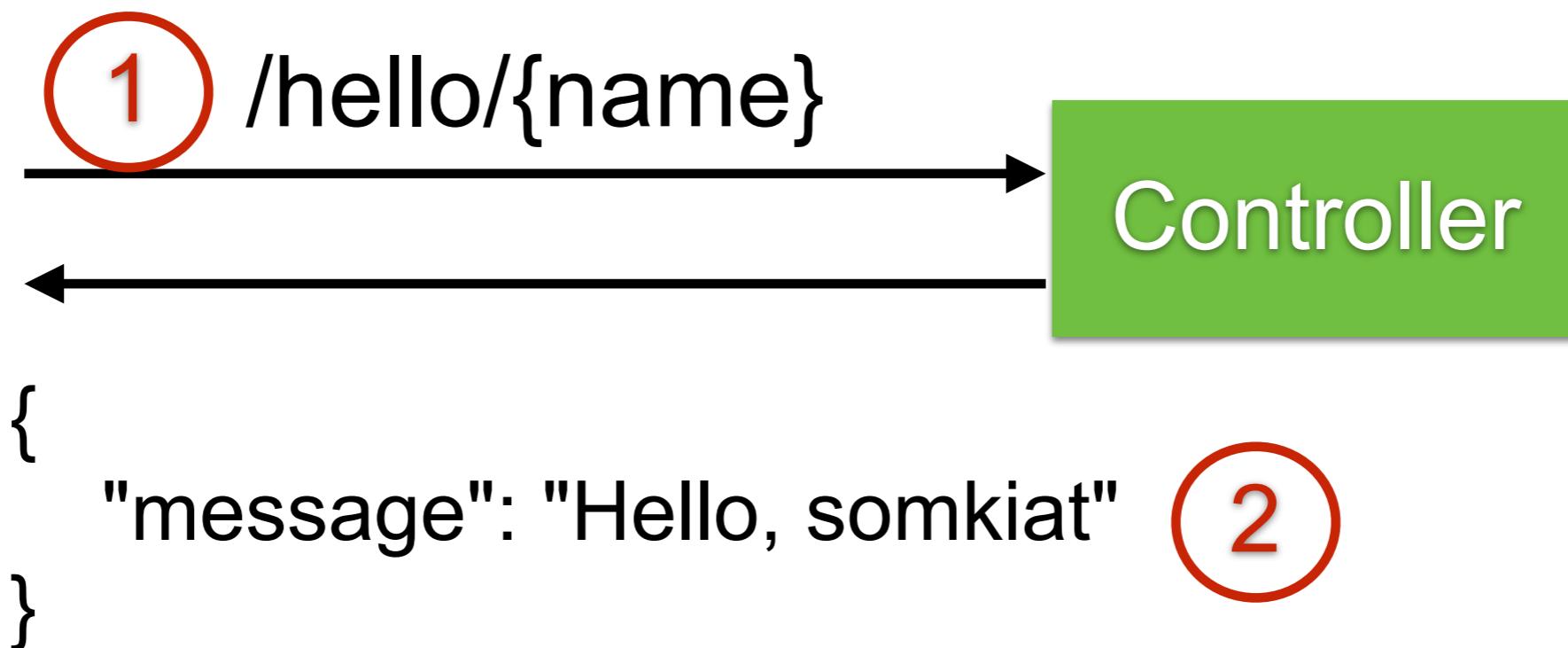
# Create first RESTful API



# Basic structure of Spring Boot



# Hello API



# 1. Create REST Controller

## HelloController.java

```
@RestController
public class HelloController {

    @GetMapping("/hello/{name}")
    public Hello sayHi(@PathVariable String name) {
        return new Hello("Hello, " + name);
    }

}
```



# 2. Create model class

## Hello.java

```
public class Hello {  
  
    private String message;  
  
    Hello(String message) {  
        this.message = message;  
    }  
  
    public String getMessage() {  
        return message;  
    }  
  
    public void setMessage(String message) {  
        this.message = message;  
    }  
}
```



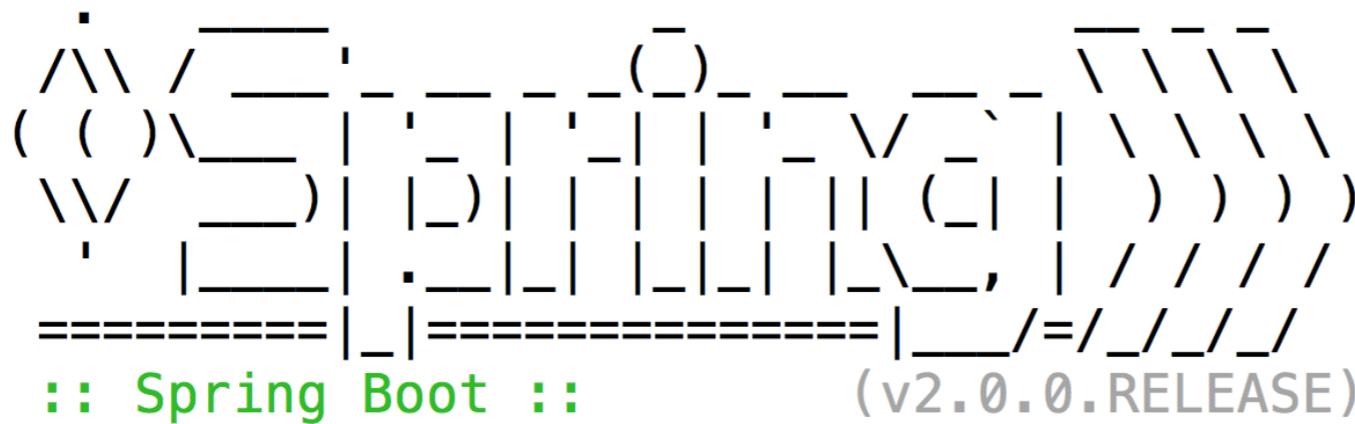
# 3. Compile and Packaging

\$mvnw clean package



# 4. Run

```
$java -jar target/hello.jar
```

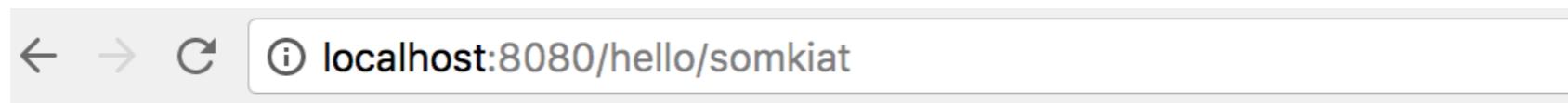


```
2018-03-05 23:37:18.018 INFO 30560 --- [           mair
          : Starting HelloApplication v1.0-SNAPSHOT
th PID 30560 (/Users/somkiat/data/slide/microservice/sl:
op/course-microservice/slide/4days-workshop/workshop/hei
  by somkiat in /Users/somkiat/data/slide/microservice/si
hop/course-microservice/slide/4days-workshop/workshop/he
2018-03-05 23:37:18.023 INFO 30560 --- [           mair
          : No active profile set, falling back to
2018-03-05 23:37:18.138 INFO 30560 --- [           mair
```



# 5. Open in browser

<http://localhost:8080/hello/somkiat>

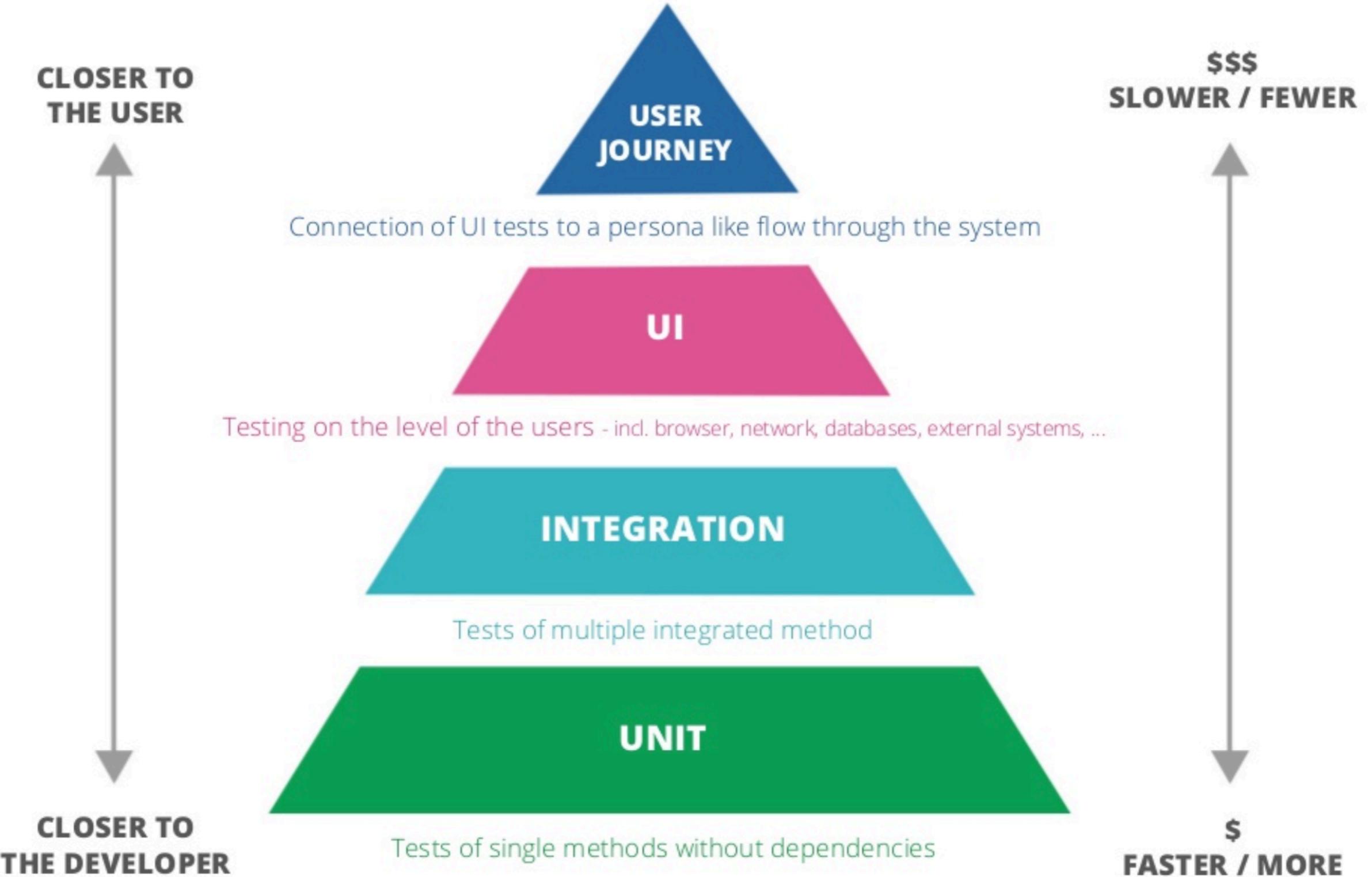


{ "message": "Hello somkiat" }



# How to test the Hello service ?





# Unit tests

How to use model ?

```
public class HelloTest {  
  
    @Test  
    public void success_to_create_model_with_constructor() {  
        Hello hello = new Hello("Somkiat");  
        assertEquals( expected: "Somkiat", hello.getMessage());  
    }  
  
}
```



# Controller testing

## How to testing with Spring Boot ?



# Testing in Spring Boot

`@SpringBootTest`  
`@WebMVCIntegrationTest`  
`@JsonTest`  
`@DataJpaTest`  
`@RestClientTest`

<https://docs.spring.io/spring-boot/docs/current/reference/html/boot-features-testing.html>



# Testing in Spring Boot

@SpringBootTest

@WebMVC Test

@JsonTest

@DataJpaTest

@RestClientTest

**Slice testing**



# SpringBootTest

```
@RunWith(SpringRunner.class)
@SpringBootTest(webEnvironment
        = SpringBootTest.WebEnvironment.RANDOM_PORT)
public class HelloControllerTest {

    @Autowired
    private TestRestTemplate testRestTemplate;

    @Test
    public void sayHi() {
        // Action :: Call controller
        Hello actualResult
            = testRestTemplate.getForObject("/hello/somkiat",
                                            Hello.class);

        // Assertion :: Check result with expected result
        assertEquals("Hello, somkiat", actualResult.getMessage());
    }
}
```



# Compile with testing

\$mvn clean test

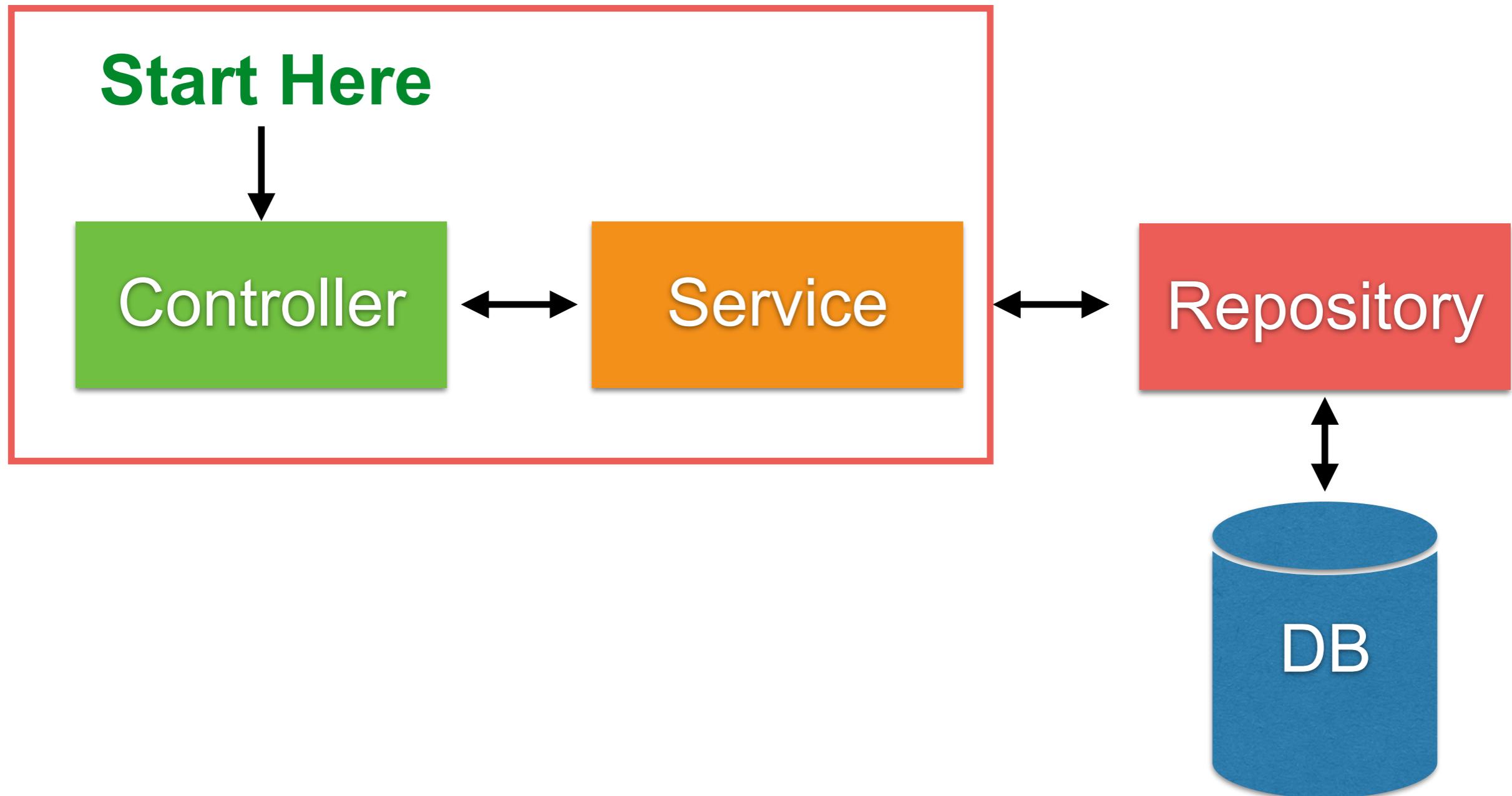
```
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 2, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO]
```



# Move business logic to service

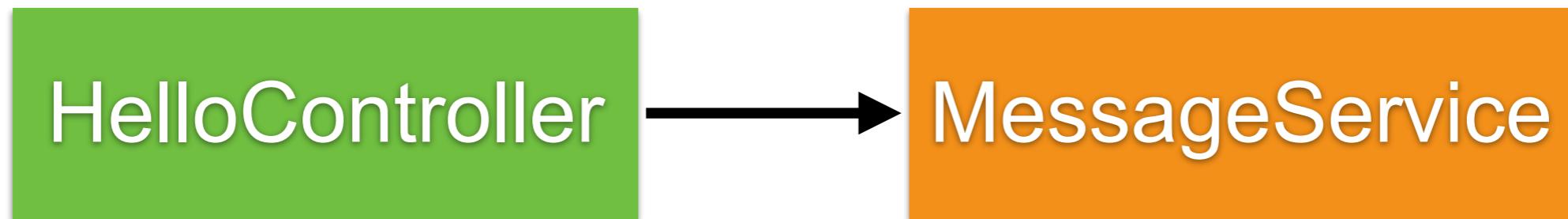


# Working with service



# Move business logic to service

MessageService.java



# MessageService.java

```
@Service  
public class MessageService {  
  
    public String concat(String name) {  
        return "Hello, " + name;  
    }  
  
}
```



# Edit HelloController.java

Call method from service

```
@Autowired  
private MessageService messageService;
```

```
@GetMapping("/hello/{name}")  
public HelloResponse  
    sayHi(@PathVariable String name) {  
        String result = messageService.concat(name);  
        return new HelloResponse(result);  
    }
```



# **Run all tests !!**

**\$mvnw clean test**



# Fix fail test cases



# Testing controller with service

## Working with @WebMvcTest

```
@Autowired  
private MockMvc mockMvc;
```

```
@MockBean  
private MessageService messageService; 1
```

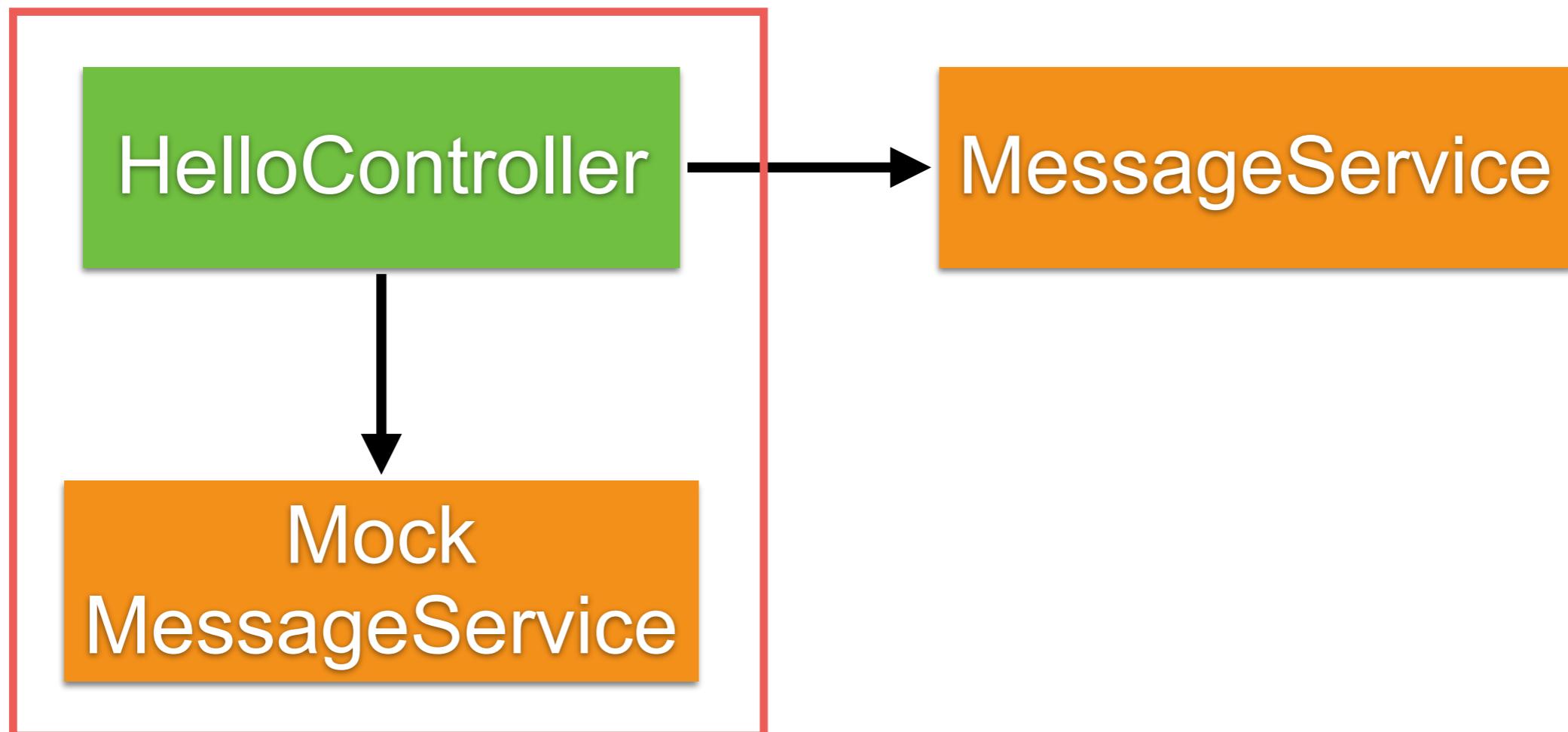
```
private JacksonTester<HelloResponse> tester;
```

```
@Before  
public void initial() {  
    JacksonTester.initFields(this, new ObjectMapper());  
    given(messageService.concat("somkiat"))  
        .willReturn("Hello, somkiat"); 2  
}
```



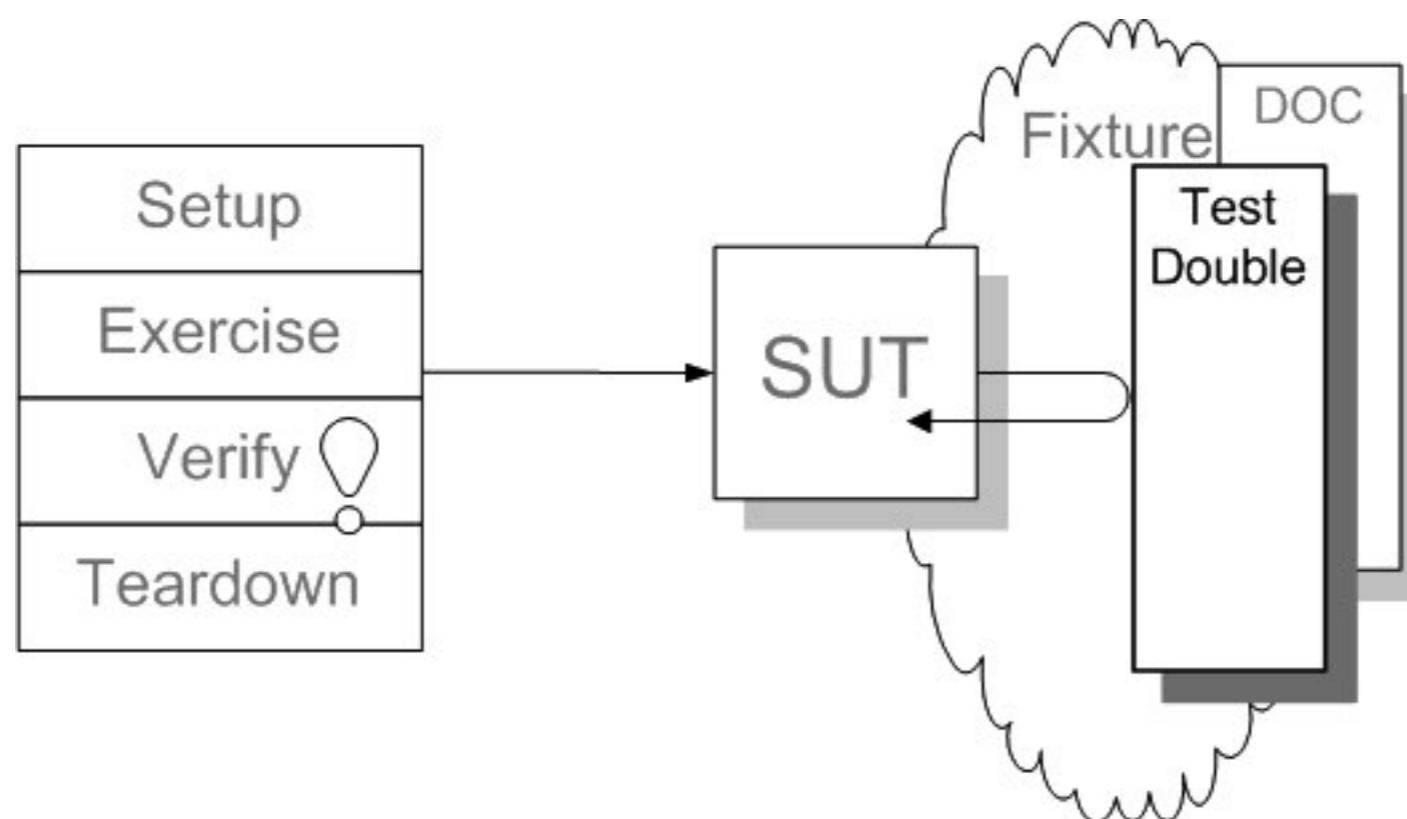
# Testing controller with service

Try to mocking service with Mockito



# Test Double

How can we verify logic independently ?  
How can we avoid Slow tests ?



<http://xunitpatterns.com/Test%20Double.html>



# Types of Test Double

Based on **how** and **why** we use it !!

Dummy  
object

Test  
stub

Test  
spy

Mock  
object

Fake  
object



# Run all tests !!

\$mvnw clean test

```
[INFO]
[INFO] Results:
[INFO]
[WARNING] Tests run: 10, Failures: 0, Errors: 0,
[INFO]
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 18.299 s
[INFO] Finished at: 2018-08-20T23:36:31+07:00
[INFO] -----
```

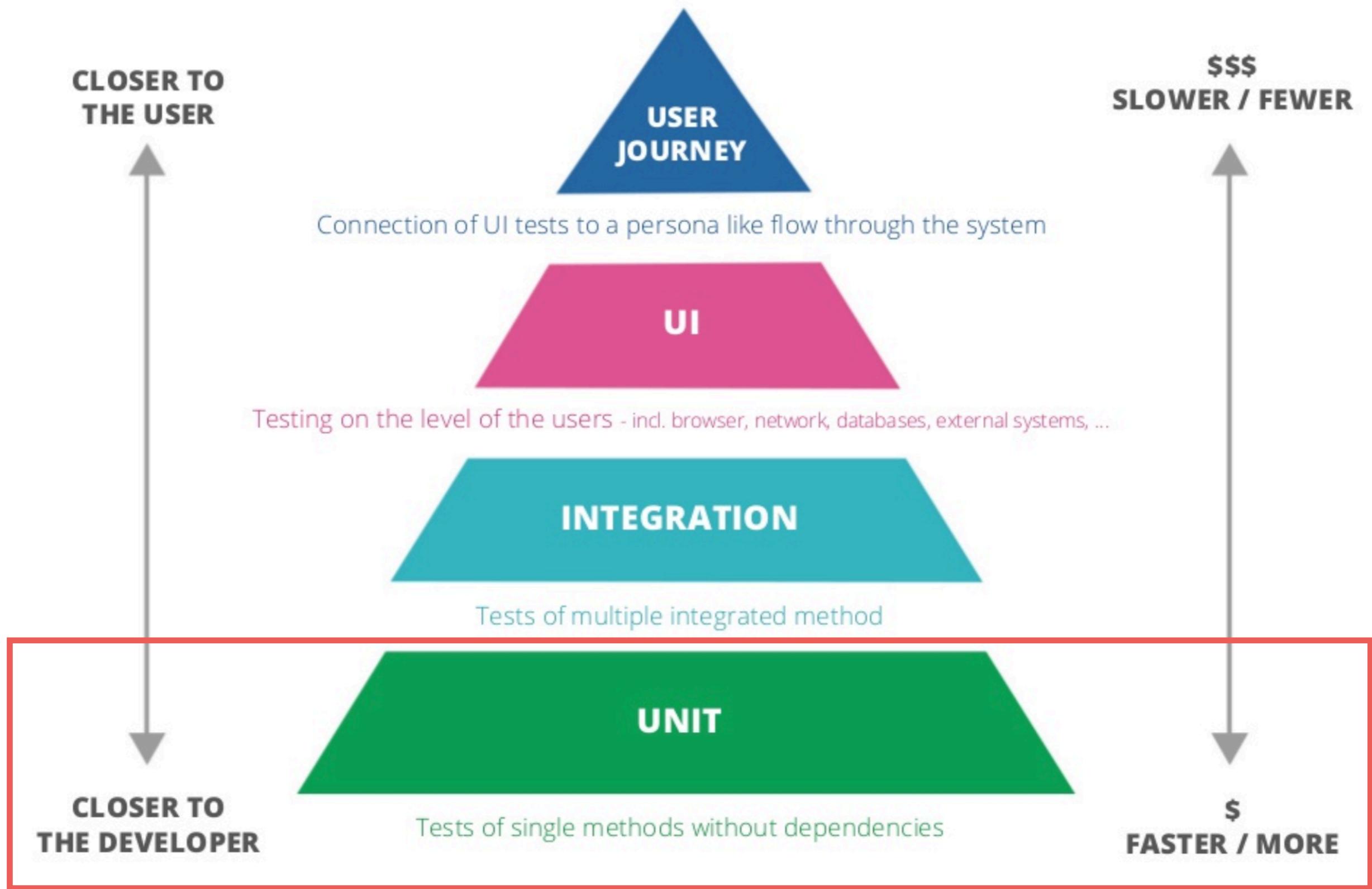


# **Slow Test ?**



# **How to improve the speed of testing ?**





# Unit testing with Controller



# Unit test

Use Test Double

In java, use Mockito library



<http://site.mockito.org/>



# Unit test with Mockito (1)

## Setup dependency and controller

```
public class HelloControllerUnitTest {
```

```
    private HelloController helloController;
```

```
    @Mock  
    private MessageService messageService;
```

1

```
@Before  
public void initial() {  
    initMocks(this);  
    helloController = new HelloController();  
    helloController.setMessageService(messageService);  
}
```

2



# Unit test with Mockito (2)

Testing by call method from controller

```
@Test  
public void test() {  
    given(messageService.concat("somkiat"))  
        .willReturn("Hello, somkiat");  
  
    HelloResponse response  
        = helloController.sayHi("somkiat");  
  
    assertEquals("Hello, somkiat", response.getMessage());  
}
```



# Code coverage



**"Code coverage can show the high risk areas in a program, but never the risk-free."**

Paul Reilly, 2018, Kotlin TDD with Code Coverage



# Code coverage

A tool to measure how much of your code is covered by tests that break down into classes, methods and lines.



# Code coverage

But 100% of code coverage **does not mean** that your code is 100% correct



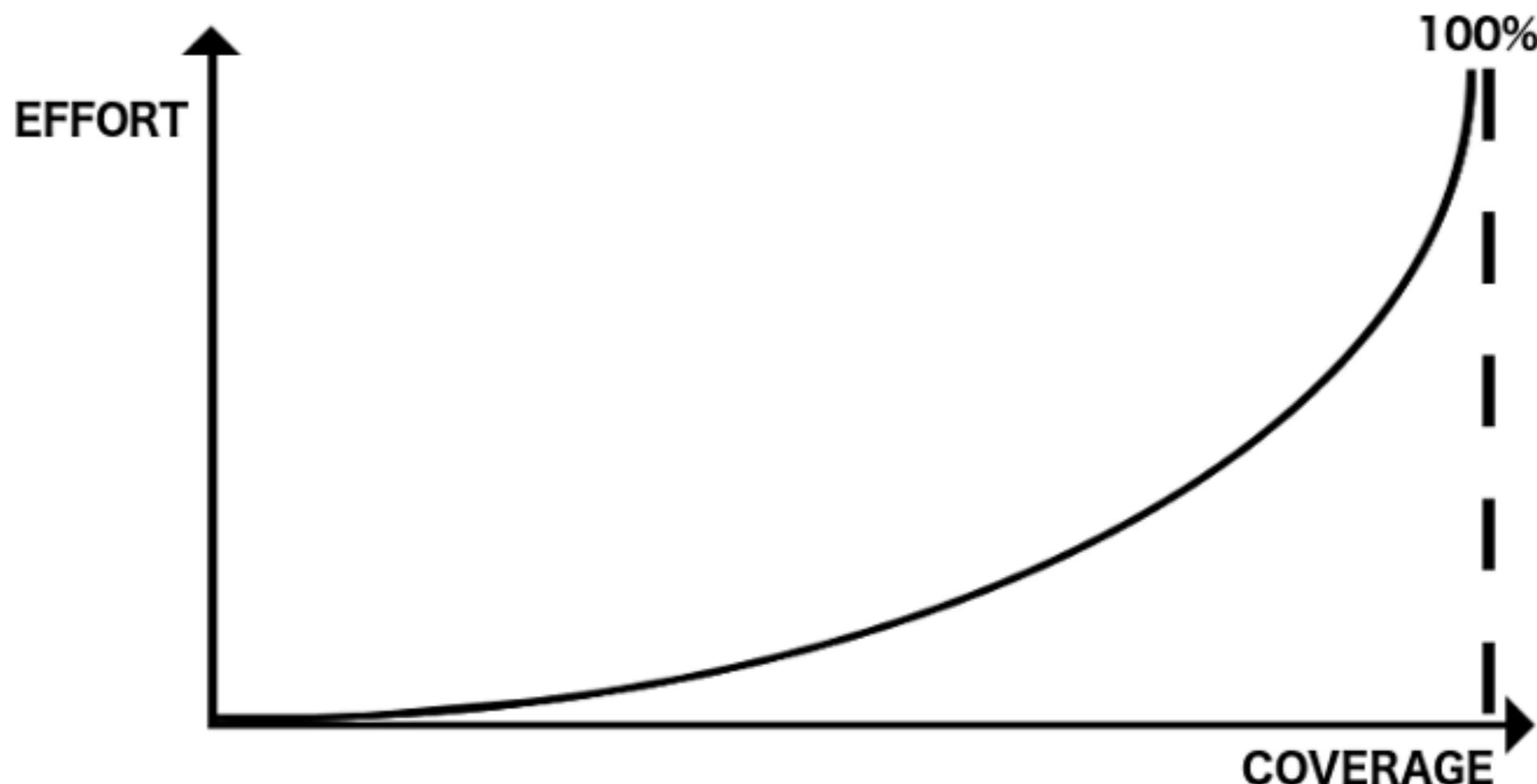
# Code coverage

Powerful tool to improve the quality of your code

**Code coverage != quality of tests**



# Code coverage 100% ?



# % of Code/Test coverage



# Code coverage with Java

Cobertura  
Jacoco

<https://gist.github.com/up1/f0d156f103dc7884baf637e4eeef8bf4>



# Add coverage to pom.xml (1)

```
<build>
    <finalName>hello</finalName>
    <plugins>
        <plugin>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-maven-plugin</artifactId>
        </plugin>

        <plugin>
            <artifactId>maven-compiler-plugin</artifactId>
            <version>2.5.1</version>
            <configuration>
                <source>${java.version}</source>
                <target>${java.version}</target>
                <encoding>${project.build.sourceEncoding}</encoding>
            </configuration>
        </plugin>
    </plugins>
</build>
```



# Add coverage to pom.xml (2)

```
<plugin>
  <groupId>org.codehaus.mojo</groupId>
  <artifactId>cobertura-maven-plugin</artifactId>
  <version>2.7</version>
  <configuration>
    <formats>
      <format>html</format>
      <format>xml</format>
    </formats>
  </configuration>
  <executions>
    <execution>
      <phase>package</phase>
      <goals>
        <goal>cobertura</goal>
      </goals>
    </execution>
  </executions>
  <dependencies>
    <dependency>
      <groupId>org.ow2.asm</groupId>
      <artifactId>asm</artifactId>
      <version>5.0.3</version>
    </dependency>
  </dependencies>
</plugin>
```



# Run test again

\$mvn clean package

Cobertura Report generation was successful.

Cobertura 2.1.1 - GNU GPL License (NO WARRANTY) - See COPYRIGHT file

Cobertura: Loaded information on 3 classes.

time: 125ms

Cobertura Report generation was successful.

---

BUILD SUCCESS

---



# Coverage report

open target/site/cobertura/index.html

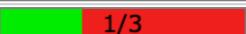
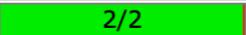
**Packages**

[All](#)  
[hello](#)  
[hello.controller](#)  
[hello.domain](#)

---

**All Packages**

**Coverage Report - All Packages**

Package	# Classes	Line Coverage	Branch Coverage	Complexity
All Packages	3	63% 	N/A	N/A
hello	1	33% 	N/A	N/A
hello.controller	1	100% 	N/A	N/A
hello.domain	1	66% 	N/A	N/A

Report generated by [Cobertura](#) 2.1.1 on 3/6/18 12:40 AM.

---

**Classes**

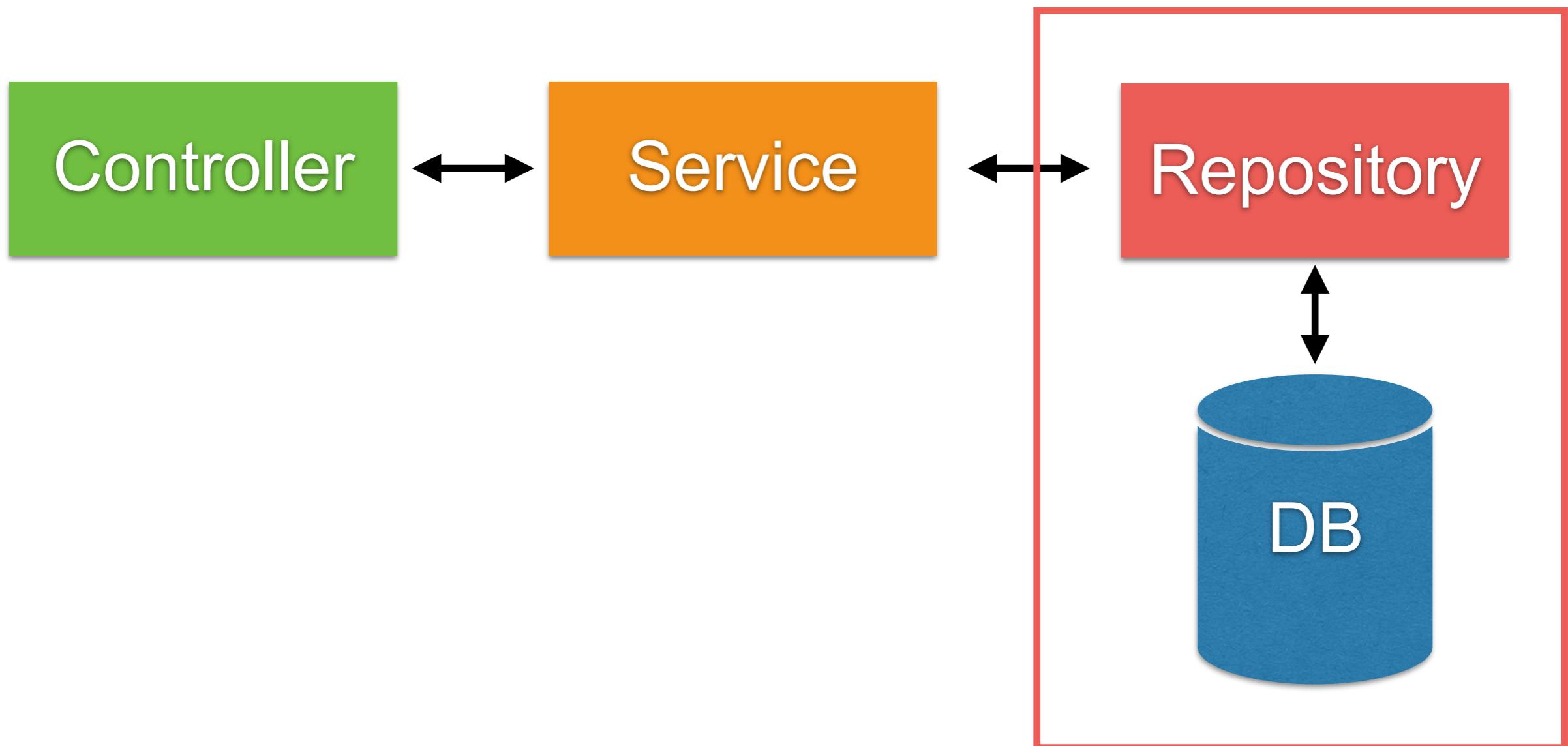
[Hello \(66%\)](#)  
[HelloApplication \(33%\)](#)  
[HelloController \(100%\)](#)



# Working with Repository

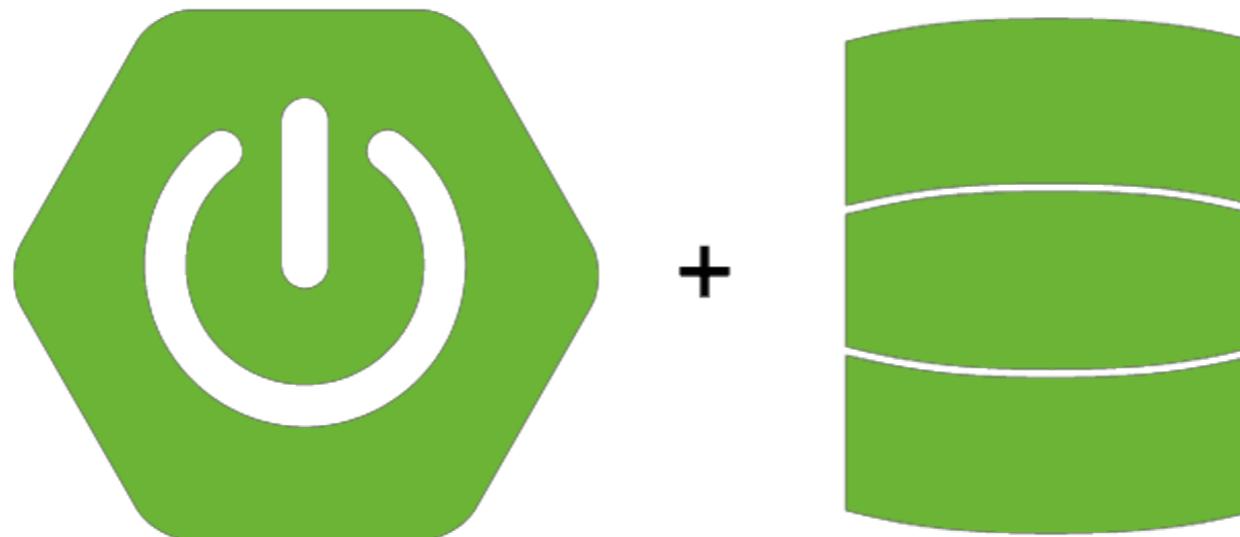


# Working with repository



# Working with repository

We're using Spring Data



<http://projects.spring.io/spring-data/>



# 1. Modify pom.xml

Add library of Spring Data such as JPA

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-data-jpa</artifactId>
</dependency>
```

<https://docs.spring.io/spring-data/jpa/docs/current/reference/html/>



## 2. Modify pom.xml (1)

Add library of data store such as **MariaDB**

```
<dependency>
    <groupId>org.mariadb.jdbc</groupId>
    <artifactId>mariadb-java-client</artifactId>
    <version>2.2.6</version>
</dependency>
```

<https://mvnrepository.com/artifact/org.mariadb.jdbc/mariadb-java-client>



## 2. Modify pom.xml (2)

Add library of data store such as PostgreSQL

```
<dependency>
    <groupId>org.postgresql</groupId>
    <artifactId>postgresql</artifactId>
    <version>42.2.4</version>
</dependency>
```

<https://mvnrepository.com/artifact/org.postgresql/postgresql>



# 3. Add configuration of database

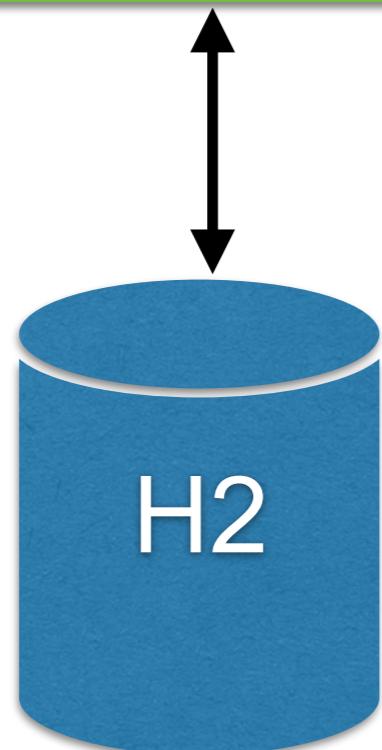
In src/main/resources/application.properties

```
spring.jpa.hibernate.ddl-auto=create
spring.datasource.url=<url>
spring.datasource.username=<user>
spring.datasource.password=<password>
```

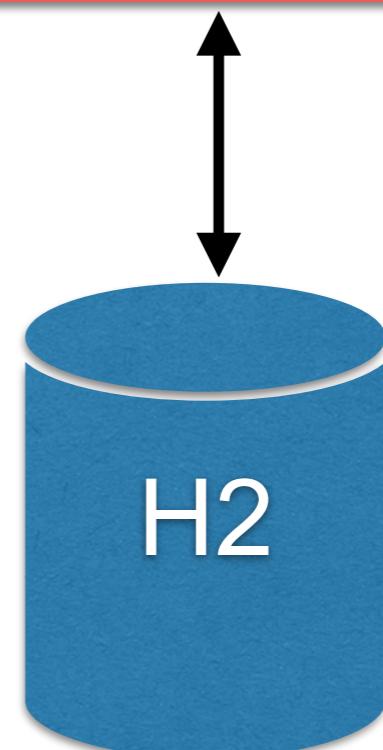


# Working with H2 Database

Production



Testing



# Working with H2 database

Add library of data store such as H2

```
<dependency>
    <groupId>com.h2database</groupId>
    <artifactId>h2</artifactId>
    <version>1.4.197</version>
</dependency>
```

<https://mvnrepository.com/artifact/com.h2database/h2/>



# Create repository with JPA

## MessageRepository.java

```
public interface MessageRepository  
    extends CrudRepository<Message, Integer> {  
  
}
```



# Create Entity class

## Message.java

```
@Entity  
public class Message {  
  
    @Id  
    @GeneratedValue(strategy = GenerationType.AUTO)  
    private int id;  
    private String data;
```

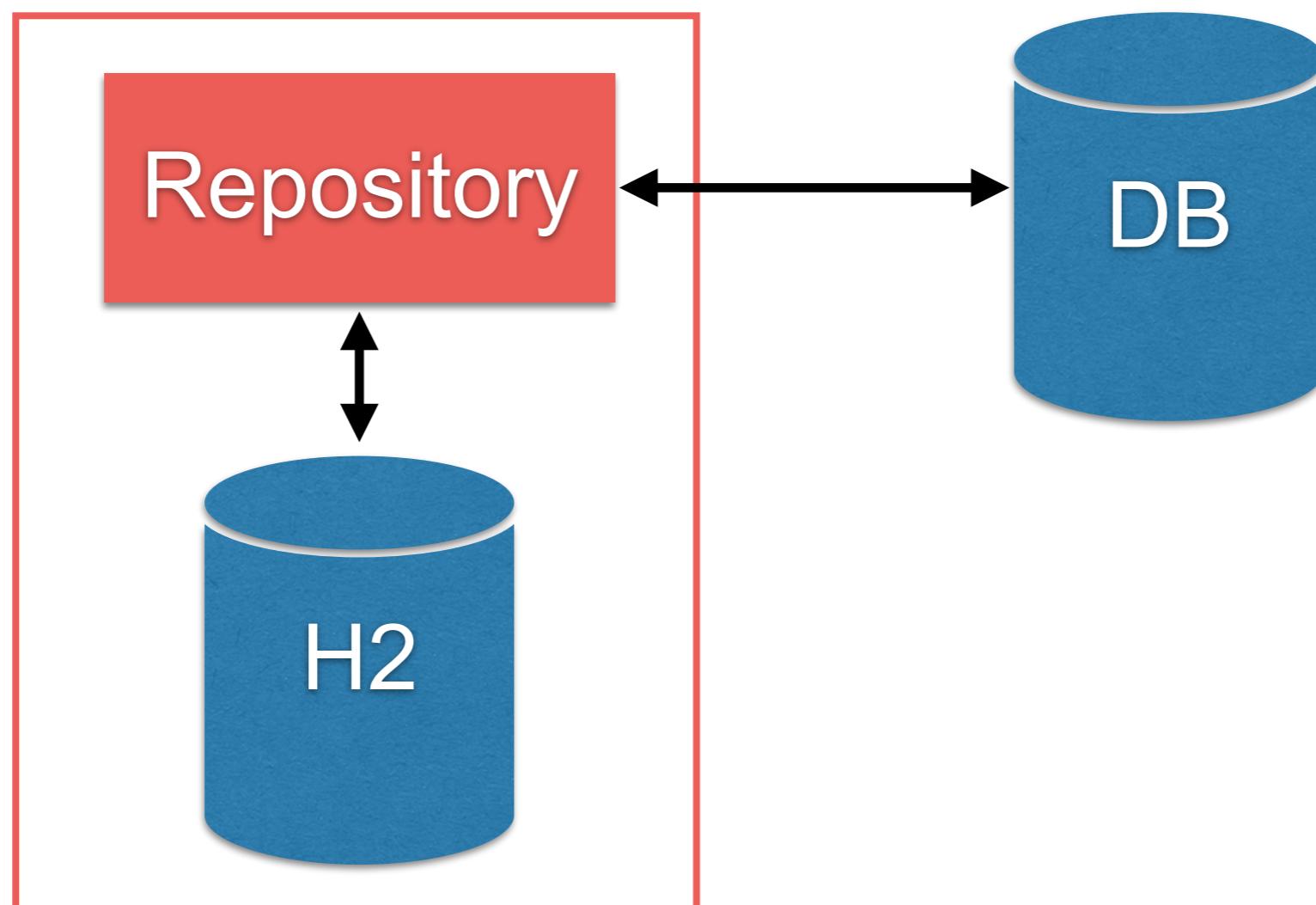


# How to testing repository ?



# Repository Testing

Using `@DataJpaTest` (slice testing)



Working with In-memory database



# Spring boot provide DataJpaTest

should be add H2 library to pom.xml

```
<dependency>
    <groupId>com.h2database</groupId>
    <artifactId>h2</artifactId>
    <scope>test</scope>
</dependency>
```



# Repository Testing (1)

Setup test with `@DataJpaTest`

```
@RunWith(SpringRunner.class)
@DataJpaTest
public class MessageRepositoryTest {

    @Autowired
    private MessageRepository messageRepository;
```



# Repository Testing (2)

## Add a test case

```
@Test  
public void create_and_fetch_message() {  
    Message newMessage = new Message("somkiat");  
    messageRepository.save(newMessage);  
  
    Optional<Message> actualMessage  
        = messageRepository.findById(1);  
  
    assertEquals("somkiat", actualMessage.get().getData());  
}
```



# Repository Testing (3)

Add a test case

```
@Test  
public void create_and_count_message() {  
    Message newMessage = new Message("somkiat");  
    messageRepository.save(newMessage);  
  
    long allRecord = messageRepository.count();  
    assertEquals(1, allRecord);  
}
```



# Run test

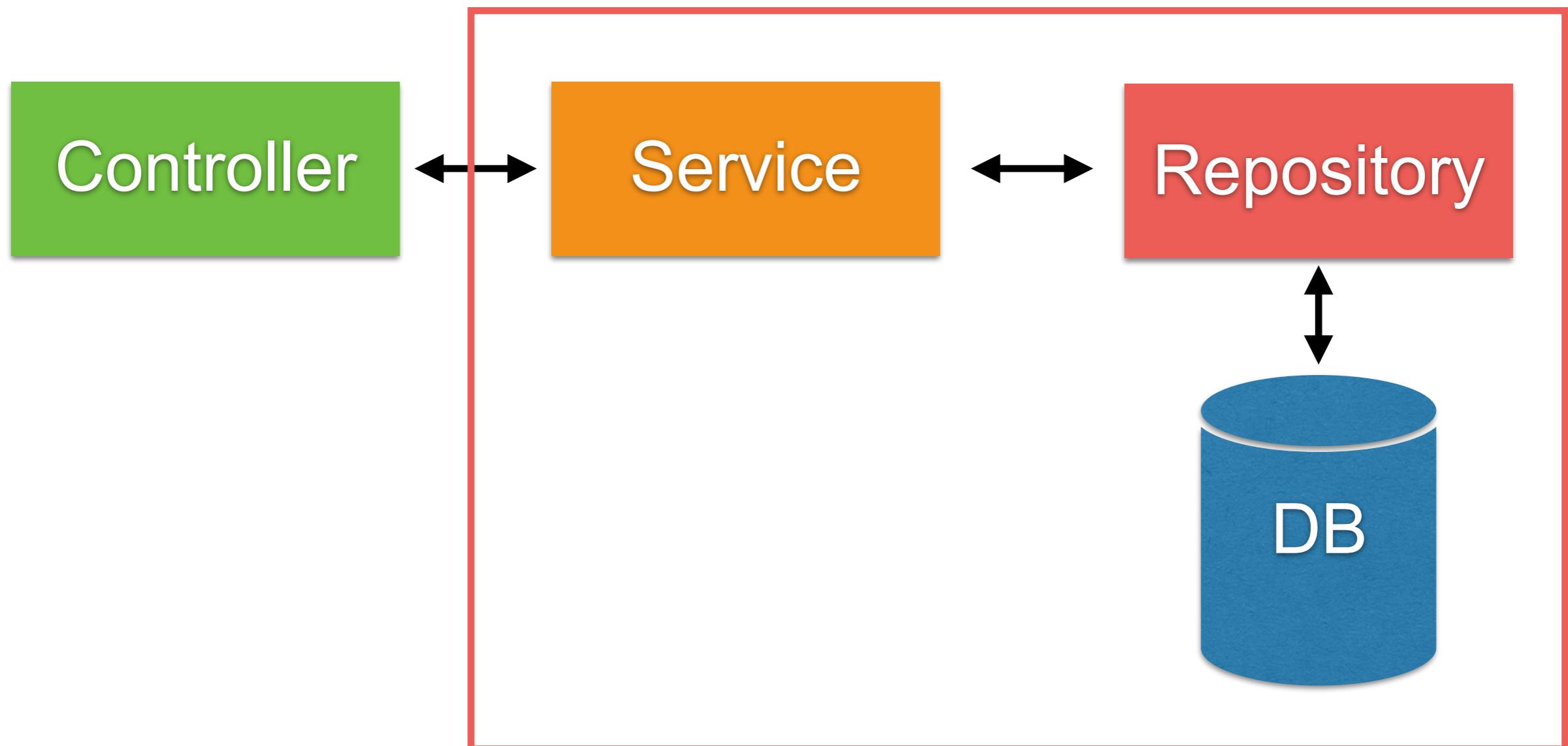
\$mvn clean test



# Integrate repository with service



# Service use repository ?



# Service call repository

## Edit MessageService.java

```
@Service
public class MessageService {

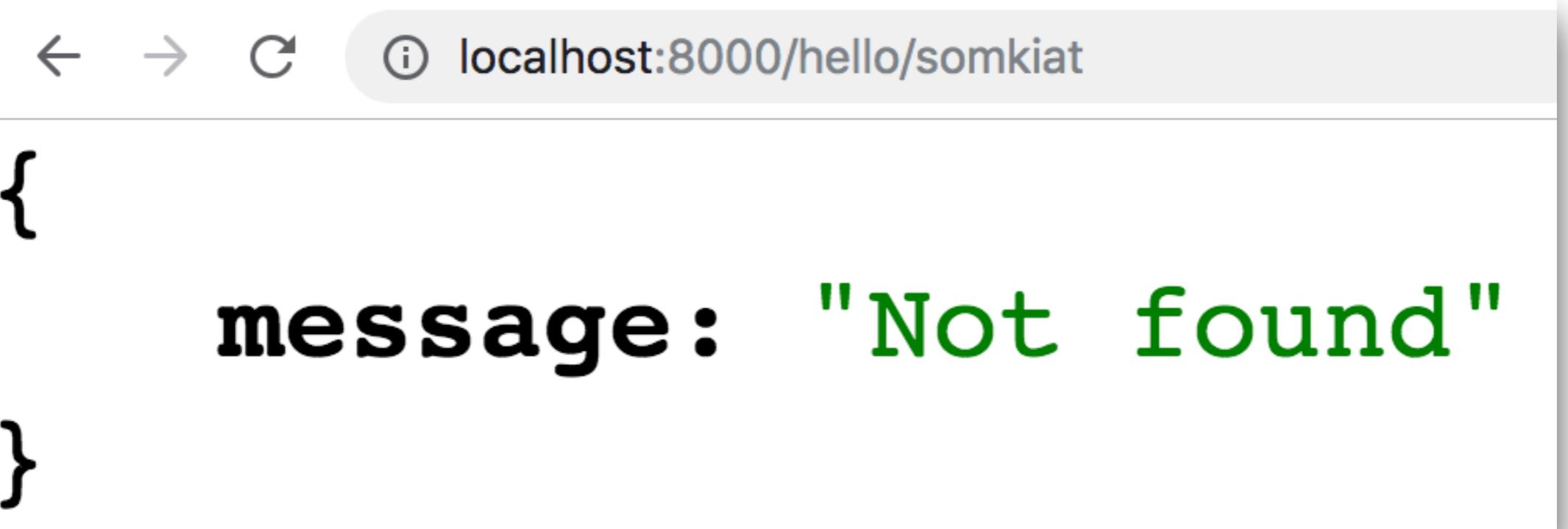
    @Autowired
    private MessageRepository messageRepository;

    public String concat(String name) {
        Optional<Message> result = messageRepository.findById(1);
        if(result.isPresent()) {
            return "Hello, " + name;
        } else {
            return "Not found";
        }
    }
}
```



# Run spring boot

\$mvnw spring-boot:run



```
{  
  message: "Not found"  
}
```



# Service call repository

```
@Service  
public class UserService {  
  
    private AccountRepository accountRepository;  
  
    @Autowired  
    public UserService(AccountRepository accountRepository) {  
        this.accountRepository = accountRepository;  
    }  
  
    public Account getAccount(int id) {  
        Optional<Account> account = accountRepository.findById(id);  
        if(account.isPresent()) {  
            return account.get();  
        }  
        throw new MyAccountNotFoundException(  
            String.format("Account id=[%d] not found", id));  
    }  
}
```

1



# Service call repository

```
@Service  
public class UserService {  
  
    private AccountRepository accountRepository;  
  
    @Autowired  
    public UserService(AccountRepository accountRepository) {  
        this.accountRepository = accountRepository;  
    }  
  
    public Account getAccount(int id) {  
        Optional<Account> account = accountRepository.findById(id);  
        if(account.isPresent()) {  
            return account.get();  
        }  
        throw new MyAccountNotFoundException(  
            String.format("Account id=[%d] not found", id));  
    }  
}
```

2



# Initial data in database



# Initial database #1

## Using @Bean and CommandLineRunner

```
@Bean  
public CommandLineRunner initData(MessageRepository repository) {  
    return new CommandLineRunner() {  
  
        @Override  
        public void run(String... args) throws Exception {  
            repository.save(new Message("somkiat1"));  
            repository.save(new Message("somkiat2"));  
        }  
    };  
}
```



# Initial database #2

## Using @PostConstruct

```
@PostConstruct  
public void initData() {  
    Account account1 = new Account();  
    account1.setAccountId("01");  
    accountRepository.save(account1);  
    Account account2 = new Account();  
    account2.setAccountId("02");  
    accountRepository.save(account2);  
}
```



# Initial database #3

**Schema** (resources/schema.sql)

**Data** (resources/data.sql)

## Schema.sql

```
CREATE TABLE account(  
    id BIGINT AUTO_INCREMENT PRIMARY KEY,  
    account_Id VARCHAR(16) NOT NULL UNIQUE,  
    mobile_No VARCHAR(10),  
    name VARCHAR(50),  
    account_Type CHAR(2)  
);
```

## Data.sql

```
INSERT INTO account (account_Id) VALUES ('01');  
INSERT INTO account (account_Id) VALUES ('02');
```



# Initial database 2

Disable auto generate DDL from JPA in file application.yml

```
spring:  
  jpa:  
    show-sql: true  
    hibernate:  
      ddl-auto: none
```



# Initial database 2

## Problem with naming strategy !!

```
spring:  
  jpa:  
    show-sql: true  
    hibernate:  
      ddl-auto: none  
      naming:  
        physical-strategy:  
          org.springframework.boot.orm.jpa.hibernate.SpringPhysicalNamingStrategy  
        implicit-strategy:  
          org.springframework.boot.orm.jpa.hibernate.SpringImplicitNamingStrategy
```

<https://github.com/spring-projects/spring-boot/tree/master/spring-boot-project/spring-boot/src/main/java/org/springframework/boot/orm/jpa/hibernate>



# Run and see from logging

Execute file schema.sql and data.sql

```
.datasource.init.ScriptUtils      : Executing SQL script from URL [file:/Users/somki...  
.datasource.init.ScriptUtils      : Executed SQL script from URL [file:/Users/somki...  
.datasource.init.ScriptUtils      : Executing SQL script from URL [file:/Users/somki...  
.datasource.init.ScriptUtils      : Executed SQL script from URL [file:/Users/somki...
```



# Run spring boot

\$mvnw spring-boot:run

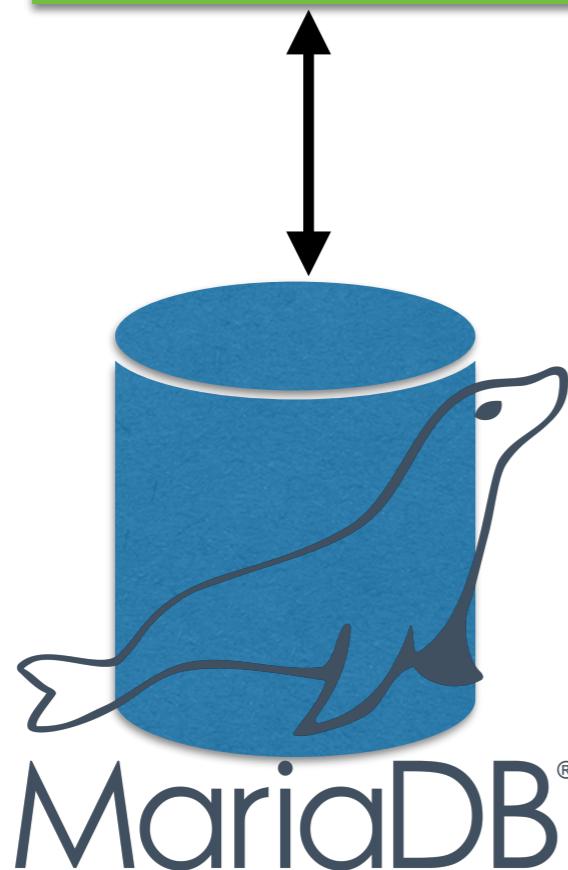


```
{  
  message: "Hello, somkiat"  
}
```

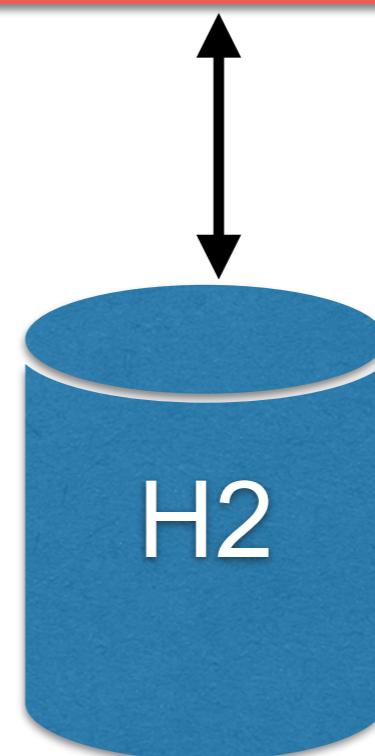


# Working with Real Database

Production



Testing



# Add database dependency

In file pom.xml

```
<dependency>
    <groupId>org.mariadb.jdbc</groupId>
    <artifactId>mariadb-java-client</artifactId>
    <version>2.2.5</version>
</dependency>
```

<https://mariadb.com/kb/en/library/about-mariadb-connector-j/>



# Config database in application

In file resources/application.yml

```
spring:
  datasource:
    driver-class-name: org.mariadb.jdbc.Driver
    url: jdbc:mariadb://35.198.254.195:3306/account
    username: user01
    password: password

    initialization-mode: always
    testWhileIdle: true
    validationQuery: SELECT 1

  jpa:
    show-sql: true
    properties:
      hibernate:
        dialect: org.hibernate.dialect.MySQL5InnoDBDialect
```



# Config database in application

## Required config for database

```
spring:
```

```
  datasource:  
    driver-class-name: org.mariadb.jdbc.Driver  
    url: jdbc:mariadb://35.198.254.195:3306/account  
    username: user01  
    password: password
```

```
    initialization-mode: always
```

```
    testWhileIdle: true
```

```
    validationQuery: SELECT 1
```

```
jpa:
```

```
  show-sql: true
```

```
  properties:
```

```
    hibernate:
```

```
      dialect: org.hibernate.dialect.MySQL5InnoDBDialect
```



# Config database in application

## Required config for database

```
spring:  
  datasource:  
    driver-class-name: org.mariadb.jdbc.Driver  
    url: jdbc:mariadb://35.198.254.195:3306/account  
    username: user01  
    password: password
```

```
  initialization-mode: always
```

```
  testWhileIdle: true  
  validationQuery: SELECT 1  
  
  jpa:  
    show-sql: true  
    properties:  
      hibernate:  
        dialect: org.hibernate.dialect.MySQL5InnoDBDialect
```

By default not run



# Run service

http://localhost:8080/account/0868696209

```
← → ⌂ ⓘ localhost:8080/account/0868696209
[
  - {
    accountNo: "01",
    mobileNo: null,
    name: "",
    accountType: ""
  },
  - {
    accountNo: "02",
    mobileNo: null,
    name: "",
    accountType: ""
  }
]
```



# **Check your test ?**

**\$mvnw clean test**



# Step to test

Stop the real database before run test

See result



# Error ?

Spring boot try to connect to the real database ?



# Fix Error

Create file /resources/application.yml in test folder

spring: **config of H2 database**

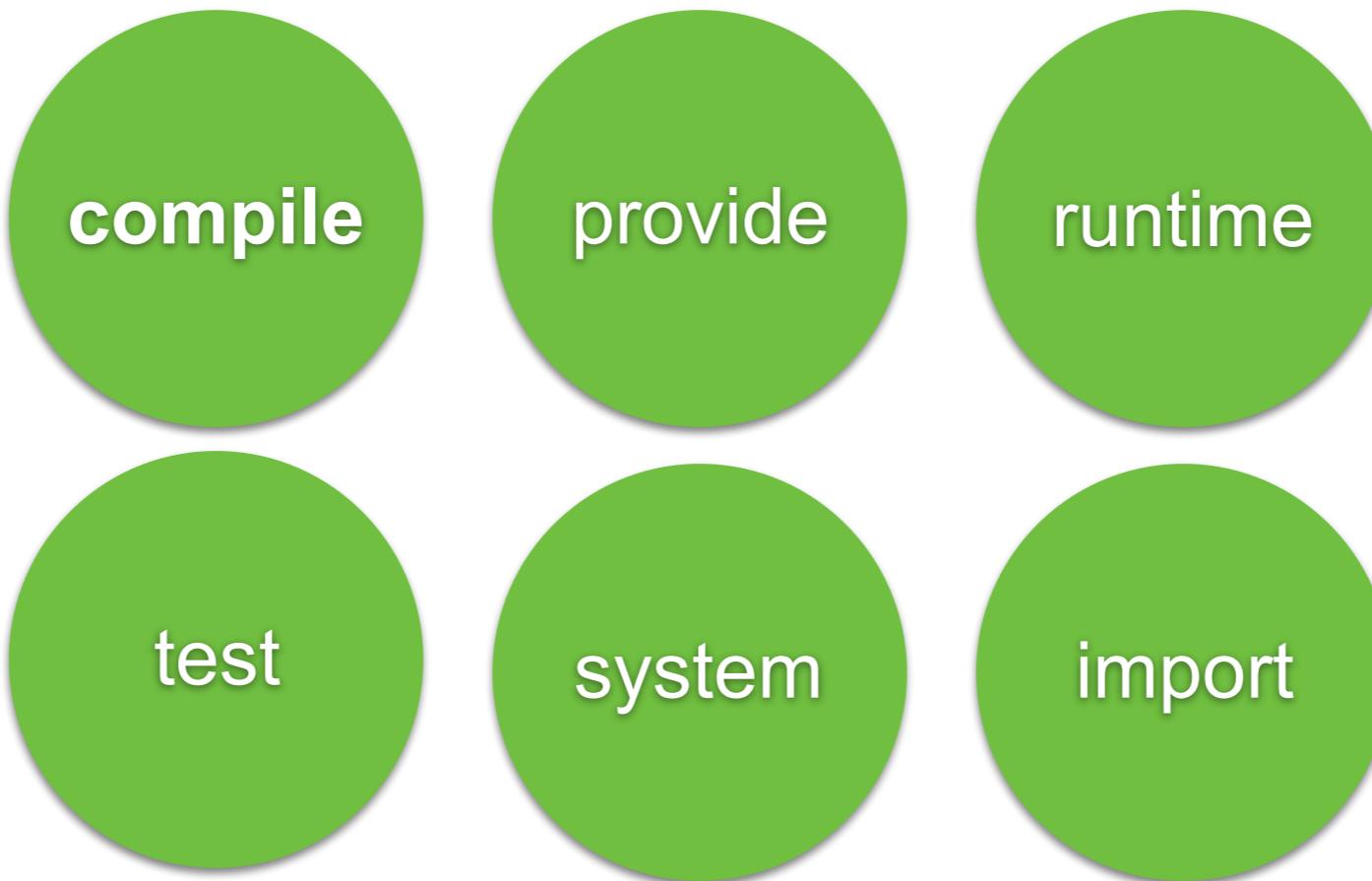
```
datasource:  
  driver-class-name: org.h2.Driver  
  url: jdbc:h2:mem:db;DB_CLOSE_DELAY=-1  
  username: sa  
  password: sa
```

```
jpa:  
  show-sql: true  
  properties:  
    hibernate:  
      dialect: org.hibernate.dialect.H2Dialect
```



# Question ?

Dependency Scopes in file pom.xml  
runtime ?



[https://maven.apache.org/guides/introduction/introduction-to-dependency-mechanism.html#Dependency\\_Scope](https://maven.apache.org/guides/introduction/introduction-to-dependency-mechanism.html#Dependency_Scope)



# Question ?

show\_sql=true not working ?

```
jpa:  
  show_sql: true  
  hibernate:  
    ddl-auto: none  
properties:  
  hibernate:  
    format_sql: true  
dialect: org.hibernate.dialect.MySQL5InnoDBDialect
```

```
logging:  
  level:  
    org.hibernate.SQL: DEBUG
```

**Beautiful sql format**

**Default log level = INFO**

<http://www.baeldung.com/sql-logging-spring-boot>



# Result of logging message

```
13:18:36.336  INFO 79616 --- [nio-8080-exec-2] o.s.web.servlet.DispatcherServlet : initialization completed in 37 ms
13:18:36.864  INFO 79616 --- [nio-8080-exec-2] o.h.h.i.QueryTranslatorFactory      : HHH000397: Using ASTQueryTranslatorFactory
13:18:37.138 DEBUG 79616 --- [nio-8080-exec-2] org.hibernate.SQL
select account0_.id as id1_0_, account0_.account_id as account_2_0_, ac
 as account_3_0_, account0_.mobile_no as mobile_n4_0_, account0_.name a
unt account0_
```



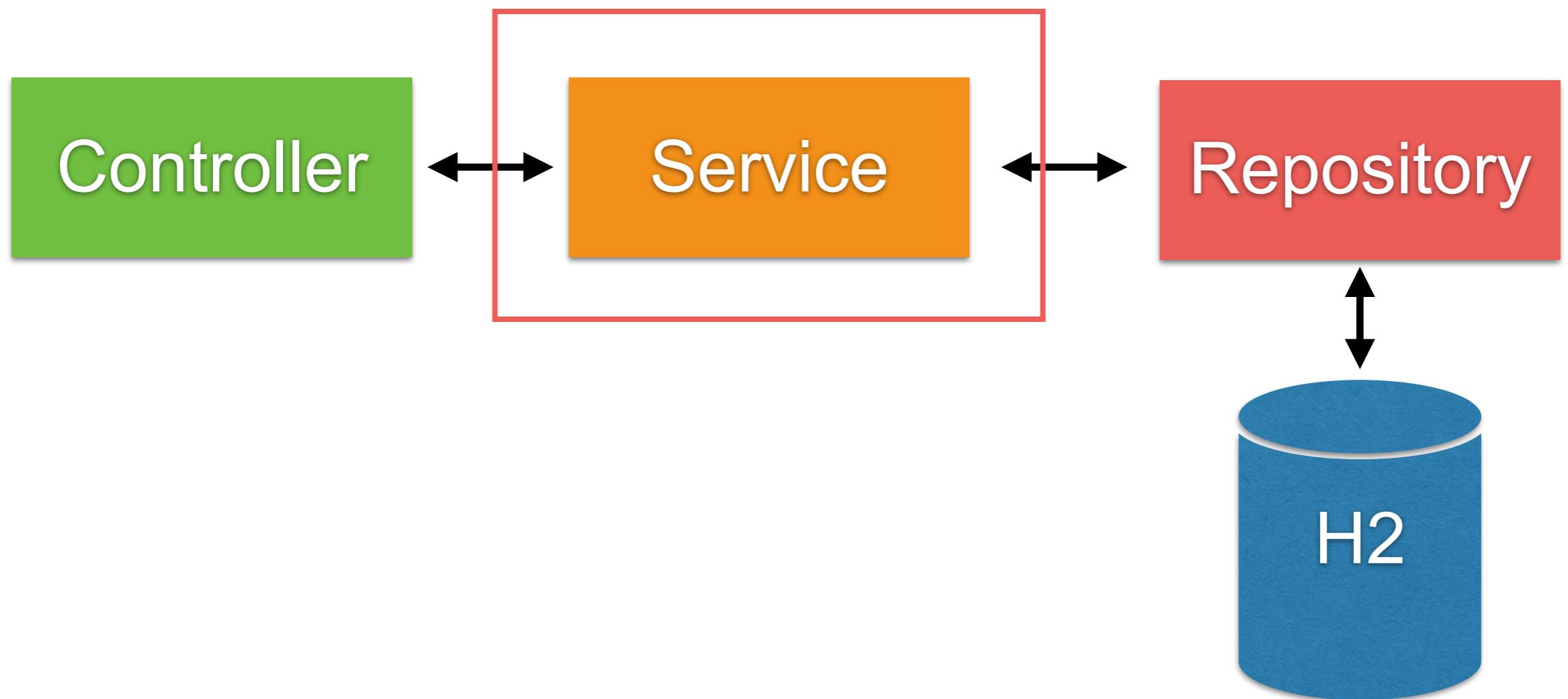
# Log Levels

Level	Color
FATAL	Red
ERROR	Red
WARN	Yellow
<b>INFO (default)</b>	Green
DEBUG	Green
TRACE	Green

<https://docs.spring.io/spring-boot/docs/current/reference/html/boot-features-logging.html>



# Service Testing ?



# Service Testing

```
@RunWith(MockitoJUnitRunner.class)
public class UserServiceTest {

    @Mock
    private AccountRepository accountRepository;

    @Test
    public void getAccount() {
        // Stub
        Account account = new Account();
        account.setUserName("user");
        account.setPassword("pass");
        account.setSalary(1000);
        given(accountRepository.findById(1))
            .willReturn(Optional.of(account));

        UserService userService = new UserService(accountRepository);
        Account actualAccount = userService.getAccount(1);
        assertNotNull(actualAccount);
    }
}
```

1



# Service Testing

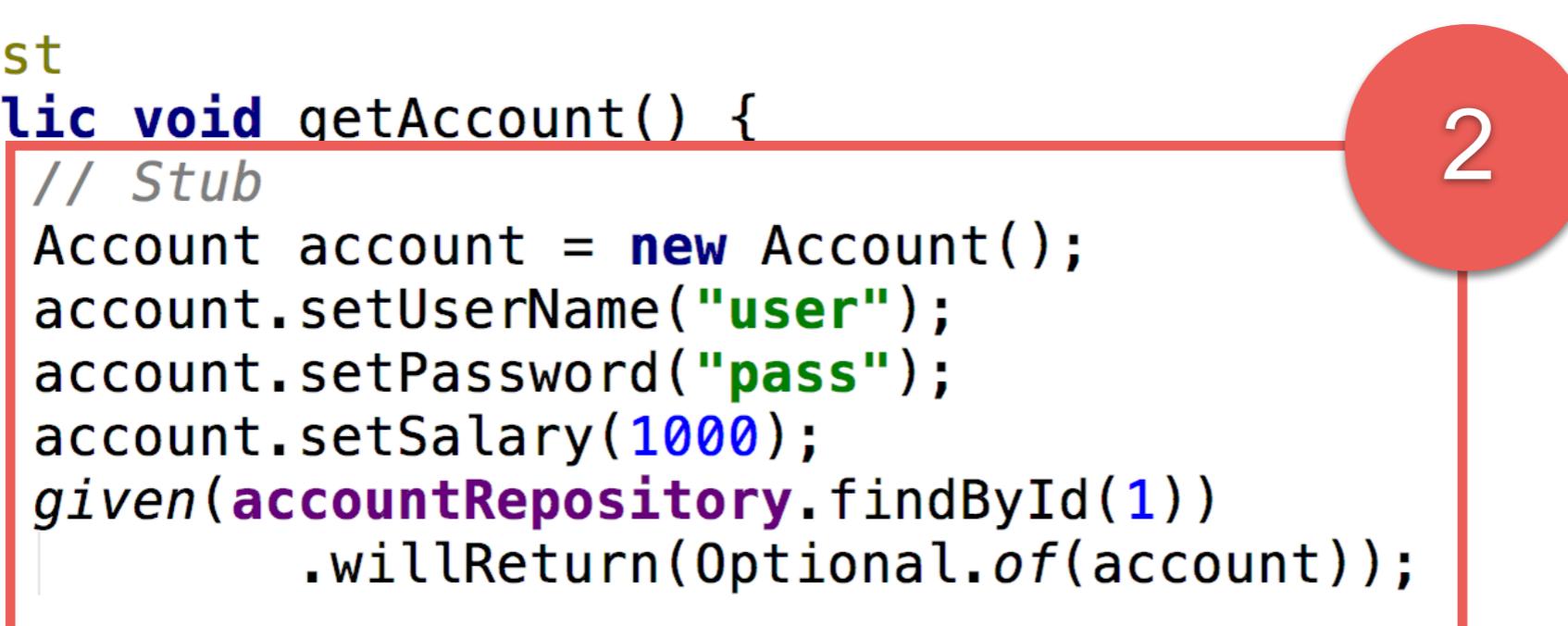
```
@RunWith(MockitoJUnitRunner.class)
public class UserServiceTest {

    @Mock
    private AccountRepository accountRepository;

    @Test
    public void getAccount() {
        // Stub
        Account account = new Account();
        account.setUserName("user");
        account.setPassword("pass");
        account.setSalary(1000);
        given(accountRepository.findById(1))
            .willReturn(Optional.of(account));
    }

    UserService userService = new UserService(accountRepository);
    Account actualAccount = userService.getAccount(1);
    assertNotNull(actualAccount);
}

}
```



The diagram shows a red rectangular box highlighting the stub code within the `getAccount()` method. A red line connects this box to a red circle containing the number 2, which is positioned above the code block.



# Error handling



# Error handling

```
@Service
public class UserService {

    private AccountRepository accountRepository;

    @Autowired
    public UserService(AccountRepository accountRepository) {
        this.accountRepository = accountRepository;
    }

    public Account getAccount(int id) {
        Optional<Account> account = accountRepository.findById(id);
        if(account.isPresent()) {
            return account.get();
        }
        throw new MyAccountNotFoundException(
            String.format("Account id=[%d] not found", id));
    }
}
```



# MyAccountNotFoundException

```
public class MyAccountNotFoundException  
    extends RuntimeException {
```

```
    public MyAccountNotFoundException(String message) {  
        super(message);  
    }
```

```
}
```



# Response Status

404 = Not Found

Status	Description
400	Request body doesn't meet API spec
401	Authentication/Authorization fail
403	User can't perform the operation
404	Resource does not exist
405	Unsupported operation
500	Error on server



# Handling error in Spring Boot

```
@RestControllerAdvice  
public class AccountControllerHandler {  
  
    @ExceptionHandler(MyAccountNotFoundException.class)  
    public ResponseEntity<ExceptionResponse> accountNotFound(  
        MyAccountNotFoundException exception) {  
  
        ExceptionResponse response =  
            new ExceptionResponse(exception.getMessage(),  
                "More detail");  
  
        return new ResponseEntity<ExceptionResponse>(response,  
            HttpStatus.NOT_FOUND);  
    }  
}
```

1



# Handling error in Spring Boot

```
@RestControllerAdvice  
public class AccountControllerHandler {  
  
    @ExceptionHandler(MyAccountNotFoundException.class)  
    public ResponseEntity<ExceptionResponse> accountNotFound(  
        MyAccountNotFoundException exception) {  
  
        ExceptionResponse response =  
            new ExceptionResponse(exception.getMessage(),  
                "More detail");  
  
        return new ResponseEntity<ExceptionResponse>(response,  
            HttpStatus.NOT_FOUND);  
    }  
}
```

2



# ExceptionResponse

Response format of error

```
public class ExceptionResponse{  
  
    private Date timestamp = new Date();  
    private String message;  
    private String detail;  
  
    public ExceptionResponse(String message, String detail) {  
        this.message = message;  
        this.detail = detail;  
    }  
}
```



# Result of API

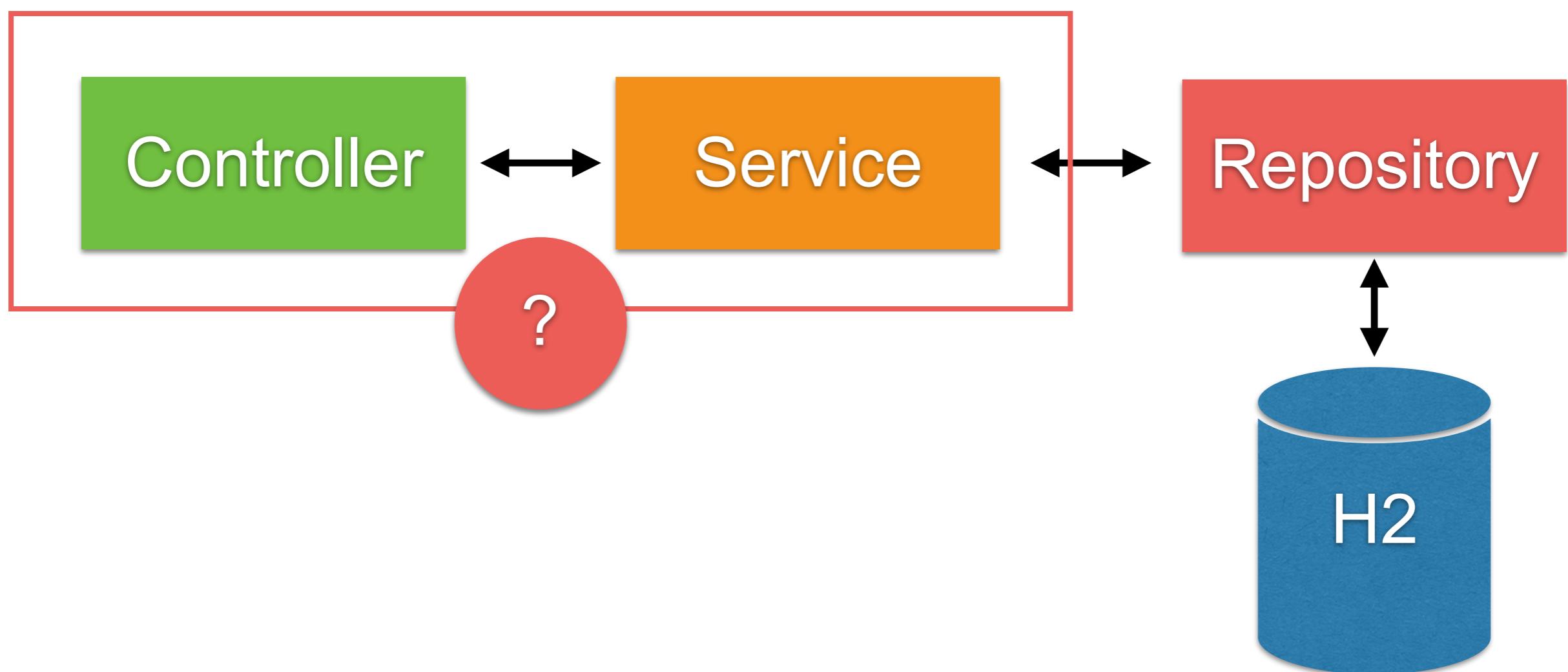
```
← → ⌂ ⓘ localhost:8888/account/2 🔍 ☆
{
  timestamp: "2018-09-15T15:25:44.776+0000",
  message: "Account id=[ 2 ] not found",
  detail: "More detail"
}
```



# How to test ?



# How to test with Error/Exception ?



# Testing with WebMvcTest and MockMvc

Try to check data in response

```
@Test  
public void getByIdWithNotFoundAccount() throws Exception {  
    // Stub  
    given(userService.getAccount(2))  
        .willThrow(new MyAccountNotFoundException("Not found"));  
  
    mockMvc.perform(  
        get("/account/2")  
            .accept(MediaType.APPLICATION_JSON))  
        .andExpect(status().isNotFound())  
        .andExpect(content().contentType(MediaType.APPLICATION_JSON_UTF8))  
        .andExpect(jsonPath("$.message", is("Not found")));  
}
```

1



# Testing with WebMvcTest and MockMvc

Try to check data in response

```
@Test  
public void getByIdWithNotFoundAccount() throws Exception {  
    // Stub  
    given(userService.getAccount(2))  
        .willThrow(new MyAccountNotFoundException("Not found"));  
  
    mockMvc.perform(  
        get("/account/2")  
            .accept(MediaType.APPLICATION_JSON))  
        .andExpect(status().isNotFound())  
        .andExpect(content().contentType(MediaType.APPLICATION_JSON_UTF8))  
        .andExpect(jsonPath("$.message", is("Not found")));  
}
```

2



# Testing with Service

Try to check exception

```
@RunWith(MockitoJUnitRunner.class)
public class UserServiceWithExceptionTest {

    @Mock
    private AccountRepository accountRepository;

    @Test(expected = MyAccountNotFoundException.class)
    public void getAccountWithException() {
        // Stub
        given(accountRepository.findById(1))
            .willThrow(new MyAccountNotFoundException("Not found"));

        UserService userService = new UserService(accountRepository);
        Account actualAccount = userService.getAccount(1);
    }
}
```



# Separate tests with jUnit



<https://semaphoreci.com/community/tutorials/how-to-split-junit-tests-in-a-continuous-integration-environment>



# Separate test with jUnit



# Working with jUnit Category

Create interface in each category

```
package com.lotto.lotto.category;  
  
public interface UnitTest {  
}
```

```
package com.lotto.lotto.category;  
  
public interface SlicingTest {  
}
```

```
package com.lotto.lotto.category;  
  
public interface IntegrationTest {  
}
```



# Add category in each test class

```
@RunWith(SpringRunner.class)
@SpringBootTest(webEnvironment = WebEnvironment.RANDOM_PORT)
@Category(IntegrationTest.class)
public class AccountControllerTest {
```

```
    @Autowired
    private TestRestTemplate testRestTemplate;
```

```
    @MockBean
    private UserService userService;
```



# Run with category

```
$mvnw clean test  
-Dgroups="com.lotto.lotto.category.UnitTest"
```



# More ...

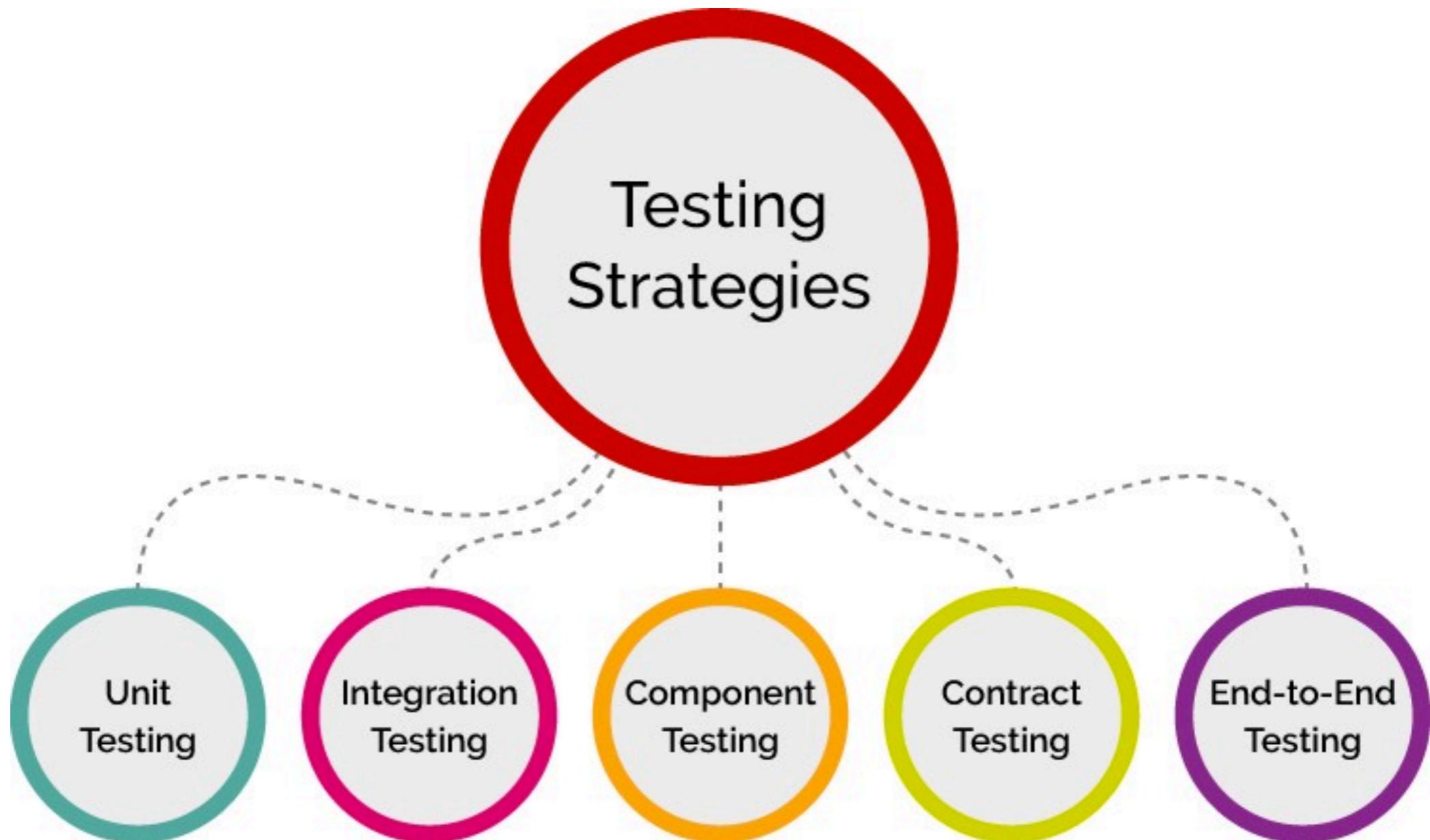


# Working with transaction !!

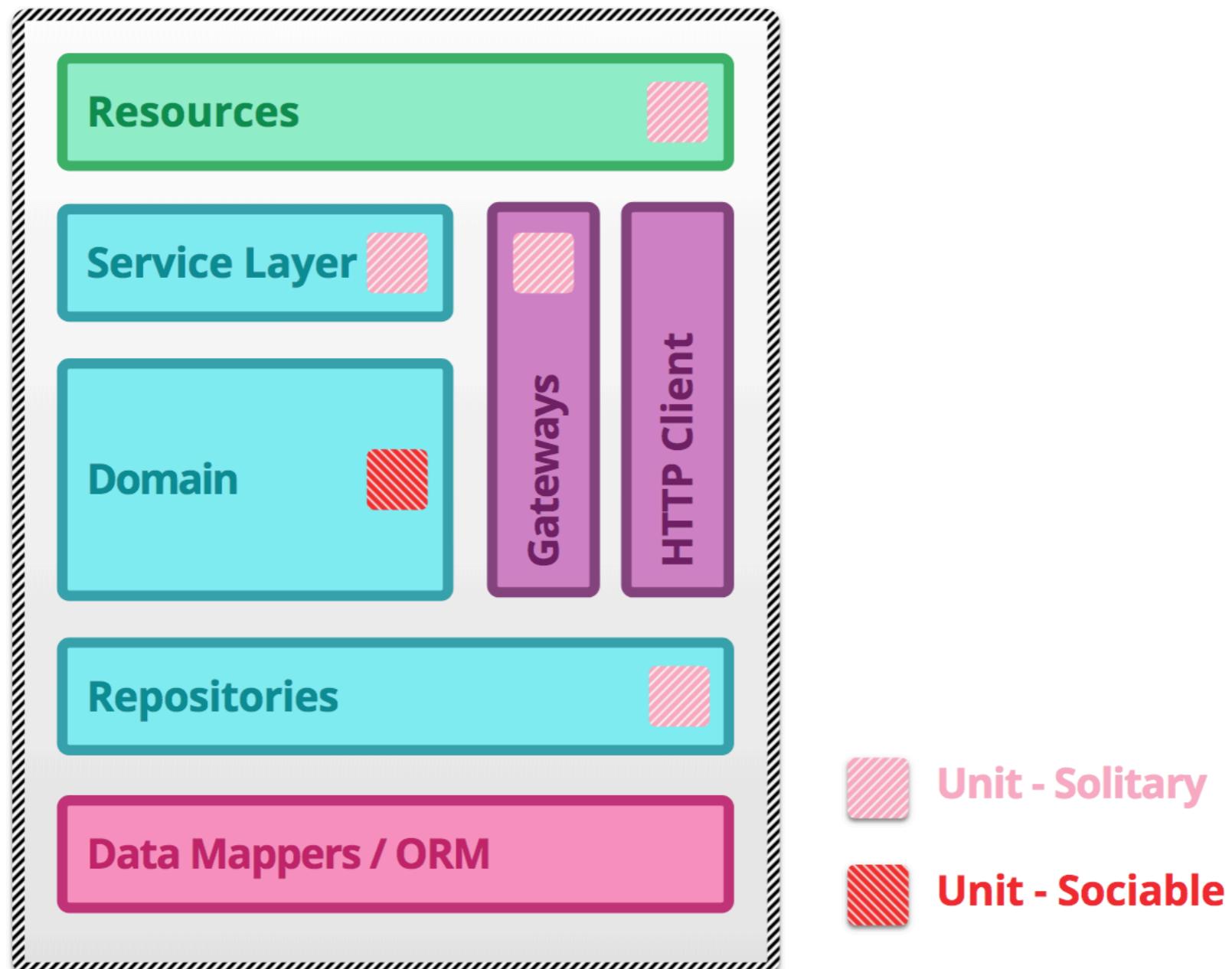


# Testing Strategies

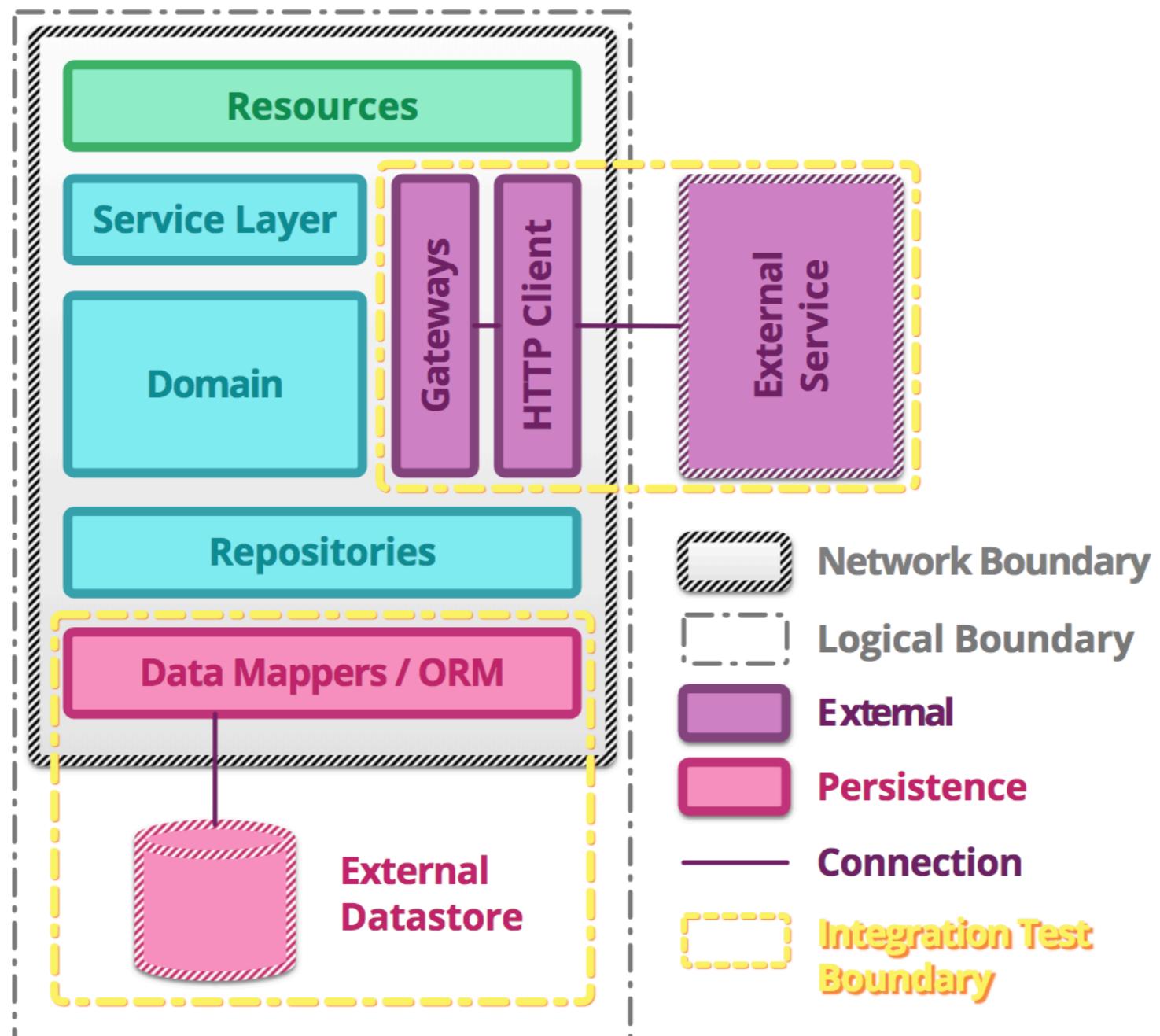




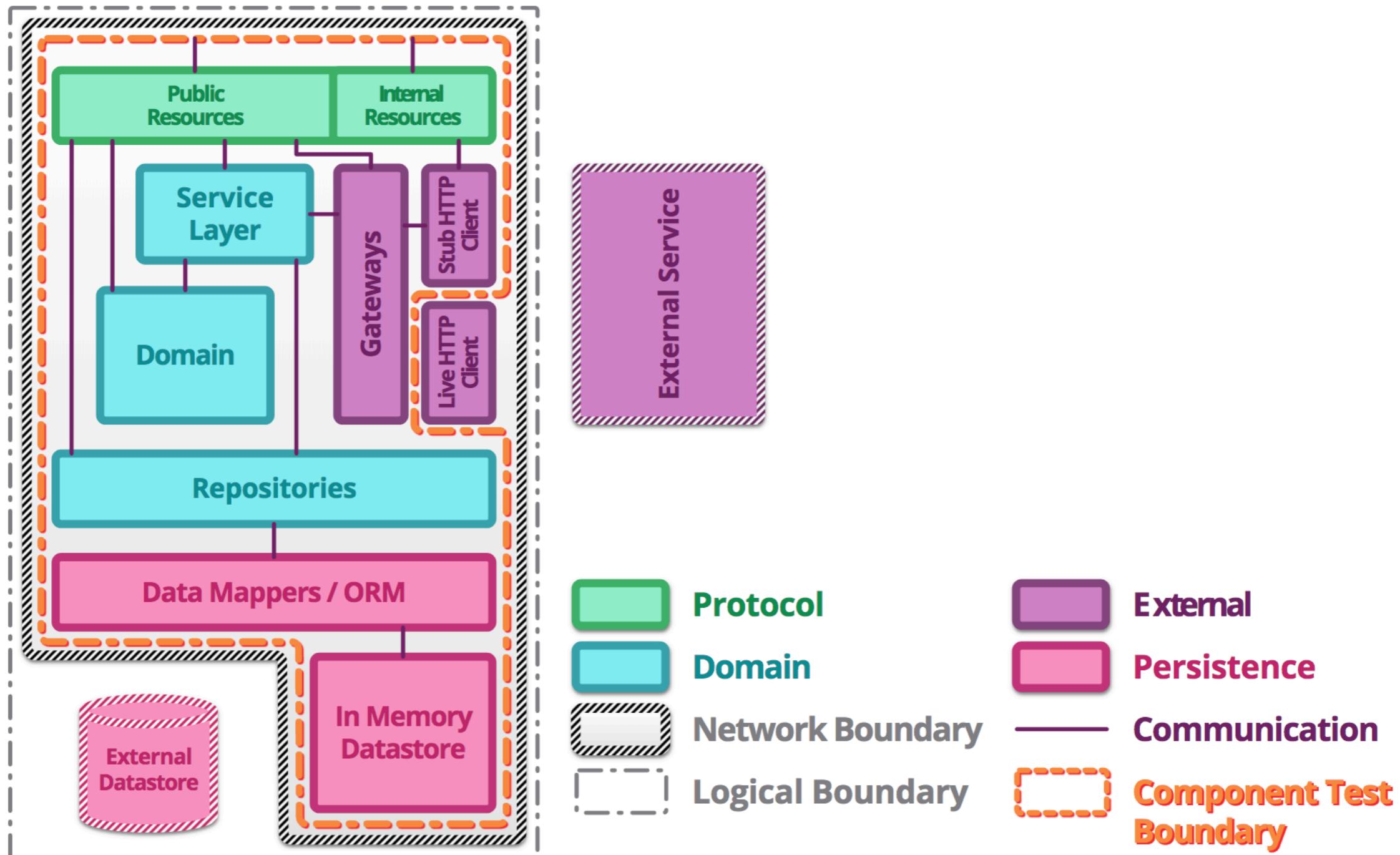
# Unit testing



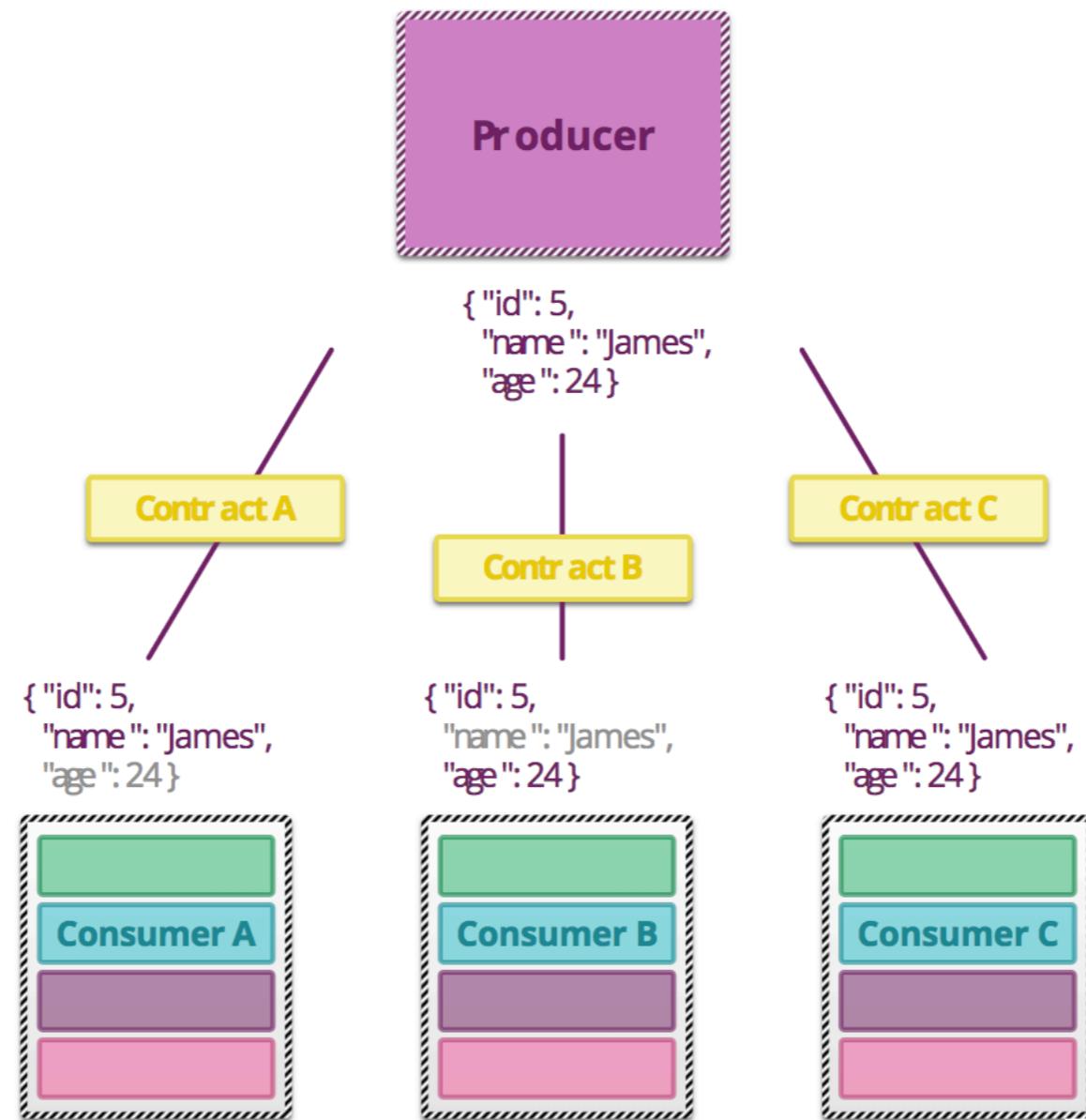
# Integration testing



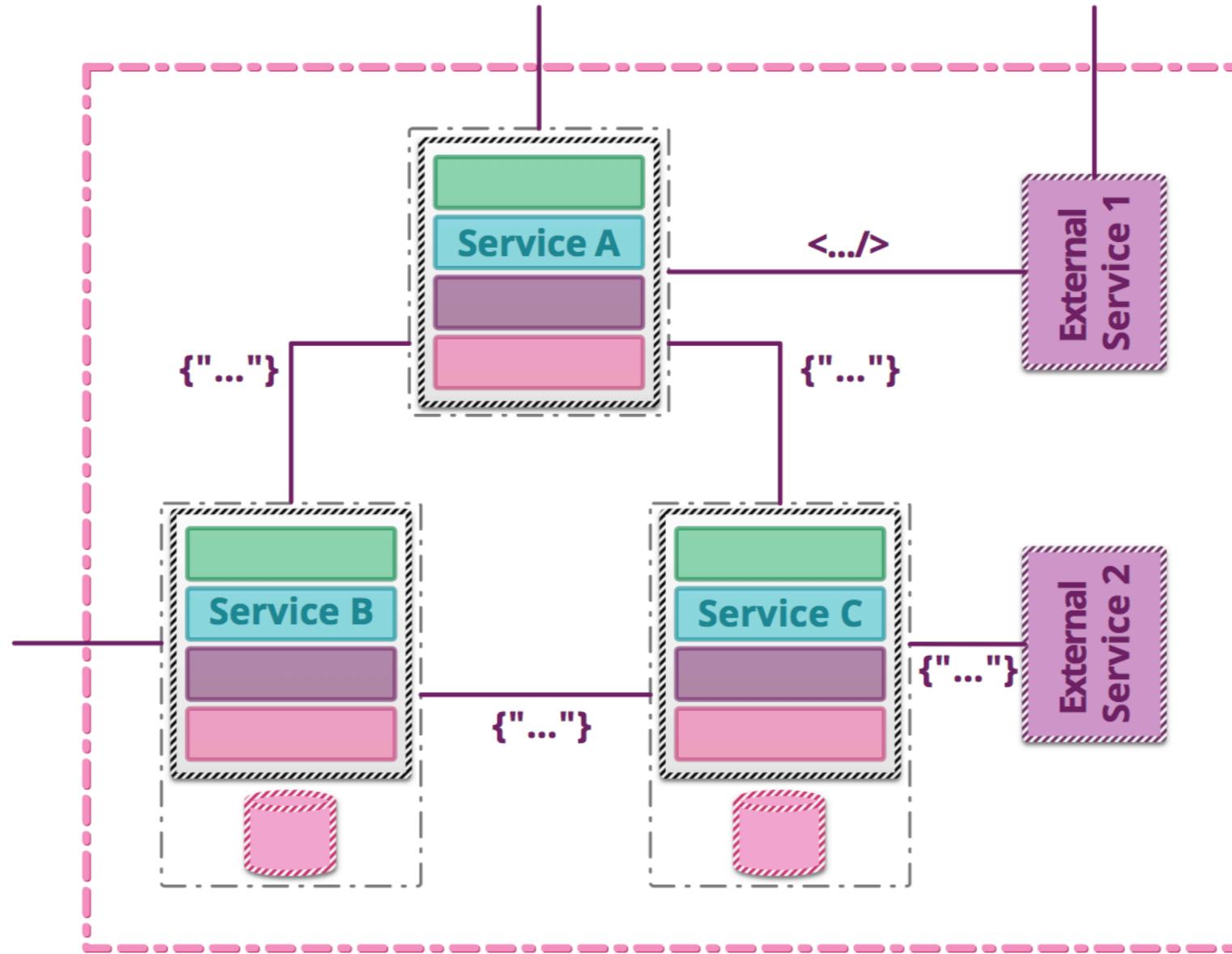
# Component testing



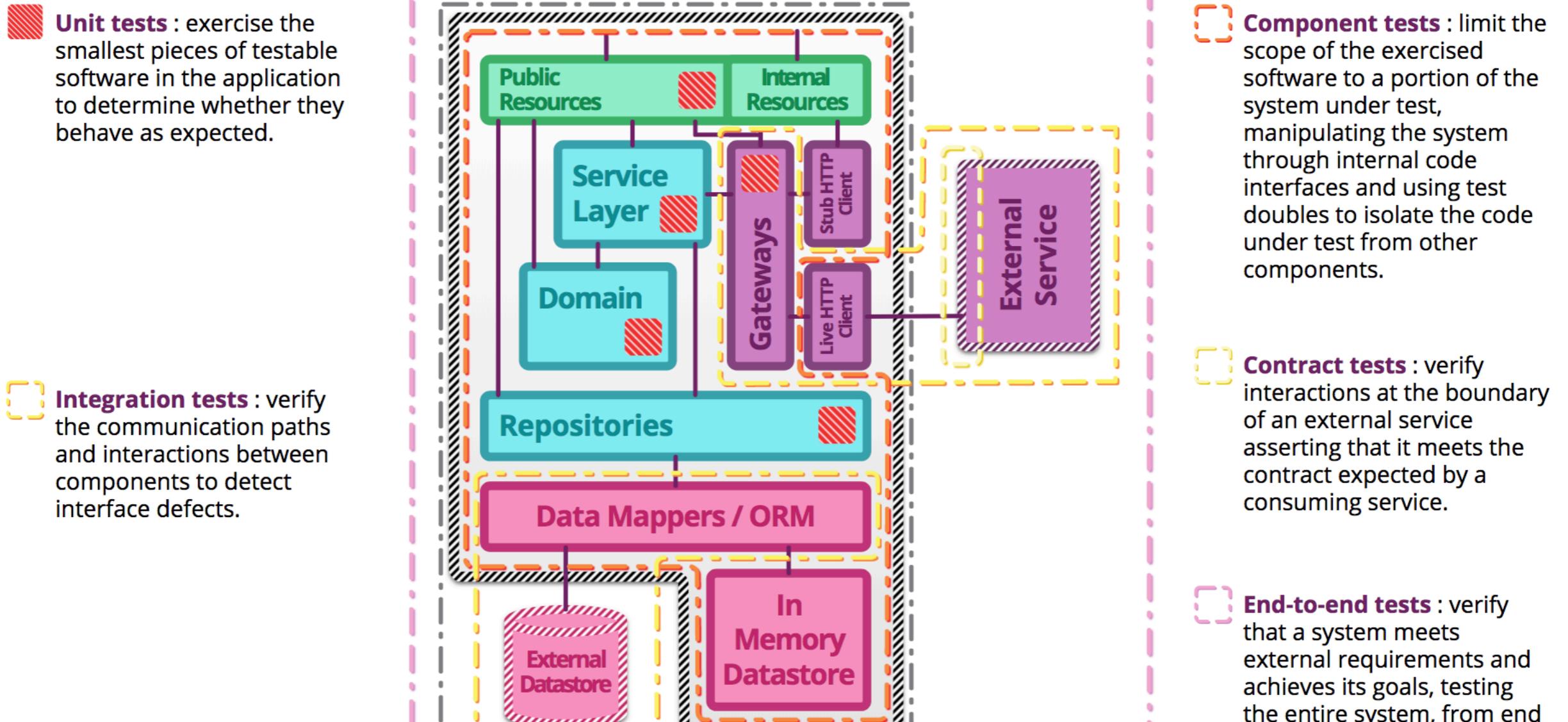
# Contract testing

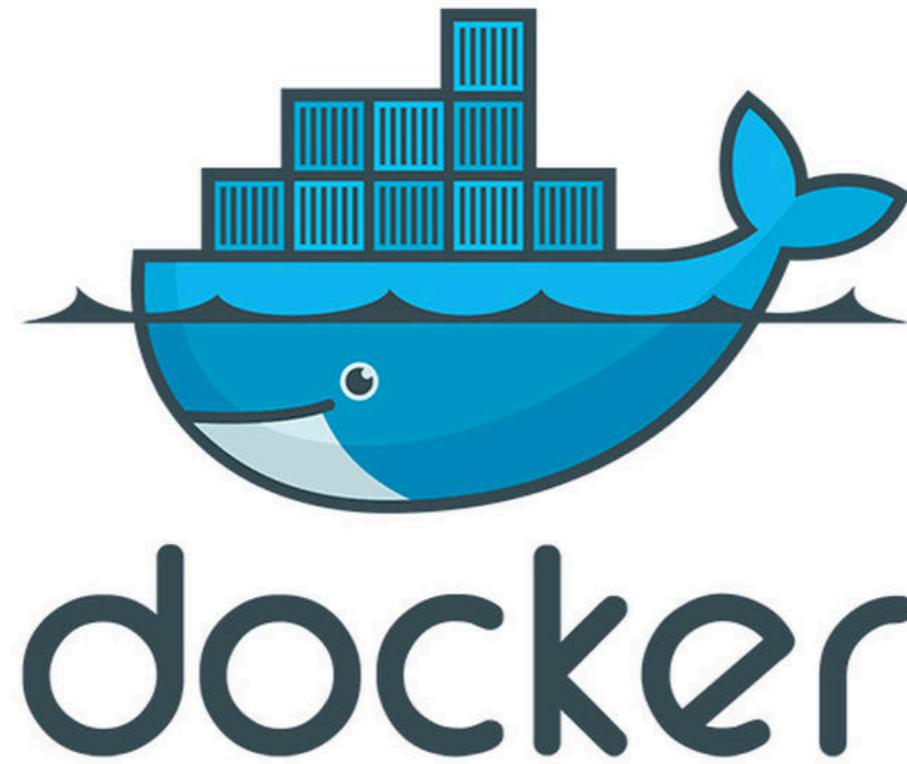


# End-to-End testing



# Summary



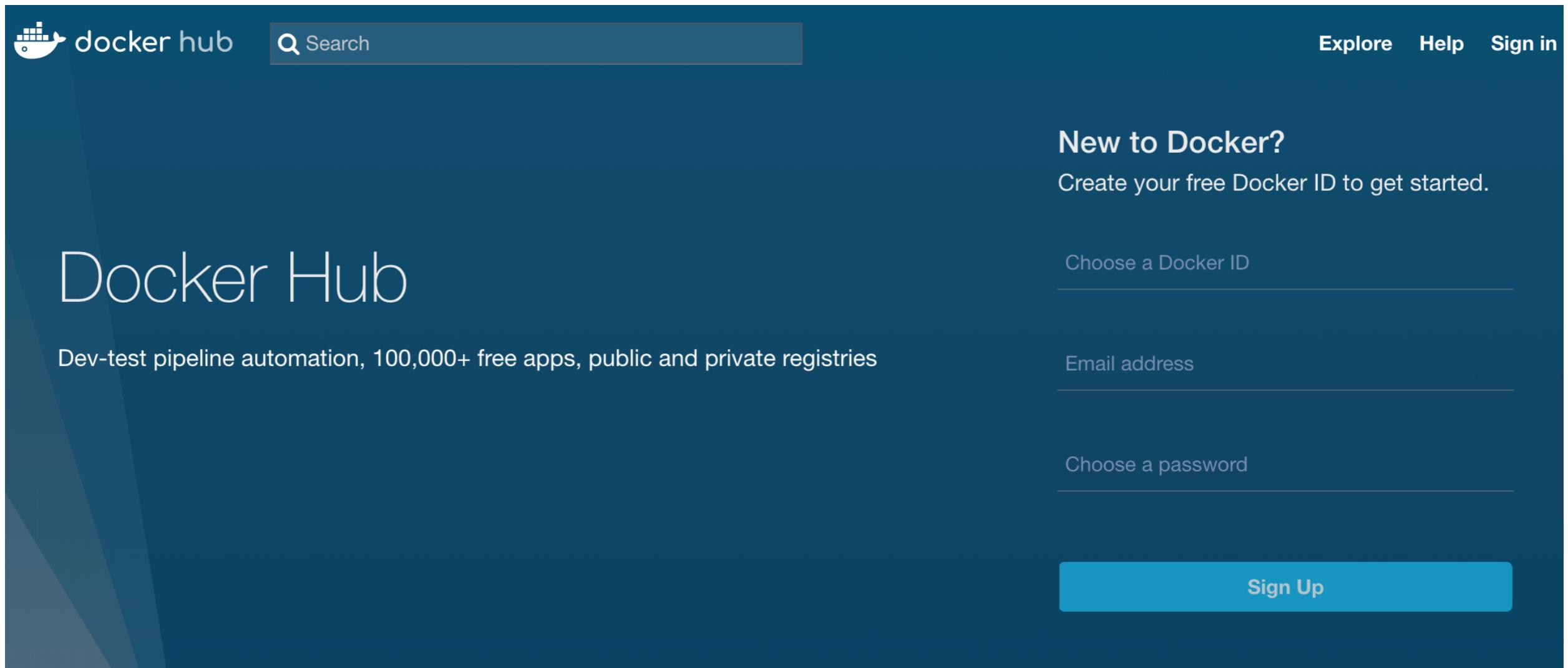


# Working with container



# Choose your image

from hub.docker.com



The screenshot shows the Docker Hub sign-up page. At the top left is the Docker Hub logo and a search bar with the placeholder "Search". To the right are links for "Explore", "Help", and "Sign in". The main title "Docker Hub" is centered above a subtitle: "Dev-test pipeline automation, 100,000+ free apps, public and private registries". On the right side, there's a "New to Docker?" section with a "Create your free Docker ID to get started." link, followed by input fields for "Choose a Docker ID", "Email address", and "Choose a password", each with a horizontal line underneath. A large blue "Sign Up" button is at the bottom.

docker hub  Search

Explore Help Sign in

# Docker Hub

Dev-test pipeline automation, 100,000+ free apps, public and private registries

New to Docker?  
Create your free Docker ID to get started.

Choose a Docker ID

Email address

Choose a password

**Sign Up**



# Images in workshop

Java => OpenJDK

Apache Maven => maven

PostgreSQL => progrès



# Java => OpenJDK

OFFICIAL REPOSITORY

openjdk 

Last pushed: 5 days ago

Repo Info Tags

## Short Description

OpenJDK is an open-source implementation of the Java Platform, Standard Edition

## Docker Pull Command



```
docker pull openjdk
```

## Full Description

Supported tags and respective [Dockerfile](#) links

## Simple Tags

- [10-ea-32-jre-experimental](#), [10-ea-jre-experimental](#), [10-jre-](#)

[https://hub.docker.com/\\_/openjdk/](https://hub.docker.com/_/openjdk/)



# Apache Maven => maven

OFFICIAL REPOSITORY

maven 

Last pushed: 5 days ago

Repo Info Tags

Short Description

Apache Maven is a software project management and comprehension tool.

Docker Pull Command 

`docker pull maven`

Full Description

Supported tags and respective [Dockerfile](#) links

- `3.5.2-jdk-7-alpine` ([jdk-7-alpine/Dockerfile](#))
- `3.5.2-jdk-7-slim` ([jdk-7-slim/Dockerfile](#))
- `3.5.2-slim` ([3.5.2/Dockerfile](#))

[https://hub.docker.com/\\_/maven/](https://hub.docker.com/_/maven/)



# PostgreSQL => progrès

OFFICIAL REPOSITORY

postgres 

Last pushed: 3 days ago

Repo Info Tags

## Short Description

The PostgreSQL object-relational database system provides reliability and data integrity.

## Docker Pull Command



```
docker pull postgres
```

## Full Description

Supported tags and respective [Dockerfile](#) links

- [10.3](#) , [10](#) , [latest](#) ([10/Dockerfile](#))  
[10.3/Dockerfile](#) [10/Dockerfile](#) [latest/Dockerfile](#)

[https://hub.docker.com/\\_/postgres/](https://hub.docker.com/_/postgres/)



# Pull images from Docker Hub

\$docker image pull openjdk:<tag>

\$docker image pull maven:<tag>

\$docker image pull postgres:<tag>



# Create container to run Spring Boot



# Create container to run Spring Boot

```
$docker container run -d  
-v $(pwd)/target/toystore.jar:/xxx/toystore.jar  
-p 8080:8080  
--name web  
openjdk:8-jre java -jar /xxx/toystore.jar
```



# Create new Docker image (1)

from Dockerfile

```
FROM openjdk:8-jre  
COPY ./target/toystore.jar /xxx/toystore.jar  
CMD java -jar /xxx/toystore.jar
```



# Create new Docker image (2)

Build image from Dockerfile

```
$ docker image build -t toystore:0.1 .
```

```
Sending build context to Docker daemon 33.47MB
```

```
Step 1/3 : FROM openjdk:8-jre
```

```
---> e956268fd4ed
```

```
Step 2/3 : COPY ./target/toystore.jar /xxx/toystore.jar
```

```
---> 3dd837b158eb
```

```
Step 3/3 : CMD java -jar /xxx/toystore.jar
```

```
---> Running in 06994b290e74
```

```
Removing intermediate container 06994b290e74
```

```
---> 0bc2054f4ba8
```

```
Successfully built 0bc2054f4ba8
```

```
Successfully tagged toystore:0.1
```



# Create container to run Spring Boot

```
$ docker container run -d -p 8080:8080 toystore:0.1
```

```
$ docker container run -d -p 8081:8080 toystore:0.1
```

```
$ docker container run -d -p 8082:8080 toystore:0.1
```



# **Create container to run Build Maven Project**



# Create container to build

```
$docker container run --rm  
-v $(pwd):/xxx  
-w /xxx  
maven:3.5.2-alpine mvn clean package
```



# Run your application

```
$java -jar target/hello.jar
```

org.postgresql.util.PSQLException: Connection to 127.0.0.1:15432 refused. Check that the postmaster is accepting TCP/IP connections.



# Create container of PostgreSQL



# Create container of PostgreSQL

```
$ docker container run  
  -p 15432:5432  
  -e POSTGRES_USER=user  
  -e POSTGRES_PASSWORD=password  
  postgres:10.3-alpine
```



# Run your application again !!

\$java -jar target/hello.jar

```
Hibernate: create sequence hibernate_sequence start 1 increment 1
Hibernate: create table person (id int4 not null, first_name varchar(255), last_name varchar(255), primary key (id))
2018-03-06 19:45:41.115  INFO 56495 --- [           main] o.h.t.schema.internal.SchemaCreatorImpl : HHH000476: Executing import script 'org.hibernate.tool.schema.internal.exec.ScriptSourceNonExistentImpl@7e990ed7'
2018-03-06 19:45:41.121  INFO 56495 --- [           main] j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityManagerFactory for persistence unit 'default'
2018-03-06 19:45:42.592  INFO 56495 --- [           main] s.w.s.m.m.a.RequestMappingHandlerAdapter : Looking for @ControllerAdvice: org.springframework.boot.web.servlet.context.AnnotationConfigServletWebServerApplicationContext@7cef4e59: startup date [Tue Mar 06 19:45:34 ICT 2018]; root of context hierarchy
2018-03-06 19:45:42.746  WARN 56495 --- [           main] aWebConfiguration$JpaWebMvcConfiguration : spring.jpa.open-in-view is enabled by default. Therefore, database queries may be performed during view rendering. Explicitly configure spring.jpa.open-in-view to disable this warning
2018-03-06 19:45:42.857  INFO 56495 --- [           main] s.w.s.m.m.a.RequestMappingHandlerMapping : Mapped "{[/hello/{name}],methods=[GET]}" onto public toystore.domain.Hello toystore.controller.HelloController.sayHi(java.lang.String)
2018-03-06 19:45:42.881  INFO 56495 --- [           main] s.w.s.m.m.a.RequestMappingHandlerMapping : Mapped "{[/hello/data/{name}],methods=[GET]}" onto public toystore.domain.Hello toystore.controller.HelloWithRepositoryController.sayHi(java.lang.String)
2018-03-06 19:45:42.889  INFO 56495 --- [           main] s.w.s.m.m.a.RequestMappingHandlerMapping : Mapped "{[/error]}" onto public org.springframework.http.ResponseEntity<java.util.Map<java.lang.String, java.lang.Object>> org.springframework.boot.autoconfigure.web.servlet.error.BasicErrorController.error(javax.servlet.http.HttpServletRequest)
2018-03-06 19:45:42.893  INFO 56495 --- [           main] s.w.s.m.m.a.RequestMappingHandlerMapping : Mapped "{[/error],produces=[text/html]}" onto public org.springframework.web.servlet.ModelAndView org.springframework.boot.autoconfigure.web.servlet.error.BasicErrorController.errorHtml(javax.servlet.http.HttpServletRequest,javax.servlet.http.HttpServletResponse)
2018-03-06 19:45:43.046  INFO 56495 --- [           main] o.s.w.s.handler.SimpleUrlHandlerMapping : Mapped URL path [/webjars/**] onto handler of type [class org.springframework.web.servlet.resource.ResourceHttpRequestHandler]
2018-03-06 19:45:43.046  INFO 56495 --- [           main] o.s.w.s.handler.SimpleUrlHandlerMapping : Mapped URL path [/**] onto handler of type [class org.springframework.web.servlet.resource.ResourceHttpRequestHandler]
2018-03-06 19:45:43.168  INFO 56495 --- [           main] o.s.w.s.handler.SimpleUrlHandlerMapping : Mapped URL path [/**/favicon.ico] onto handler of type [class org.springframework.web.servlet.resource.ResourceHttpRequestHandler]
2018-03-06 19:45:43.783  INFO 56495 --- [           main] o.s.j.e.a.AnnotationMBeanExporter : Registering beans for JMX exposure on startup
2018-03-06 19:45:43.785  INFO 56495 --- [           main] o.s.j.e.a.AnnotationMBeanExporter : Bean with name 'dataSource' has been autodetected for JMX exposure
2018-03-06 19:45:43.793  INFO 56495 --- [           main] o.s.j.e.a.AnnotationMBeanExporter : Located MBean 'dataSource': registering with JMX server as MBean [com.zaxxer.hikari:name=dataSource,type=HikariDataSource]
2018-03-06 19:45:43.873  INFO 56495 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 8080 (http) with context path ''
2018-03-06 19:45:43.877  INFO 56495 --- [           main] toystore.ToyStoreApplication : Started ToyStoreApplication in 10.18 seconds (JVM running for 10.912)
2018-03-06 19:45:47.875  INFO 56495 --- [nio-8080-exec-1] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring FrameworkServlet 'dispatcherServlet'
2018-03-06 19:45:47.876  INFO 56495 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet : FrameworkServlet 'dispatcherServlet': initialization started
2018-03-06 19:45:47.917  INFO 56495 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet : FrameworkServlet 'dispatcherServlet': initialization completed in 41 ms
```



# Monitoring and Metric



# Metric in Spring Boot

**Spring Boot Actuator for Spring Boot 1.x**  
**MicroMeter for Spring Boot 2.0**



# Spring Boot Actuator (1)

Add library to pom.xml

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-actuator</artifactId>
</dependency>
```



# Spring Boot Actuator (2)

Enabled endpoint in application.properties

```
info.app.name=Toy Store
```

```
info.app.description=This is my first spring boot application
```

```
info.app.version=1.0.0
```

```
management.endpoints.web.exposure.include=health,info,metrics,httptrace
```



# Spring Boot Actuator (3)

List of endpoints = /actuator/

```
← → ⌂ ⓘ localhost:8080/actuator/
```

```
{  
  - _links: {  
    - self: {  
        href: "http://localhost:8080/actuator",  
        templated: false  
      },  
    - health: {  
        href: "http://localhost:8080/actuator/health",  
        templated: false  
      },  
    - info: {  
        href: "http://localhost:8080/actuator/info",  
        templated: false  
      },  
    - metrics-requiredMetricName: {  
        href: "http://localhost:8080/actuator/metrics/{requiredMetricName}",  
        templated: true  
      },  
    - metrics: {  
        href: "http://localhost:8080/actuator/metrics",  
        templated: false  
      },  
    - httptrace: {  
        href: "http://localhost:8080/actuator/httptrace",  
        templated: false  
      }  
  }  
}
```



# Spring Boot Actuator (4)

Info endpoint = /actuator/info

```
← → ⌂ ⓘ localhost:8080/actuator/info

{
  - app: {
      name: "Toy Store",
      description: "This is my first spring boot application",
      version: "1.0.0"
    }
}
```



# Spring Boot Actuator (5)

Info endpoint = /actuator/info

```
← → ⌂ ⓘ localhost:8080/actuator/info

{
  - app: {
      name: "Toy Store",
      description: "This is my first spring boot application",
      version: "1.0.0"
    }
}
```



# Spring Boot Actuator (6)

## Info endpoint = /actuator/httptrace

```
← → ⌂ ⓘ localhost:8080/actuator/httptrace

{
  - traces: [
    - {
      timestamp: "2018-03-06T13:33:02.800Z",
      principal: null,
      session: null,
      - request: {
        method: "GET",
        uri: "http://localhost:8080/prometheus",
        - headers: {
          - host: [
            "localhost:8080"
          ],
          - user-agent: [
            "Prometheus/2.0.0"
          ],
          - accept: [
            "text/plain;version=0.0.4;q=1,*/*;q=0.1"
          ],
          - accept-encoding: [
            "gzip"
          ],
          - x-prometheus-scrape-timeout-seconds: [
            "5.000000"
          ]
        },
        remoteAddress: null
      },
    }
  ]
}
```



# Spring Boot Actuator (7)

List of metrics endpoint = /actuator/metrics

```
← → ⌂ ⓘ localhost:8080/actuator/metrics

{
  - names: [
    "jvm.buffer.memory.used",
    "jvm.memory.used",
    "jvm.gc.memory.allocated",
    "jvm.memory.committed",
    "http.server.requests",
    "jdbc.connections.min",
    "tomcat.sessions.created",
    "tomcat.sessions.expired",
    "hikaricp.connections.usage",
    "tomcat.global.request.max",
    "tomcat.global.error",
    "jvm.gc.max.data.size",
    "logback.events",
    "system.cpu.count",
    "jvm.memory.max",
    "jdbc.connections.active",
    "jvm.buffer.total.capacity",
    "jvm.buffer.count",
    "process.files.max",
    "jvm.threads.daemon",
```



# Spring Boot Actuator (8)

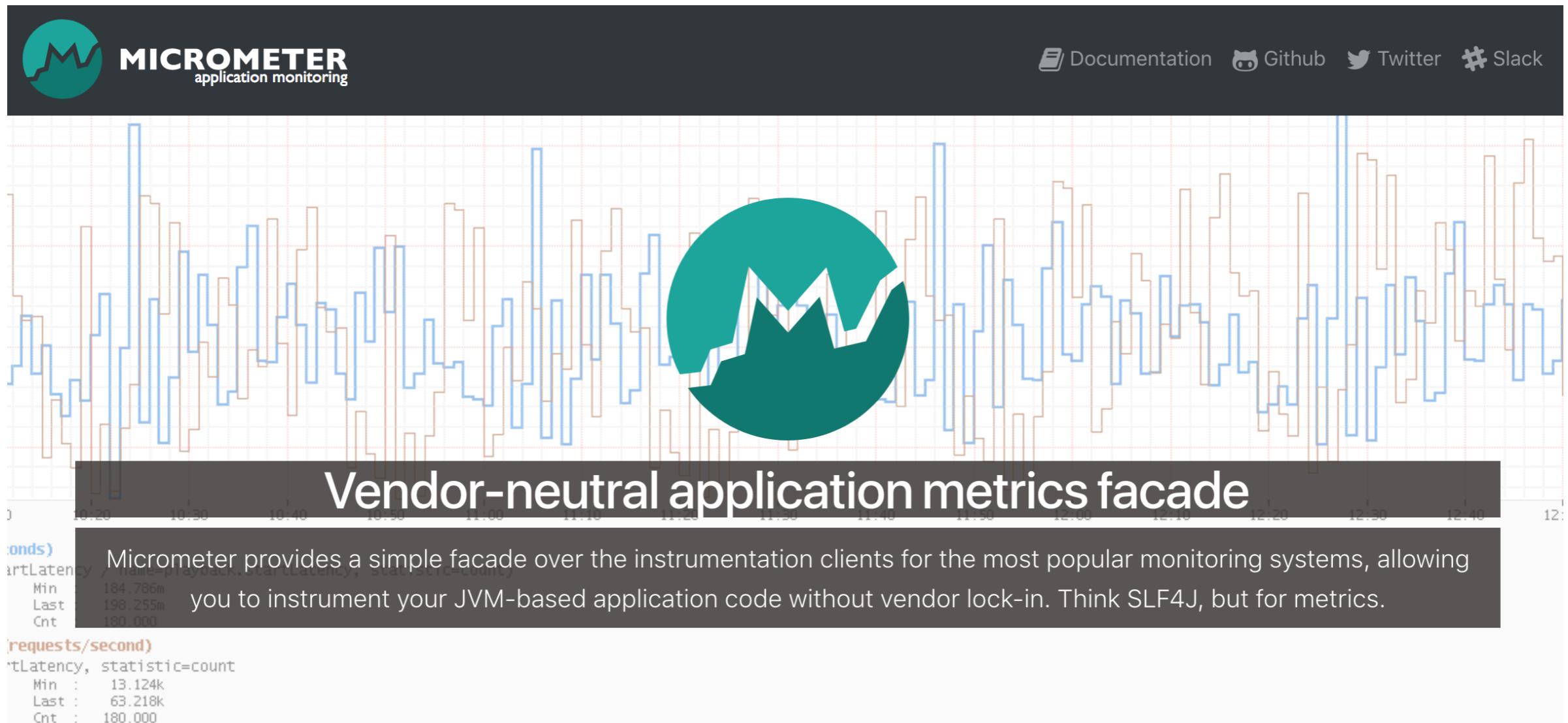
## /actuator/metrics/http.server.requests

```
← → ⌂ ⓘ localhost:8080/actuator/metrics/http.server.requests

{
  name: "http.server.requests",
  - measurements: [
    - {
      statistic: "COUNT",
      value: 269
    },
    - {
      statistic: "TOTAL_TIME",
      value: 1.1072010200000002
    },
    - {
      statistic: "MAX",
      value: 0.04373569
    }
  ],
  - availableTags: [
    - {
      tag: "exception",
      - values: [
        "None"
      ]
    },
    - {
      tag: "method",
      - values: [
        "GET"
      ]
    },
  ],
}
```



# Spring Boot 2.0 with MicroMeter



<https://micrometer.io/>



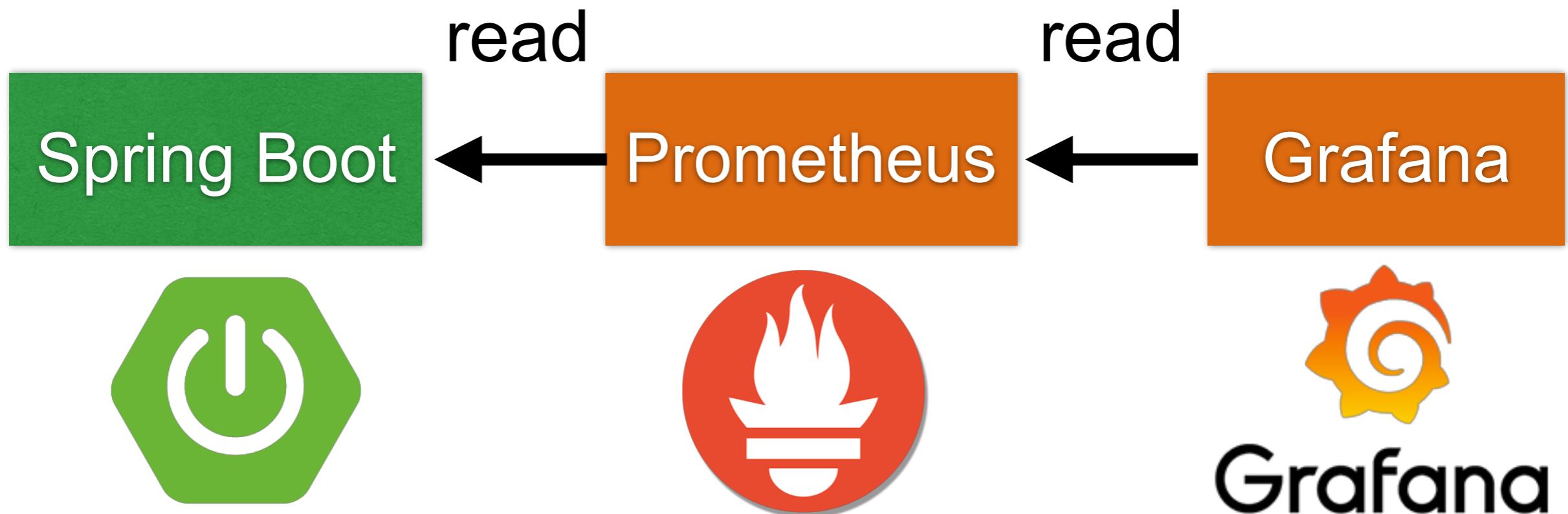
# Keep and Visualize Metric of Spring Boot Services



# Sample Architecture



# Sample Architecture



# Service metric for Prometheus



# Enable Prometheus (1)

Add library to pom.xml

```
<dependency>
    <groupId>io.micrometer</groupId>
    <artifactId>micrometer-registry-prometheus</artifactId>
    <version>1.0.1</version>
</dependency>
```



# Enable Prometheus (2)

Enabled endpoint in application.properties

```
management.endpoints.web.exposure.include  
=....,prometheus
```



# Enable Prometheus (3)

New endpoint = actuator/prometheus

```
← → ⌂ ⓘ localhost:8080/actuator/prometheus

# HELP jvm_memory_used_bytes The amount of used memory
# TYPE jvm_memory_used_bytes gauge
jvm_memory_used_bytes{area="nonheap",id="Code Cache",} 1.49056E7
jvm_memory_used_bytes{area="nonheap",id="Metaspace",} 5.6766712E7
jvm_memory_used_bytes{area="nonheap",id="Compressed Class Space",} 7617096.0
jvm_memory_used_bytes{area="heap",id="PS Eden Space",} 1.7135864E7
jvm_memory_used_bytes{area="heap",id="PS Survivor Space",} 1.6235192E7
jvm_memory_used_bytes{area="heap",id="PS Old Gen",} 2.1936456E7
# HELP hikaricp_connections_idle Idle connections
# TYPE hikaricp_connections_idle gauge
hikaricp_connections_idle{pool="HikariPool-1",} NaN
# HELP tomcat_threads_config_max
# TYPE tomcat_threads_config_max gauge
tomcat_threads_config_max{name="http-nio-8080",} 200.0
# HELP tomcat_servlet_error_total
# TYPE tomcat_servlet_error_total counter
tomcat_servlet_error_total{name="default",} 0.0
# HELP jvm_threads_peak The peak live thread count since the Java virtual machine start
# TYPE jvm_threads_peak gauge
jvm_threads_peak 28.0
# HELP hikaricp_connections_pending Pending threads
# TYPE hikaricp_connections_pending gauge
hikaricp_connections_pending{pool="HikariPool-1",} NaN
# HELP system_cpu_count The number of processors available to the Java virtual machine
```



# Keep data in Prometheus

<https://prometheus.io/>



# Prometheus



Prometheus

DOCS

DOWNLOAD

COMMUNITY

BLOG



## From metrics to insight

Power your metrics and alerting with a leading  
open-source monitoring solution.

GET STARTED

DOWNLOAD

Prometheus v2.0 is available now — [Read the announcement blog post!](#)

<https://prometheus.io/>



Spring Boot

© 2017 - 2018 Siam Chamnankit Company Limited. All rights reserved.

214

# Prometheus

PUBLIC | AUTOMATED BUILD

[prom/prometheus](#) 

Last pushed: 17 hours ago

[Repo Info](#) [Tags](#) [Dockerfile](#) [Build Details](#)

## Short Description

Short description is empty for this repo.

## Full Description

Prometheus 

   
   
docker pulls 

Visit [prometheus.io](https://prometheus.io) for the full documentation, examples and guides.

Prometheus is a systems and service monitoring system. It collects metrics

## Docker Pull Command

`docker pull prom/prometheus`

## Owner



prom

## Source Repository

 [prometheus/prometheus](#)

<https://hub.docker.com/r/prom/prometheus/>



# Create container of Prometheus

```
$ docker container run --rm  
  -p 9090:9090  
  -v $(pwd)/prometheus.yml:/etc/prometheus/  
prometheus.yml  
  --name monitor prom/prometheus
```



# Check Data in Prometheus

http://localhost:9090/

The screenshot shows the Prometheus web interface at the URL `http://localhost:9090/graph`. The interface has a dark header bar with the Prometheus logo and navigation links for Alerts, Graph, Status, and Help. Below the header is a checkbox labeled "Enable query history". A text input field is labeled "Expression (press Shift+Enter for newlines)" with the placeholder "- insert metric at cursor -". There are two tabs: "Graph" (which is selected) and "Console". A table below shows one row with the "Element" column containing "no data" and the "Value" column empty. At the bottom left is a blue button labeled "Add Graph".



# Check Target in Prometheus

Status -> Targets

The screenshot shows the Prometheus web interface at the URL `localhost:9090/targets`. The top navigation bar includes links for Prometheus, Alerts, Graph, Status, and Help. The main section is titled "Targets" and features a checkbox labeled "Only unhealthy jobs" which is unchecked. Below this, a summary row indicates "spring-boot (1/1 up)" with a "show less" button. A table lists one target endpoint: `http://10.10.99.59:8080/actuator/prometheus`, which is marked as "UP" and has the label `instance="10.10.99.59:8080"`. The last scrape was 2.355s ago.

Endpoint	State	Labels	Last Scrape	Error
<a href="http://10.10.99.59:8080/actuator/prometheus">http://10.10.99.59:8080/actuator/prometheus</a>	UP	instance="10.10.99.59:8080"	2.355s ago	



# Show data in Grafana

<https://grafana.com/>



# Grafana

The screenshot shows the Grafana Labs website. At the top, there's a navigation bar with links for Plugins (currently installed), Grafana, GrafanaCloud, Services, Dashboards, Plugins, and Get Grafana. Below the navigation, a large central text area features the slogan "The open platform for beautiful analytics and monitoring". To the left, there's a card for "Grafana TestData" by Grafana Project, which includes a network icon. To the right, there are cards for "Kentik Connect Pro" (with a Kentik logo) and "NS1 for Grafana" by NS1. Further down, there's a card for "kubernetes" by Raintank Inc. (with a Kubernetes logo). A prominent orange button labeled "Get Grafana" is centered in the middle. At the bottom of the main content area, the Grafana logo is displayed.

<https://grafana.com/>



# Grafana

PUBLIC REPOSITORY

[grafana/grafana](#) 

Last pushed: 25 minutes ago

[Repo Info](#)

[Tags](#)

## Short Description

The official Grafana docker container

## Full Description

### Grafana Docker image

This project builds a Docker image with the latest master build of Grafana.

## Running your Grafana container

Start your container binding the external port 3000 .

```
docker run -d --name=grafana -p 3000:3000 grafana/grafana
```

## Docker Pull Command

```
docker pull grafana/grafana
```

## Owner



grafana

<https://hub.docker.com/r/grafana/grafana/>



# Create container of Grafana

```
$docker container run  
--name=grafana  
-p 3000:3000 grafana/grafana
```



# Grafana Dashboard

<https://grafana.com/dashboards/4701>

All dashboards » [JVM \(Micrometer\)](#)



**JVM (Micrometer)** by [mweirauch](#)

[DASHBOARD](#)

Dashboard for Micrometer instrumented applications (Java, Spring Boot)  
Last updated: 21 days ago

[Overview](#) [Revisions](#)



**Get this dashboard:**

[4701](#) [Copy ID to Clipboard](#)

A dashboard for [Micrometer](#) instrumented applications (Java, Spring Boot).

**Features**

- JVM memory
- Process memory (provided by [micrometer-jvm-extras](#))
- CPU-Usage, Load, Threads, File Descriptors, Log Events
- JVM Memory Pools (Heap, Non-Heap)
- Garbage Collection

**Dependencies:**

 [GRAFANA 4.6.3](#)

 [GRAPH](#)



# Take to your home

Always improve, always practice



# Are you too busy to improve?



# Thank you Q/A

