Actividad:
Uso de JUnit 5 y RSpec

Comparaciones:

1

Lenguaje de Programación:

JUnit5: Diseñado en Java
RSpec: Específicamente diseñado para Ruby.
Obs: La preferencia dependerá del lenguaje de programación que prefieras.

2

Sintaxis:

JUnit5: Usa una estructura convencional de Java utilizando anotaciones.
RSpec: Tiene una sintaxis más natural y también más legible (es decir alguien sin experiencia programando puede entender fácilmente el código)
Obs: Si eres una persona a la cual se le complica entender la estructura de Java, Ruby podría ser una buena opción para hacer testing framework.

3

Estructura de Pruebas:

JUnit5: Su estructura se basa en las anotaciones (como @Test, @AfterEach, @Nested).
RSpec: Permite organizar el código de forma jerárquica y detallada a través de anidaciones de contextos y especificaciones.

Obs: JUnit5 te permite poner anotaciones ya prestablecidas, mientras que RSpec te permite poner la etiqueta que mejor te guste, permitiendo ser más flexible, lo mismo sucede con las especificaciones.

4

Comunidad:

JUnit5: Concentrados en el desarrollo de creación de anotaciones.
RSpec: Concentrados en el desarrollo de la escritura para pruebas que sean legibles y expresivas.
Obs: Ambos tienen una comunidad fuerte, al igual que ambos son proyectos de código abierto, por lo que la misma comunidad ayuda a desarrollar y mantener activas estos marcos de pruebas.

5

Herramientas de Integración:

JUnit5: Al ser parte del ecosistema de Java se integra facilmente con otras herramientas desarrolladas en Java como Maven y Gradle, esto facilita la funcionalidad, configuración y ejecución de pruebas en proyectos de Java.
RSpec: Al ser parte del ecosistema Ruby, se utiliza normalmente en Ruby on Rails, con herramientas de administración de gemas de Ruby como Bundler, este facilita la gestión de dependencias; también Capybara te permite configurar o testear en proyectos webs.

> Obs: Siempre va a depender en que ecosistemas quieras desarrollar tu proyecto.

Convirtiendo de JUnit 5 a RSpec

En JUnit5:

```
@Test
@DisplayName("test executes only on Saturday")
public void testAssumeTrueSaturday() {
LocalDateTime dt = LocalDateTime.now();
 assumeTrue(dt.getDayOfWeek().getValue() == 6);
System.out.println("further code will execute only if above assumption holds true");
```

En RSpec:

```
It “Ejecuta solo si es Sábado” do
        dt = Time.now
        skip(“Saltar si no es sábado”) unless dt.saturday?

        expect(some_condition).to be truthy
```

Convirtiendo de JUnit 5 a RSpec

En RSpec:

```
class StringCalculator
        def self.add(input)
                if input.empty?
                0
                else numbers = input.split(",").map { |num| num.to_i }
                numbers.inject(0) { |sum, number| sum + number }
                end
        end
end

$ bundle exec rspec --format documentation
```

```
StringCalculator
        .add
                given an empty string
                        returns zero
                single numbers
                        given '4'
                                returns 4
                        given '10'
                                returns 10
                        two numbers given '2,4'
                                returns 6
                        given '17,100'
                                returns 117
```

En JUnit5:

```
public class StringCalculator {

    public static int add(String input) {
        if (input.isEmpty()) {
            return 0;
        } else {
            String[] numbers = input.split(",");
            int sum = 0;
            for (String number : numbers) {
                sum += Integer.parseInt(number);
            }
            return sum;
        }
    }
}

import org.junit.jupiter.api.Test;
import static org.junit.jupiter.api.Assertions.*;

public class StringCalculatorTest {

    @Test
    public void testAddEmptyString() {
        int result = StringCalculator.add("");
```

```
        assertEquals(0, result);
    }

    @Test
    public void testAddSingleNumber() {
        int result = StringCalculator.add("4");
        assertEquals(4, result);
    }

    @Test
    public void testAddTwoNumbers() {
        int result = StringCalculator.add("2,4");
        assertEquals(6, result);
    }

    @Test
    public void testAddMultipleNumbers() {
        int result = StringCalculator.add("1,2,3,4,5");
        assertEquals(15, result);
    }
}
```