Java Unit Testing com Spring Boot 3, Junit 5 e Mockito

Theory

0203 O que são Testes Unitários?

Leandro Costa

https://www.erudio.com.br/ https://www.youtube.com/c/ErudioTraining

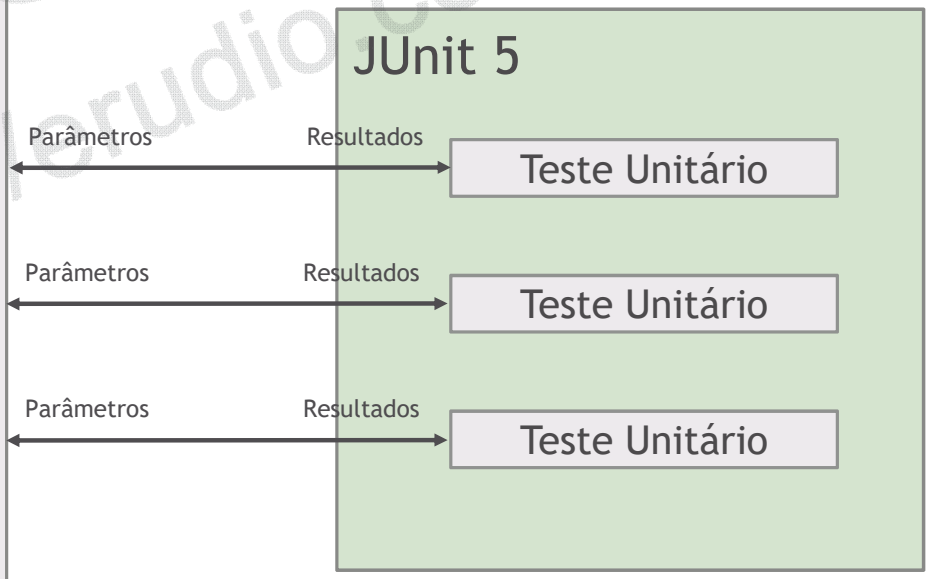
https://www.semeru.com.br/ https://hub.docker.com/u/leandrocgsi/

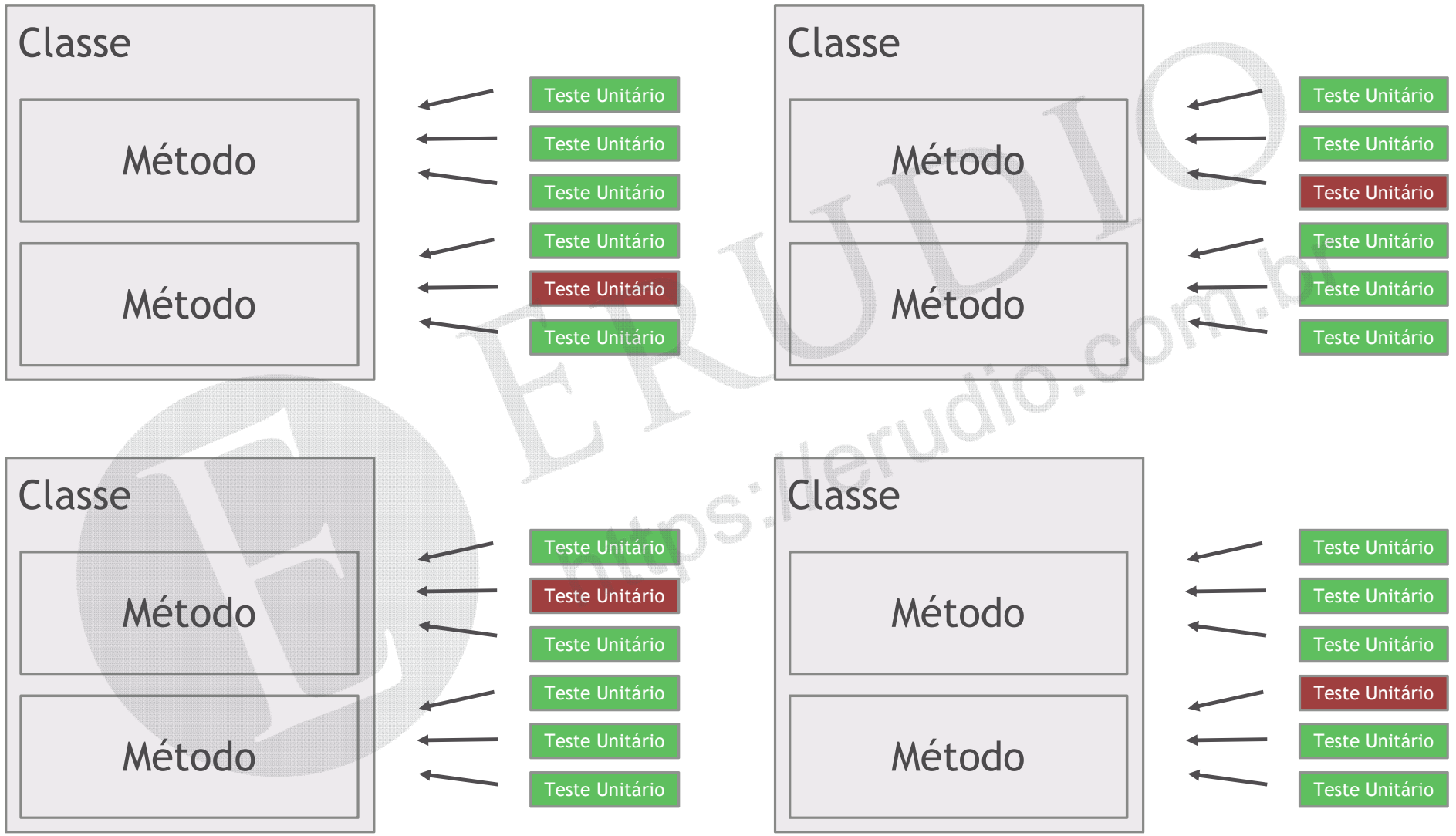
https://github.com/leandrocgsi/automated-tests-with-java-erudio

O que são Testes Unitários?

Um **teste unitário** é um pequeno método que você escreve para testar alguma parte do seu código.

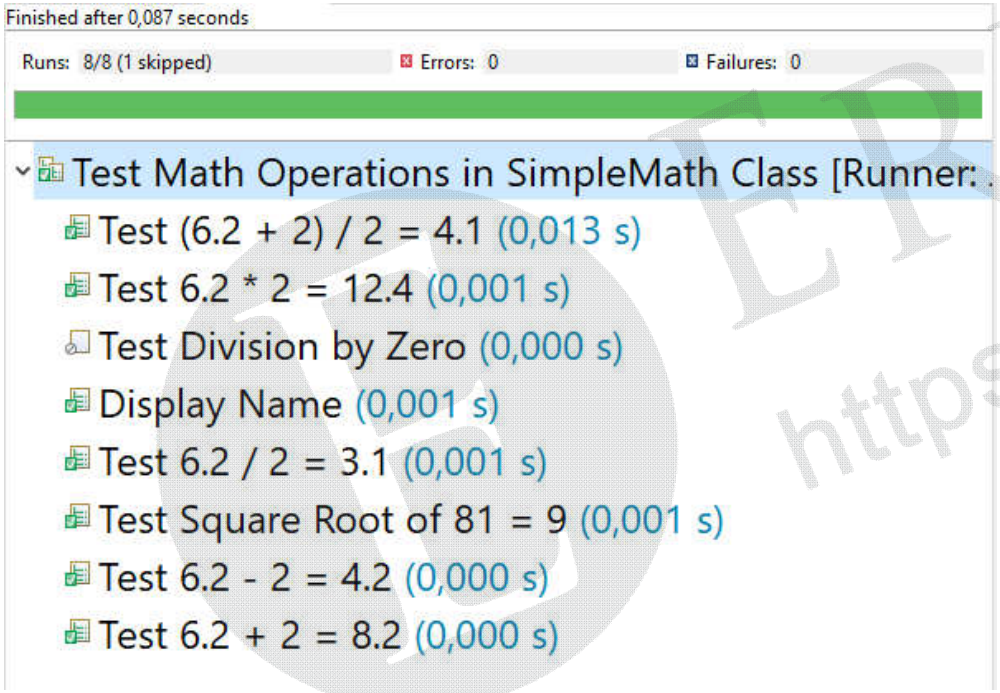
```
public String generateCoupon(String couseId) {

    // Code under test

}
```

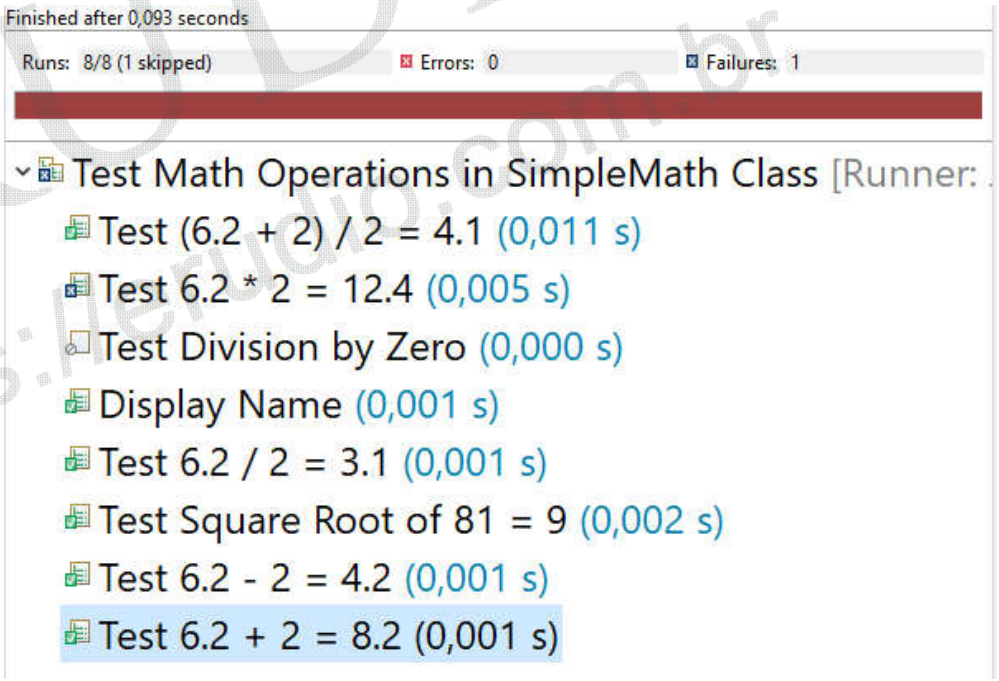


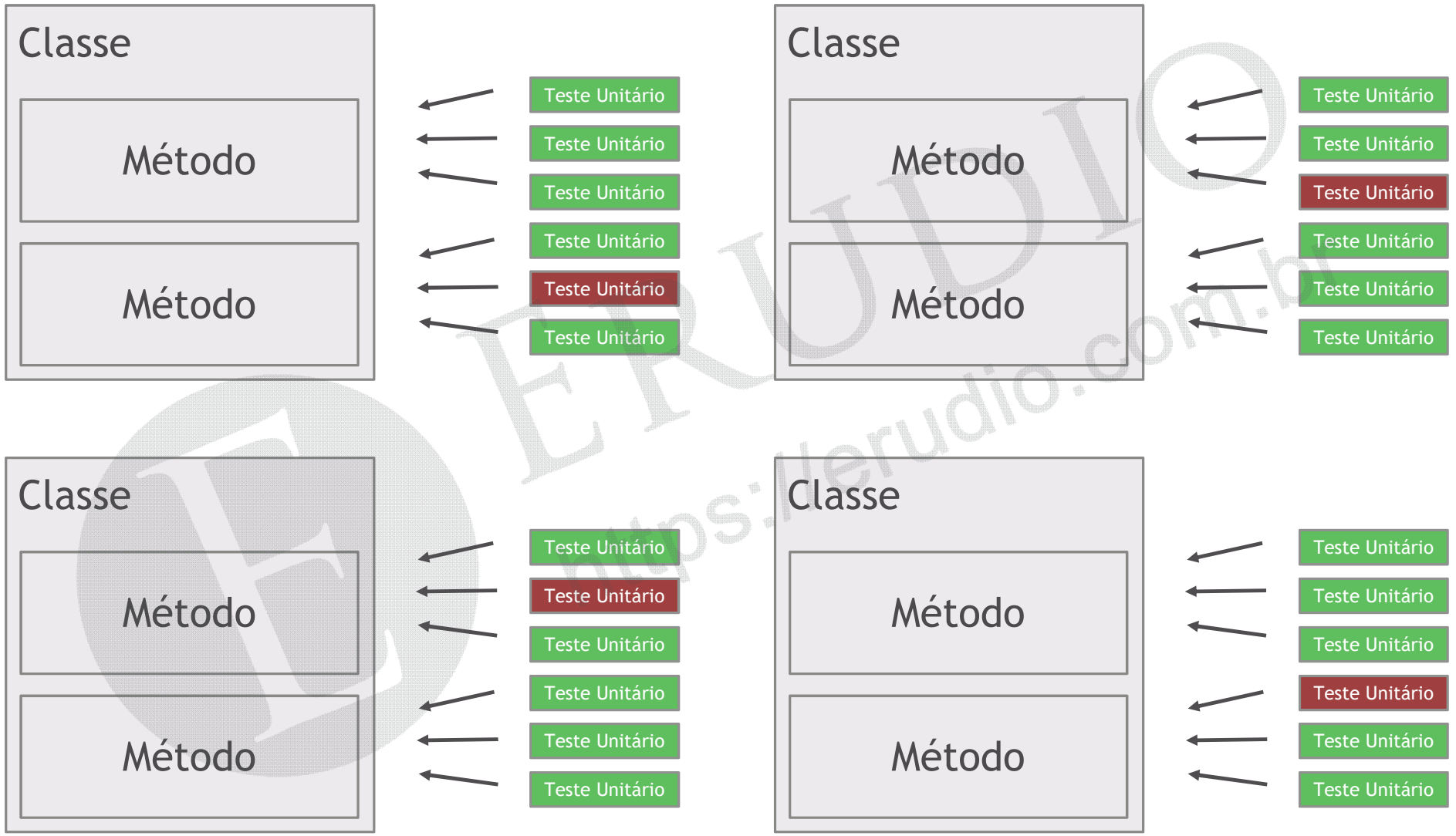

Classe
Método
Método
Teste Unitário
Teste Unitário
Teste Unitário
Teste Unitário
Teste Unitário
Teste Unitário
Classe
Método
Método
Teste Unitário
Teste Unitário
Teste Unitário
Teste Unitário
Teste Unitário
Teste Unitário
Classe
Método
Método
Teste Unitário
Teste Unitário
Teste Unitário
Teste Unitário
Teste Unitário
Teste Unitário
Classe
Método
Método
Teste Unitário
Teste Unitário
Teste Unitário
Teste Unitário
Teste Unitário
Teste Unitário

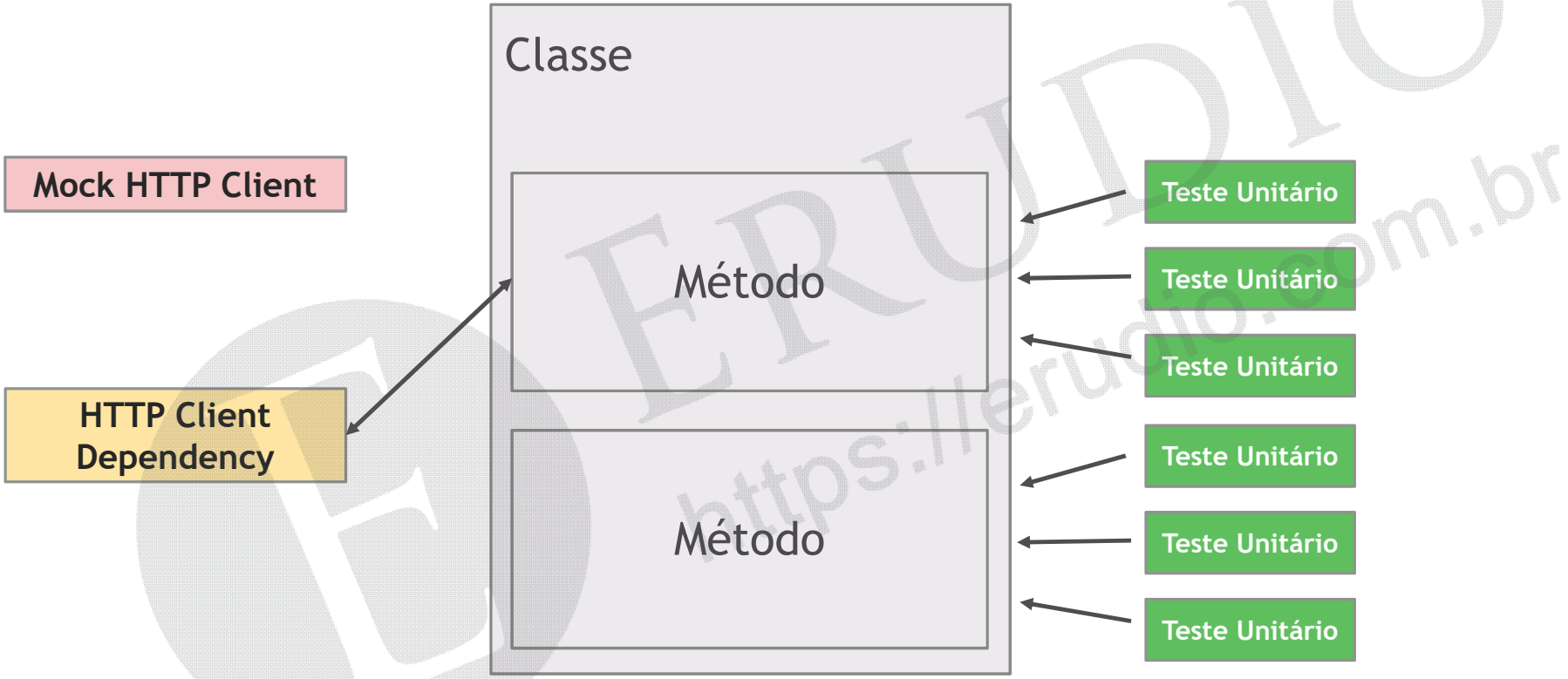
```java
public class SimpleMath {

    public Double sum(Double firstNumber, Double secondNumber)
    {
        return firstNumber + secondNumber;
    }
}

@Test
void testSum_WhenSixDotTwoIsAddedByTwo_ShouldReturnEightDotTwo() {
    // Given / Arrange
    SimpleMath math = new SimpleMath();

    // When / Act
    Double actual = math.sum(6.2D, 2D);

    // Then / Assert
    assertEquals(8.2D, actual, () -> "6.2 + 2 did not produce 8.2!");
}
```

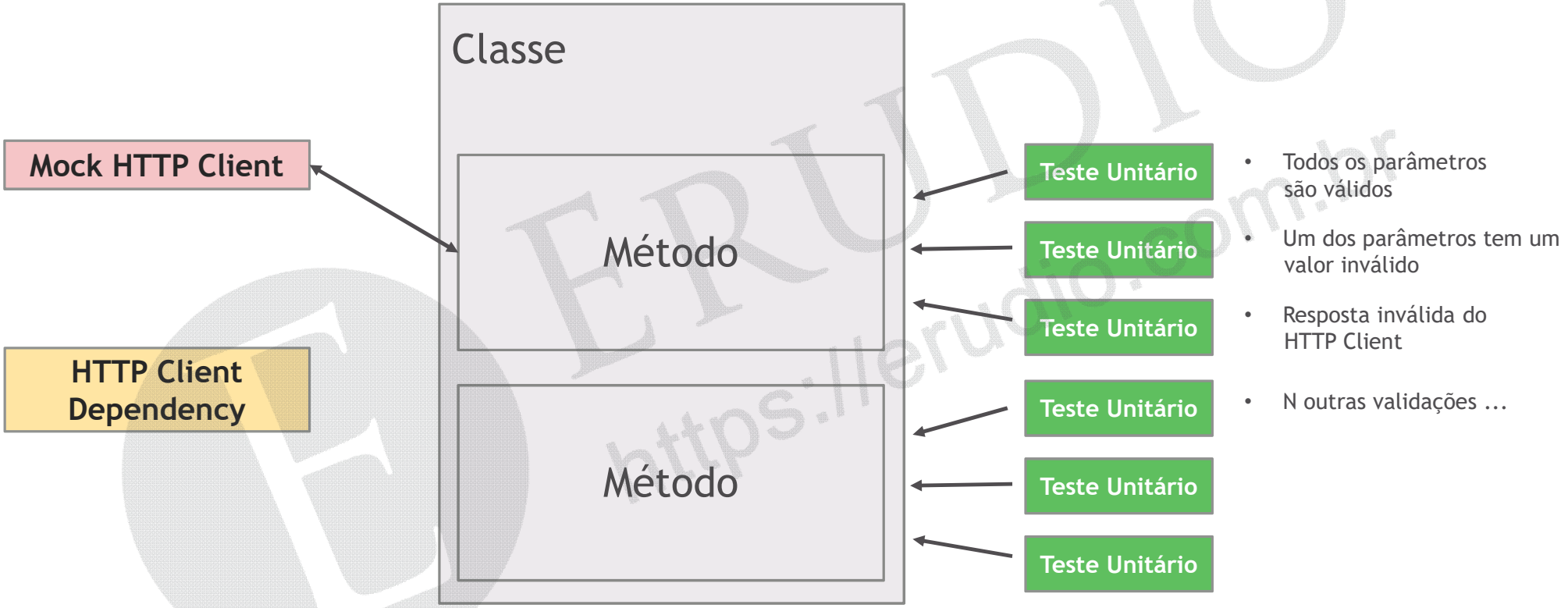
Test Reports






Classe
Método
Método
Teste Unitário
Teste Unitário
Teste Unitário
Teste Unitário
Teste Unitário
Teste Unitário
Classe
Método
Método
Teste Unitário
Teste Unitário
Teste Unitário
Teste Unitário
Teste Unitário
Teste Unitário
Classe
Método
Método
Teste Unitário
Teste Unitário
Teste Unitário
Teste Unitário
Teste Unitário
Teste Unitário
Classe
Método
Método
Teste Unitário
Teste Unitário
Teste Unitário
Teste Unitário
Teste Unitário
Teste Unitário


Classe
Mock HTTP Client
Método
HTTP Client Dependency
Método
Teste Unitário
Teste Unitário
Teste Unitário
Teste Unitário
Teste Unitário
Teste Unitário


Classe
Mock HTTP Client
HTTP Client Dependency
Método
Método
Teste Unitário
Teste Unitário
Teste Unitário
Teste Unitário
Teste Unitário
Teste Unitário
Todos os parâmetros são válidos
Um dos parâmetros tem um valor inválido
Resposta inválida do HTTP Client
N outras validações ...

0204 Por que escrever Testes Unitários?

Leandro Costa

https://www.erudio.com.br/ https://www.youtube.com/c/ErudioTraining

https://www.semeru.com.br/ https://hub.docker.com/u/leandrocgsi/

https://github.com/leandrocgsi/automated-tests-with-java-erudio

Por que escrever Testes Unitários?

- Ter certeza de que o código funciona;
- O código funciona tanto com parâmetros válidos quanto inválidos ;
- O código funciona hoje e vai continuar funcionando no futuro;
- O código continuará funcionando inclusive depois que fizermos mudanças.

0205 O Princípio F.I.R.S.T.

Leandro Costa

https://www.erudio.com.br/ https://www.youtube.com/c/ErudioTraining

https://www.semeru.com.br/ https://hub.docker.com/u/leandrocgsi/

https://github.com/leandrocgsi/automated-tests-with-java-erudio

O Princípio F.I.R.S.T.

- **F***ast* – os testes unitários devem executar rapidamente;
- **I***ndependent* – os testes unitários devem ser independentes uns dos outros;
- **R***epeatable* – os testes unitários devem ser repetíveis;
- **S***elf-validating* – os testes unitários devem se auto validarem;
- **T***horough & Timely* – os testes unitários devem cobrir casos extremos.

O Princípio F.I.R.S.T.

- **F***ast*
- **I***ndependent*
- **R***epeatable*
- **S***elf-validating*
- **T***horough & Timely*

0206 Testando Código em Isolamento

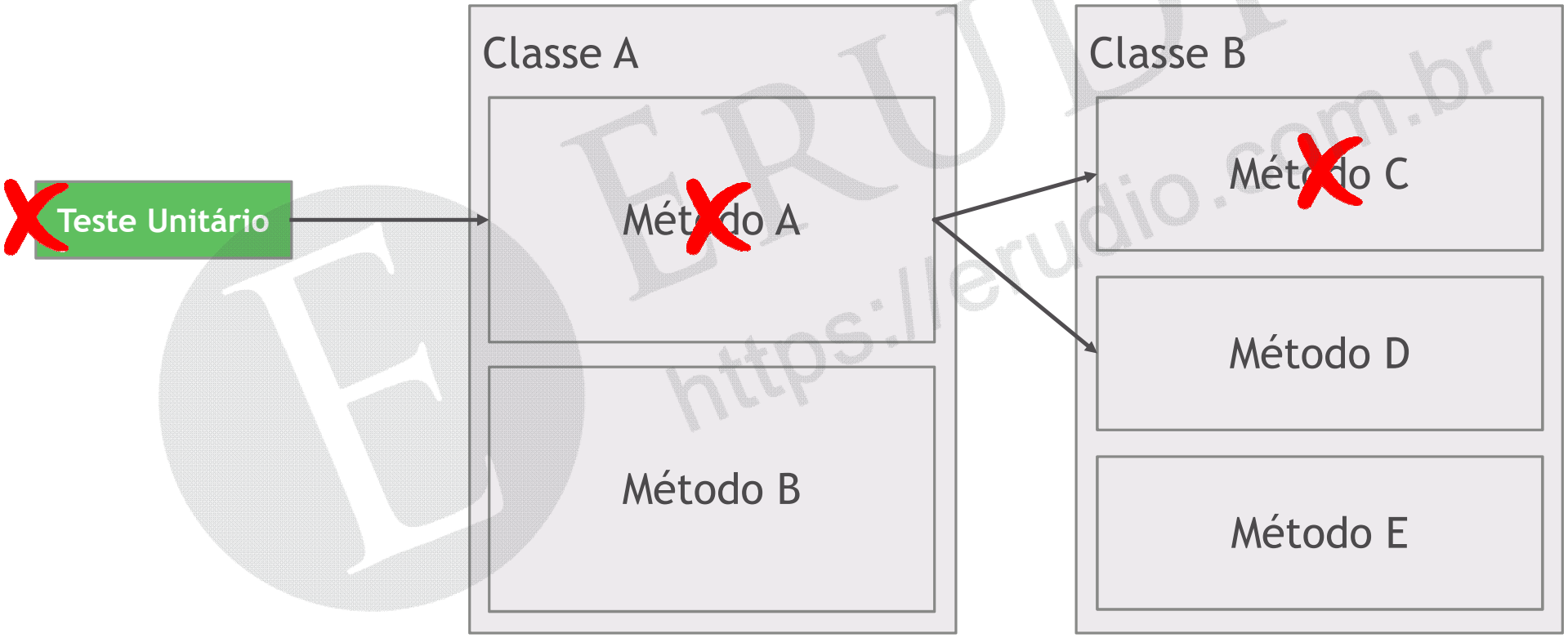
Leandro Costa

https://www.erudio.com.br/ https://www.youtube.com/c/ErudioTraining

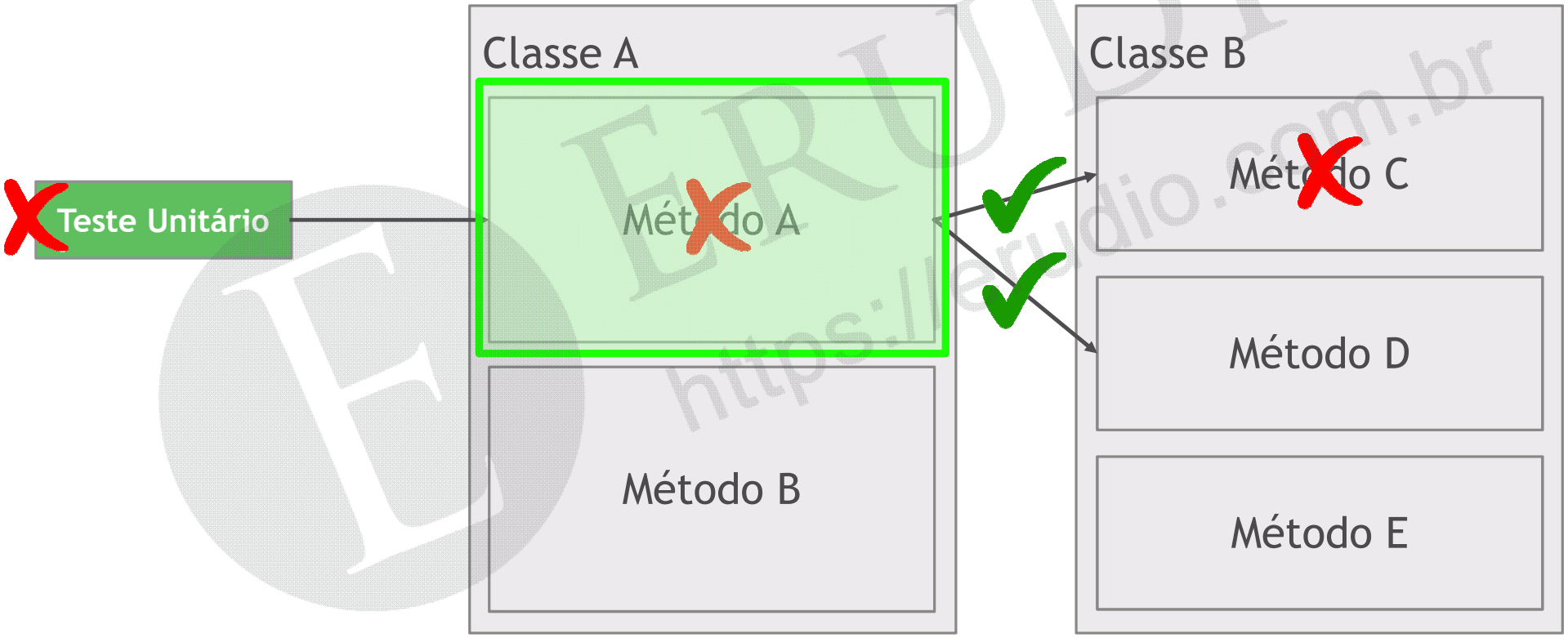
https://www.semeru.com.br/ https://hub.docker.com/u/leandrocgsi/

https://github.com/leandrocgsi/automated-tests-with-java-erudio

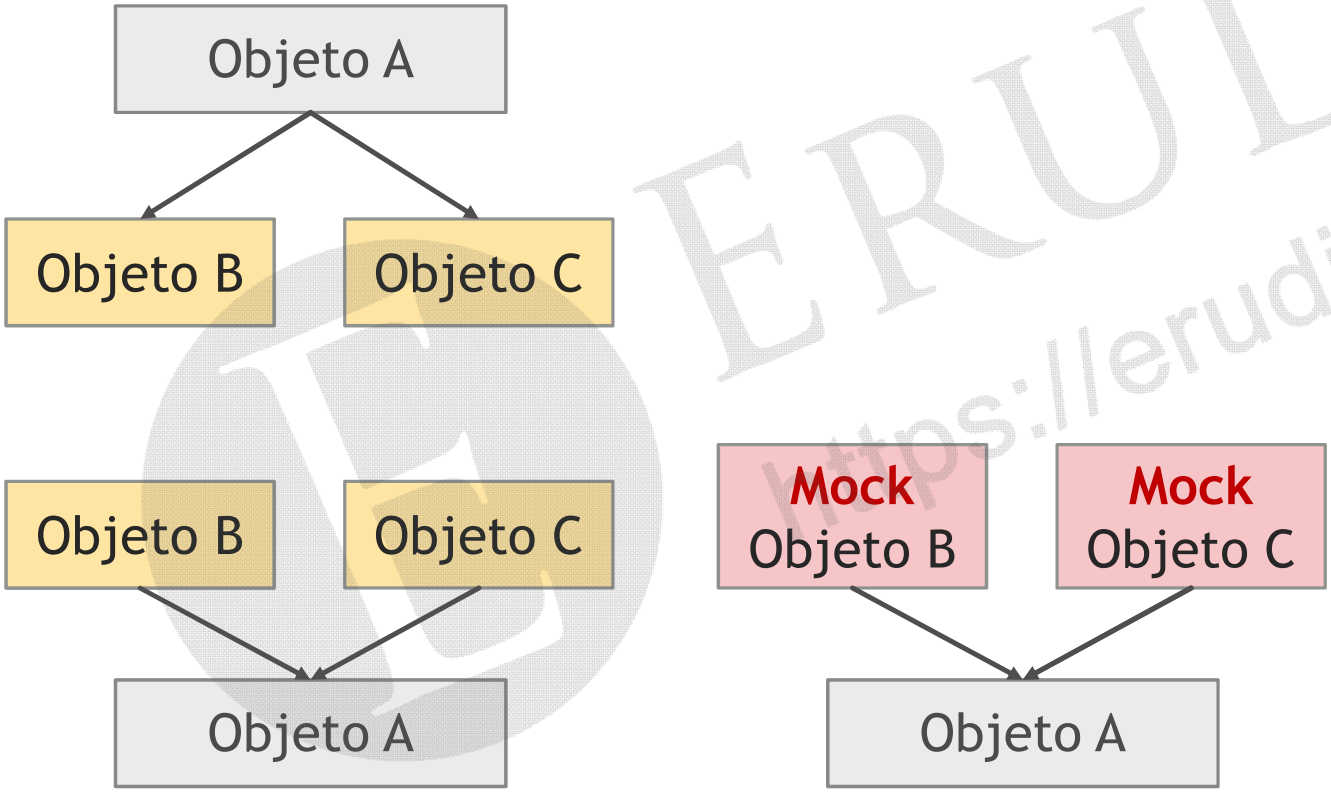
Testando Código em Isolamento



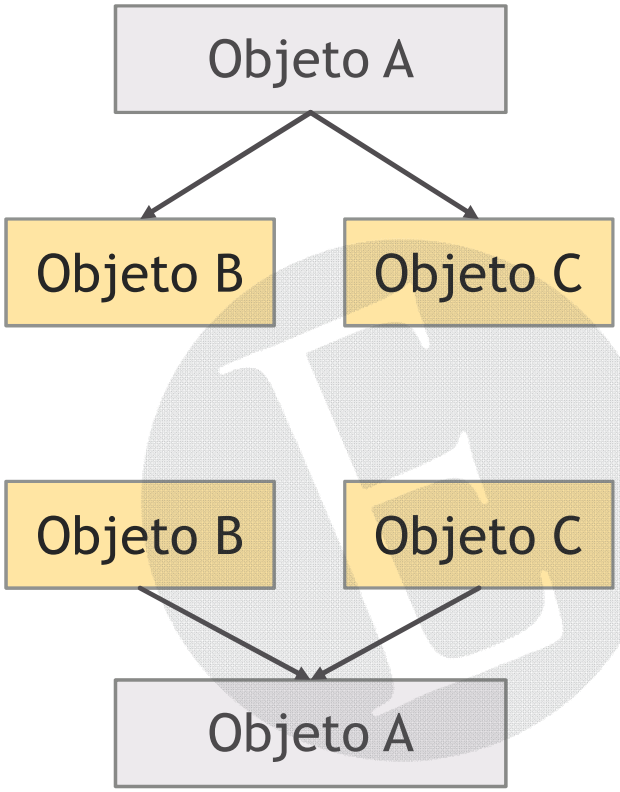
Testando Código em Isolamento

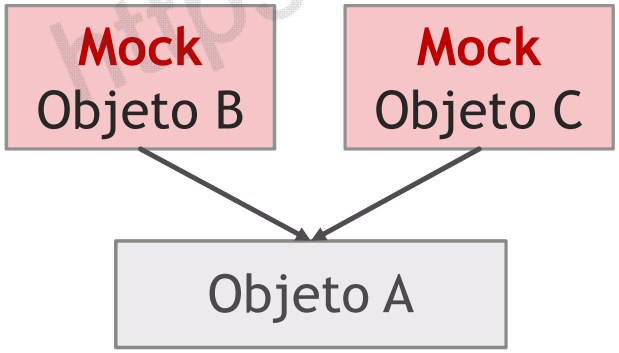


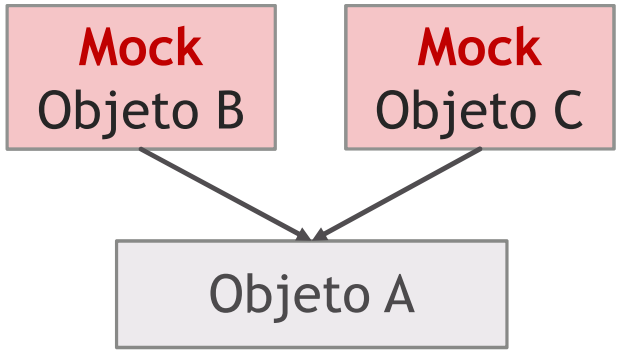
Injeção de Dependências



Injeção de Dependências

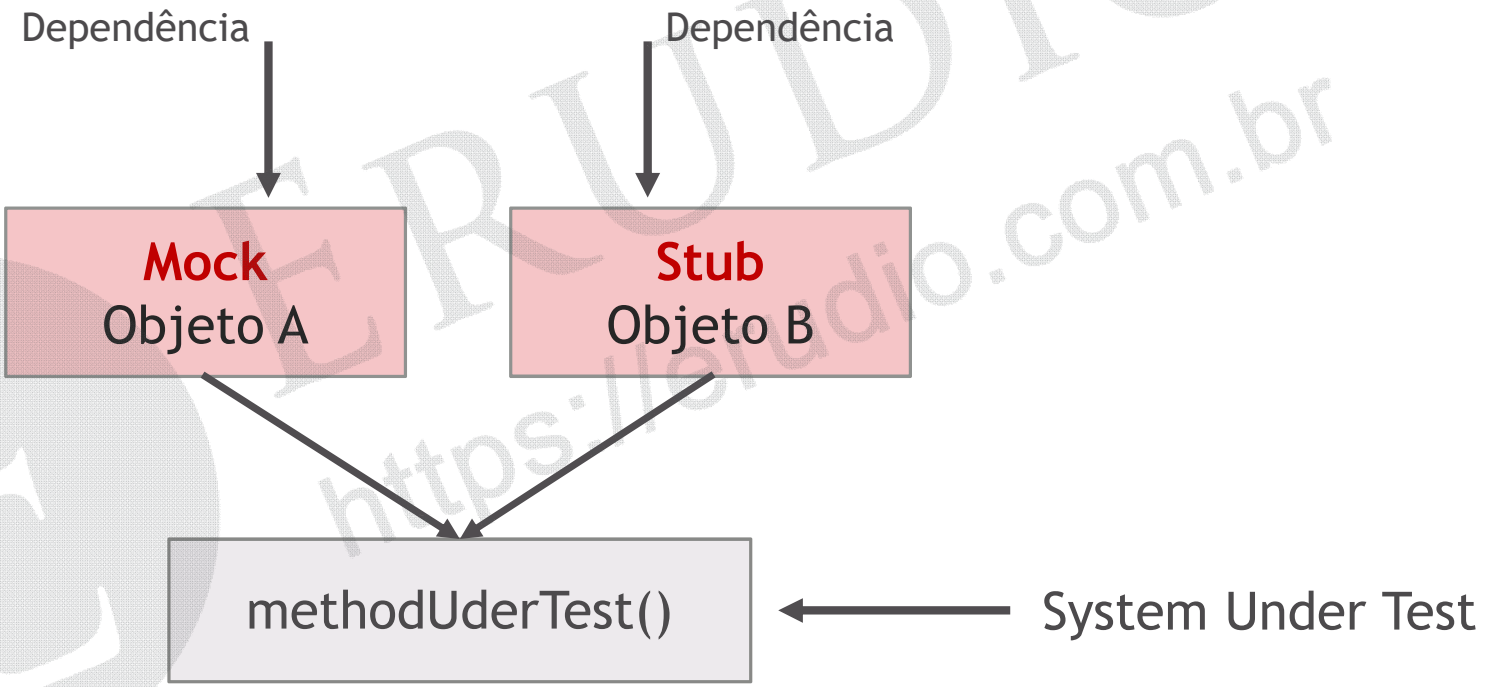






Injeção de Dependências

```
public void checkout(Cart cart) {
   PayoutProcessor payoutProcessor = new PayoutProcessor();
   payoutProcessor.proccessPayment(cart);
}

PayoutProcessor payoutProcessor;

public CartCheckout(PayoutProcessor payoutProcessor) {
   this.payoutProcessor = payoutProcessor;
}

public void Checkout(Cart cart, PayoutProcessor payoutProcessor)
{
   payoutProcessor.proccessPayment(cart);
}
```



0207 A Pirâmide de Testes

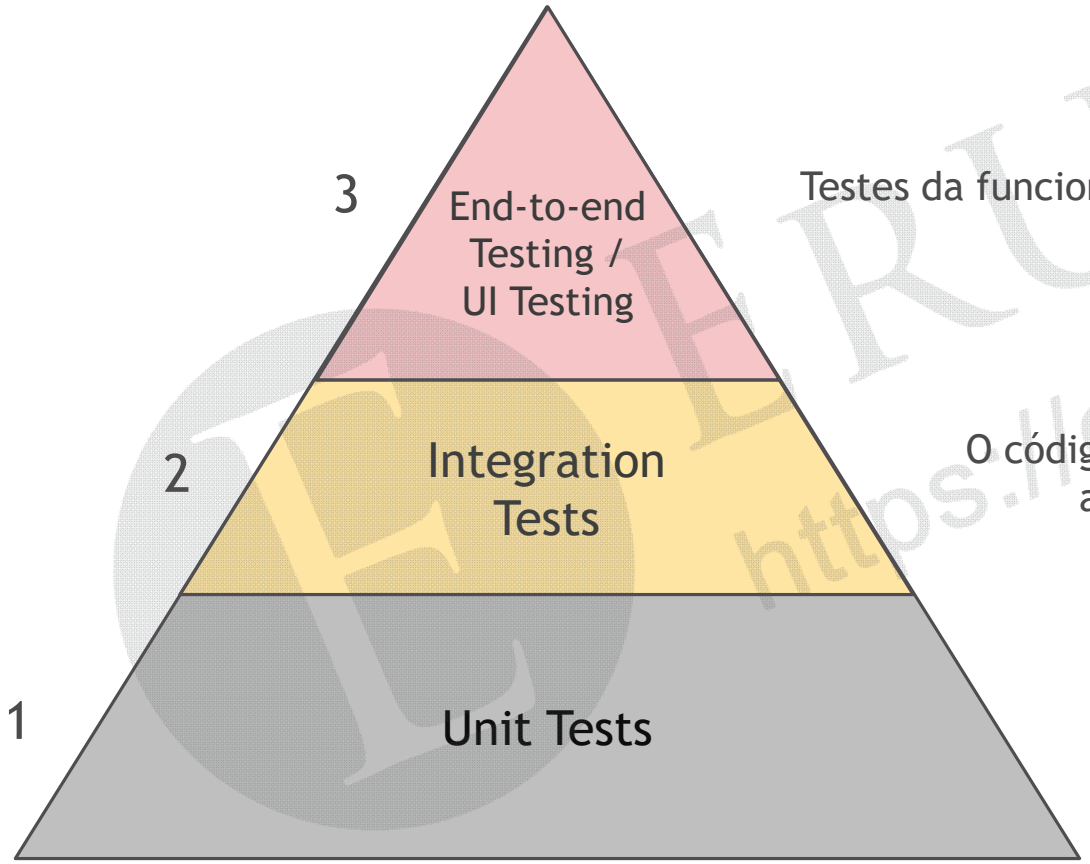
Leandro Costa

https://www.erudio.com.br/ https://www.youtube.com/c/ErudioTraining

https://www.semeru.com.br/ https://hub.docker.com/u/leandrocgsi/

https://github.com/leandrocgsi/automated-tests-with-java-erudio

A Pirâmide de Testes



Testes da funcionalidade de software de ponta a ponta

O código da aplicação é testado sem mockar acessos ao banco de dados e/ou conexões HTTP

Testes de pequenos trechos de código isolados com dependências stubs ou mockadas

0208 O que é JUnit 5

Leandro Costa

https://www.erudio.com.br/ https://www.youtube.com/c/ErudioTraining

https://www.semeru.com.br/ https://hub.docker.com/u/leandrocgsi/

https://github.com/leandrocgsi/automated-tests-with-java-erudio

O que é JUnit 5

JUnit 5 = JUnit Platform + JUnit Jupiter + JUnit Vintage

- ***JUnit Platform*** – A plataforma JUnit é o principal framework para testes na JVM, sendo a base que fundamenta a maioria das novas ferramentas de testes;
- ***JUnit Jupiter*** – É a combinação do novo modelo de programação e extensão para escrever testes e extensões para o JUnit 5;
- ***JUnit Vintage*** – TestEngine de backward compatibility para a execução de testes escritos em JUnit 3 e JUnit 4.

0209 JUnit e Ferramentas de Build

Leandro Costa

https://www.erudio.com.br/ https://www.youtube.com/c/ErudioTraining

https://www.semeru.com.br/ https://hub.docker.com/u/leandrocgsi/

https://github.com/leandrocgsi/automated-tests-with-java-erudio

JUnit e Ferramentas de Build

- *IDE's:*
  - Spring Tool Suite;
  - Eclipse;
  - IntelliJ IDEA;
  - NetBeans;
  - Visual Studio Code
  - etc.

- *Ferramentas de Build:*
  - Maven;
  - Gradle;
  - Ant;
  - etc.

0317 Ciclo de Vida dos Testes no JUnit 5

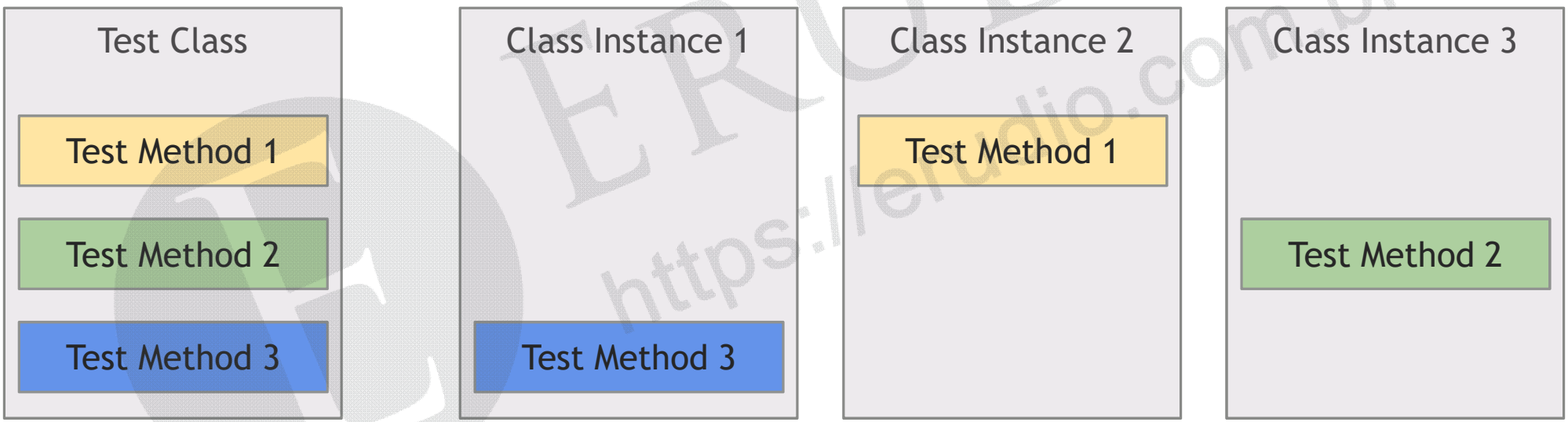
Leandro Costa

https://www.erudio.com.br/ https://www.youtube.com/c/ErudioTraining

https://www.semeru.com.br/ https://hub.docker.com/u/leandrocgsi/

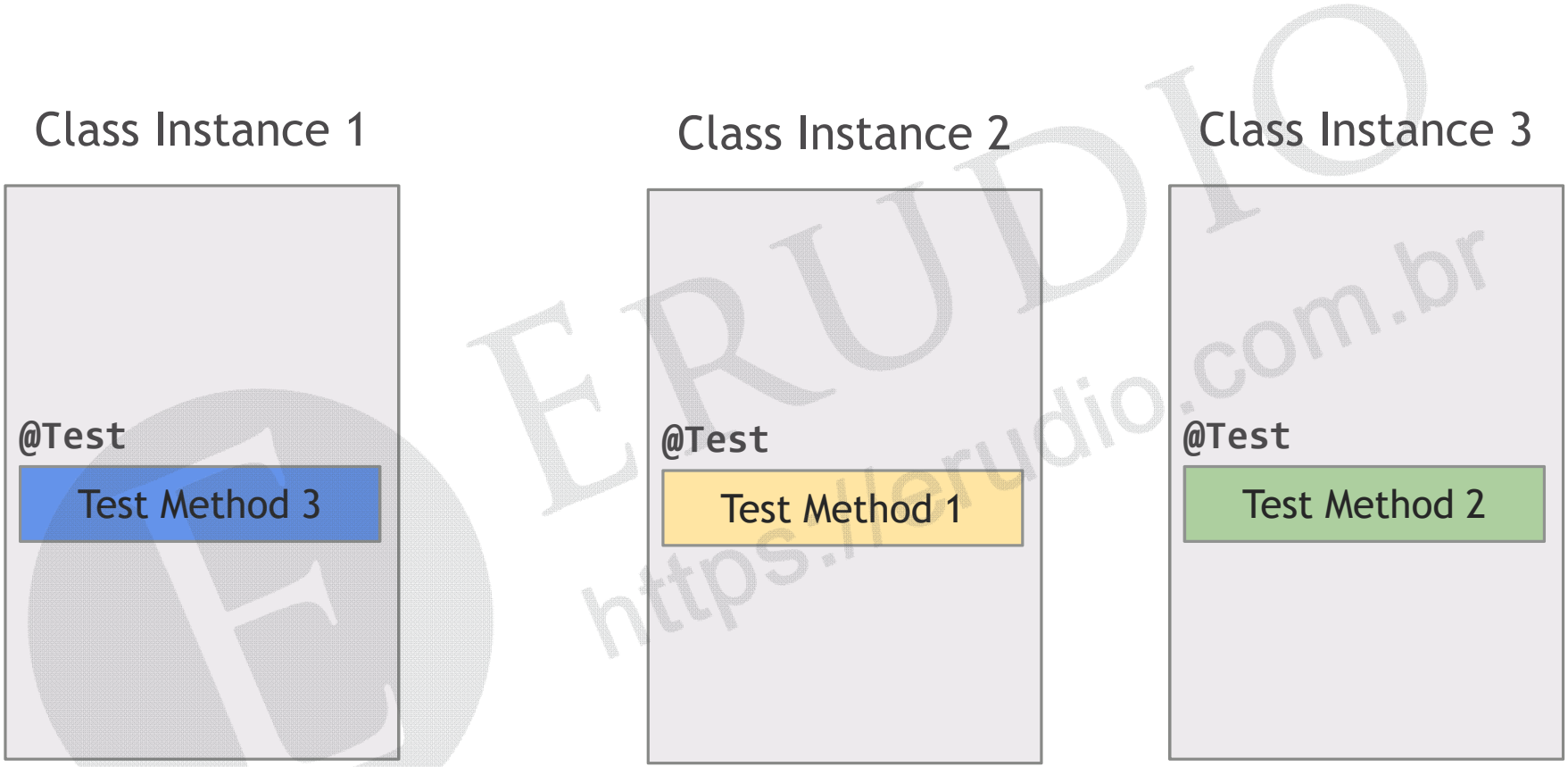
https://github.com/leandrocgsi/automated-tests-with-java-erudio

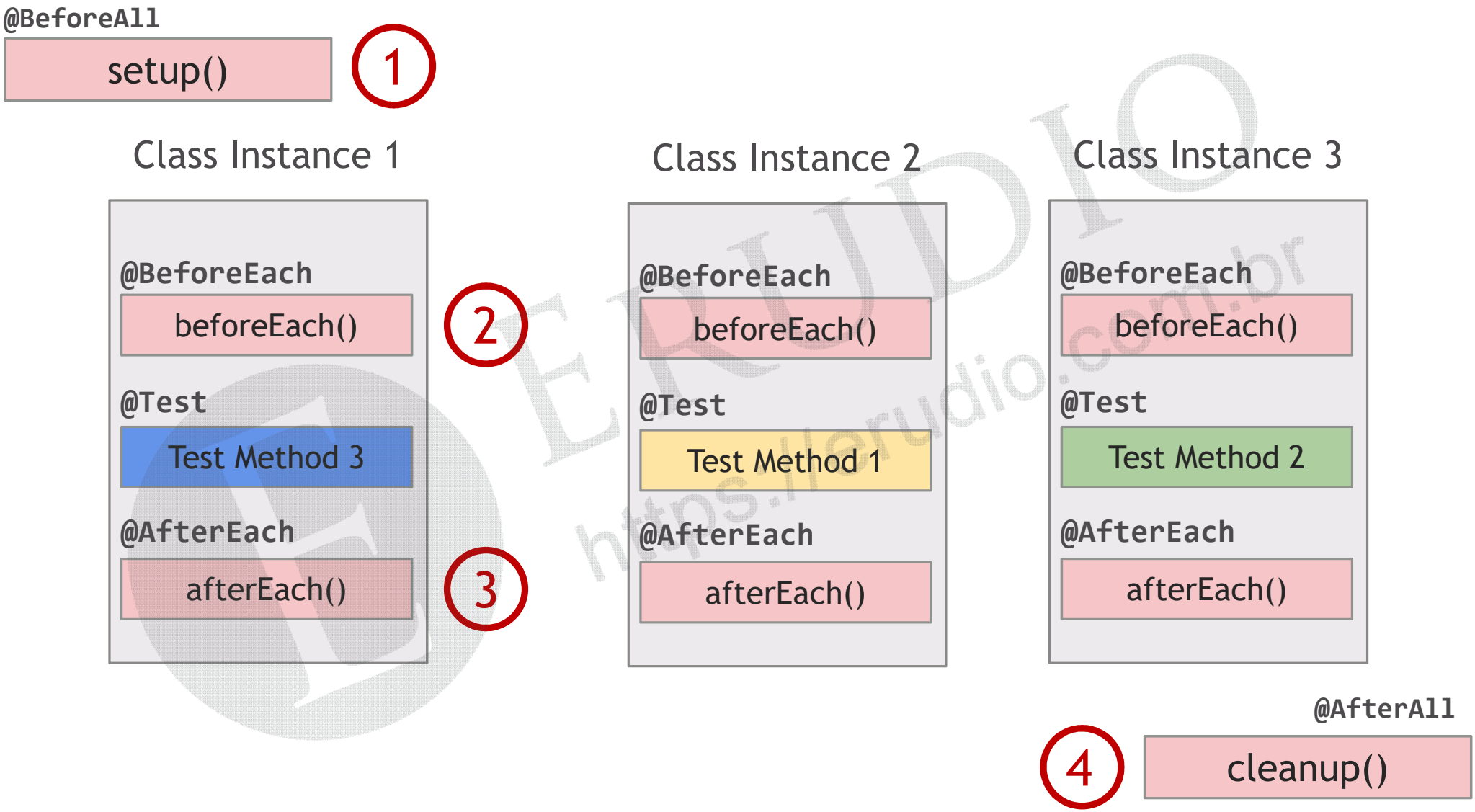
Ciclo de Vida dos Testes no JUnit 5



Ciclo de Vida dos Testes no JUnit 5

```
@Test
void testSum() {

    SimpleMath math = new SimpleMath();
    Double actual = math.sum(6.2D, 2D);
    assertEquals(8.2D, actual);
}
```




@BeforeAll
setup()
1
Class Instance 1
@BeforeEach
beforeEach()
2
@Test
Test Method 3
@AfterEach
afterEach()
3
Class Instance 2
@BeforeEach
beforeEach()
@Test
Test Method 1
@AfterEach
afterEach()
Class Instance 3
@BeforeEach
beforeEach()
@Test
Test Method 2
@AfterEach
afterEach()
@AfterAll
4
cleanup()

0411 Introdução ao Ciclo de Vida do Test Instance

Leandro Costa

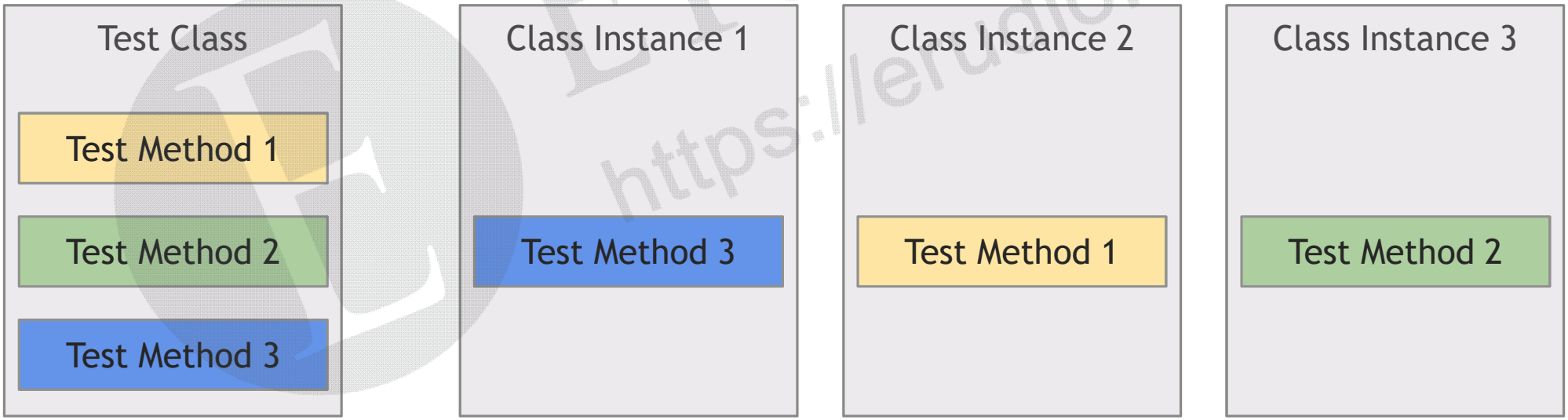
https://www.erudio.com.br/ https://www.youtube.com/c/ErudioTraining

https://www.semeru.com.br/ https://hub.docker.com/u/leandrocgsi/

https://github.com/leandrocgsi/automated-tests-with-java-erudio

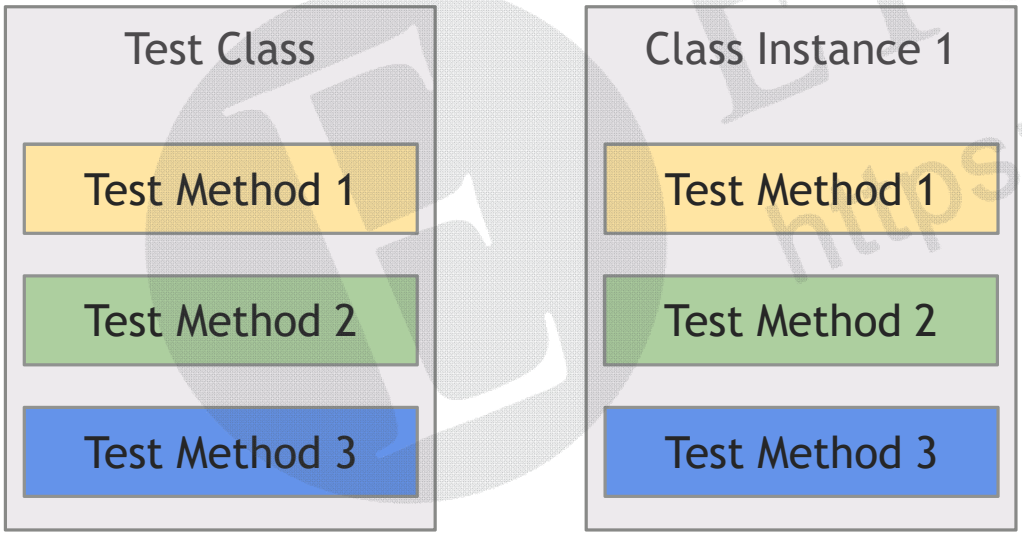
Ciclo de Vida dos Testes no JUnit 5

- Por padrão, o ciclo de vida da instância de testes é "por método“;
- A ordem de execução é determinística, mas intencionalmente não óbvia.



Ciclo de Vida dos Testes no JUnit 5

```
@TestInstance(Lifecycle.PER_CLASS)
```



0501 Apresentação da Seção - Test Driven Development (TDD)

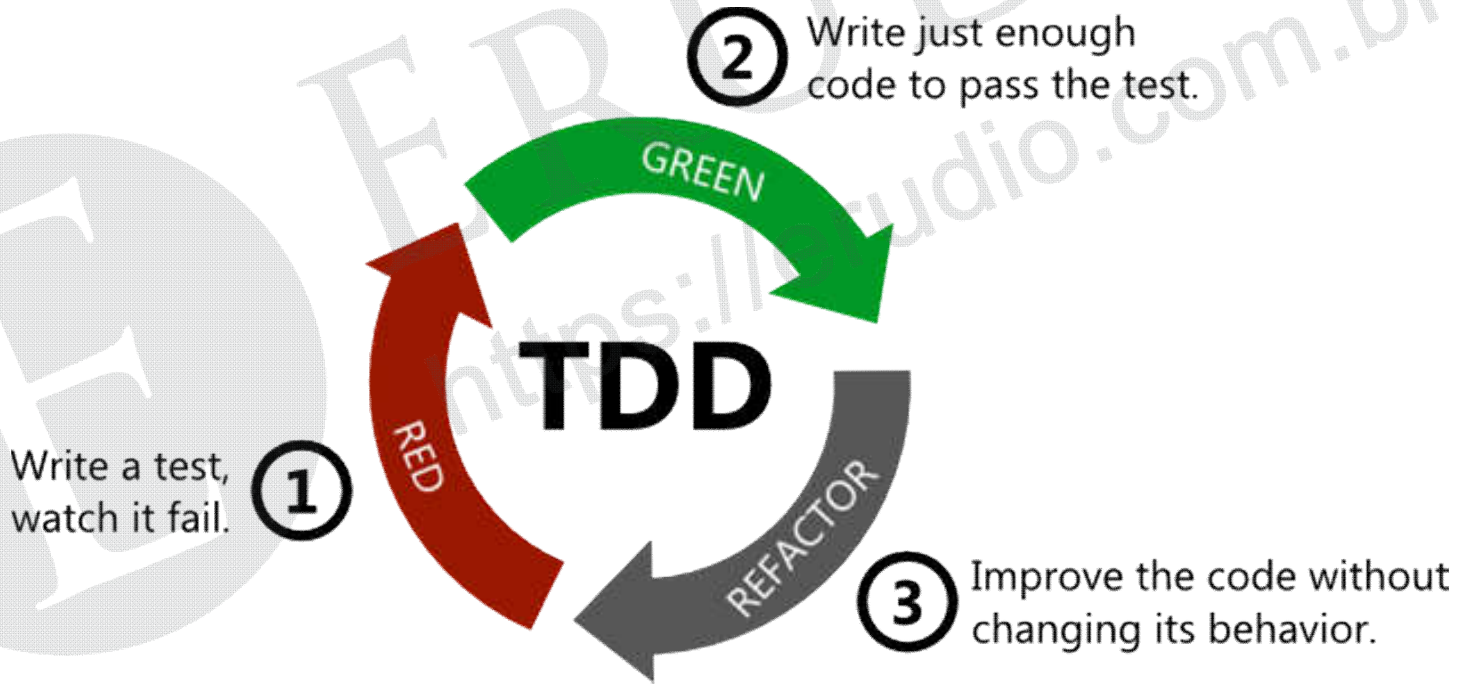
Leandro Costa

https://www.erudio.com.br/ https://www.youtube.com/c/ErudioTraining

https://www.semeru.com.br/ https://hub.docker.com/u/leandrocgsi/

https://github.com/leandrocgsi/automated-tests-with-java-erudio

Ciclo de Vida dos Testes no JUnit 5



1002 Conhecendo a Annotation @DataJpaTest do Spring Boot

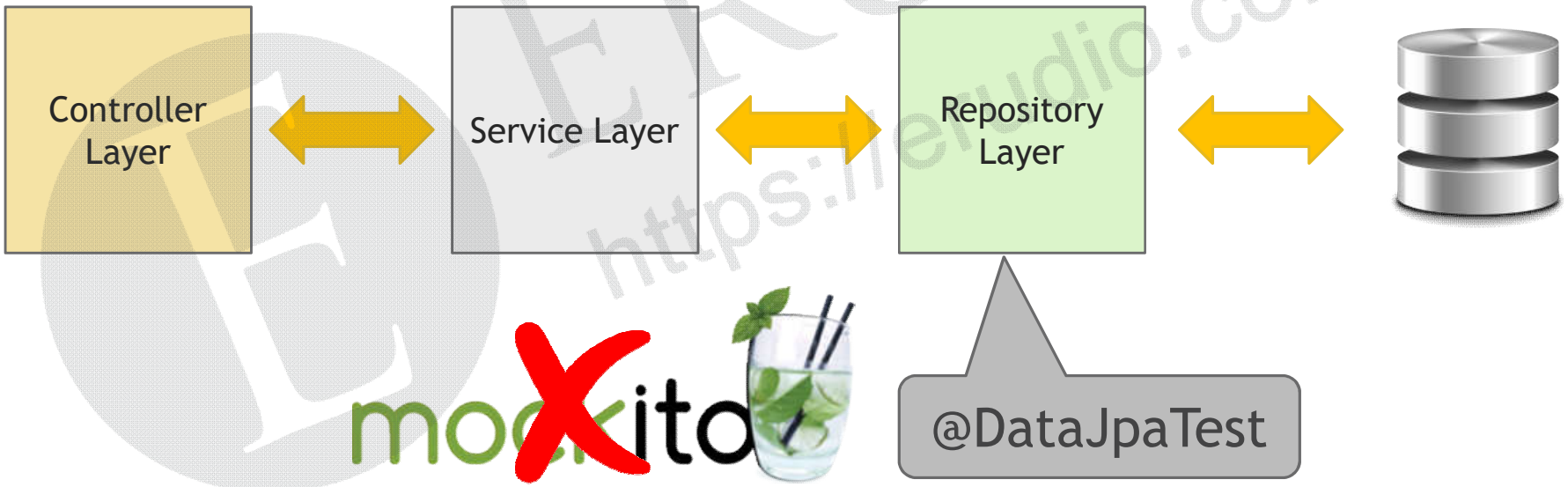
Leandro Costa

https://www.erudio.com.br/ https://www.youtube.com/c/ErudioTraining

https://www.semeru.com.br/ https://hub.docker.com/u/leandrocgsi/

https://github.com/leandrocgsi/automated-tests-with-java-erudio

Testando a Camada de Repositórios de uma Aplicação Spring Boot



@DataJpaTest annotation

- O Spring Boot fornece a anotação @DataJpaTest para testar os componentes da camada de persistência, que configurarão automaticamente um banco de dados embarcado em memória para fins de teste;
- A anotação @DataJpaTest não carrega outros beans do Spring (@Components, @Controller, @Service e beans anotados) no ApplicationContext;
- Por padrão, ele procura por classes @Entity e configura os repositórios do Spring Data JPA anotados com a anotação @Repository;
- Por padrão, os testes anotados com @DataJpaTest são transacionais e são revertidos ao final de cada teste;
- Para testar repositórios do Spring Data JPA ou qualquer outro componente relacionado ao JPA, o Spring Boot fornece a anotação @DataJpaTest.

1102 Overview e as Annotations @Mock e @InjectMocks do Mockito

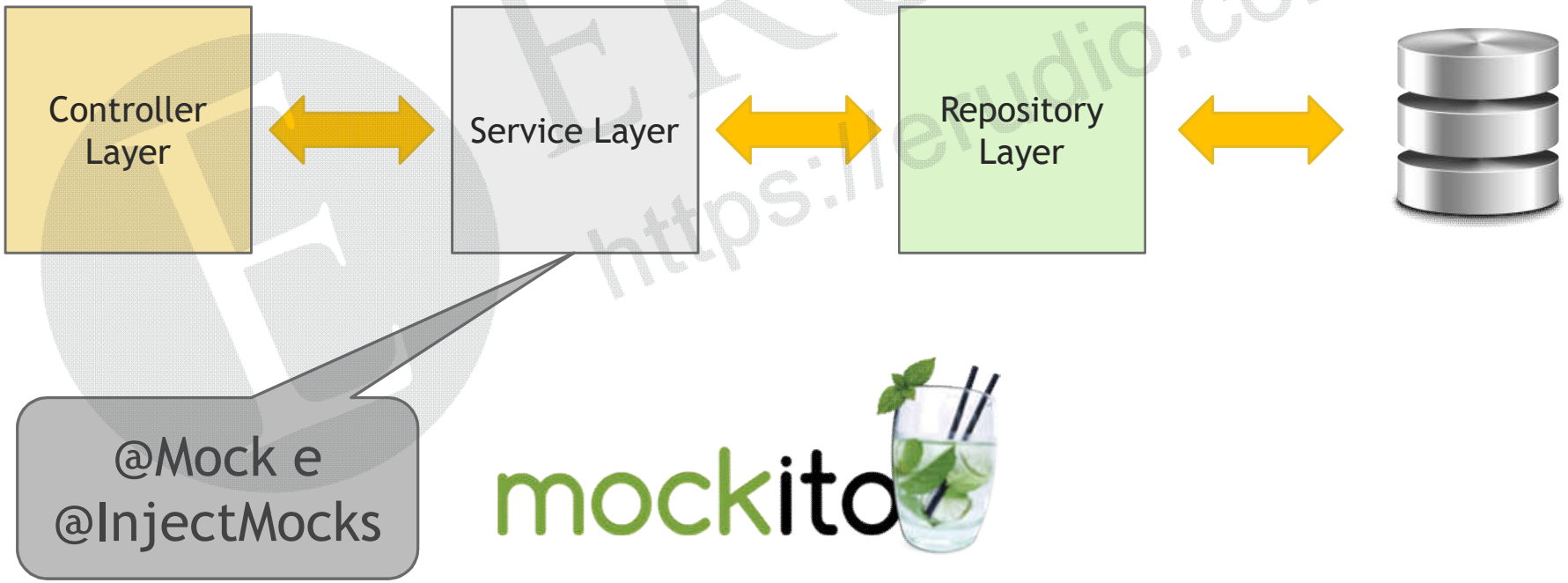
Leandro Costa

https://www.erudio.com.br/ https://www.youtube.com/c/ErudioTraining

https://www.semeru.com.br/ https://hub.docker.com/u/leandrocgsi/

https://github.com/leandrocgsi/automated-tests-with-java-erudio

Testando a Camada de Serviços de uma Aplicação Spring Boot



Mockito @InjectMocks Annotation

- Quando queremos injetar um objeto *mockado* em outro objeto *mockado*, podemos usar a *annotation* ***@InjectMocks***. A *annotation @InjectMocks* cria um *mock* da classe e injeta os *mocks* que estão marcados com as annotations @Mock nela.

```
@ExtendWith(MockitoExtension.class)
public class PersonServicesTest {

    @Mock
    private PersonRepository repository;

    @InjectMocks
    private PersonServices service;
```

1202 Overview do Teste de Controller e a Annotation @WebMvcTest

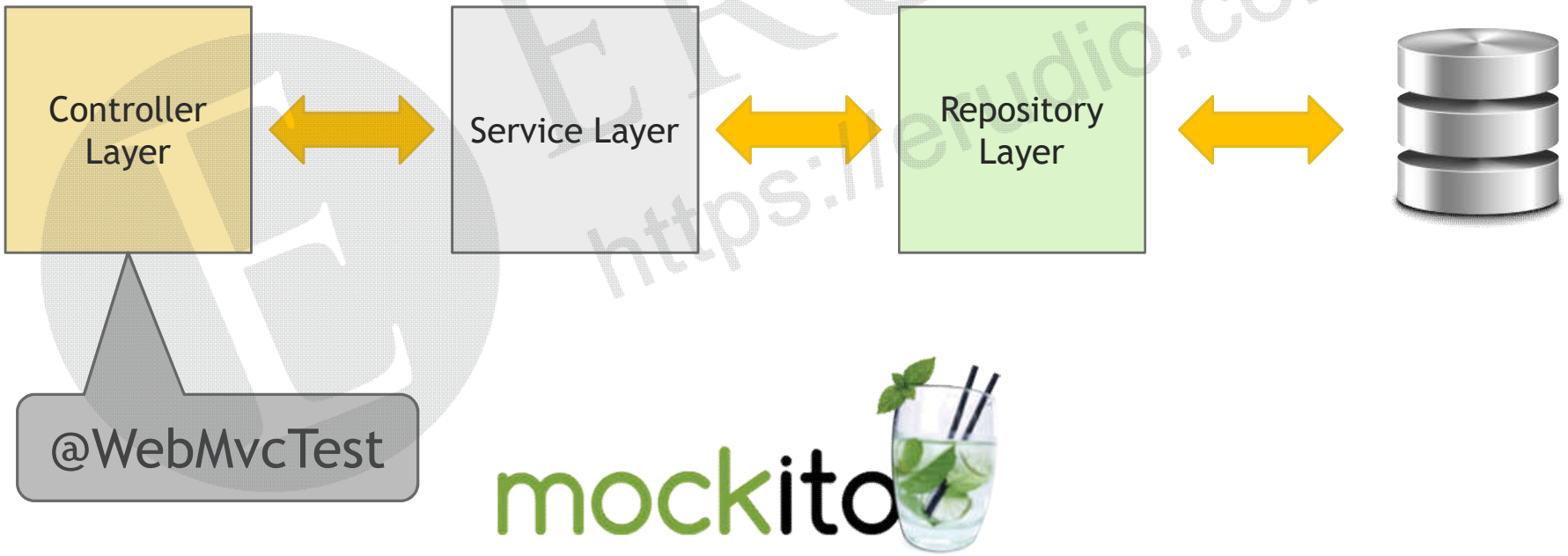
Leandro Costa

https://www.erudio.com.br/ https://www.youtube.com/c/ErudioTraining

https://www.semeru.com.br/ https://hub.docker.com/u/leandrocgsi/

https://github.com/leandrocgsi/automated-tests-with-java-erudio

Testando a Camada de Controllers de uma Aplicação Spring Boot



JsonPath Library

- Uma DSL Java para leitura de documentos JSON.
- As expressões JsonPath sempre se referem a uma estrutura JSON da mesma forma que as expressões XPath são usadas em combinação com um documento XML. O "objeto membro raiz" no JsonPath é sempre referido como $, independentemente de ser um objeto ou um array.

Json

```
{
  "firstName": "Leandro",
  "lastName": "Costa",
  "email": "leandro@erudio.com.br"
}
```

JsonPath Expressions

$ - root member de uma estrutura JSON, seja um objeto ou um array.

```
$.firstName = "Leandro",
$.lastName = "Costa",
$.email = "leandro@erudio.com.br"
```

1203 @WebMvcTest vs @SpringBootTest

Leandro Costa

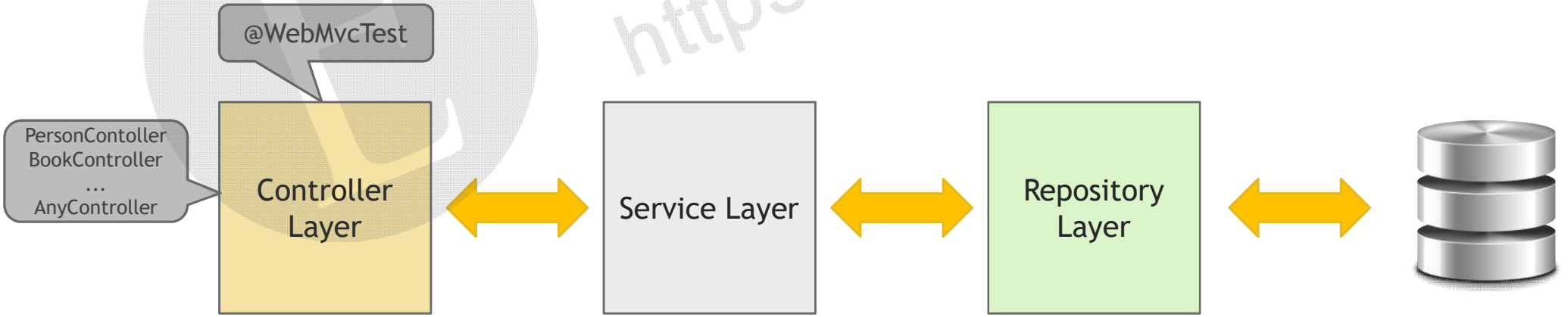
https://www.erudio.com.br/ https://www.youtube.com/c/ErudioTraining

https://www.semeru.com.br/ https://hub.docker.com/u/leandrocgsi/

https://github.com/leandrocgsi/automated-tests-with-java-erudio

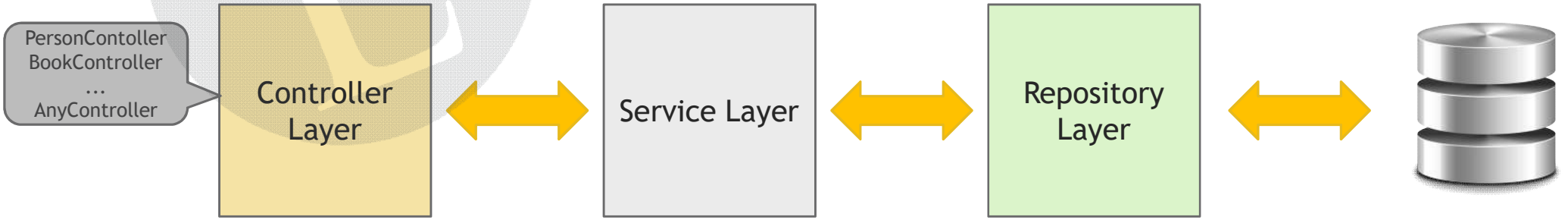
@WebMvcTest Annotation

- O Spring Boot fornece a anotação ***@WebMvcTest*** para testar controllers Spring MVC. Além disso, os testes baseados em ***@WebMvcTest*** são mais rápidos, pois carregam apenas o controller especificado e suas dependências, sem carregar a aplicação inteira;
- O Spring Boot instancia apenas a camada web em vez de todo o ***application context***. Em uma aplicação com vários controllers, você pode definir a instanciação de apenas um deles usando, por exemplo, ***@WebMvcTest(PersonController.class)***.



@WebMvcTest x @SpringBootTest

- O Spring Boot fornece a anotação @WebMvcTest para testar *controllers* do Spring MVC. Essa anotação cria um *application context* que contém todos os beans necessários para testar um controlador web do Spring;
- O Spring Boot fornece a anotação @SpringBootTest para testes de integração. Essa anotação carrega um *application context* completo;
- Teste de unidade - anotação @WebMvcTest;
- Teste de integração - anotação @SpringBootTest.



1302 Integration Testing e @SpringBootTest Annotation Overview

Leandro Costa

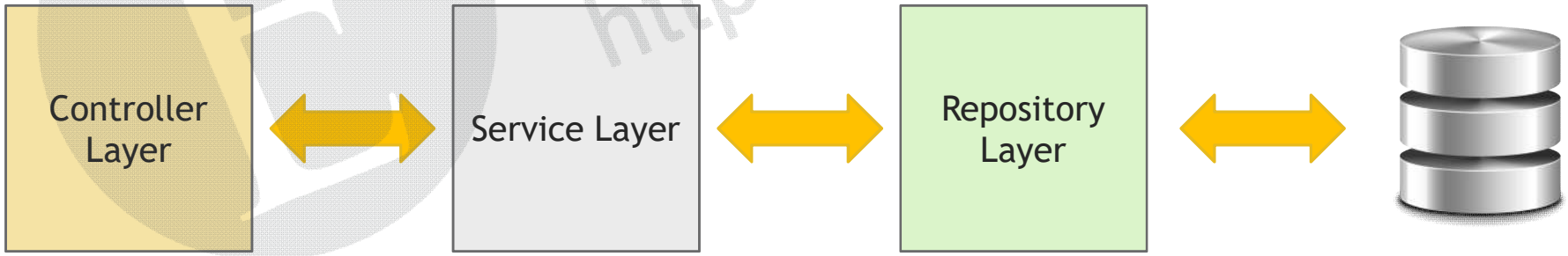
https://www.erudio.com.br/ https://www.youtube.com/c/ErudioTraining

https://www.semeru.com.br/ https://hub.docker.com/u/leandrocgsi/

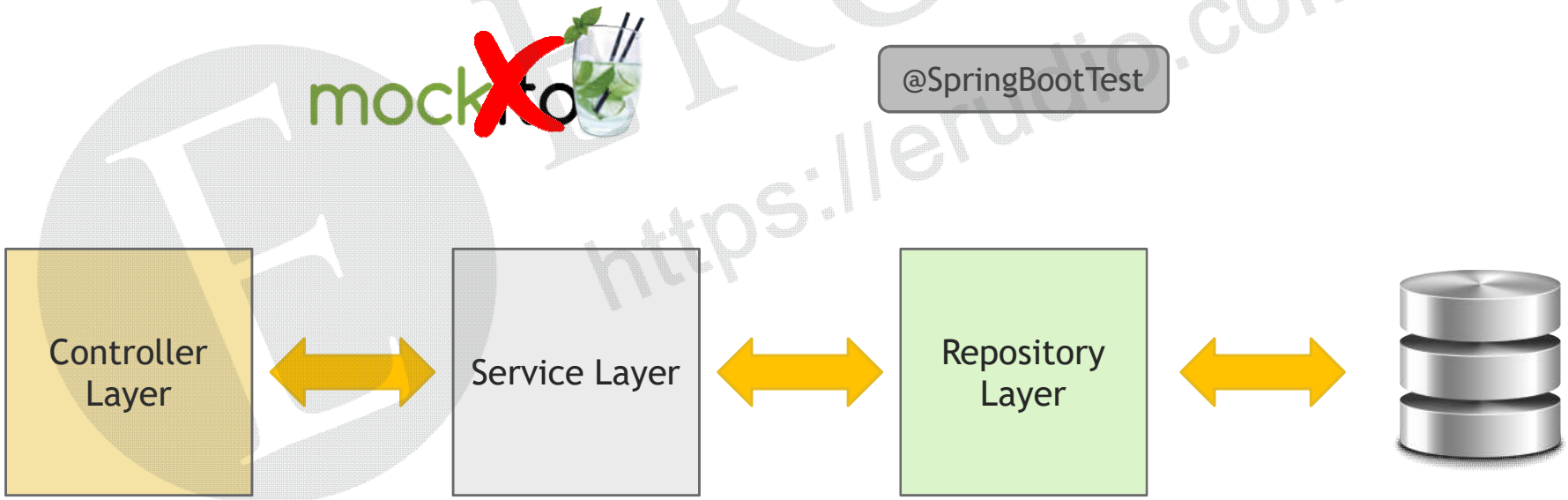
https://github.com/leandrocgsi/automated-tests-with-java-erudio

Testes de Integração em uma Aplicação Spring Boot

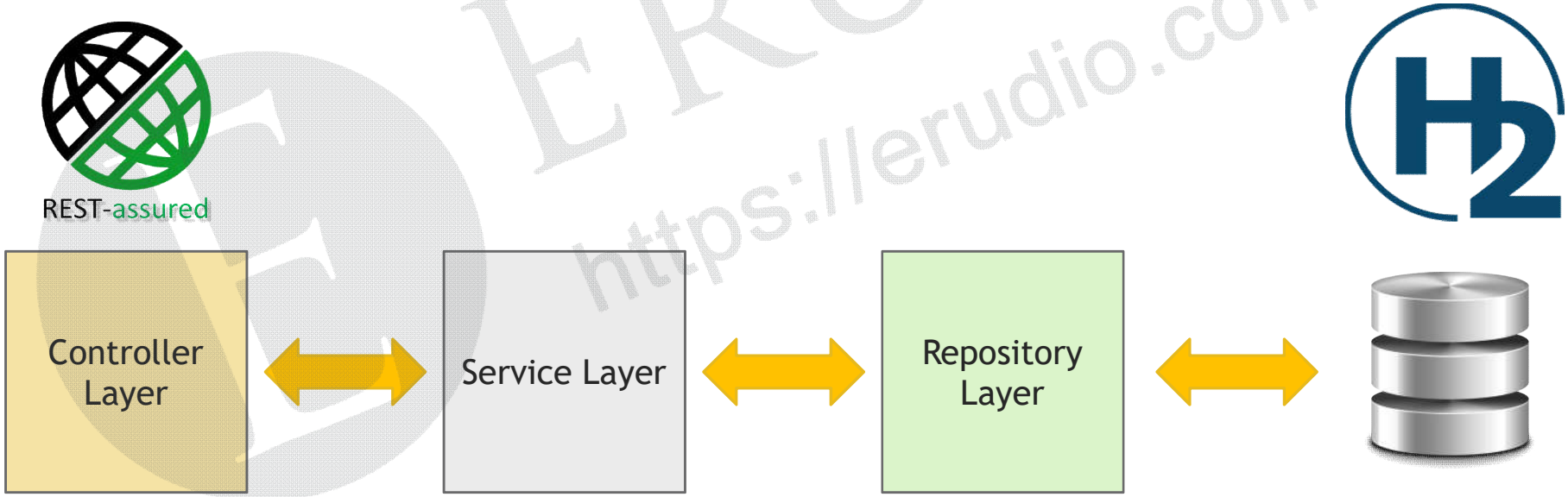
- Como o nome sugere, os testes de integração têm foco na integração de diferentes camadas da aplicação. Isso também significa que não há uso de mocks;
- Basicamente, escrevemos testes de integração para testar um *features* que podem envolver interação com múltiplos componentes.



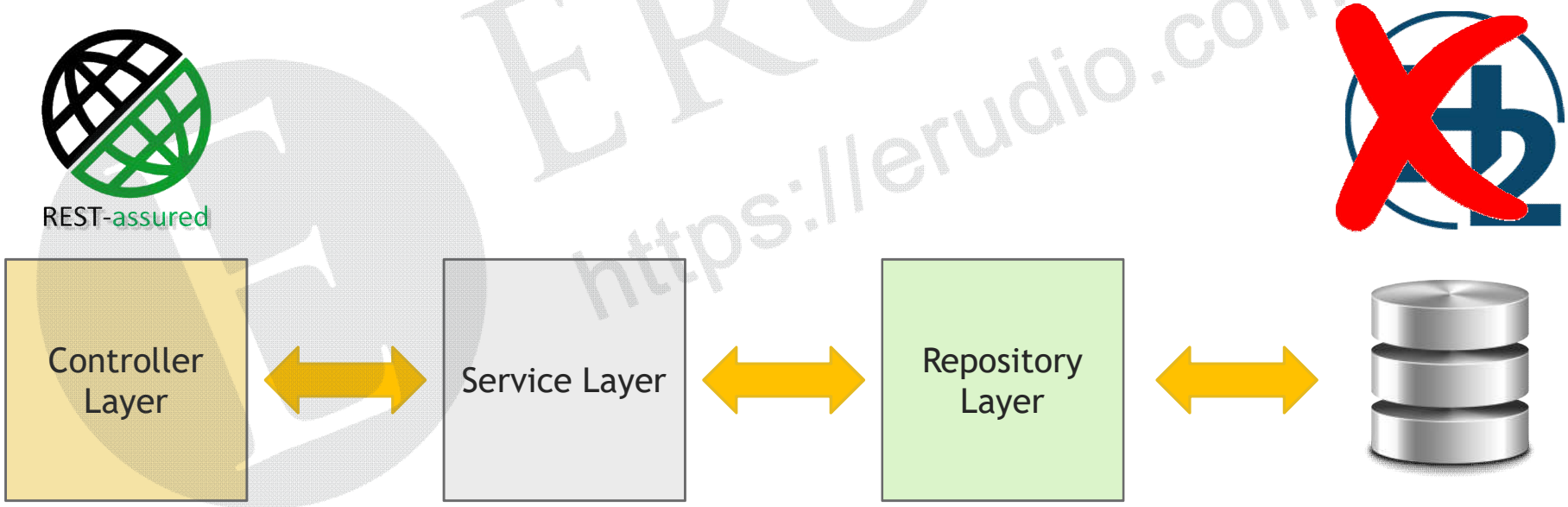
Testes de Integração em uma Aplicação Spring Boot



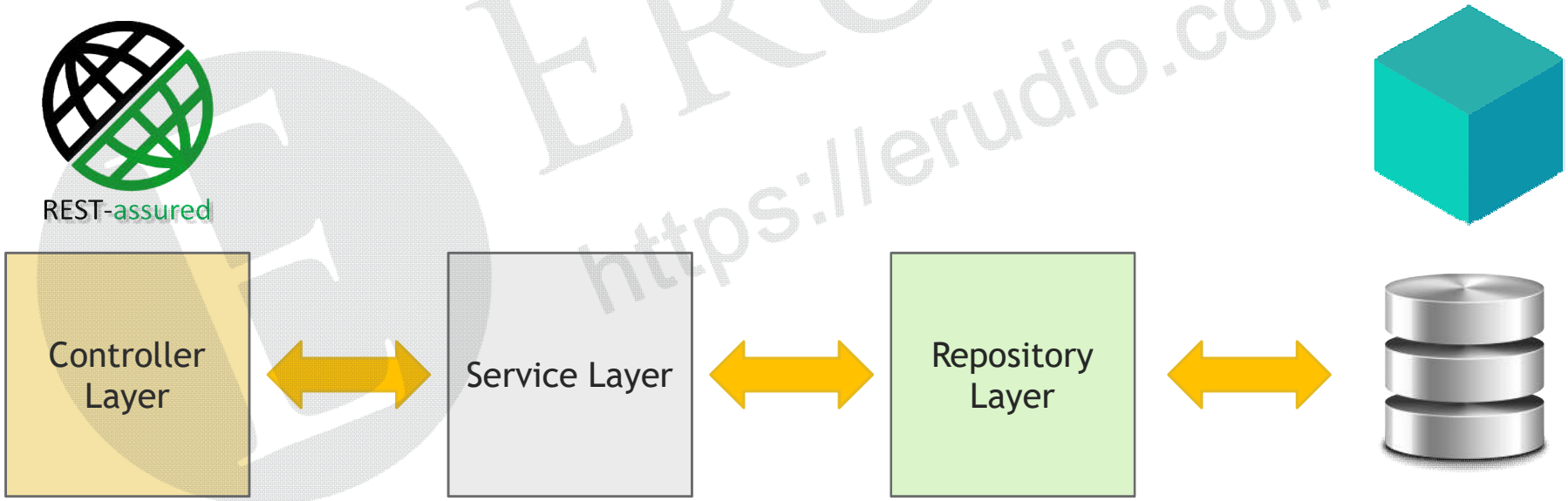
Testes de Integração em uma Aplicação Spring Boot



Testes de Integração em uma Aplicação Spring Boot



Testes de Integração em uma Aplicação Spring Boot

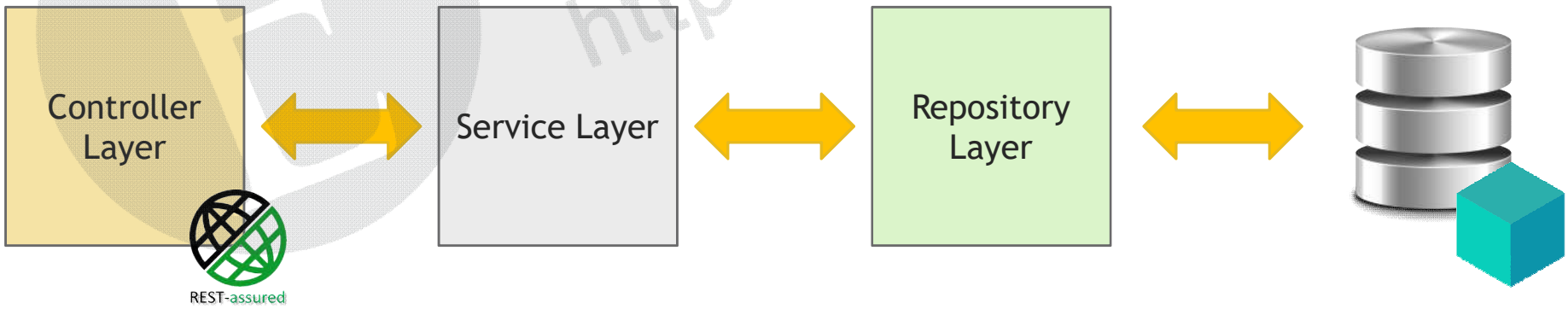


@SpringBootTest

- O Spring Boot fornece a anotação ***@SpringBootTest*** para testes de integração. Essa anotação cria um *application context* completo.

- @SpringBootTest inicializará o *application context* completo, o que significa que podemos usar a anotação @Autowire para injetar qualquer bean detectado pelo *component scan* em nosso teste.

Testes de integração - @SpringBootTest



@SpringBootTest

- Ele inicializa um servidor embarcado, cria um *web environment* e possibilita que os métodos **@Test** façam testes de integração.
- Por padrão, **@SpringBootTest** não inicia um servidor. Precisamos adicionar o atributo **WebEnvironment** para refinar ainda mais como nossos testes serão executados. Existem várias opções:
  - **MOCK** (Padrão): Carrega um *WebServerApplicationContext* e fornece um *web environment* mockado;
  - **RANDOM_PORT**: Carrega um *WebServerApplicationContext* e fornece um *web environment* real. O servidor embarcado é iniciado e exposto em uma porta aleatória. Essa opção deve ser usada para testes de integração;
  - **DEFINED_PORT**: Carrega um *WebServerApplicationContext* e fornece um *web environment* real;
  - **NONE**: Carrega um ***ApplicationContext*** usando o ***SpringApplication***, mas não fornece nenhum web environment.