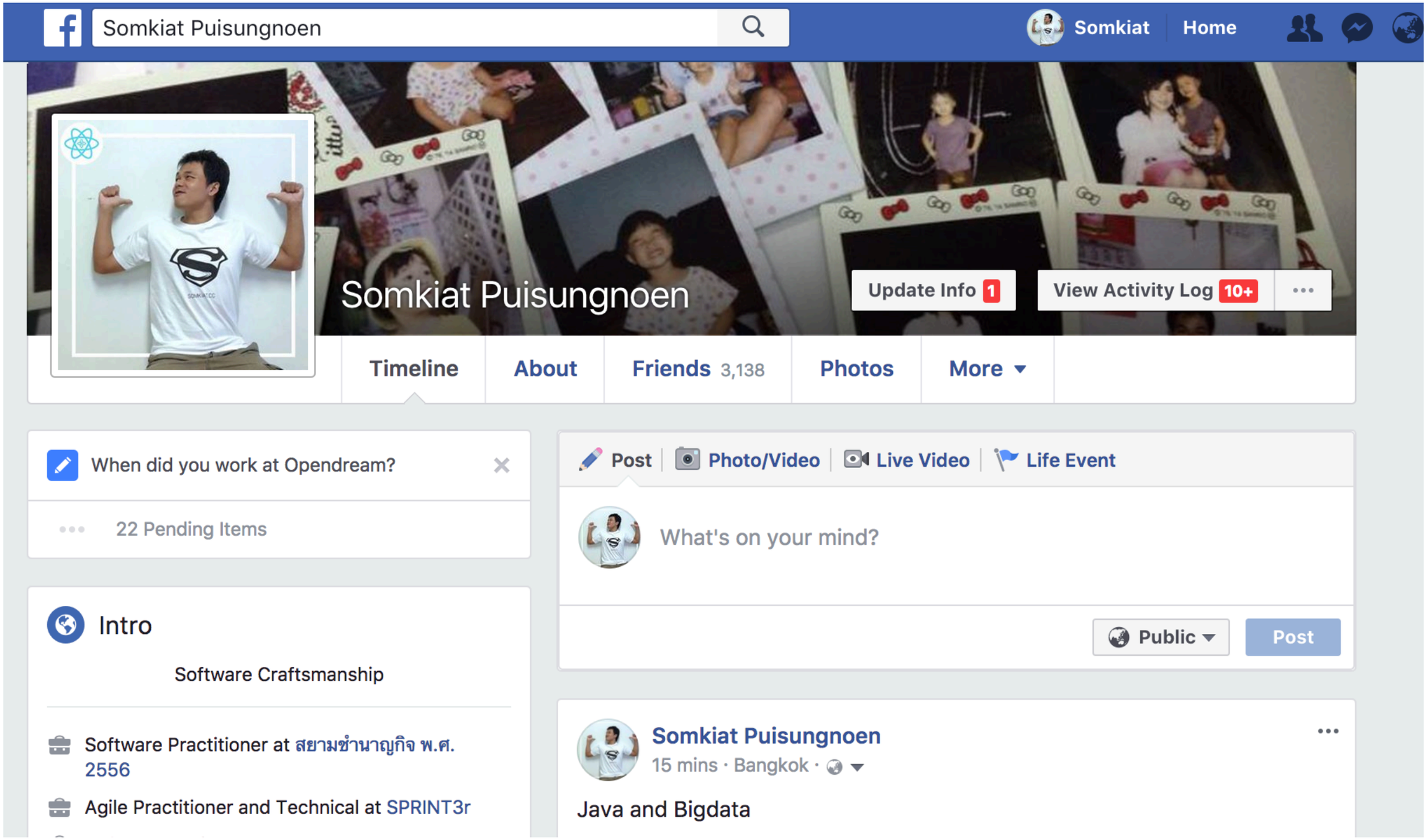
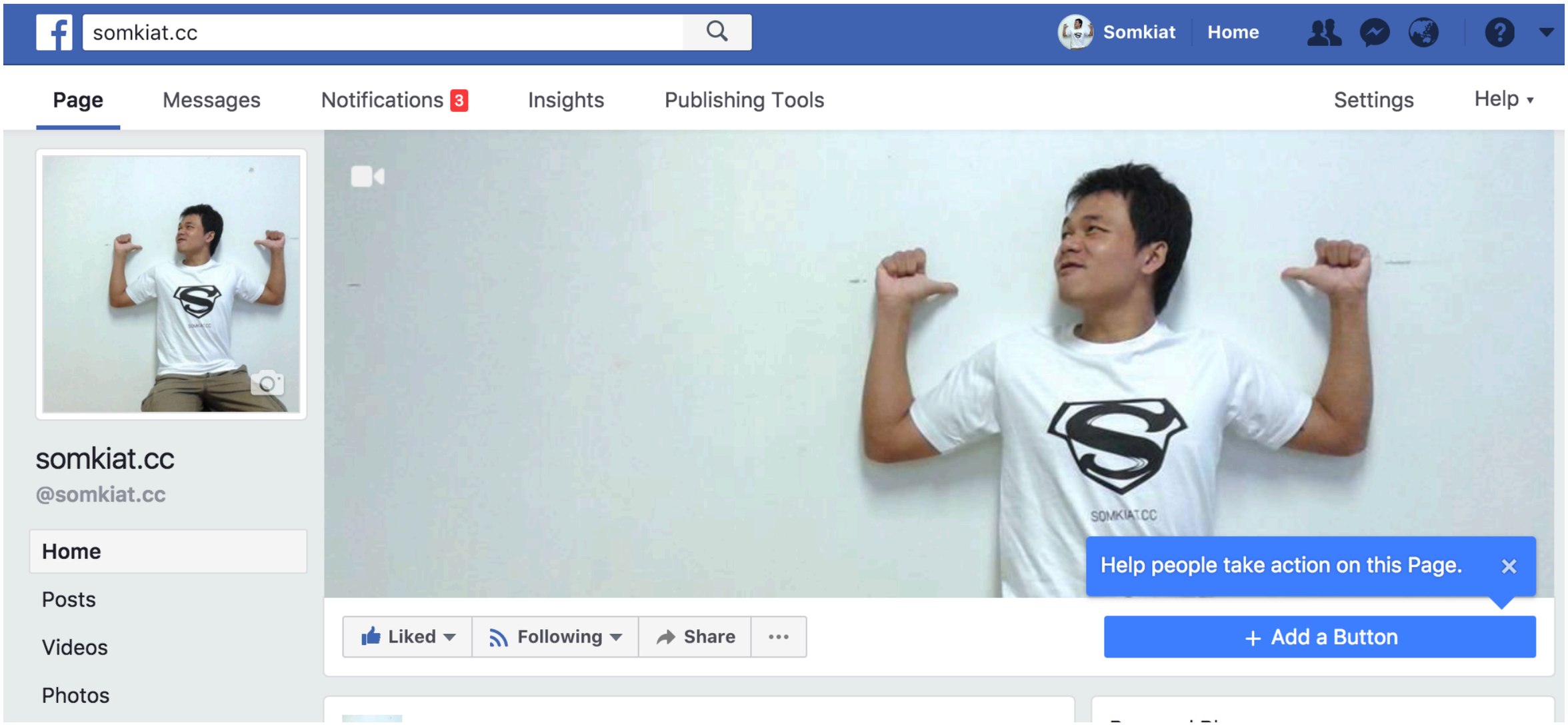
สยามจำนากุกิจ
SIAM CHAMNAN KIT

Spring Boot Workshop



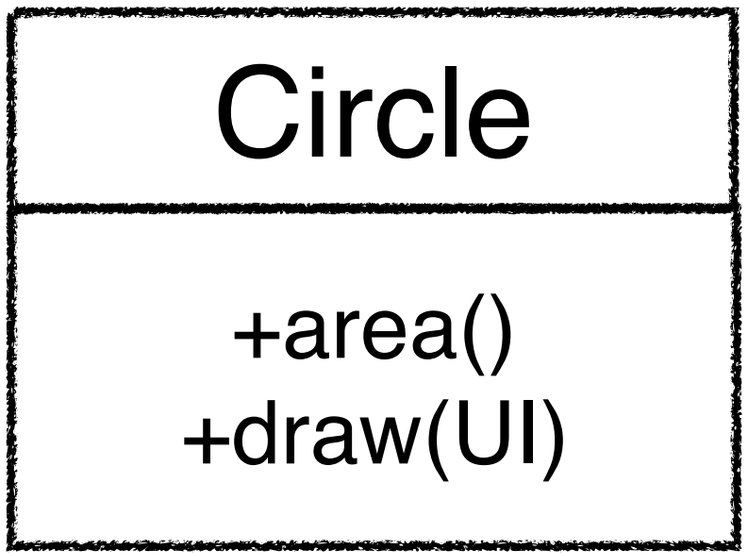


SOLID principle

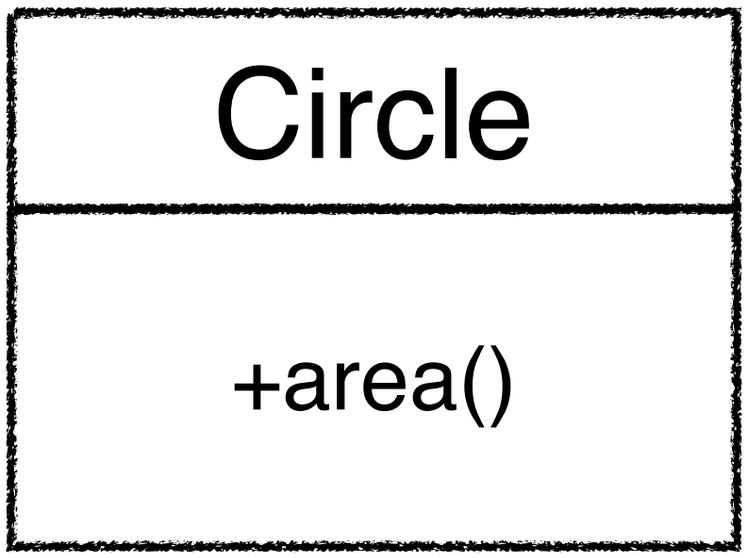
SOLID principle

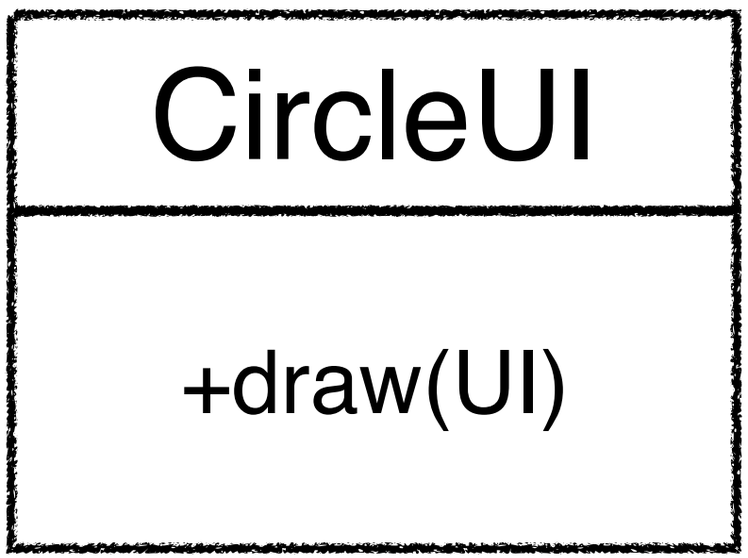
1. Single Responsibility Principle

Example



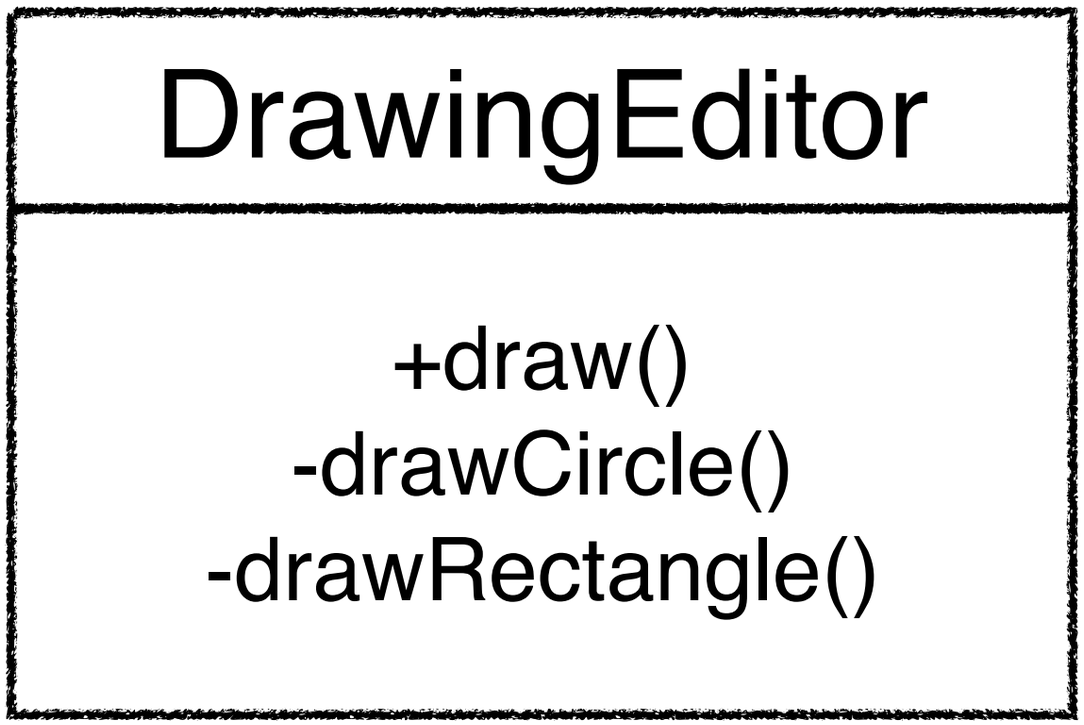
Example



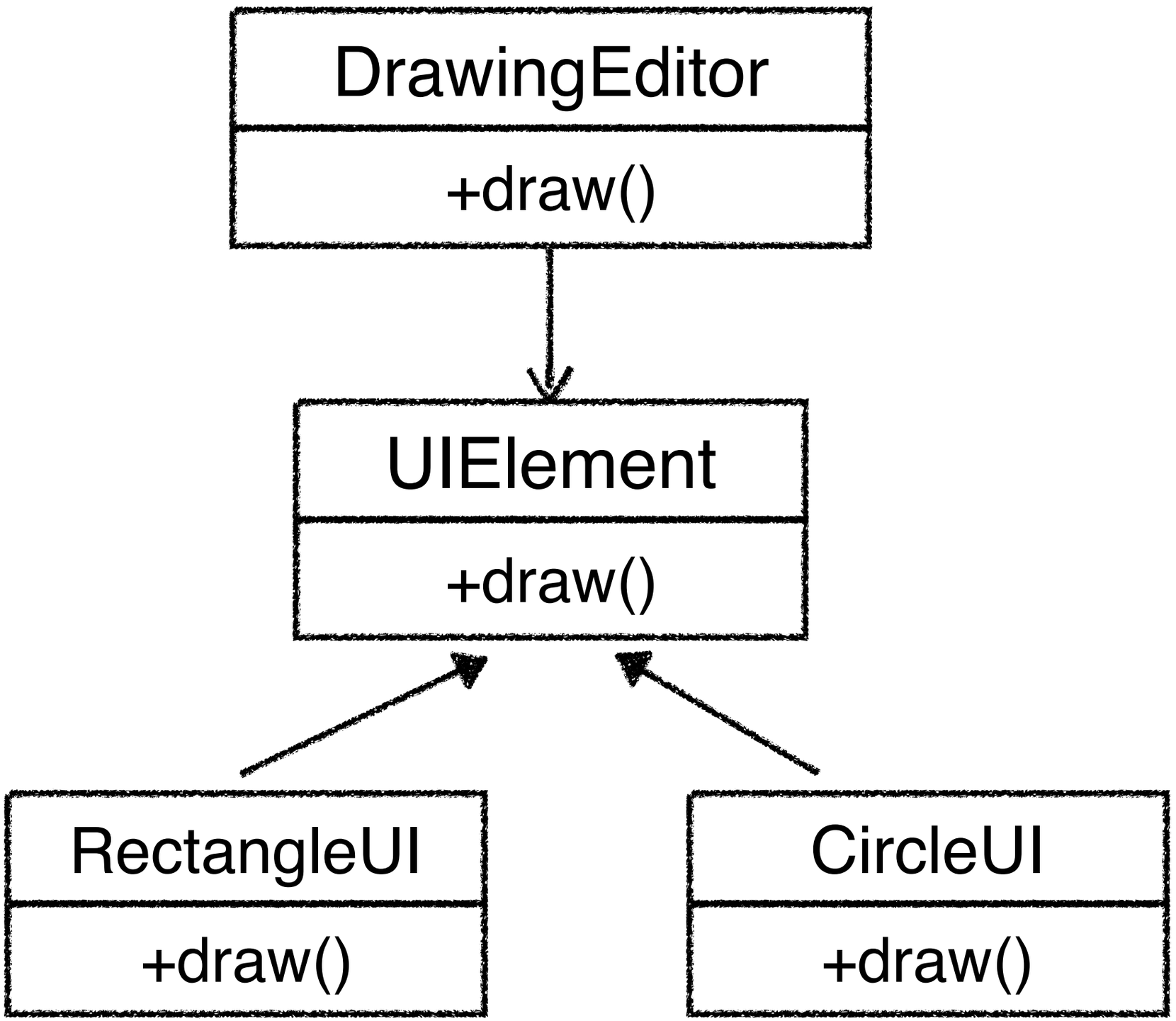


2. Open-Closed Principle

Example

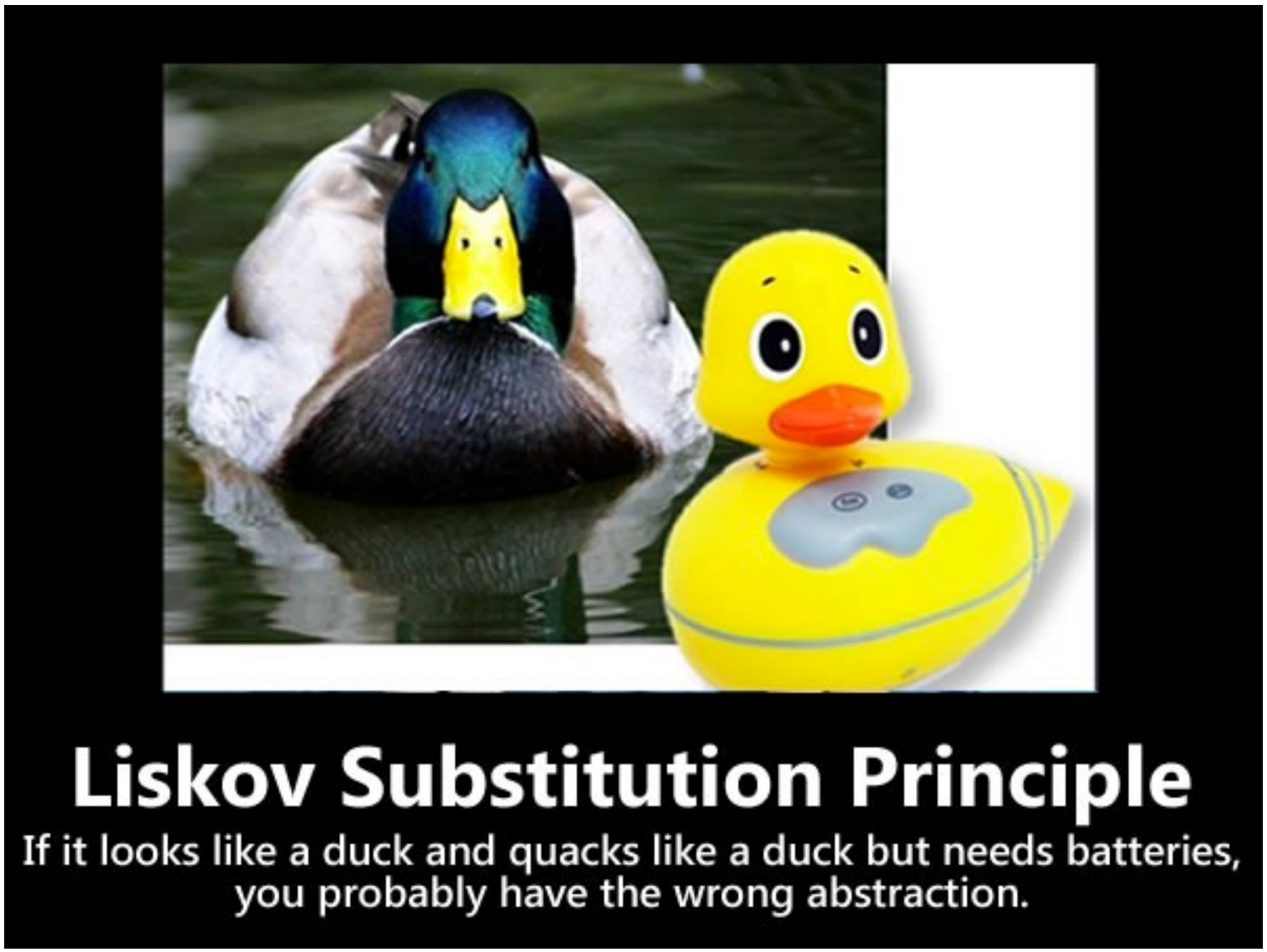


Example



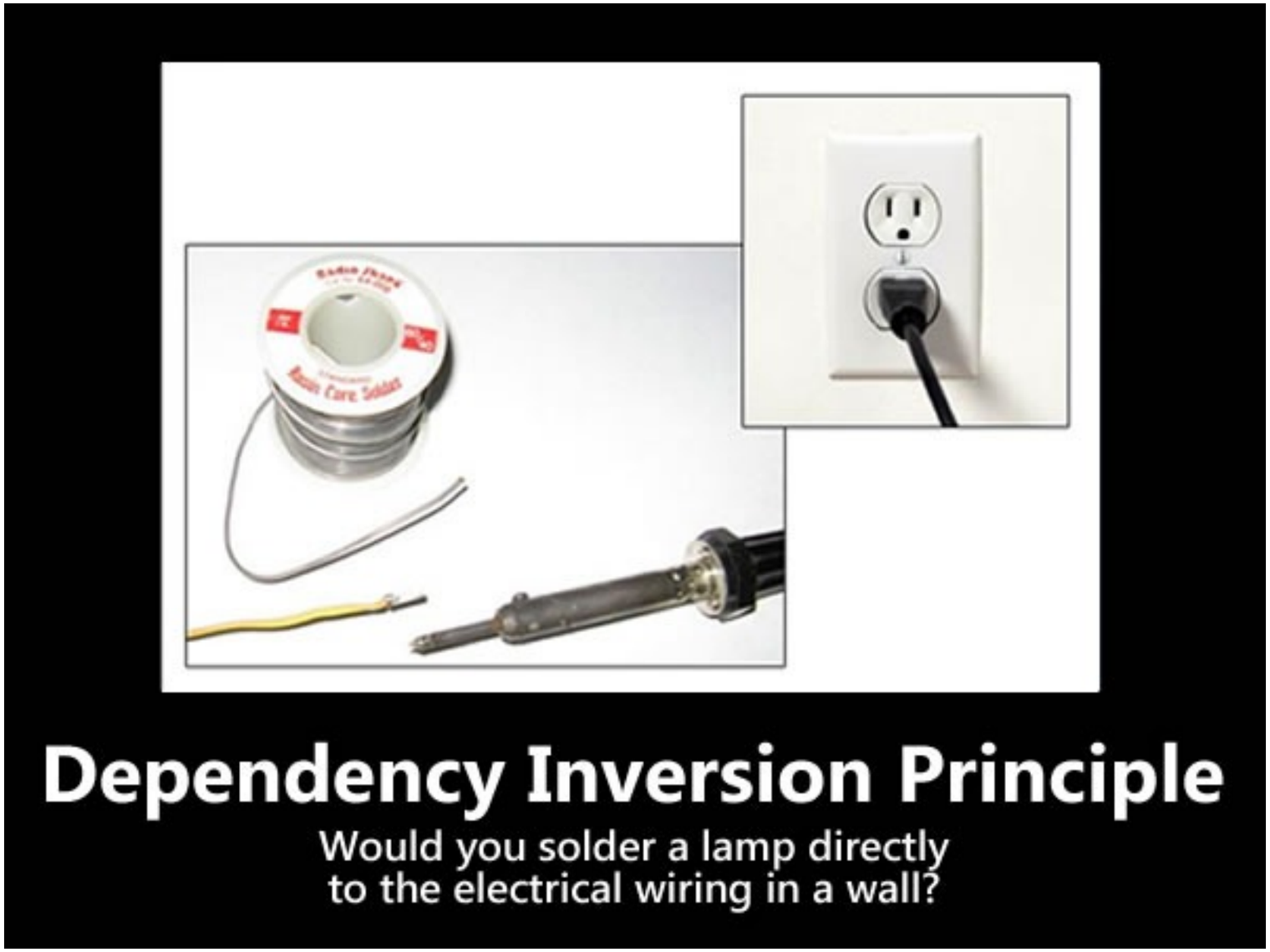
Workshop

3. Liskov Substitution Principle



4. Interface Segregation Principle

5. Dependency Inversion Principle



Spring Boot

Agenda

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

REST

REpresentation **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

Goals

Scalability
Generality of interfaces
Independent deployment of components

RESTful service

REST Request Messages

RESTful request is typically in form of **U**niform **R**esource **I**dentifiers (URI)

REST Request Messages

RESTful request is typically in form of **U**niform **R**esource **I**dentifiers (URI)

Structure of URI depend on specific service

REST Request Messages

RESTful request is typically in form of **U**niform **R**esource **I**dentifiers (URI)

Structure of URI depend on specific service

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

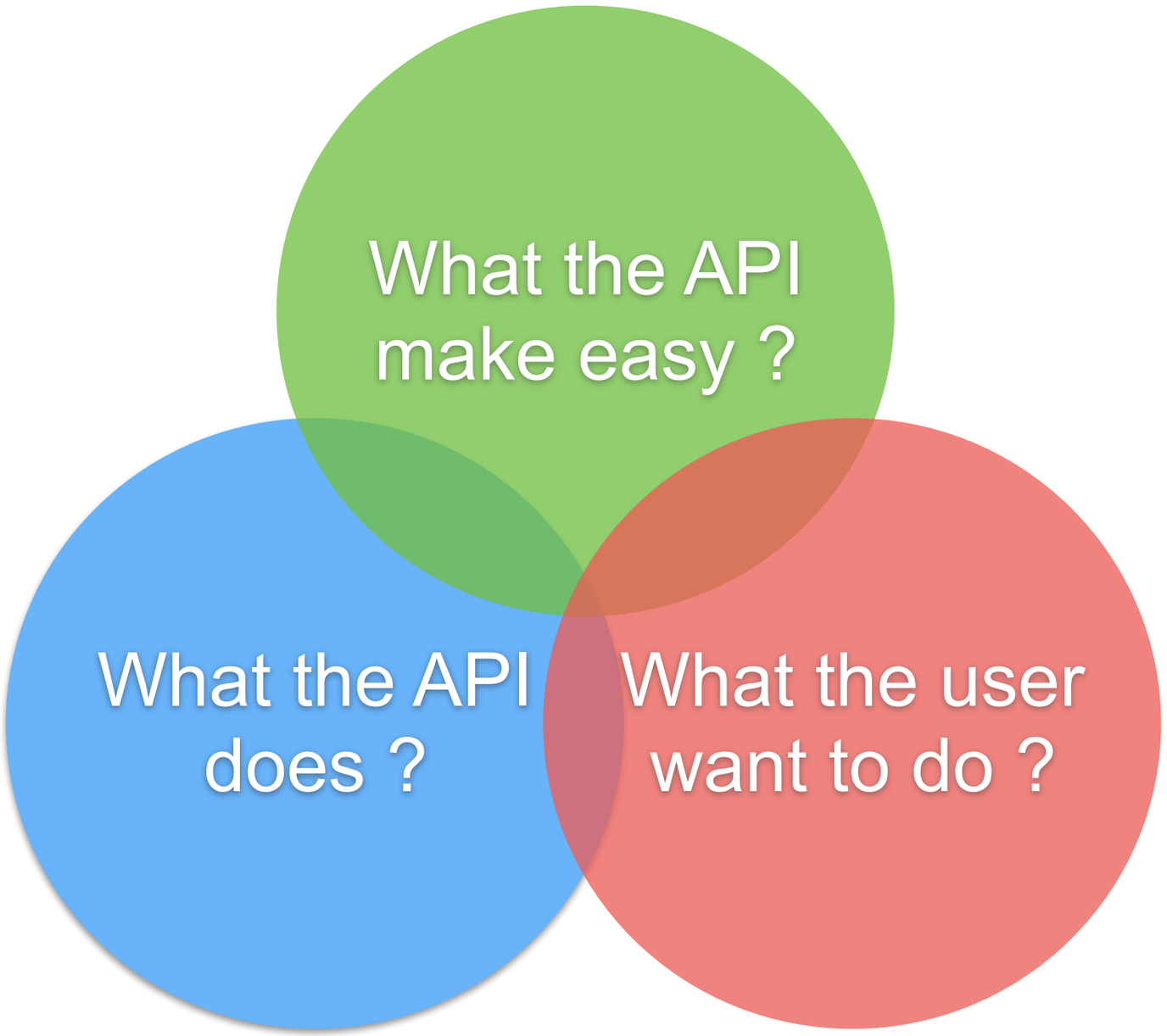
REST Request & Response

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

undertow

Jersey
RESTful Web Services in Java

WildFly SWARM

Java
J2EE

JAX-RS

Ratpack
A toolkit for JVM web applications

RESTEasy

spring
by Pivotal

Spark

Restlet

akka

Apache CXF

STRUTS

play

unirest
Lightweight HTTP Request Client Libraries

Dropwizard

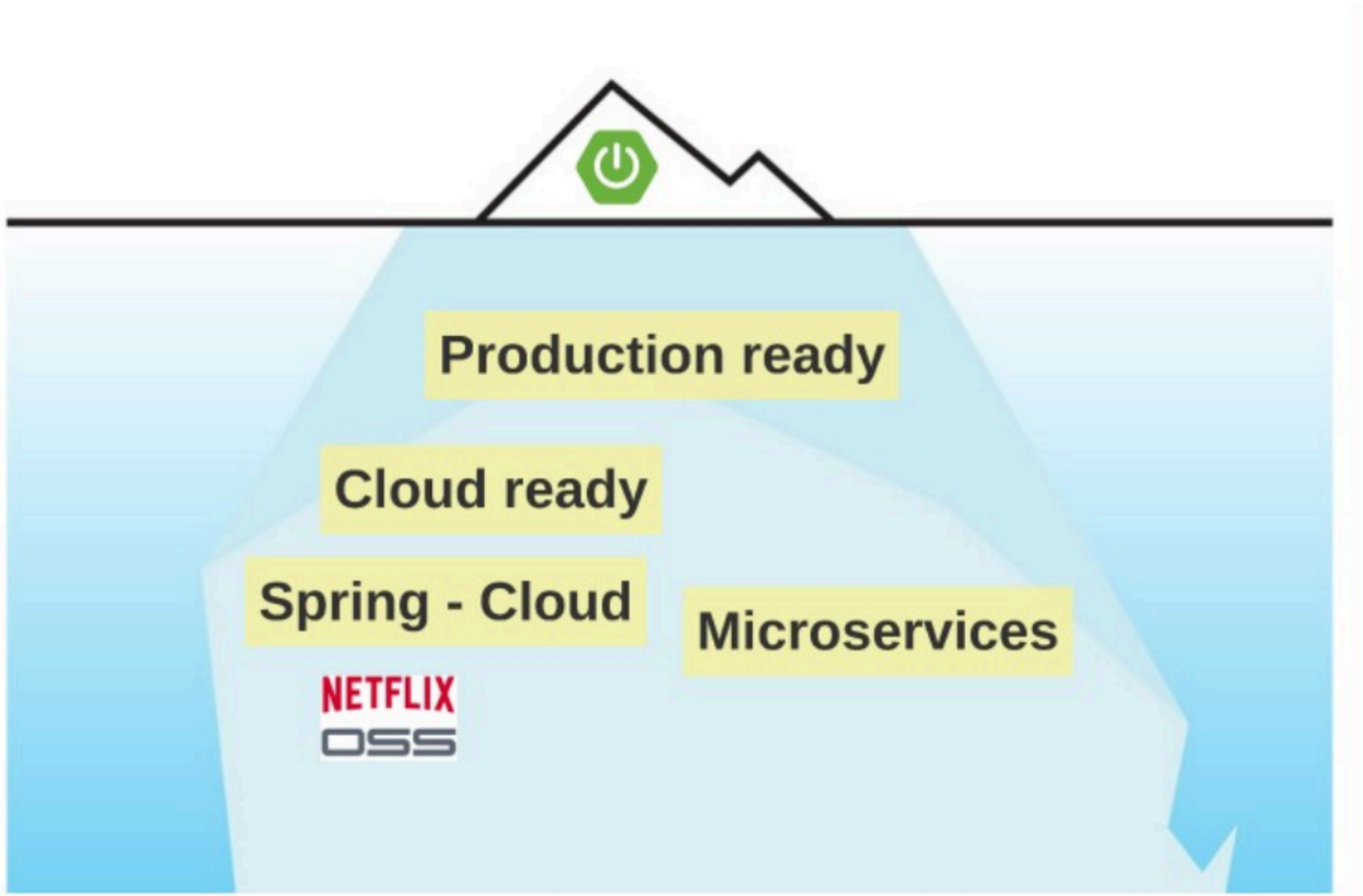
NINJA
web framework

ACT.framework

Hello Spring Boot 2.x

Why ?

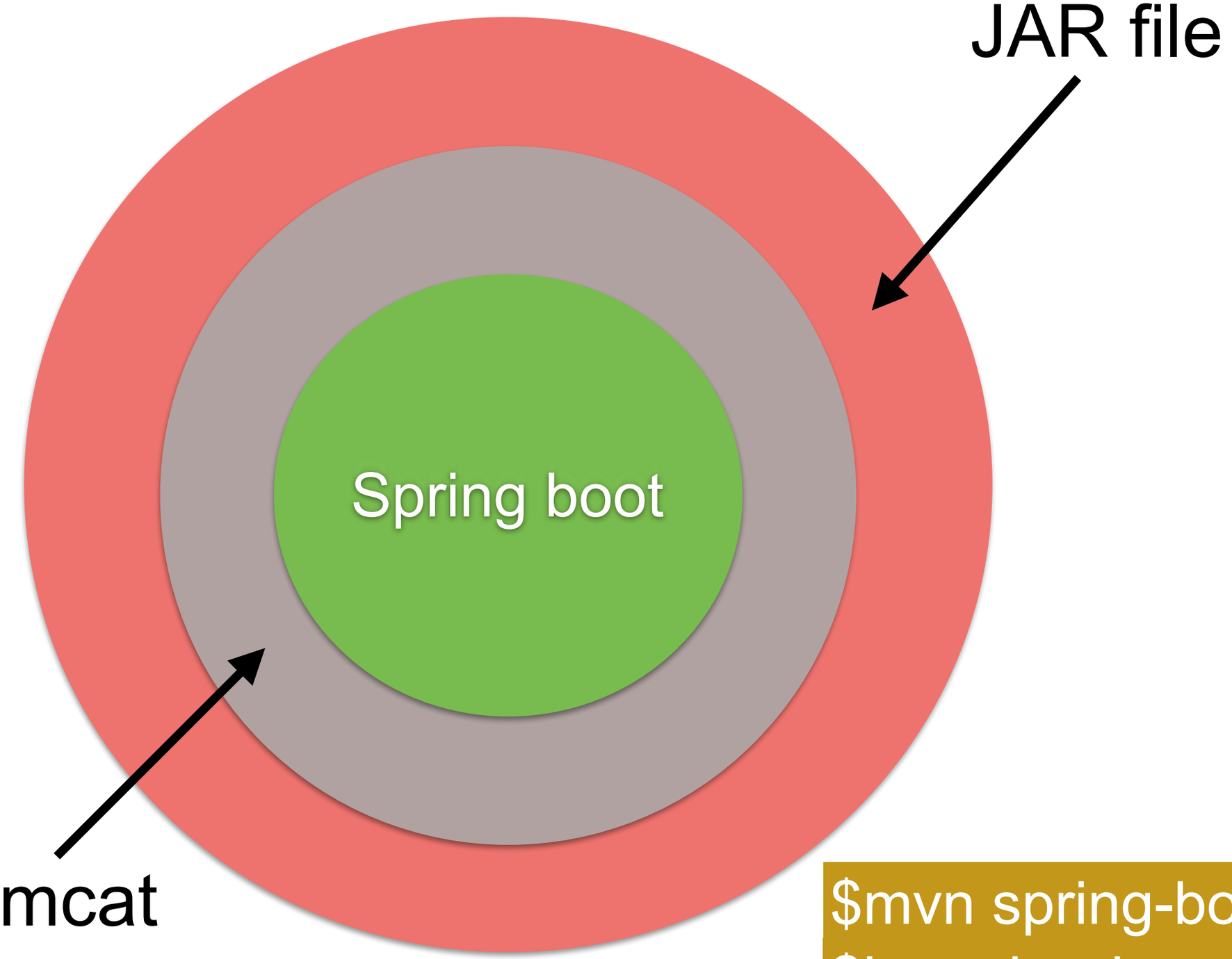
Application skeleton generator
Reduce effort to add new technologies



What ?

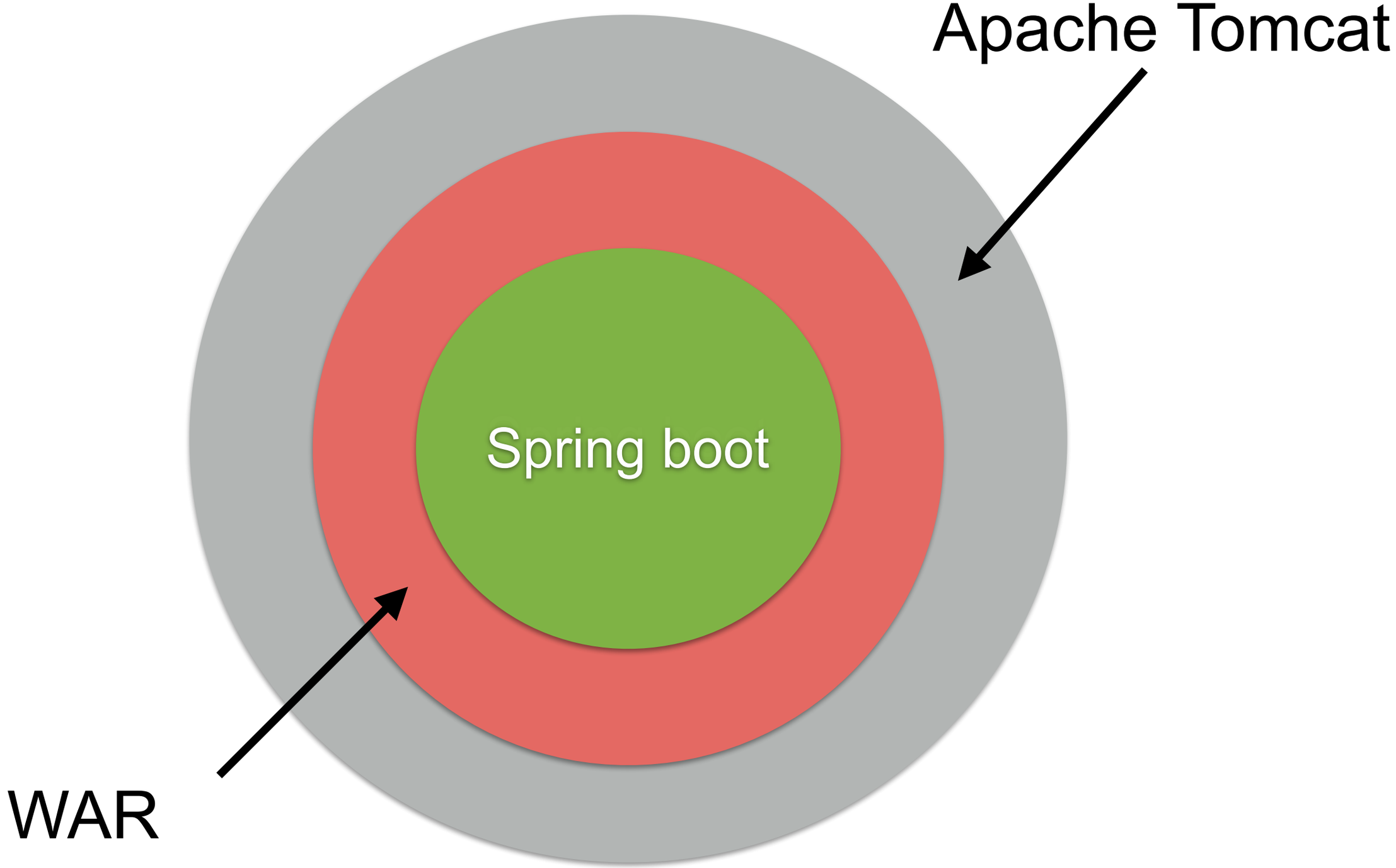
Embedded application server
Integration with tools/technologies (starter)
Production tools (monitoring, health check)
Configuration management
Dev tools
No source code generation, no XML

How ?



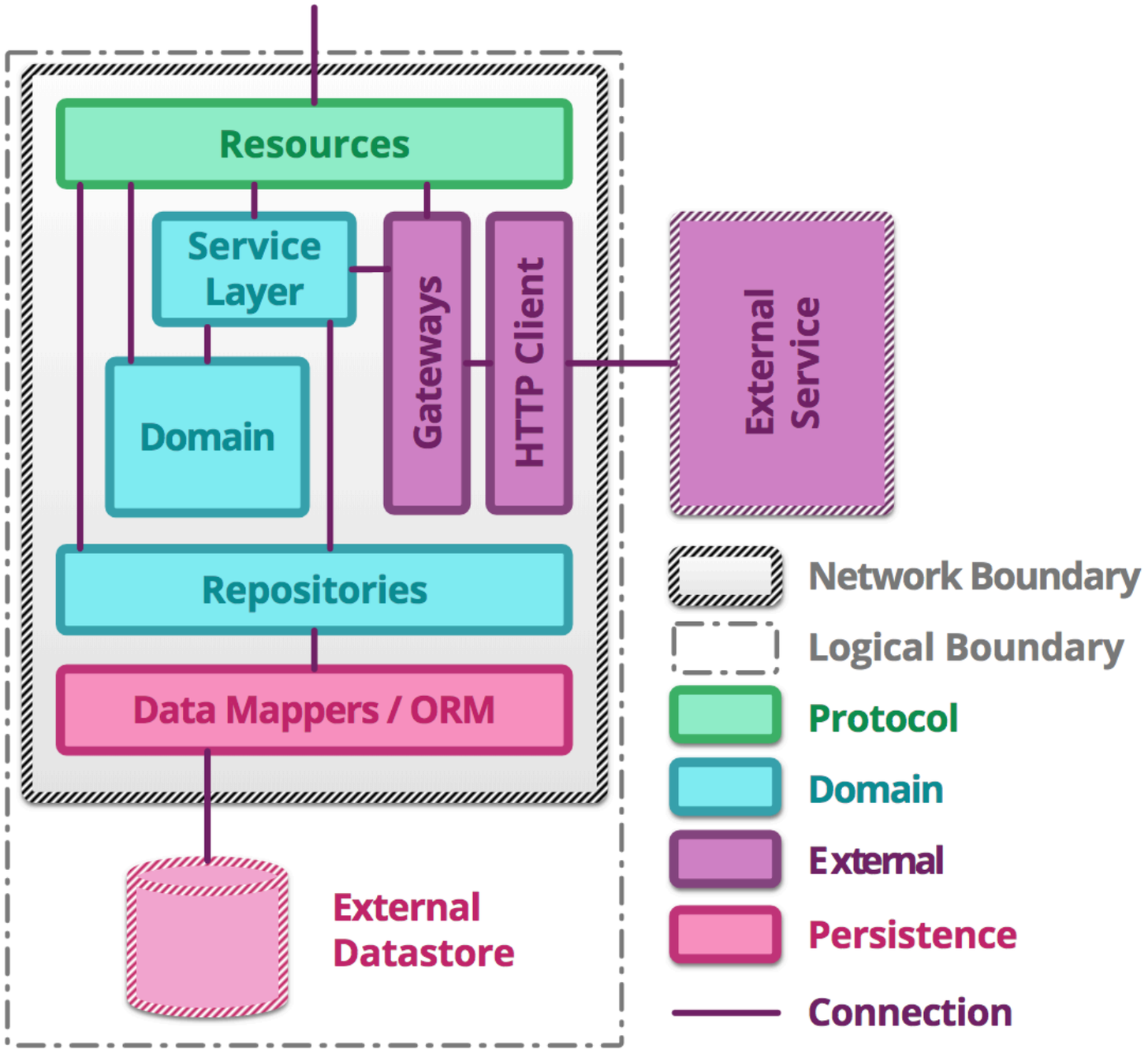

```
$mvn spring-boot:run
$java -jar demo.jar
```

How ?

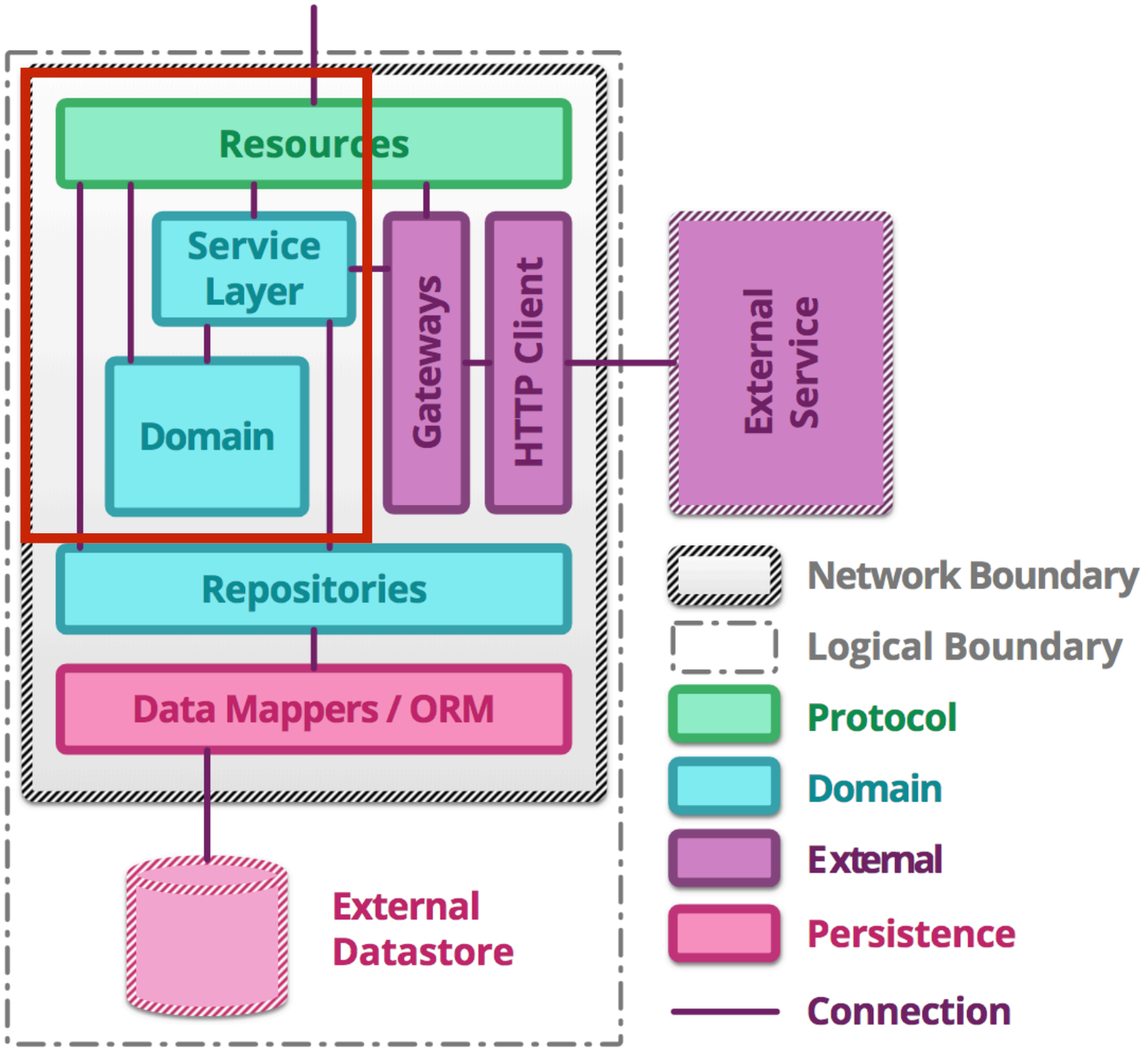


Building RESTful API with Spring Boot

Service Structure



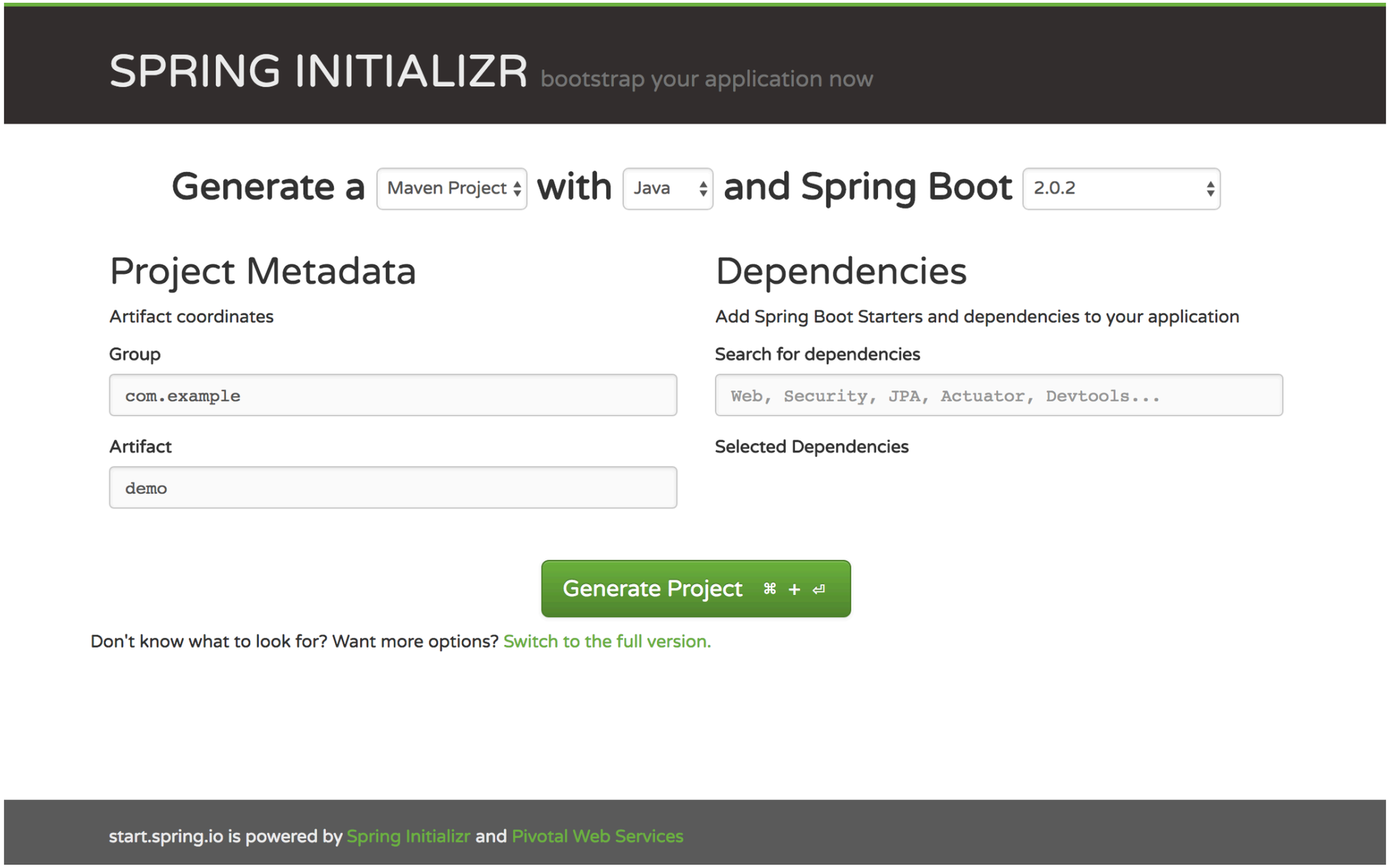
Service Structure



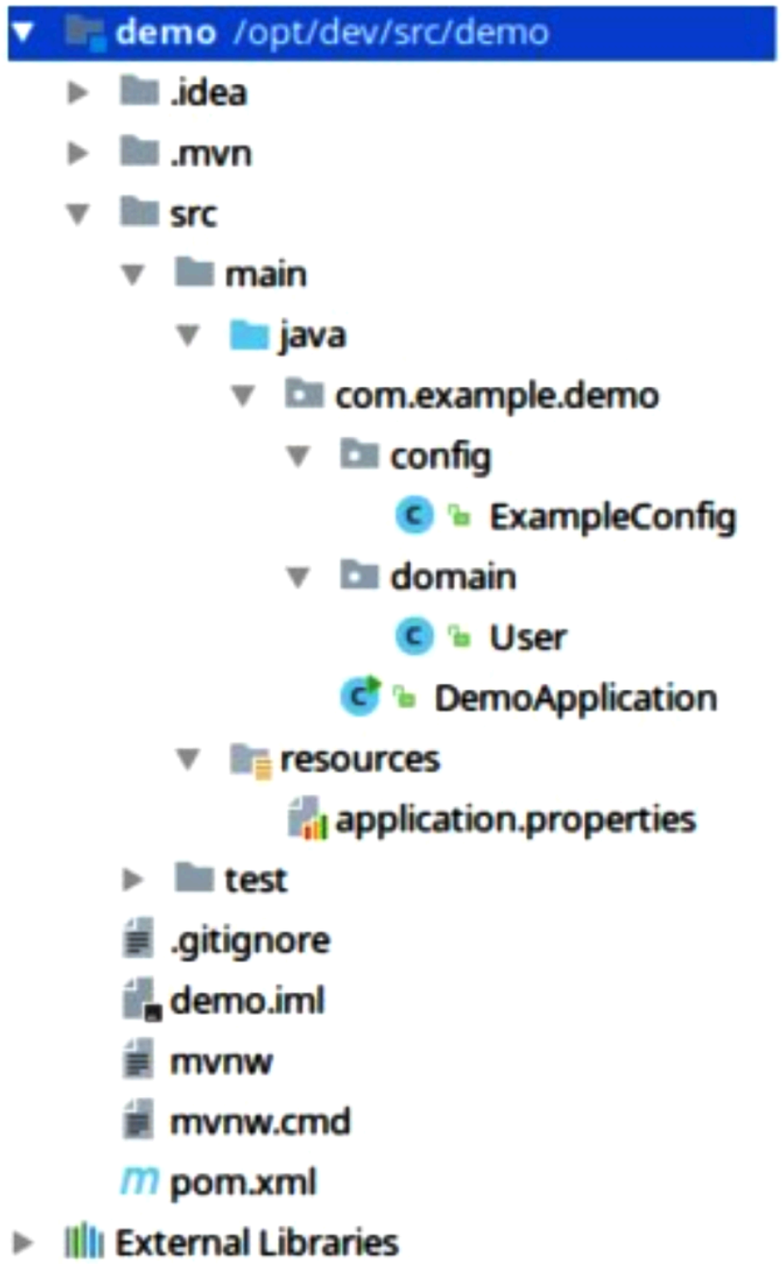
Create project with Spring Initializr

Spring Initializr

https://start.spring.io/

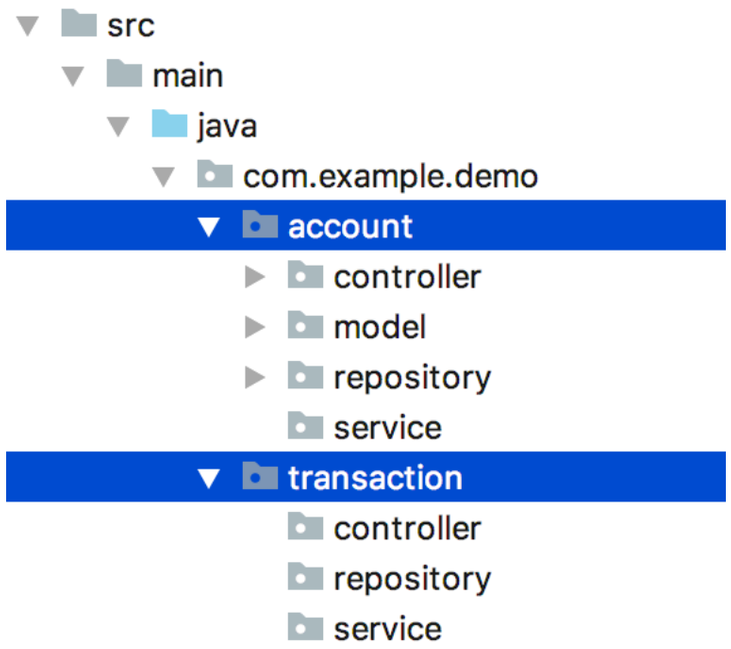


Structure of Spring Boot



Spring Boot Structure

Separate by function/domain



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

Spring Boot main class

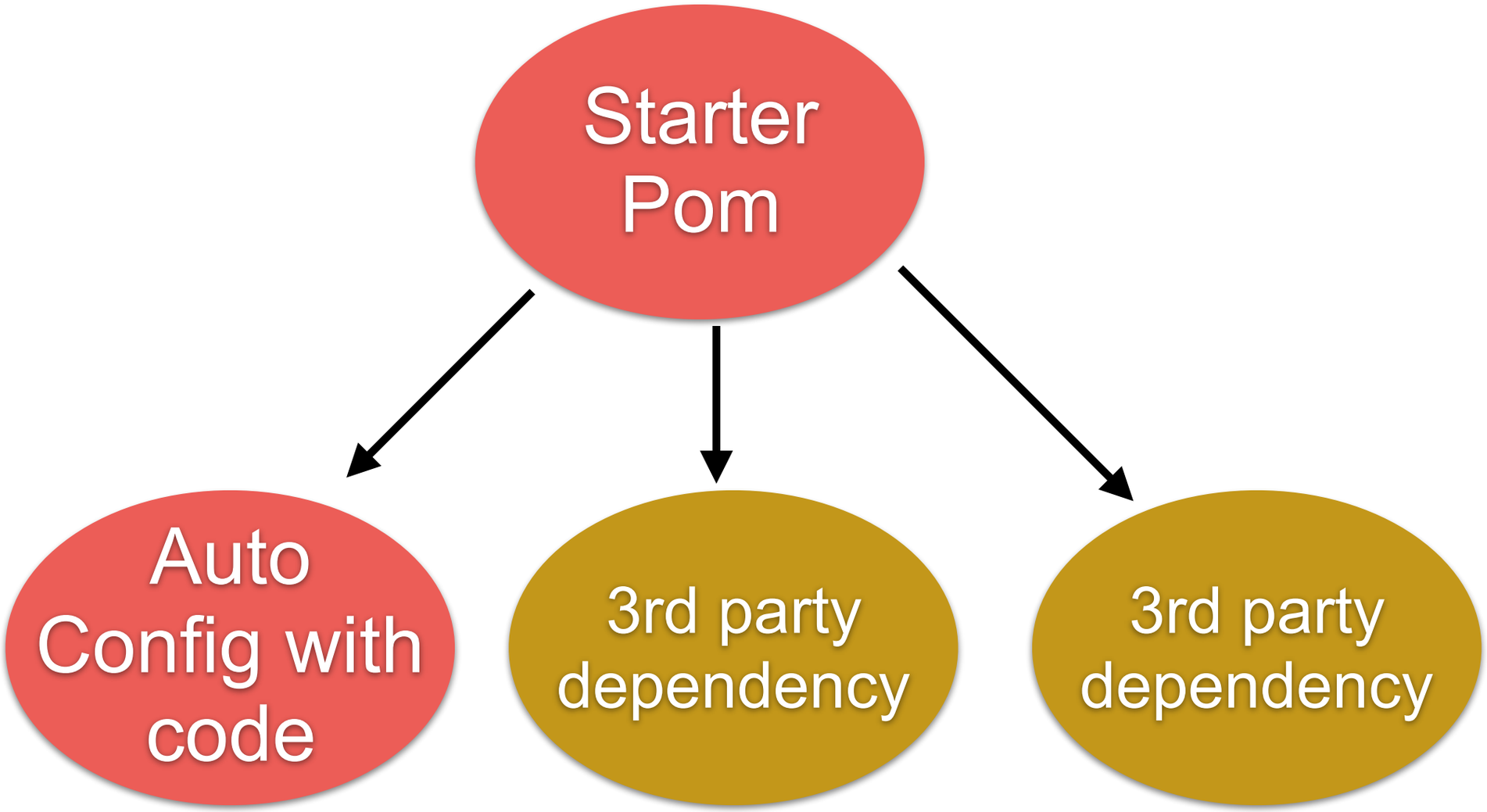
```
package com.example.demo;

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

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

```
  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '   |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::        (v2.0.2.RELEASE)

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
```

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

Create REST Controller

hello.controller.HelloController.java

```java
package hello.controller;

import hello.domain.Hello;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RestController;

@RestController
public class HelloController {

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

}
```

Compile and Packaging

$./gradlew clean bootJar
$java -jar ./build/libs/bookstore.jar

```
  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::        (v2.0.0.RELEASE)

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

Development mode

$./gradlew bootRun

Open in browser

http://localhost:8080/hello/somkiat

Create RESTFul APIs

HTTP GET

HTTP POST

HTTP PUT

HTTP DELETE

HTTP GET (Path Variable)

GET /hello/<name>

```java
@RestController
public class HelloController {

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

}
```

HTTP GET (Query String)

GET /hello?name=<name>

```
@GetMapping("/hello")
public String sayHi2(@RequestParam String name) {
    return "Hello, " + name;
}
```

HTTP POST

```
POST /hello
{
    "name": "Somkiat",
    "age": 30,
}
```

HTTP POST

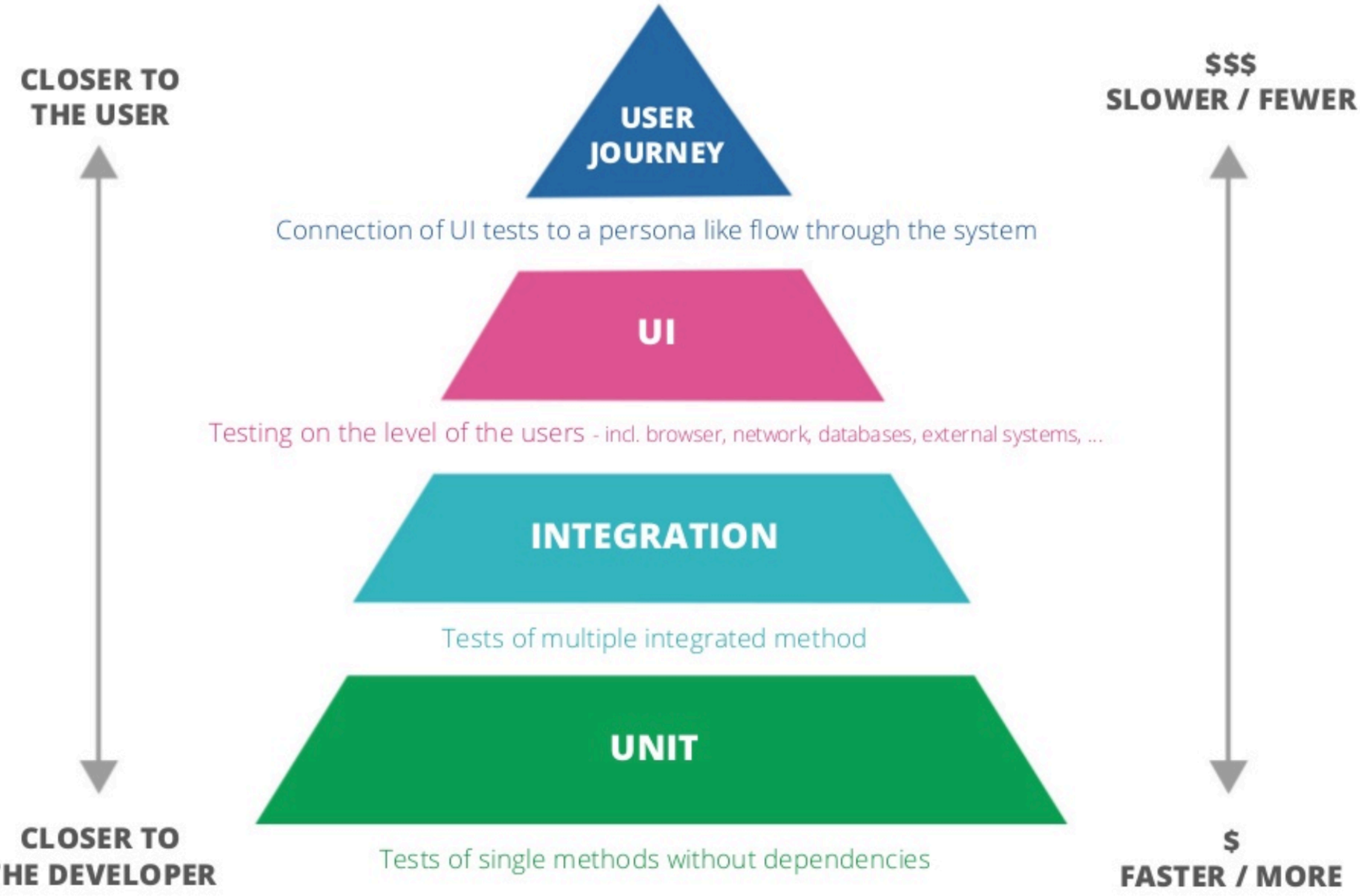
Send data in body payload (@RequestBody)

```java
@RestController
public class HelloController {

    @PostMapping("/hello")
    public String createData(
            @RequestBody RequestCreateHello request) {
        return request.toString();
    }
```

```java
public class RequestCreateHello {

    private String name;
    private int age;
```

How to test the RESTFul API ?

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



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());
    }

}
```

API/Controller tests

How to use controller ?

Spring boot provides MockMvc to test Controller

```java
package hello.controller;

import hello.domain.Hello;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RestController;

@RestController
public class HelloController {

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

}
```

API/Controller tests

```java
@RunWith(SpringRunner.class)
@WebMvcTest(controllers = HelloController.class)
public class HelloControllerTest {

    @Autowired
    private MockMvc mockMvc;

    @Test
    public void shouldReturnHelloSomkiat() throws Exception {
        mockMvc.perform(get( urlTemplate: "/hello/somkiat"))
                .andExpect(
                        jsonPath( expression: "$.message")
                                .value( expectedValue: "Hello somkiat"))
                .andExpect(status().is2xxSuccessful());
    }

}
```

API/Controller tests

With Spring Boot Testing + TestRestTemplate

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

    @Autowired
    private TestRestTemplate testRestTemplate;

    @Autowired
    private AccountRepository accountRepository;

    @Before
    public void initData() {
        accountRepository.save(new Account("01"));
        accountRepository.save(new Account("02"));
    }
```

API/Controller tests

With Spring Boot Testing + TestRestTemplate

```
@Test
public void
เรียกข้อมูลบัญชีของหมายเลข_0868696209_ต้องเจอสองบัญชีนะ() {
    List<AccountResponse> accountResponses
            = testRestTemplate.getForObject(
                    "/account/0868696209", List.class);

    assertEquals(2, accountResponses.size());
}
```

Compile with testing

$./gradlew clean test

Test Summary

1	0	0	0.029s
tests	failures	ignored	duration

100%
successful

Packages Classes

Package	Tests	Failures	Ignored	Duration	Success rate
com.example.bookstore	1	0	0	0.029s	100%

Generated by Gradle 5.6.2 at Oct 1, 2019 11:07:05 PM

% of Code/Test coverage

Build.gradle

```
plugins {
      id 'org.springframework.bo
      id 'io.spring.dependency-m
      id 'java'
      id 'jacoco'
}
```

Run test again

$./gradlew clean test
$./gradlew jacocoTestReport

bookstore

bookstore

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines
com.example.bookstore.demo		5%		n/a	9	10	11	12
com.example.bookstore		37%		n/a	1	2	2	3
Total	62 of 68	8%	0 of 0	n/a	10	12	13	15

Coverage report

open build/reports/jacoco/test/html/index.html

bookstore > com.example.bookstore.demo > HelloController.java

HelloController.java

```
1.  package com.example.bookstore.demo;
2.
3.  import org.springframework.web.bind.annotation.*;
4.
5.  @RestController
6.  public class HelloController {
7.
8.      @PostMapping("/hello")
9.      public String createData(
10.             @RequestBody RequestCreateHello request) {
11.         return request.toString();
12.     }
13.
14.     @GetMapping("/hello/{name}")
15.     public String sayHi(@PathVariable String name) {
16.         return "Hello, " + name;
17.     }
18.
19.     @GetMapping("/hello")
20.     public String sayHi2(@RequestParam String name) {
21.         return "Hello, " + name;
22.     }
23.
24. }
25.
```

How to improve % of coverage ?

Error handling

Error Handling

1xx Informational

100 Continue
101 Switching Protocols
102 Processing (WebDAV)

2xx Success

★ 200 OK
★ 201 Created
202 Accepted
203 Non-Authoritative Information
★ 204 No Content
205 Reset Content
206 Partial Content
207 Multi-Status (WebDAV)
208 Already Reported (WebDAV)
226 IM Used

3xx Redirection

300 Multiple Choices
301 Moved Permanently
302 Found
303 See Other
★ 304 Not Modified
305 Use Proxy
306 (Unused)
307 Temporary Redirect
308 Permanent Redirect (experiemental

4xx Client Error

★ 400 Bad Request
★ 401 Unauthorized
402 Payment Required
★ 403 Forbidden
★ 404 Not Found
405 Method Not Allowed
406 Not Acceptable
407 Proxy Authentication Required
408 Request Timeout
★ 409 Conflict
410 Gone
411 Length Required
412 Precondition Failed
413 Request Entity Too Large
414 Request-URI Too Long
415 Unsupported Media Type
416 Requested Range Not Satisfiable
417 Expectation Failed
418 I'm a teapot (RFC 2324)
420 Enhance Your Calm (Twitter)
422 Unprocessable Entity (WebDAV)
423 Locked (WebDAV)
424 Failed Dependency (WebDAV)
425 Reserved for WebDAV
426 Upgrade Required
428 Precondition Required
429 Too Many Requests
431 Request Header Fields Too Large
444 No Response (Nginx)
449 Retry With (Microsoft)
450 Blocked by Windows Parental Controls (Microsoft)
451 Unavailable For Legal Reasons
499 Client Closed Request (Nginx)

5xx Server Error

★ 500 Internal Server Error
501 Not Implemented
502 Bad Gateway
503 Service Unavailable
504 Gateway Timeout
505 HTTP Version Not Supported
506 Variant Also Negotiates (Experimental)
507 Insufficient Storage (WebDAV)
508 Loop Detected (WebDAV)
509 Bandwidth Limit Exceeded (Apache)
510 Not Extended
511 Network Authentication Required
598 Network read timeout error
599 Network connect timeout error

Error Handling

How to handling exception in Spring Boot ?

```java
@GetMapping("/hello/{name}")
public String sayHi(@PathVariable String name) throws Exception {
    if("400".equals(name)) {
        throw new IOException();
    } else if("404".equals(name)) {
        throw new Exception();
    }
    return "Hello, " + name;
}
```

Use RestControllerAdvice

```java
@RestControllerAdvice
public class HelloControllerAdvice {

    @ExceptionHandler(value = {IOException.class})
    @ResponseStatus(HttpStatus.BAD_REQUEST)
    public ErrorResponse badRequest(Exception ex) {
        return new ErrorResponse(400, "Bad Request");
    }

    @ExceptionHandler(value = {Exception.class})
    @ResponseStatus(HttpStatus.NOT_FOUND)
    public ErrorResponse unKnownException(Exception ex) {
        return new ErrorResponse(404, "Data Not Found");
    }

}
```

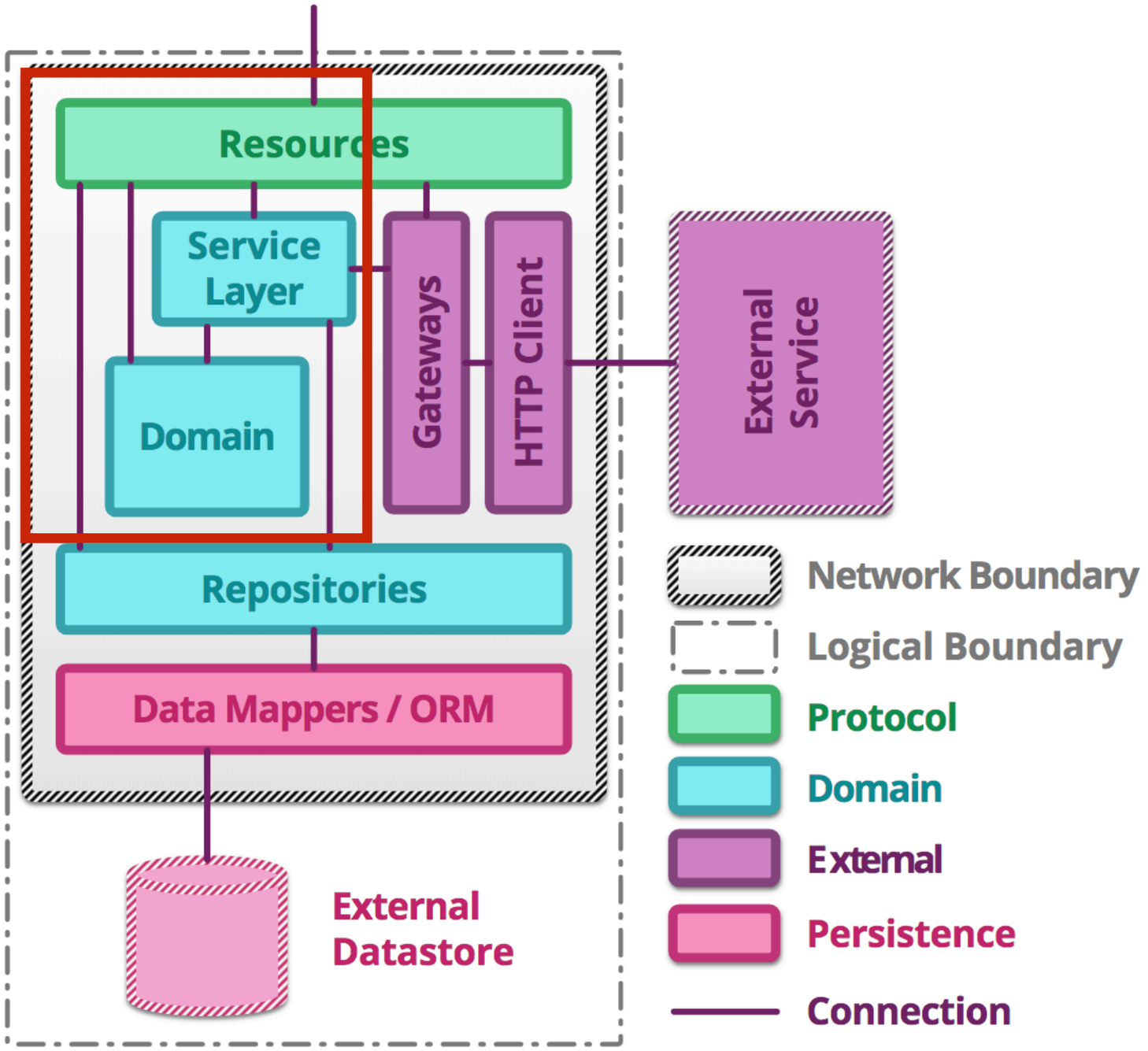
1

Use RestControllerAdvice

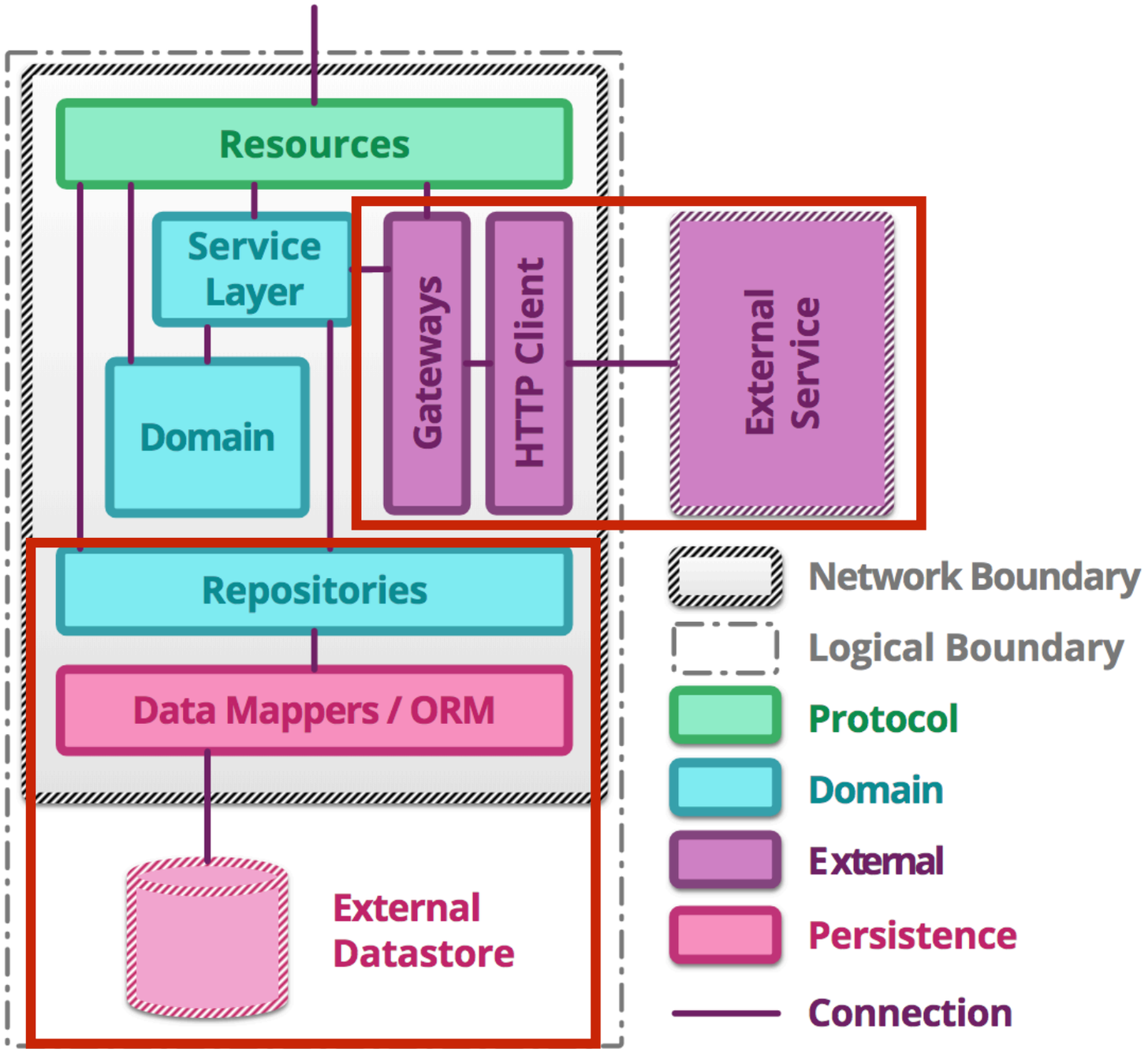
```java
@RestControllerAdvice
public class HelloControllerAdvice {

    @ExceptionHandler(value = {IOException.class})
    @ResponseStatus(HttpStatus.BAD_REQUEST)
    public ErrorResponse badRequest(Exception ex) {
        return new ErrorResponse(400, "Bad Request");
    }

    @ExceptionHandler(value = {Exception.class})
    @ResponseStatus(HttpStatus.NOT_FOUND)
    public ErrorResponse unKnownException(Exception ex) {
        return new ErrorResponse(404, "Data Not Found");
    }

}
```

2

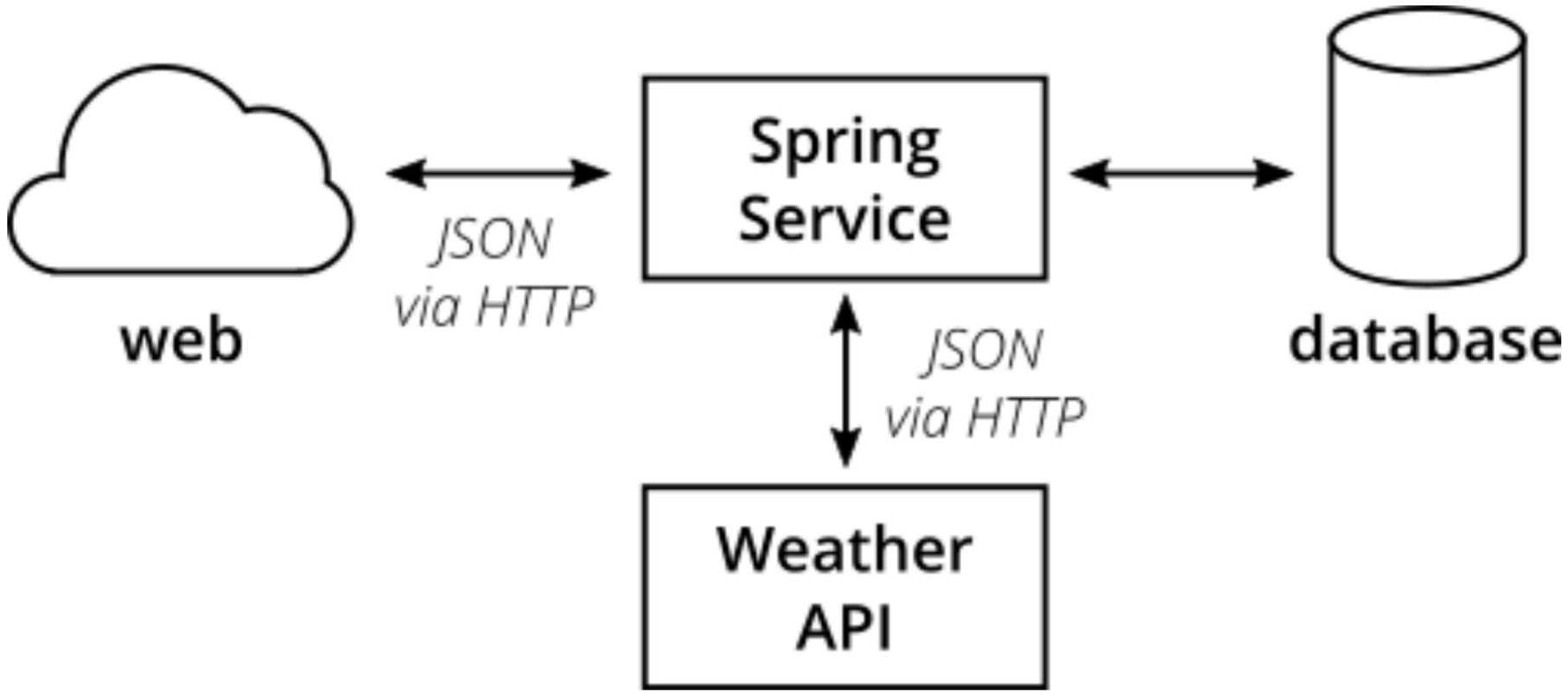
Service Structure



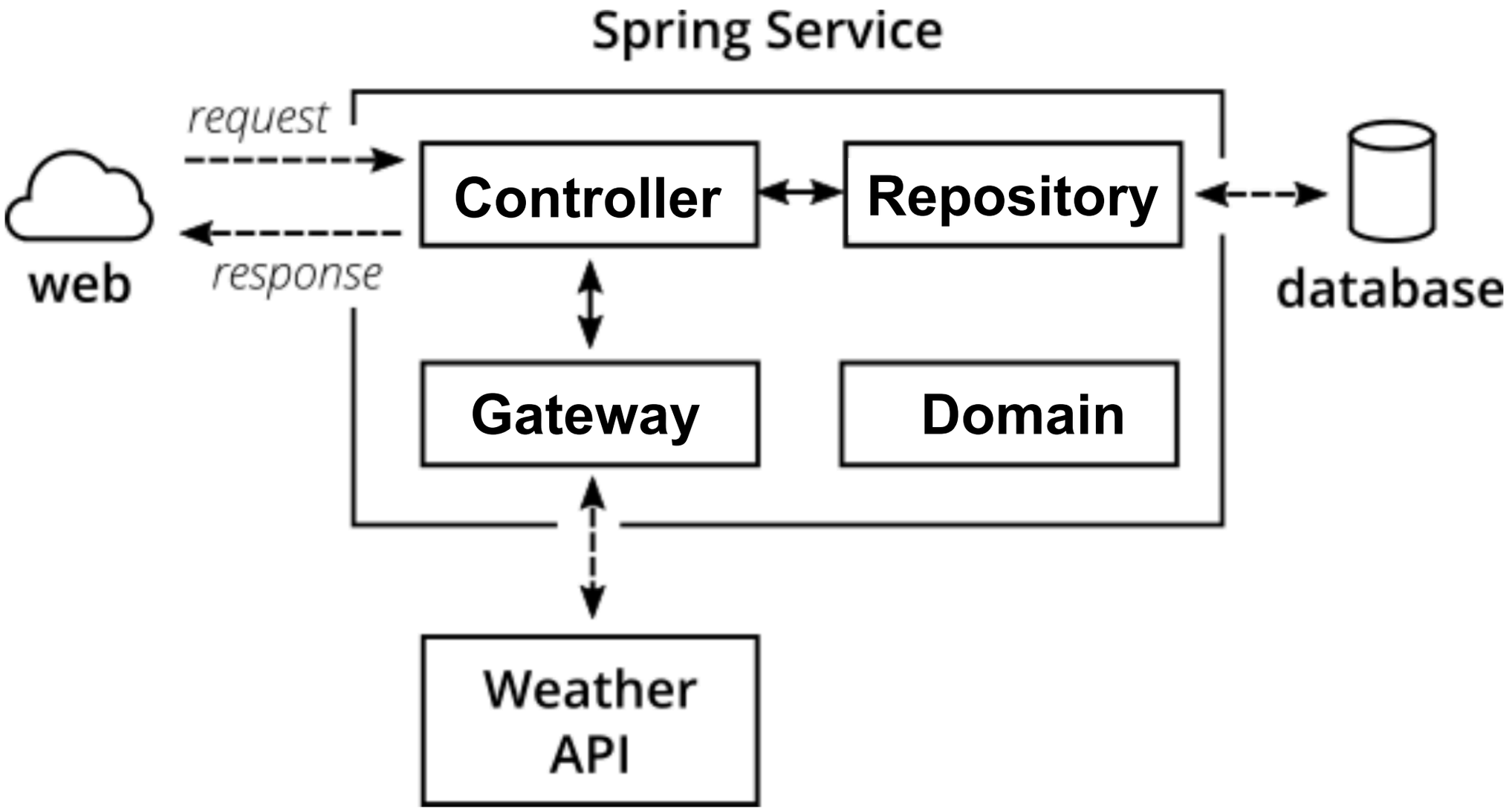
Service Structure



Sample application

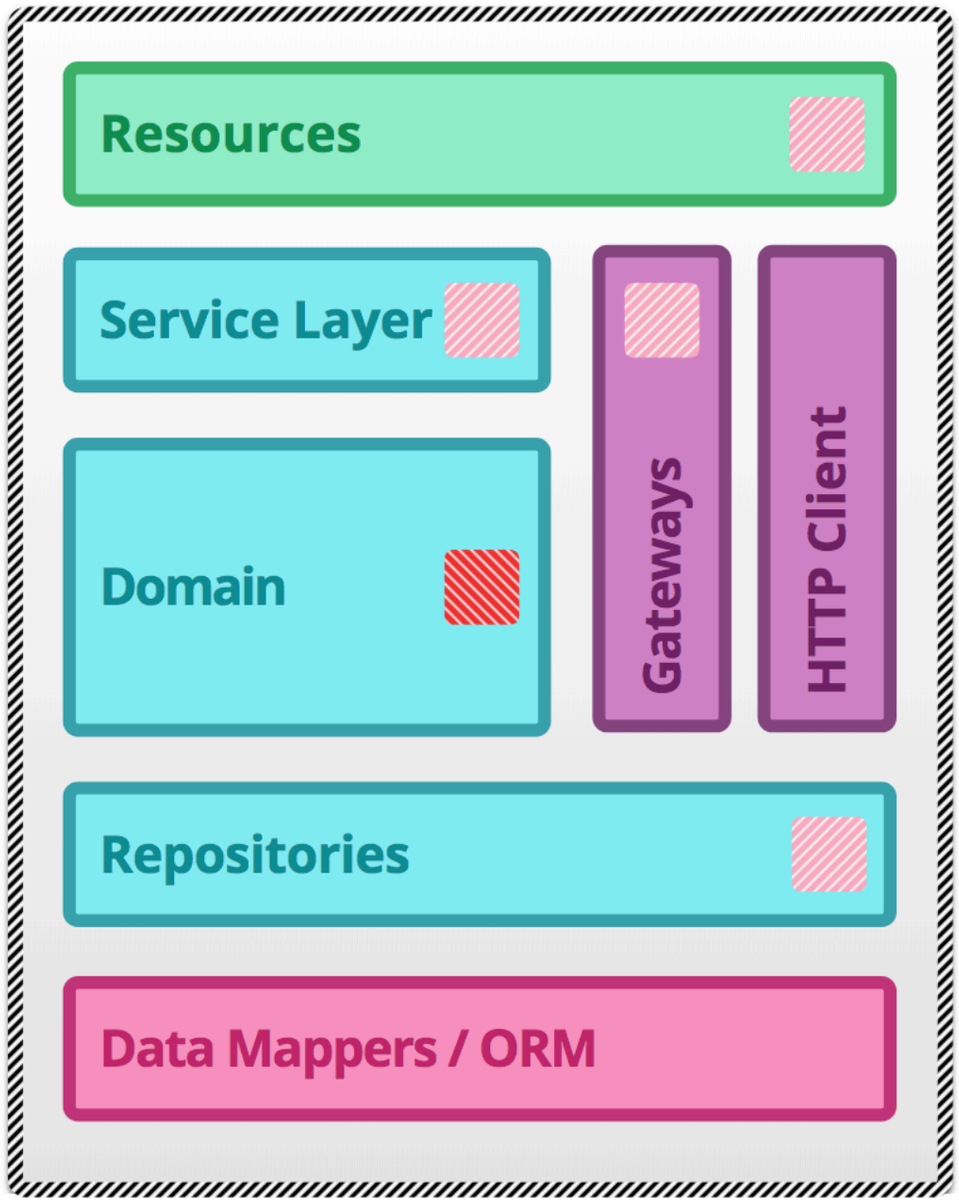


Project structure



How to test ?

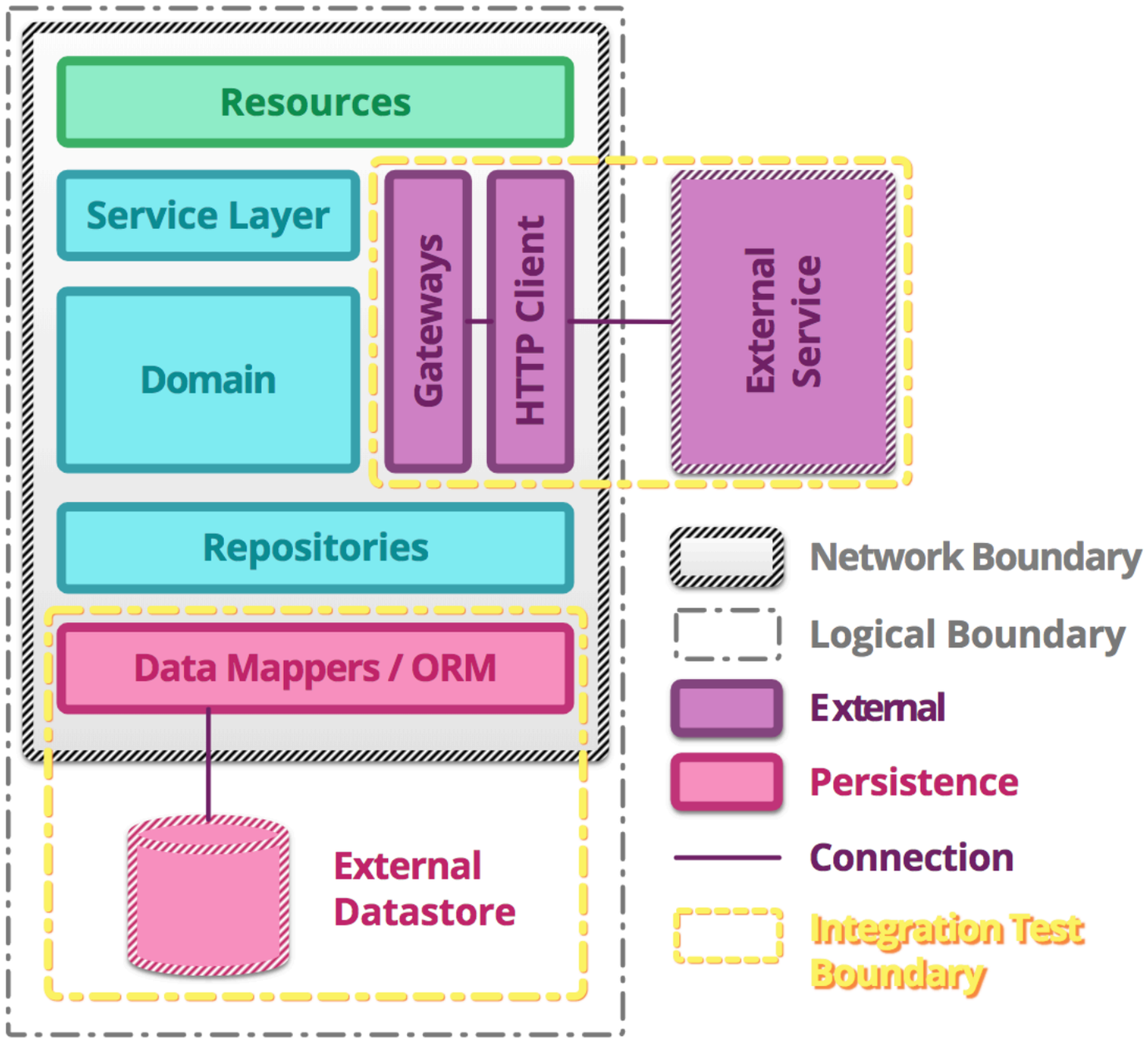
Unit testing



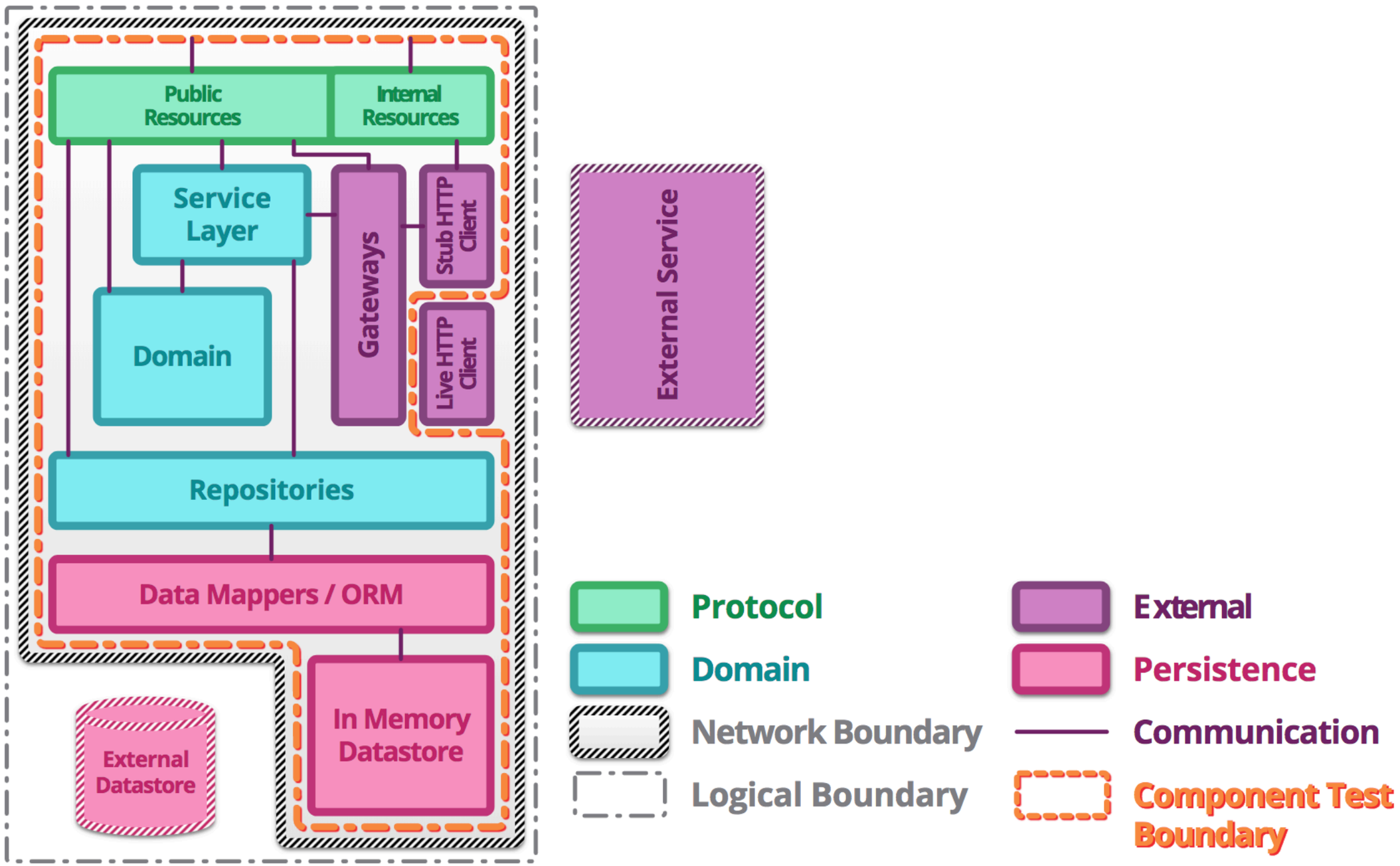
Unit - Solitary

Unit - Sociable

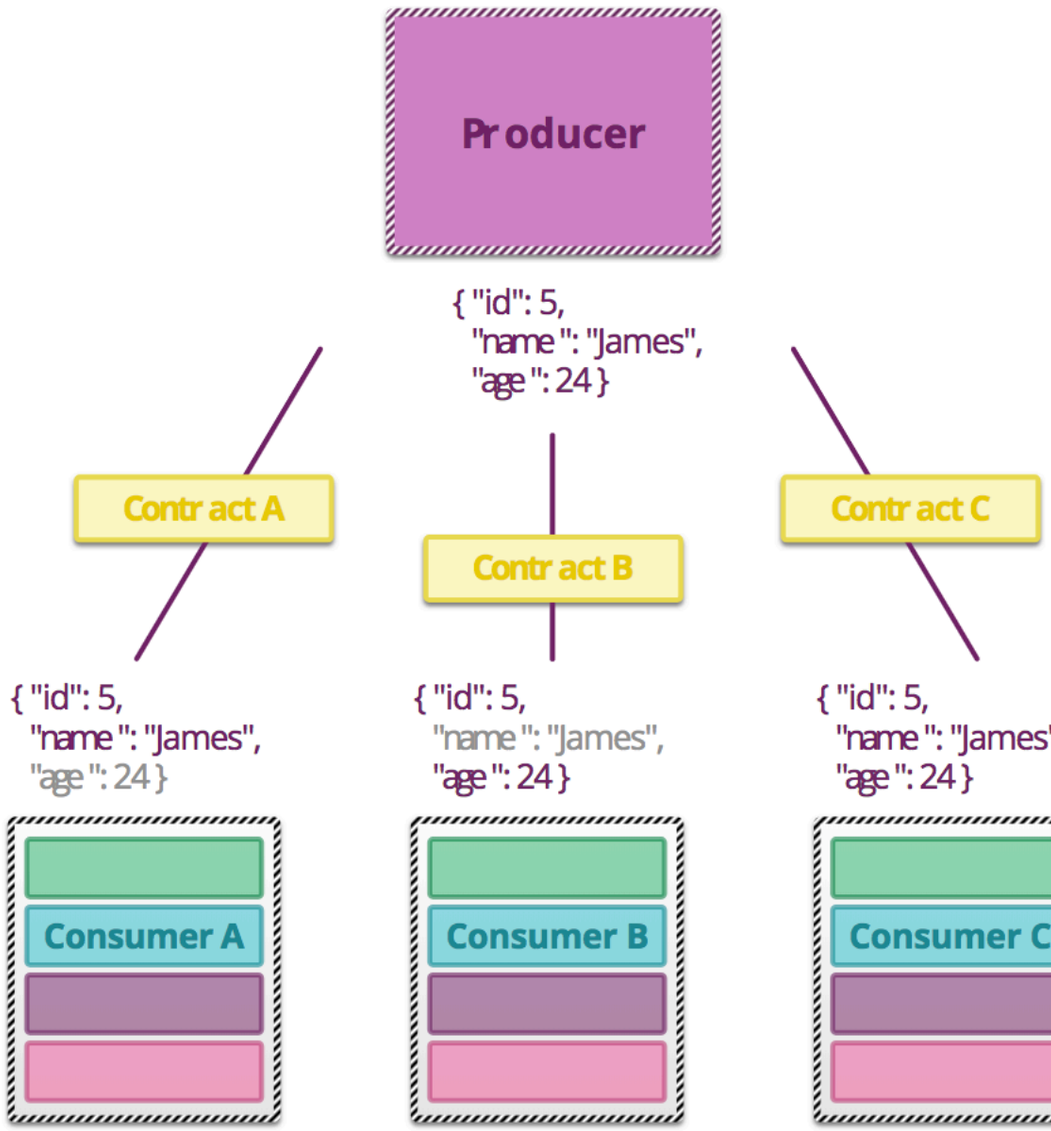
Integration testing



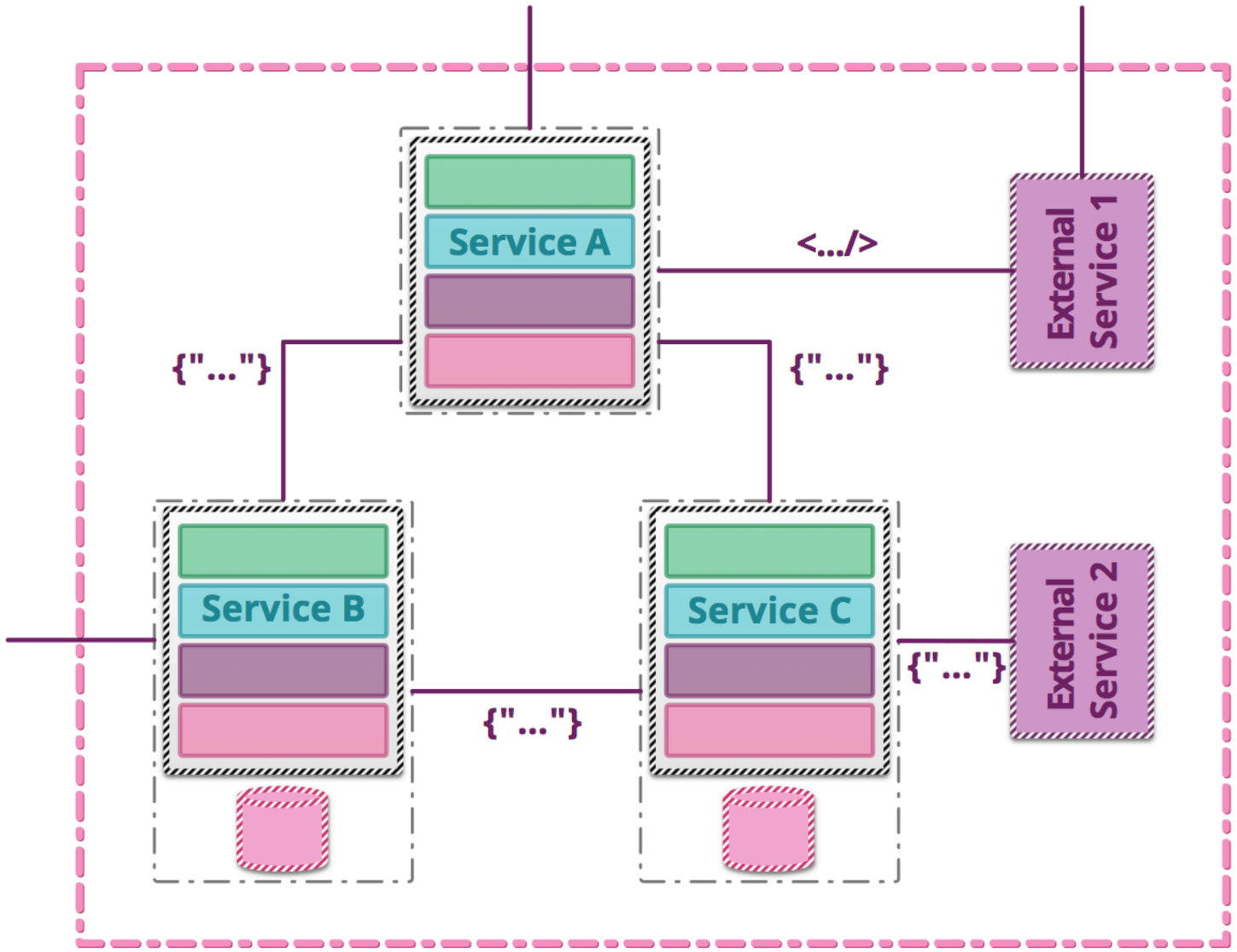
Component testing



Contract testing



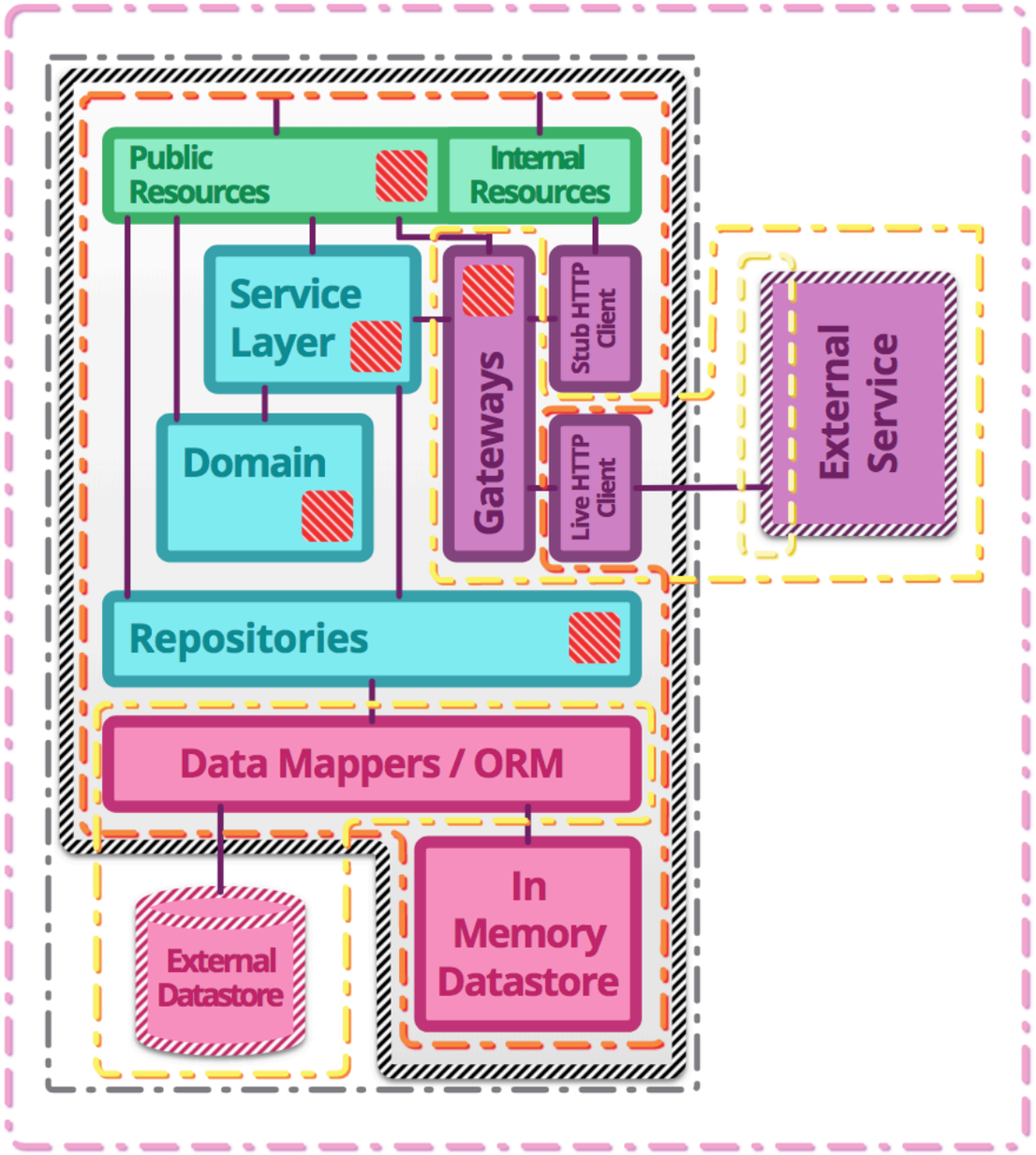
End-to-End testing



Summary

Unit tests : exercise the smallest pieces of testable software in the application to determine whether they behave as expected.

Integration tests : verify the communication paths and interactions between components to detect interface defects.



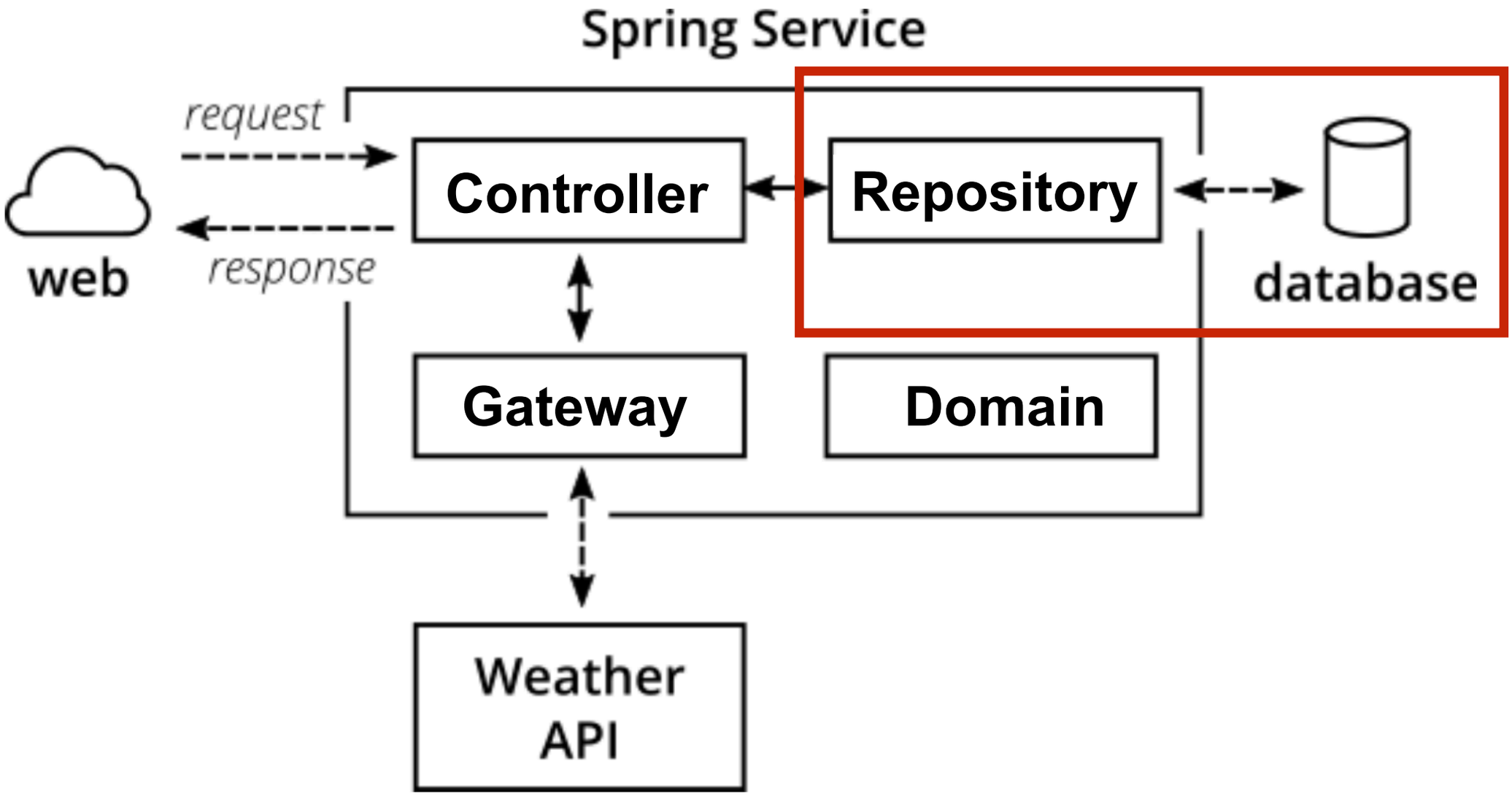
Component tests : limit the scope of the exercised software to a portion of the system under test, manipulating the system through internal code interfaces and using test doubles to isolate the code under test from other components.

Contract tests : verify interactions at the boundary of an external service asserting that it meets the contract expected by a consuming service.

End-to-end tests : verify that a system meets external requirements and achieves its goals, testing the entire system, from end to end.

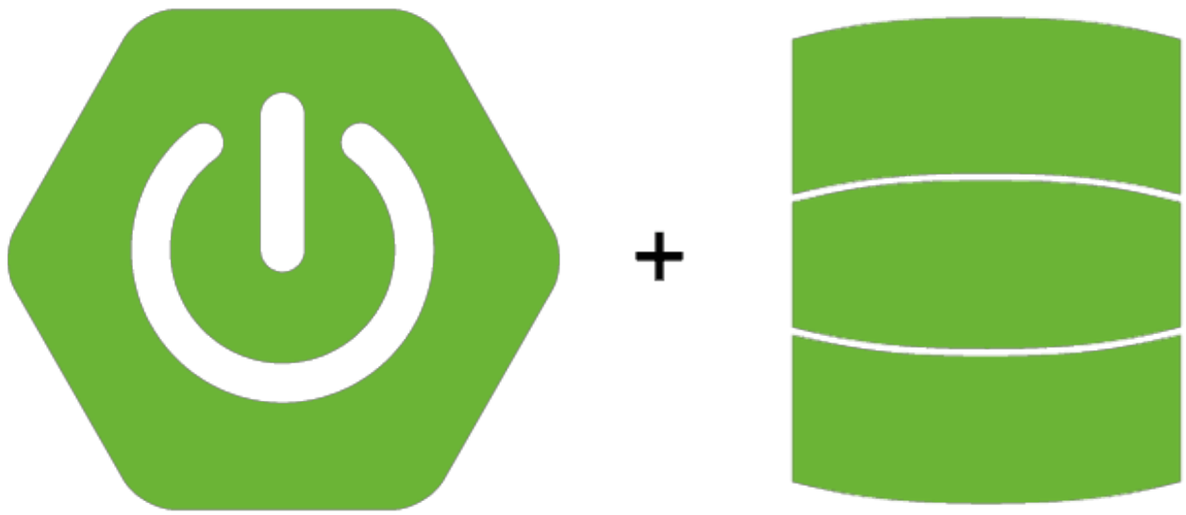
Let’s workshop with Repository

Working with repository



Working with repository

We're using Spring Data



Modify pom.xml

Add library of Spring Data

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

Modify pom.xml

Add library of Persistence/data store

```
<dependency>
    <groupId>org.postgresql</groupId>
    <artifactId>postgresql</artifactId>
    <version>42.1.1</version>
</dependency>
```

Add data store config

In src/main/resources

```
spring.datasource.url= jdbc:postgresql://127.0.0.1:15432/postgres
spring.datasource.username= user
spring.datasource.password= password
spring.datasource.platform= POSTGRESQL

spring.jpa.show-sql= true
spring.jpa.hibernate.ddl-auto= create-drop
spring.jpa.database-platform= org.hibernate.dialect.PostgreSQLDialect
```

Create repository interface

hello.repository.PersonRepository.java

```
public interface PersonRepository
        extends CrudRepository<Person, String> {

    Optional<Person> findByFirstName(String name);

}
```

Create Entity class

hello.repository.Person.java

```java
@Entity
public class Person {

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private int id;
    private String firstName;
    private String lastName;

    public Person() {
    }

    public Person(String firstName, String lastName) {
        this.firstName = firstName;
        this.lastName = lastName;
    }
```

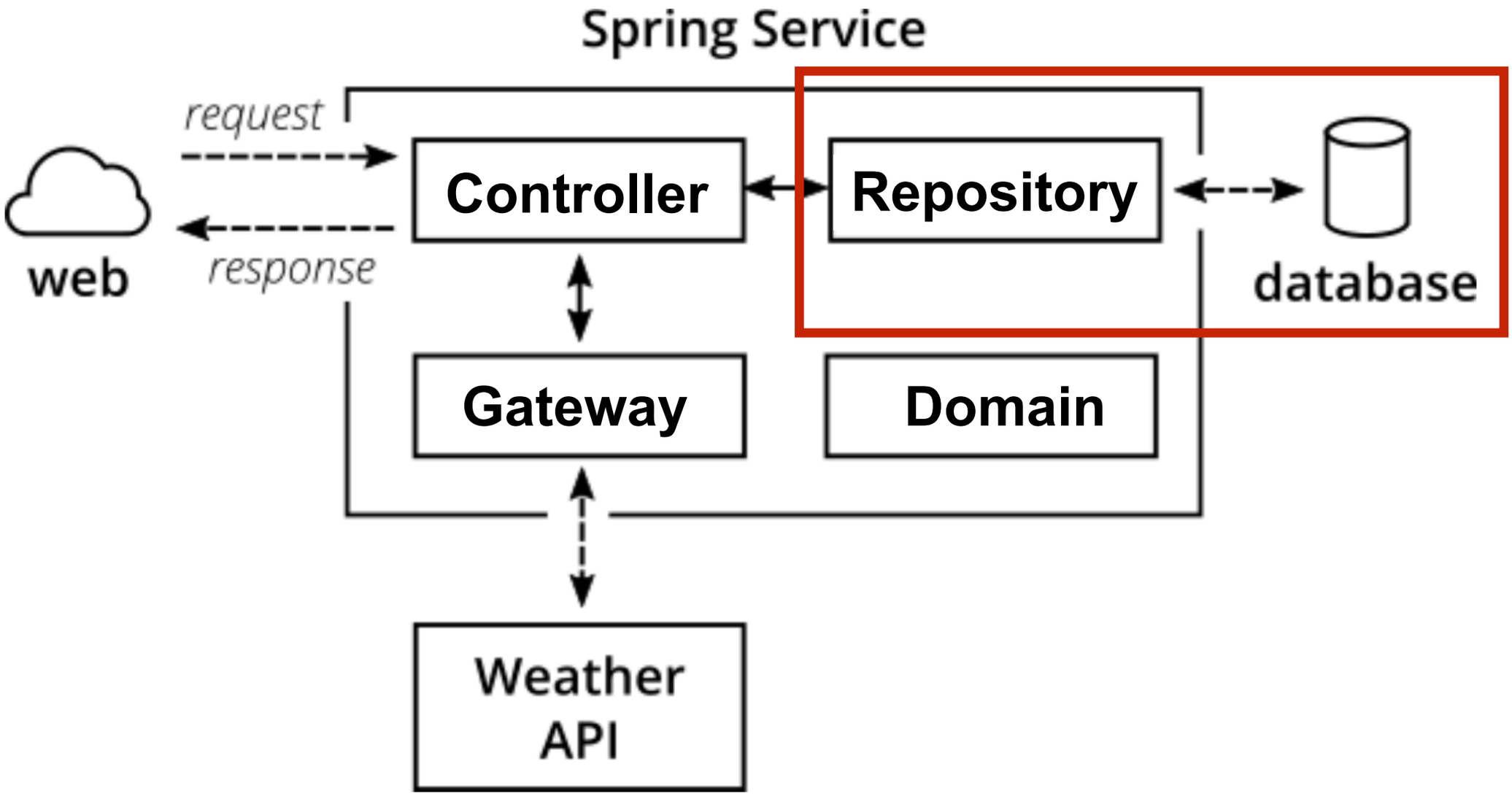
Create new controller

hello.repository.HelloControllerWithRepository.java

```java
public class HelloControllerWithRepository {

    private final PersonRepository personRepository;

    @Autowired
    public HelloControllerWithRepository(PersonRepository personRepository) {
        this.personRepository = personRepository;
    }

    @GetMapping("/hello/data/{name}")
    public Hello sayHi(@PathVariable String name) {
        Optional<Person> foundPerson = personRepository.findByName(name);
        String result = foundPerson
                .map(person -> String.format("Hello %s", person.getFirstName()))
                .orElse( other: "Data not found");
        return new Hello(result);
    }

}
```

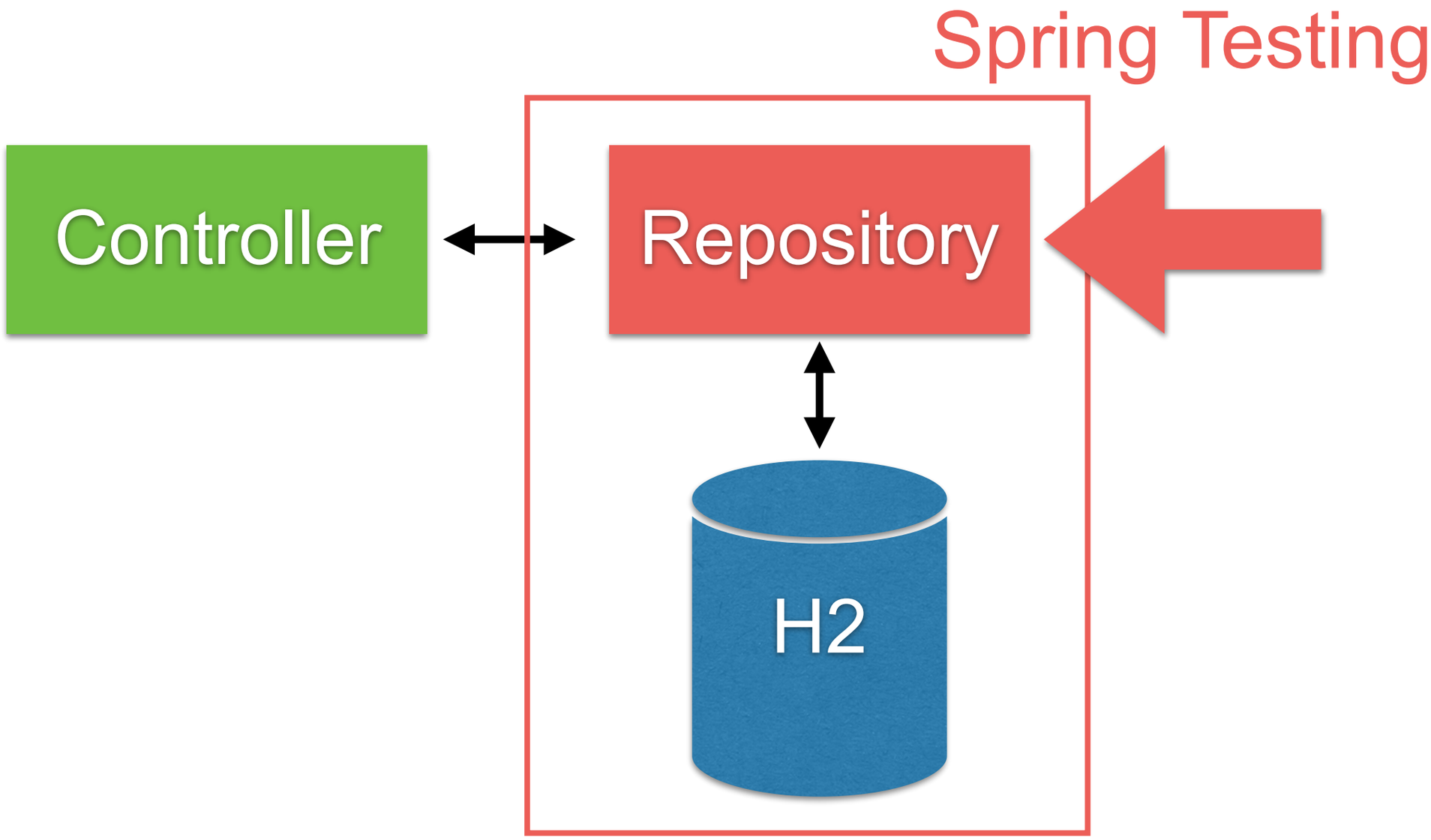
How to test ?

How to test with Repository ?



Repository Testing

Using @DataJpaTest



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)

```java
@RunWith(SpringRunner.class)
@DataJpaTest
public class PersonRepositoryTest {

    @Autowired
    private PersonRepository personRepository;

    @After
    public void clearData() {
        personRepository.deleteAll();
    }
```

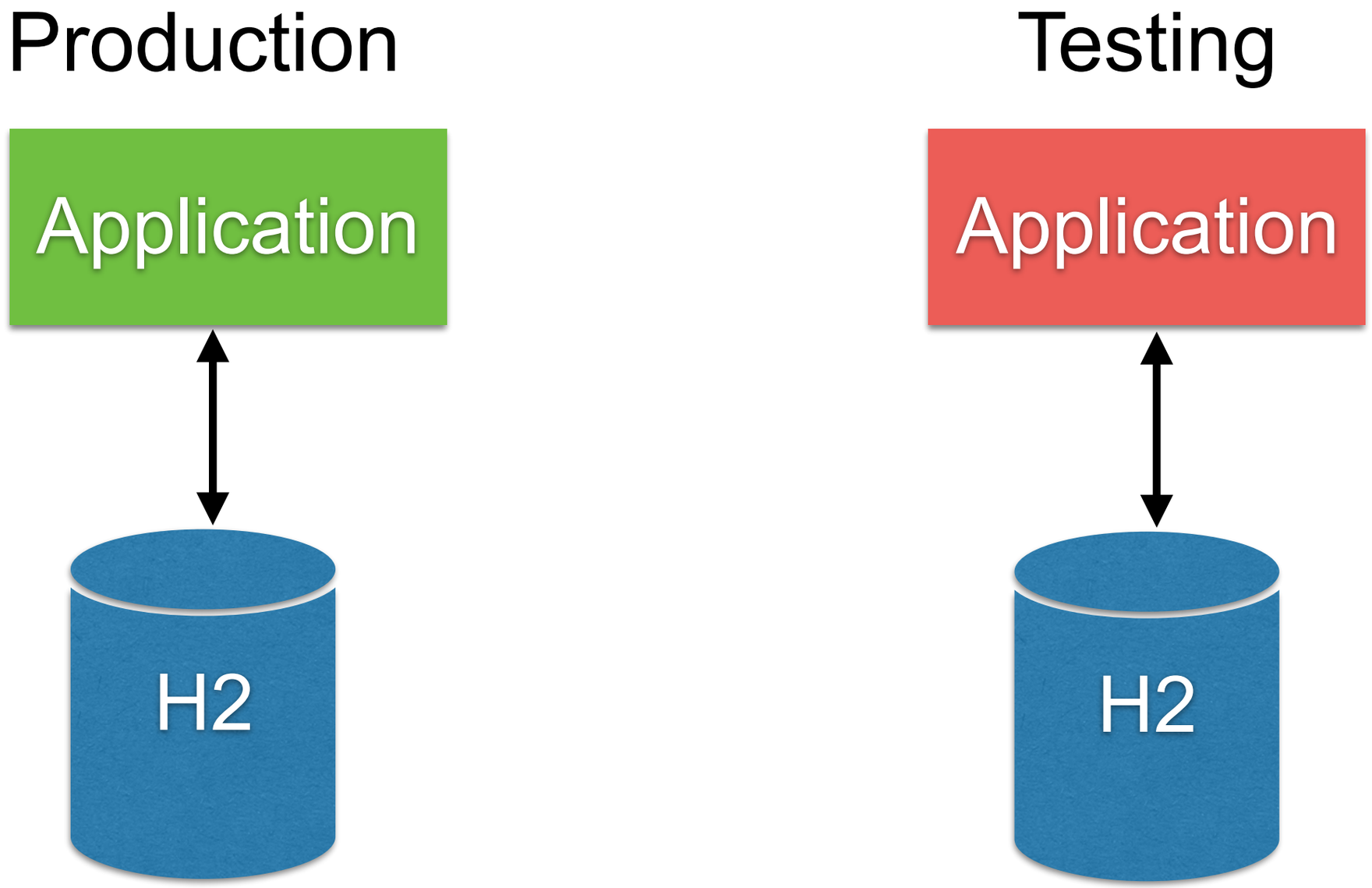
Repository Testing (2)

Add a test case

```
@Test
public void shouldSaveAndGetData() throws Exception {
    //Arrange
    Person somkiat = new Person("somkiat", "pui");
    personRepository.save(somkiat);

    Optional<Person> shouldSomkiat
            = personRepository.findByFirstName("somkiat");

    assertEquals( expected: "somkiat",
            shouldSomkiat.get().getFirstName());

}
```

Working with Real Database

Working with Database



Initial database 1

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 2

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

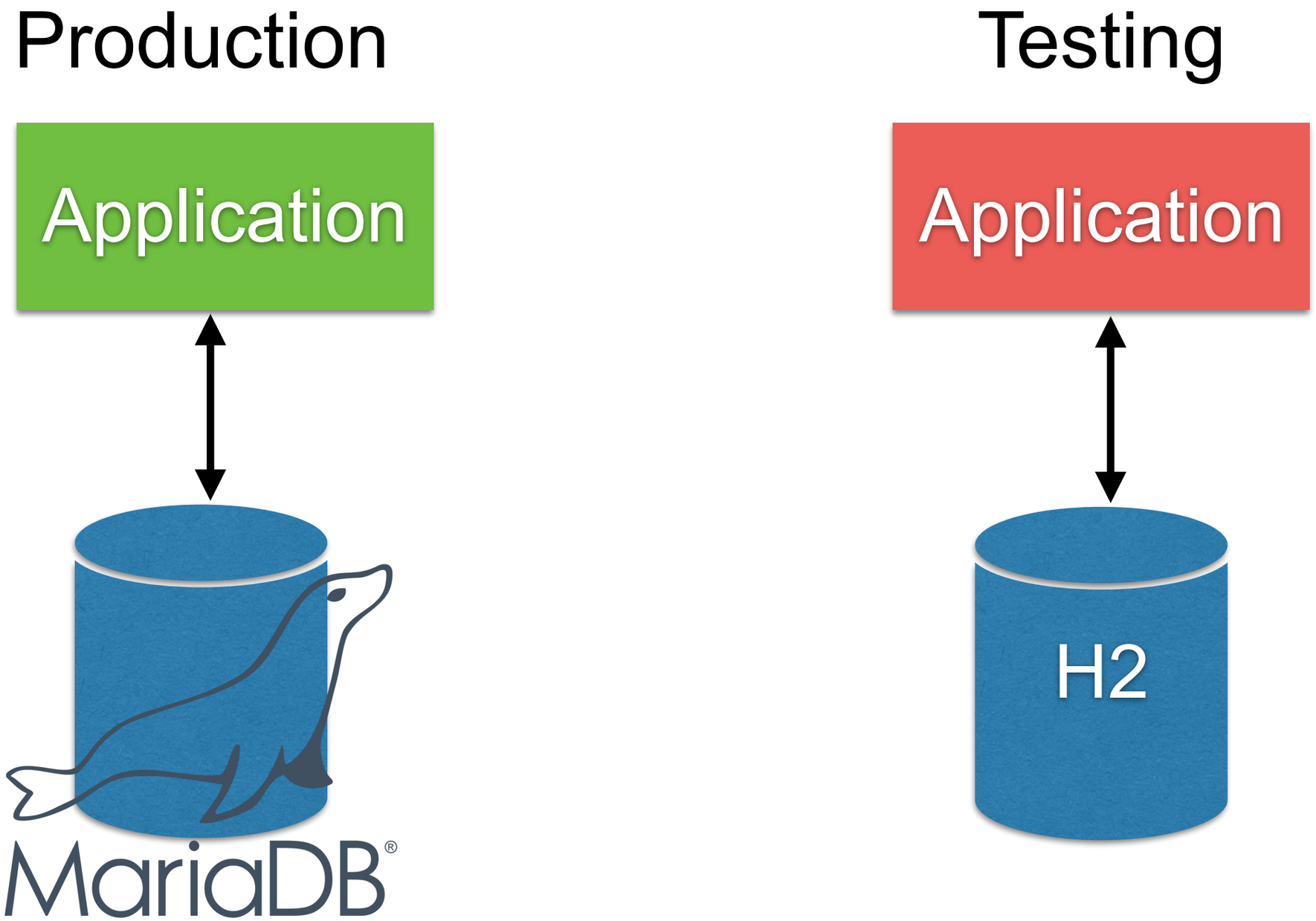
Execute file schema.sql and data.sql

```
o.s.jdbc.datasource.init.ScriptUtils     : Executing SQL script from URL [file:/Users/somkiat/dat

o.s.jdbc.datasource.init.ScriptUtils     : Executed SQL script from URL [file:/Users/somkiat/data
49 ms.
o.s.jdbc.datasource.init.ScriptUtils     : Executing SQL script from URL [file:/Users/somkiat/dat

o.s.jdbc.datasource.init.ScriptUtils     : Executed SQL script from URL [file:/Users/somkiat/data
ms.
```

Run service

http://localhost:8080/account/0868696209

Working with Database



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

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

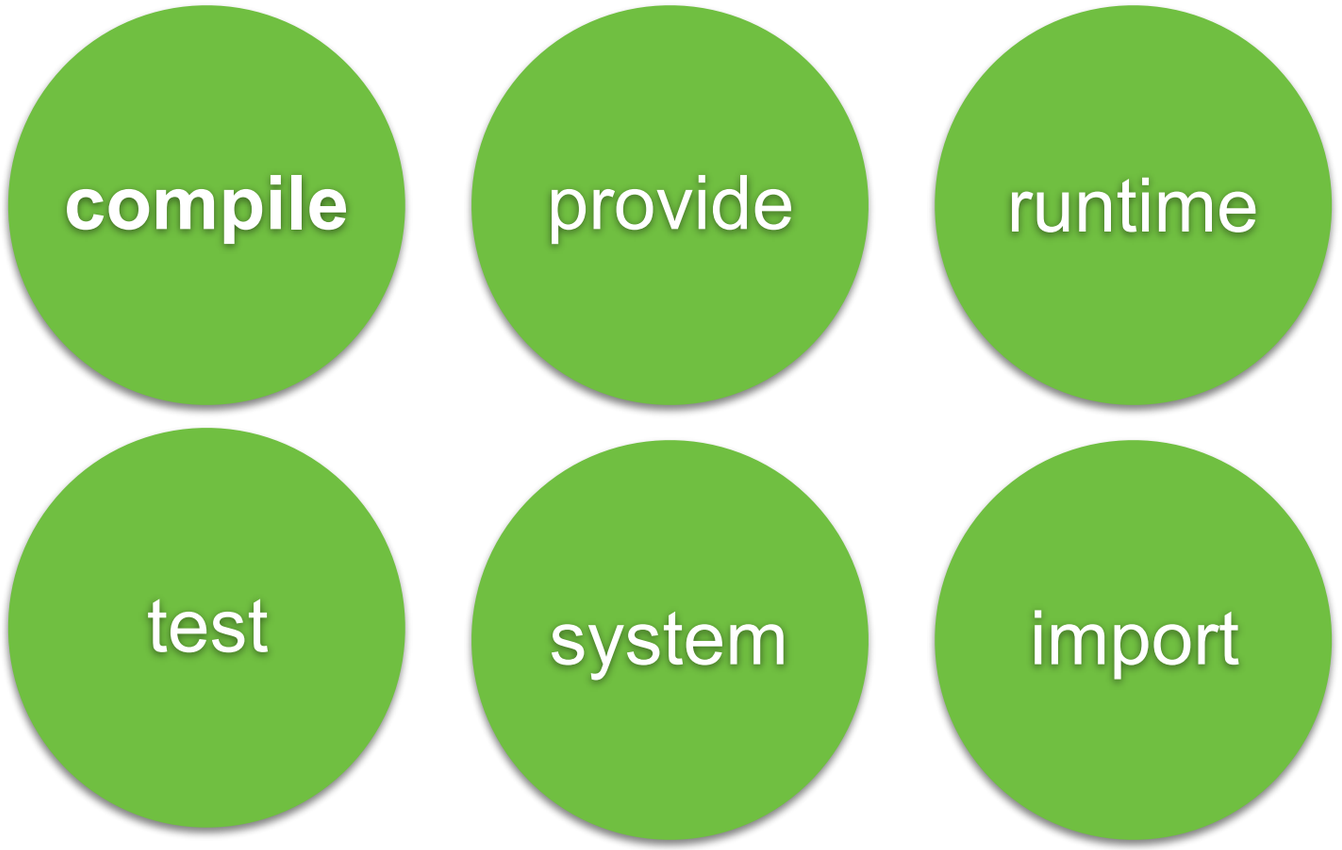
Create file /resources/application.yml in test folder

config of H2 database

```
spring:
  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

Question ?

show_sql=true not working ?

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

Beautiful sql format

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

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.Dispatc
FrameworkServlet 'dispatcherServlet': initialization completed in 37 ms
 13:18:36.864  INFO 79616 --- [nio-8080-exec-2] o.h.h.i.QueryTranslator
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
INFO (default)	Green
DEBUG	Green
TRACE	Green

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

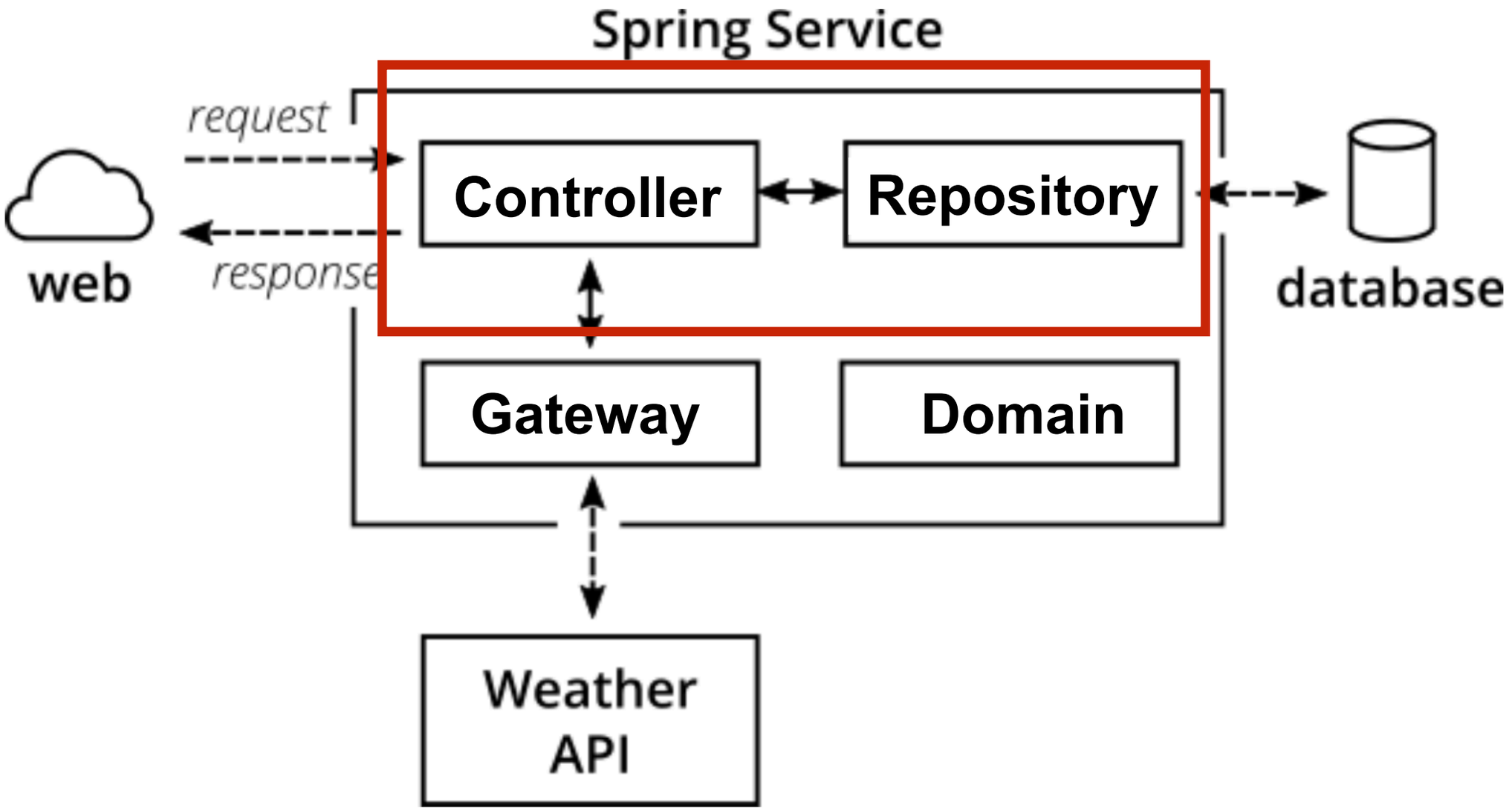
Testing with Mocking

Controller/Service Testing

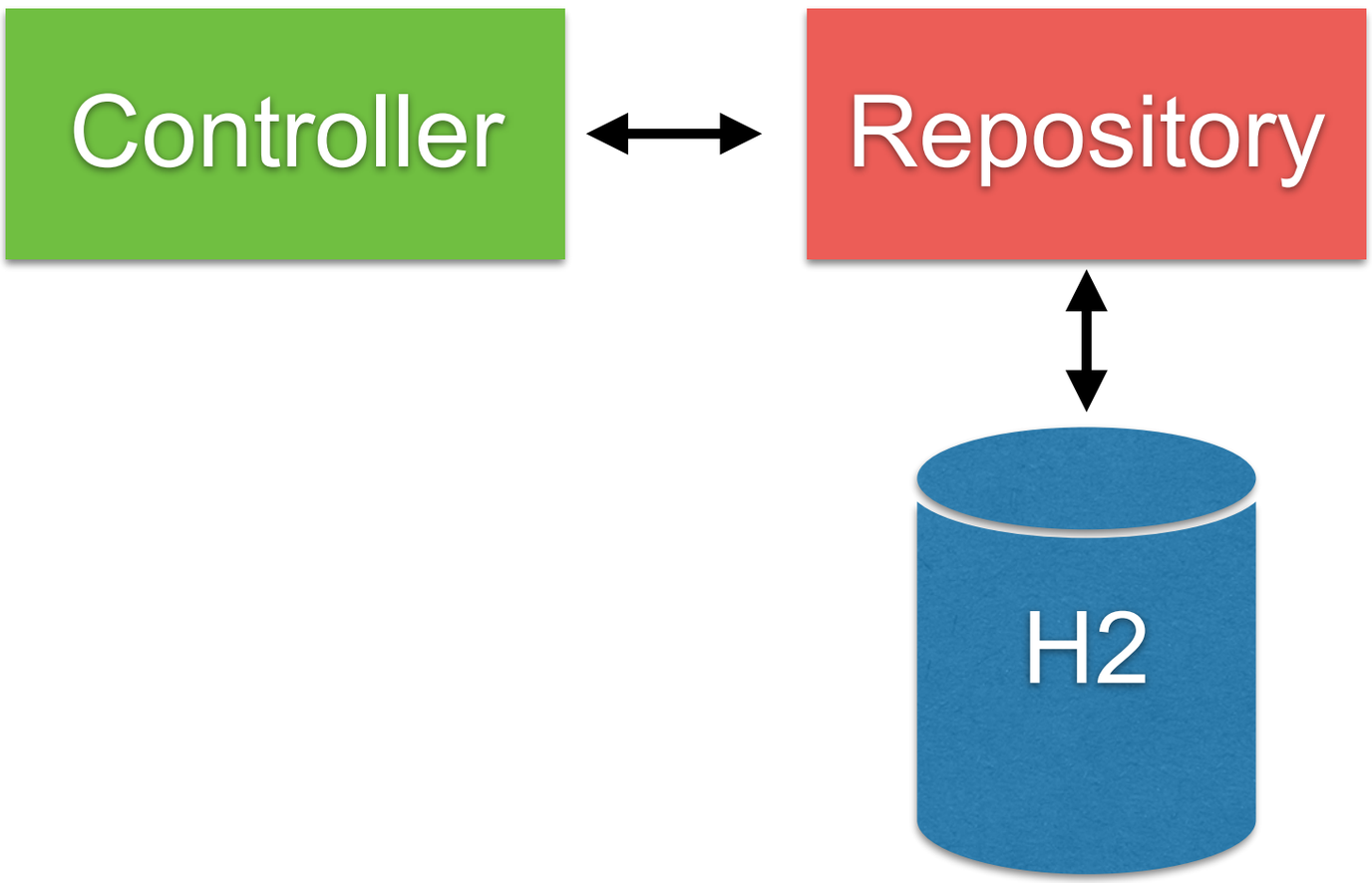
Unit testing ?

Spring Unit testing with MockMvc ?

Controller Testing with Unit test

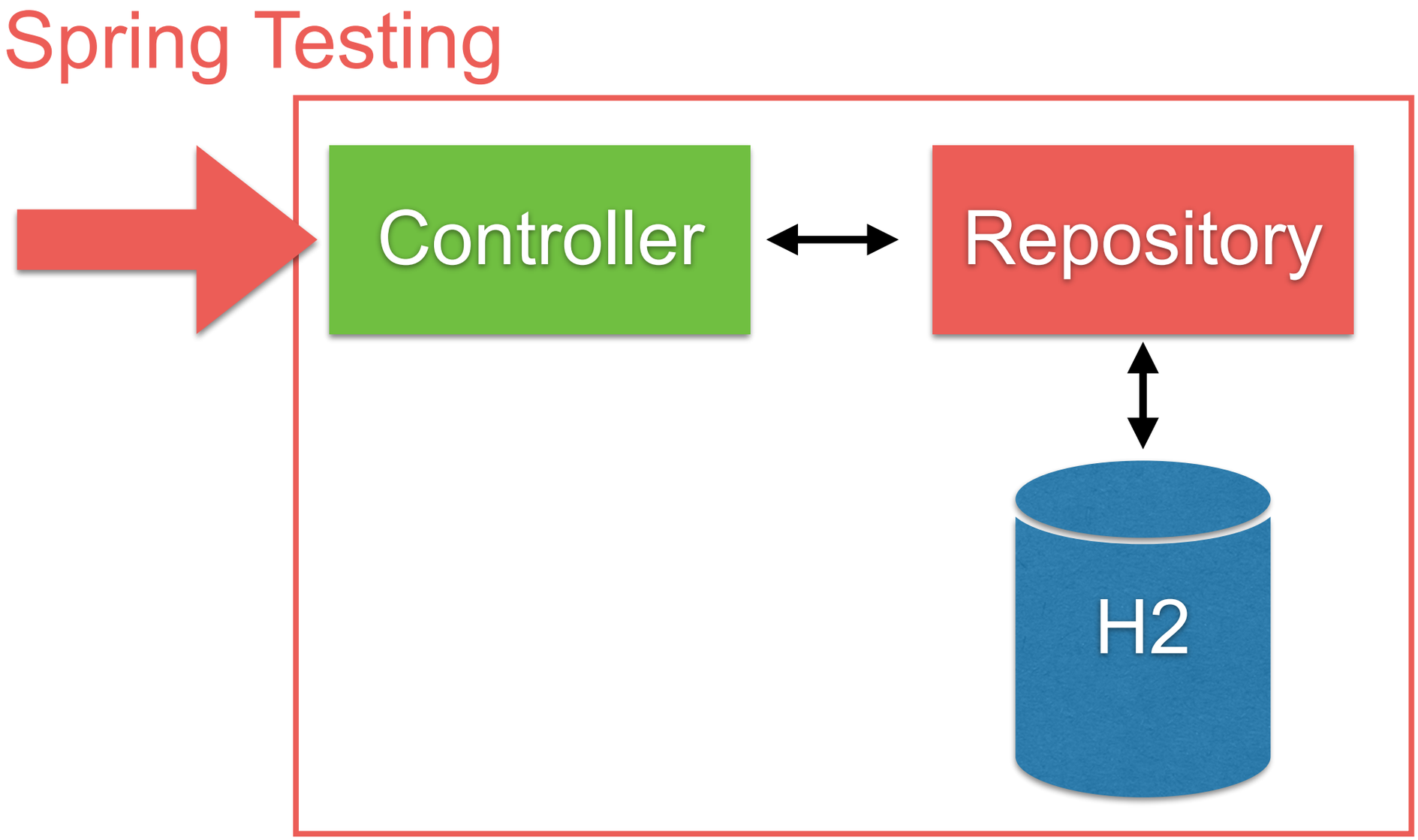


Controller Testing



Controller Testing

Using @SpringBootTest



Unit test

Use Test Double

In java, use Mockito library

http://site.mockito.org/

Unit test with Mockito (1)

```
public class HelloControllerWithRepositoryTest {

    private HelloControllerWithRepository controllerWithRepository;

    @Mock
    private PersonRepository personRepository;

    @Before
    public void init() {
        initMocks( testClass: this);
        controllerWithRepository
                = new HelloControllerWithRepository(personRepository);
    }
```

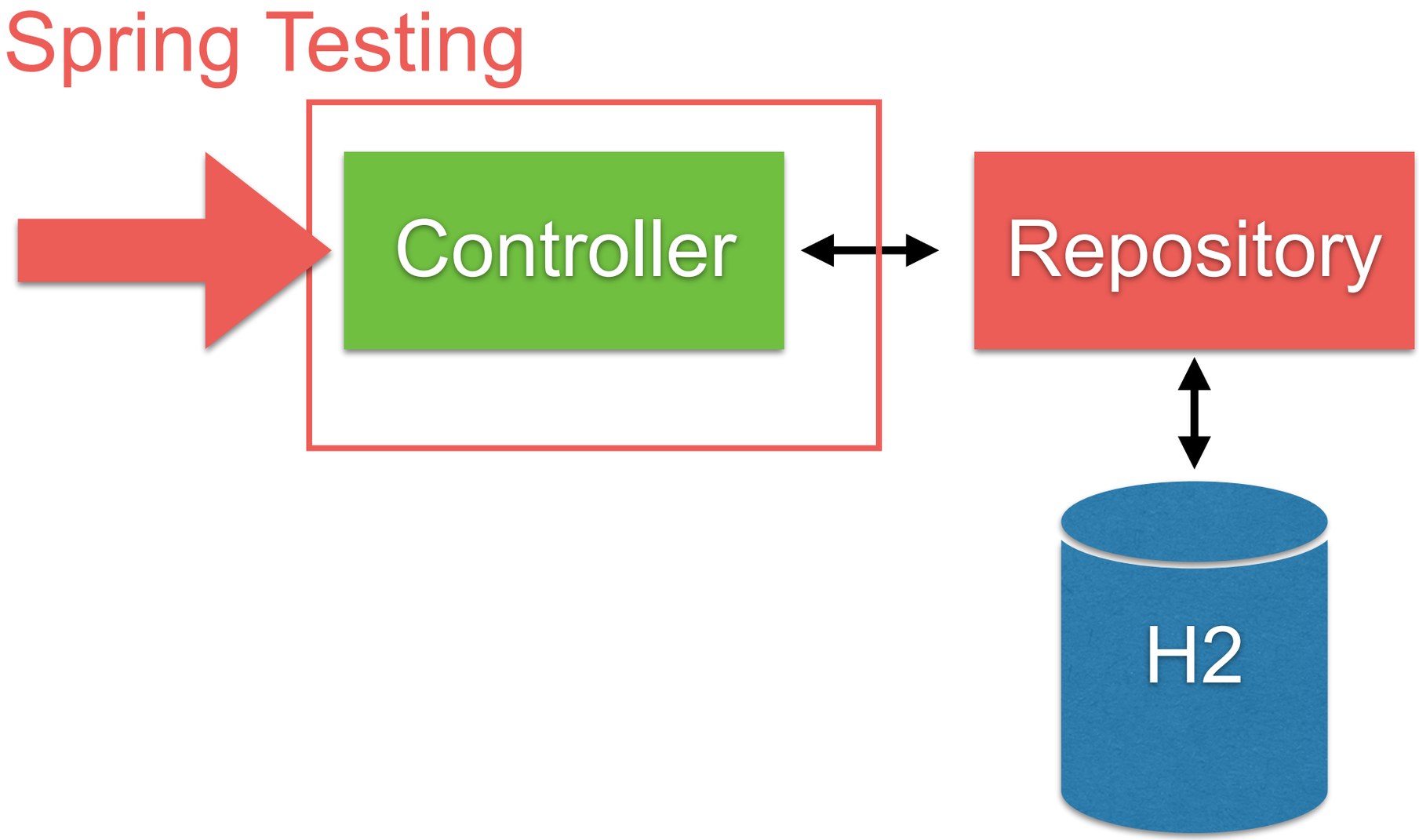
Unit test with Mockito (2)

```java
@Test
public void shouldReturnHelloSomkiat() {
    //Arrange
    Person somkiat = new Person("somkiat", "pui");
    given(personRepository.findByFirstName("somkiat"))
            .willReturn(Optional.of(somkiat));

    // Action
    Hello hello = controllerWithRepository.sayHi( name: "somkiat");

    // Assert
    assertEquals( expected: "Hello somkiat", hello.getMessage());
}
```

Try by yourself

Controller Testing with mock

Using @SpringBootTest



Controller Testing with mock (1)

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

    @Autowired
    private TestRestTemplate testRestTemplate;

    @MockBean
    private AccountRepository accountRepository;
```

Controller Testing with mock (2)

Setup datafrom repository

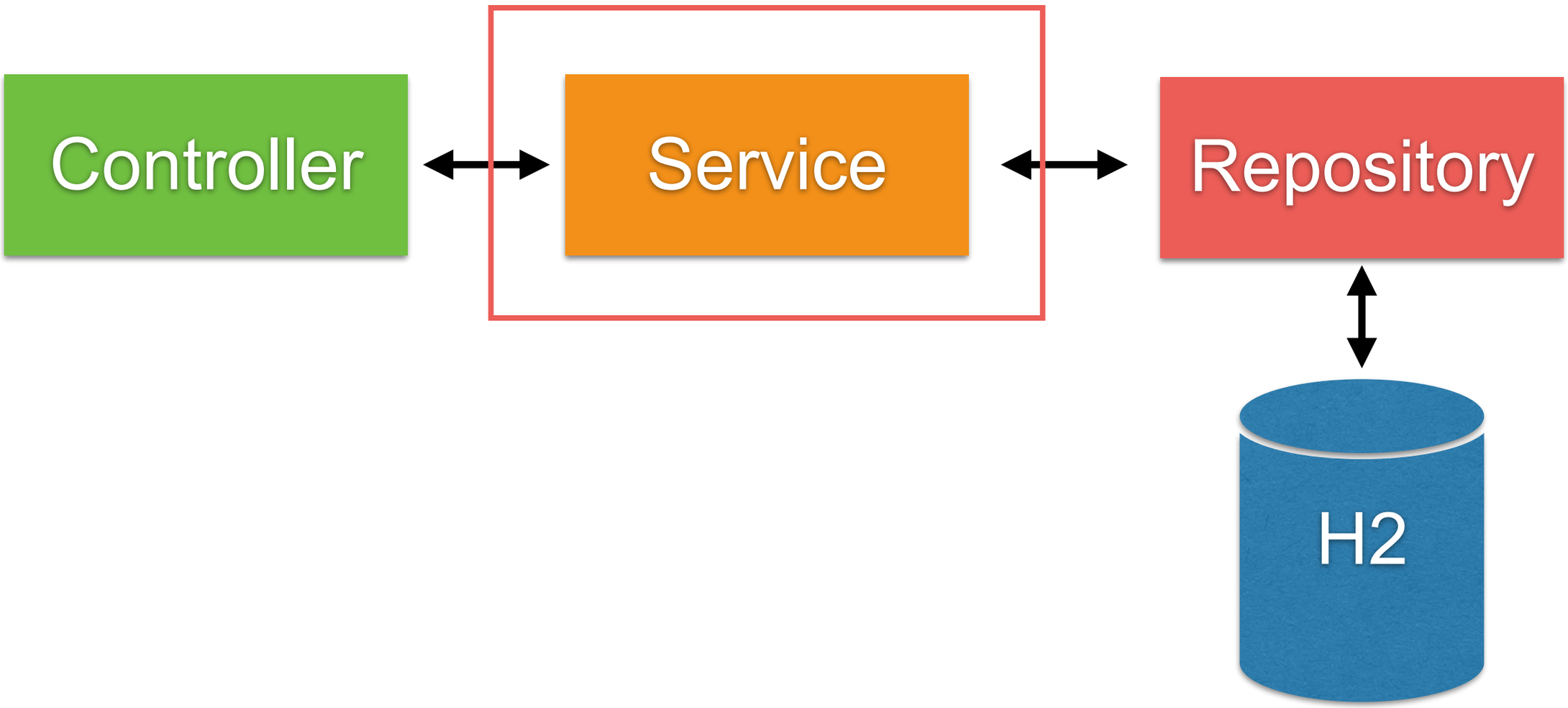
```
@Test
public void
เรียกข้อมูลบัญชีของหมายเลข_0868696209_ต้องเจอสองบัญชีนะ() {
    // Arrange
    List<Account> accountIterable = new ArrayList<>();
    accountIterable.add(new Account("01"));
    accountIterable.add(new Account("02"));
    given(accountRepository.findAll()).willReturn(accountIterable);

    // Act
    List<AccountResponse> accountResponses
            = testRestTemplate.getForObject(
                    "/account/0868696209", List.class);

    // Assert
    assertEquals(2, accountResponses.size());
}
```

Controller Testing with mock (3)

```
@Test
public void
เรียกข้อมูลบัญชีของหมายเลข_0868696209_ต้องเจอสองบัญชีนะ() {
    // Arrange
    List<Account> accountIterable = new ArrayList<>();
    accountIterable.add(new Account("01"));
    accountIterable.add(new Account("02"));
    given(accountRepository.findAll()).willReturn(accountIterable);

    // Act
    List<AccountResponse> accountResponses
            = testRestTemplate.getForObject(
                    "/account/0868696209", List.class);

    // Assert
    assertEquals(2, accountResponses.size());
}
```

Call service with rest template

Controller Testing with Unit test



Try by yourself