Automation Test

Ngọc Bản Quyền

Junit 5

Tại sao phải cần kiểm thử?

- Giảm thiểu lỗi, chi phí phải sửa chữa, khôi phục, bảo hành khi đưa sản phẩm ra thị trường
- Lập trình kiểm thử tự động giảm áp lực cho nhân sự kiểm thử thủ công. Có tính tích luỹ theo thời gian.
- Lập trình kiểm thử cũng là một cách để văn bản hoá logic.
- Buộc lập trình viên phải cải tiến code để có thể kiểm thử được.

Junit là gì?

JUnit là một Java testing framework được sử dụng phổ biến trong các dự án Java.

JUnit 5 là phiên bản mới nhất của JUnit, nó có một số cải tiến thú vị, với mục tiêu hỗ trợ các tính năng mới từ phiên bản Java 8 trở đi cũng như cho phép nhiều kiểu kiểm thử khác nhau.

Cài đặt JUnit 5 trong dự án Maven

Chúng ta cần thêm dependency của Junit vào file pom.xml

```
<dependency>
    <groupId>org.junit.jupiter</groupId>
    <artifactId>junit-jupiter-api</artifactId>
    <version>5.8.0-M1</version>
    <scope>test</scope>
</dependency>
```

Cấu tạo của Junit 5

JUnit 5 = Junit Platform + JUnit Jupiter + JUnit Vintage

JUnit Platform đóng vai trò là nền tảng khởi chạy các framework kiểm thử trên JVM

JUnit Jupiter là sự kết hợp của mô hình lập trình mới và mô hình mở rộng để viết testcase và phần mở rộng trong JUnit 5

JUnit Vintage cung cấp TestEngine để chạy cac kiểm thử dựa trên JUnit 3 và JUnit 4

Annotations

@BeforeAll: Chạy một lần trước tất cả bài test

@BeforeEach: Chạy trước mỗi bài test

@AfterEach: Chạy sau mỗi bài test

@AfterAll: Sau tất cả các bài test, chạy một lần để dọn dẹp

@ParameterizedTest: Truyền tham số vào test

@RepeatTest: lặp lại test N lần

@TestFactory: lập trình động test theo dữ liệu đầu vào

```java
//@BeforeAll dùng để chỉ định test method chạy đầu tiên
//Nó phải được đặt là phương thức tĩnh (static), nếu không chương trình sẽ không biên dịch được
@BeforeAll
static void setup() {
    System.out.println("BeforeAll");
}

//@BeforeEach chỉ định 1 method sẽ luôn được thực thi trước mỗi test method thực thi
@BeforeEach
public void beforeEach() {
    System.out.println("BeforeEach");
}
```

```java
//@AfterAll Chỉ định method sẽ được thực thi khi tất cả các test method trong class thực thi xong
//Nó phải được đặt trên static method
@AfterAll
public static void afterAll() {
    System.out.println("AfterAll");
}

//@AfterEach Chỉ định 1 method luôn thực thi sau khi 1 test method thực thi xong
@AfterEach
public void afterEach() {
    System.out.println("AfterEach");
}
```

```java
@Test
@DisplayName("My test method")
public void test() {
    System.out.println("Test");
}

//disable() method không được thực thi vì bị ngắn chặn bởi @Disabled
@Test
@Disabled
public void disable() {
    System.out.println("disable");
}
```

Kết quả thực thi:

Assertions

Assertion là gì?

Assertions chính là những method dùng để kiểm tra kết quả của đơn vị cần test có đúng với mong đợi không.

Một số assertions phổ biến

assertEquals(): dùng để xác minh giá trị mong đợi và giá trị thực tế bằng nhau

```
public int sum(int a, int b){
    return a+b;
}
```

```
@Test
void assertEqualsExample() {
    assertEquals(calculator.sum(1, 1), 2);

}
```

assertNotEquals() :dùng để xác minh giá trị mong đợi và giá trị thực tế không bằng nhau

```
@Test
void assertNotEqualsExample(){
    assertNotEquals(3,calculator.sum(1,1));
}
```

assertArrayEquals(): được áp dụng đối với mảng, nó khẳng định rằng mảng mong đợi và mảng thực tế là bằng nhau

```
@Test
void assertArrayEqualsExample(){
    assertArrayEquals(new int[]{1,2,3}, new int[]{1,2,3});
}
```

assertNull(): khẳng định rằng một object là null

assertNotnull(): khẳng định rằng object là not null

```
@Test
void assertNull_assertNotNull(){
    String nullString = null;
    String notNullString = "Techmaster";

    assertNull(nullString);
    assertNotNull(notNullString);
}
```

assertTrue(): dùng để xác minh điều kiện phải trả về *true.*

assetFalse(): dùng để xác minh điều kiện trả về là *false*

```
@Test
void assertTrue_assertFalse(){
    //Test will pass
    assertTrue(true);
    assertFalse(false);
    assertTrue(5 > 4, "5 is greater the 4");
    assertTrue(null == null, "null is equal to null");
    assertFalse(4 > 5, "5 is greater the 4" );
}
```

assertSame(): khẳng định rằng 2 object có cùng tham chiếu tới chính xác cùng một object

assertNotSame() : khẳng định rằng 2 object không tham chiếu đến cùng một đối tượng

```
@Test
    void assertSame_assertNotSame(){
        String originalObject = "Techmaster";
        String cloneObject = originalObject;
        String otherObject = "JUnit 5";
        String newObject = new String(originalObject);

        assertNotSame(originalObject, newObject);
        assertSame(originalObject, cloneObject);
        assertNotSame(originalObject, otherObject);
    }
```

AssertJ

AssertJ

- AssertJ là một thư viện để đơn giản hóa việc viết các assertions. Nó cũng cải thiện khả năng đọc các câu lệnh assertion.
- AssertJ có fluent interface cho assertions, giúp bạn dễ dàng viết code.
- Method cơ sở cho các AssertJ là assertThat.

Cài đặt assert trong dự án Maven

Để sử dụng AssertJ, bạn cần thêm dependency của JUnit và file pom.xml

```
<!-- https://mvnrepository.com/artifact/org.assertj/assertj-core -->
<dependency>
    <groupId>org.assertj</groupId>
    <artifactId>assertj-core</artifactId>
    <version>3.19.0</version>
    <scope>test</scope>
</dependency>
```

Sử dụng assertJ

Để viết assertions, bạn luôn cần bắt đầu bằng cách chuyển object của mình tới phương thức assertThat() và thực hiện theo các assertions thực tế.

```
assertThat(anyReferenceOrValue);
```

Để bắt đầu ta cần import:

```
import static org.assertj.core.api.Java6Assertions.assertThat;
```

Array Assertions

Đối với mảng, có nhiều cách để khẳng định rằng nội dung của chúng tồn tại. Một trong những assertions phổ biến nhất là kiểm tra xem mảng có chứa phần tử đã cho hay không

```
@Test
    @DisplayName("TestArray")
    void testMethod_Arrays(){
        String [] countries = new String[]{"Russia", "Viet Nam", "America", "Japan", "China"};

        assertThat(countries).isNotEmpty() //Array is not empty
                .startsWith("Russia")      //Array start with "Russia
                .contains("Viet Nam")      //contains "Viet Nam" element
                .doesNotContainNull()      //does not contains any nulls
                .containsSequence("America", "Japan")  //contains sequence of element "America", "Japan"
                .hasSize(5)                //Array length = 5
                .endsWith("China");        //Array end with "China"
    }
```

Strings Assertions: có nhiều assertions phổ biến đối với chuỗi, hãy xem ví dụ sau:

```
@Test
@DisplayName("TestStrings")
void testMethod_Strings(){
    String say = "Chị không muốn nhiêu bug nhưng mà bug nhiều nên chị phải fix";
    String sayClone = say;

    assertThat(say).isNotNull()     //String is not empty
            .startsWith("Chị")      //Start with "Chị
            .doesNotContain("Anh")  //Not contain "Anh"
            .endsWith("fix")        //End with "fix"
            .contains("bug", "Chị");

    assertThat(say).isEqualTo(sayClone);
}
```

Numbers Assertions: nhằm so sánh các giá trị số trong hoặc không có khoảng chênh lệch nhất định. Ví dụ, nếu bạn muốn kiểm tra xem hai giá trị có bằng nhau theo một độ chính xác nhất định hay không, chúng ta có thể thực hiện như sau:

```
@Test
@DisplayName("TestNumbers")
void testMethod_Numbers(){
    Double value = 12.0;
    Double value1 = 10.0;
    assertThat(value).isEqualTo(12.2, withPrecision(0.2d))
                     .isCloseTo(15.0, Offset.offset(4d))
                     .isBetween(10.0, 15.0) //value >= 10.0 and value <= 15.0
                     .isStrictlyBetween(10.0, 15.0) //value > 10.0 and value < 15.0
                     .isNotZero();
}
```