**JUnit**

Tháng Tám 19, 2010[Để lại bình luận](https://nmhblog.wordpress.com/2010/08/19/unit-test/#respond)[Go to comments](https://nmhblog.wordpress.com/2010/08/19/unit-test/#comments)

Có lẽ đa số sinh viên chúng ta đều quen với việc hoàn tất chương trình, chạy thử, thấy sai, tìm và sửa lỗi, chạy thử, v.v. Điều này không có gì là sai trái, tuy nhiên nếu đợi đến khi hoàn tất chương trình rồi mới chạy thử, thì việc tìm và sửa lỗi sẽ tương đối vất vả. Quá trình này sẽ đơn giản hơn, nếu bạn có thể chia chương trình thành từng phần nhỏ, và test chúng một cách riêng rẽ. Đó là ý nghĩa của unit test.

Tuy nhiên, để unit test hiệu quả, chúng ta cần có các công cụ để tiến hành nó một cách tự động. Các công cụ này gọi là unit test framework. Với Java, framework nổi tiếng nhất là JUnit.

Nếu bạn muốn tìm hiểu về JUnit, thì bạn nên đọc “JUnit in action”. Dưới đây là một ví dụ nhỏ về JUnit để bạn tham khảo.

**Kiểm tra với một test case**

Giả sử chúng ta có một class Number như dưới đây. Mỗi object của class đại diện cho một số nguyên. Method isEven kiểm tra tính chẵn lẻ của số nguyên đó. Mục tiêu của chúng ta là kiểm tra tính đúng đắn của method isEvent.

|  |  |
| --- | --- |
| 1  2  3  4  5 | public class Number {      private int number;      public Number(int number) { this.number = number; }      public boolean isEven() { return (number % 2 == 0)?true:false; }  } |

Việc đầu tiên là tải tập tin [junit-4.x.x.jar](http://www.junit.org/) và cho nó vào CLASSPATH. Đoạn mã dưới đây sẽ kiểm tra isEvent với 2.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | import static org.junit.Assert.\*;  import org.junit.Test;  public class NumberTest {      @Test public void testIsEven() {          Number number = new Number(2);          assertTrue(number.isEven());      }  } |

Dịch hai đoạn mã trên và chạy lệnh java org.junit.runner.JUnitCore NumberTest. Khi đó JUnit sẽ chạy các method được đánh dấu @Test. Kết quả là:

|  |  |
| --- | --- |
| 1  2  3  4 | JUnit version 4.8.2  .  Time: 0.008  OK (1 test) |

**Kiểm tra với nhiều test case**

Bây giờ ta muốn kiếm tra với nhiều số hơn. Chúng ta sẽ không chèn thêm assertTrue vào testIsEven (vì như vậy sẽ không thể biết được số nào làm cho test bị sai). Thay vào đó, chúng ta sẽ tham số hóa method testIsEven. Các tham số được lấy từ bộ test tạo ra bởi method getTestParameters.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24 | import static org.junit.Assert.\*;  import org.junit.Test;  import org.junit.runner.RunWith;  import org.junit.runners.Parameterized;  import org.junit.runners.Parameterized.Parameters;  import java.util.Arrays;  import java.util.Collection;    @RunWith(value=Parameterized.class)  public class NumberParameterizedTest {      private int value;      private boolean expected;      public NumberParameterizedTest(int value, boolean expected) {          this.value = value;          this.expected = expected;      }      @Parameters public static Collection<Object[]>getTestParameters() {          return Arrays.asList(new Object[][] { {2, true}, {3, false}, {5, true}, });      }      @Test public void testIsEven() {          Number number = new Number(value);          assertEquals(number.isEven(), expected);      }  } |

Lần này, Junit sẽ lần lượt lấy từng mẫu ra để kiểm tra. Kết quả là (trích):

|  |  |
| --- | --- |
| 1  2  3  4  5 | JUnit version 4.8.2  ...E  Time: 0.022  There was 1 failure:  1) testIsEven[2](NumberParameterizedTest)... |

Như vậy là test sai với mẫu thứ ba (5, true) (vì đếm từ 0).

**Kiểm tra Exception**

Trong một số trường hợp, một method sẽ throw Exception thay vì trả về kết quả tính toán. Để đảm bảo hành vi của method đúng như mong đợi, chúng ta phải test cả những trường hợp này để xem chúng có throw đúng Exception cần throw không. @Test(expected=YourException.class) là annotation phục vụ cho mục đích này.

**Dùng test làm hướng dẫn sử dụng**

Hãy nhìn lại đoạn code trên. Khi đọc test (nếu bạn hiểu JUnit), bạn sẽ suy ra ngay cách sử dụng method isEven: đầu tiên tạo một object Number với constructor nhận vào giá trị int, sau đó gọi isEven(). Vậy là, test code còn một tác dụng khác là làm hướng dẫn sử dụng (tương tự như các đoạn code trong tutorial trên mạng). Ưu điểm của việc lấy test làm hướng dẫn sử dụng là nhanh gọn và bám sát theo thay đổi của chương trình.