

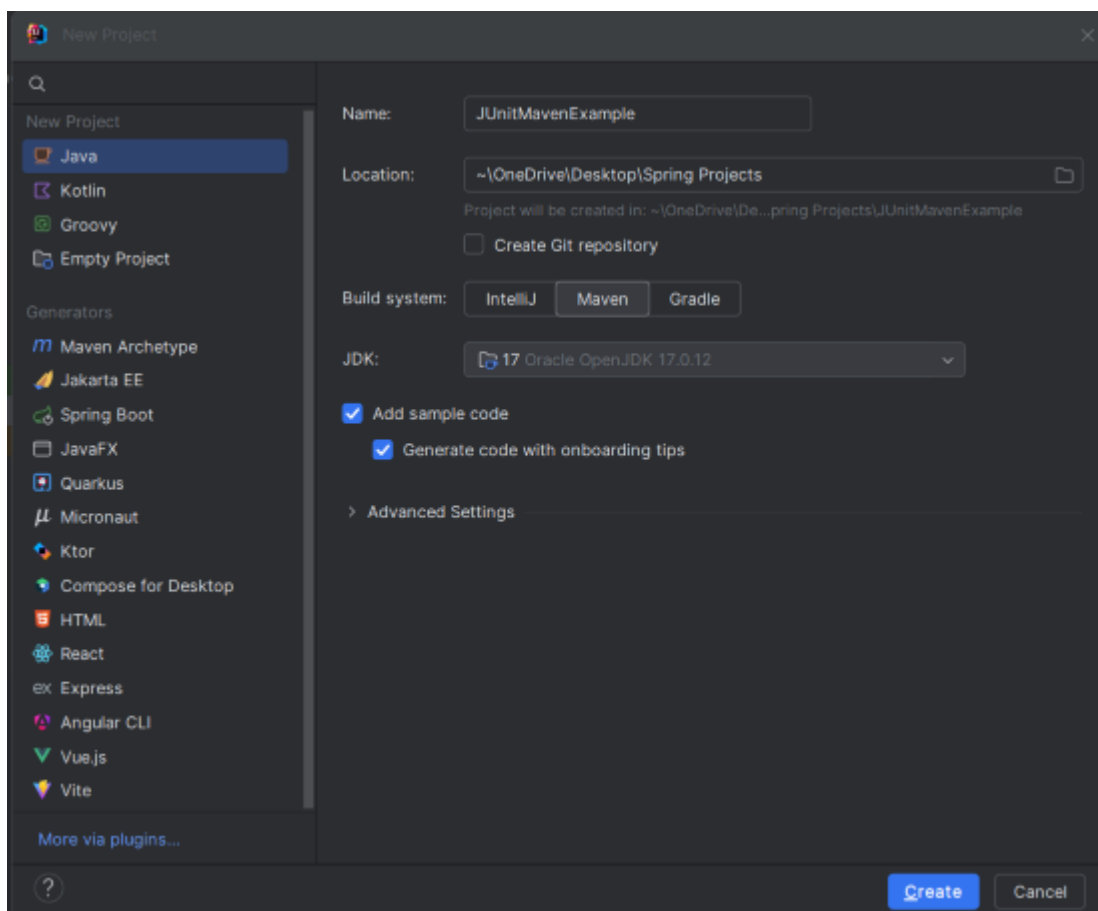
Pruebas JUnit con Maven

Este proyecto de ejemplo demuestra cómo configurar pruebas JUnit con Maven. El proyecto incluye una clase Java simple (Calculator) con métodos para sumar y restar números, junto con la clase de prueba JUnit correspondiente (CalculatorTest) para validar estos métodos.

Paso 1: Crear un nuevo proyecto Maven

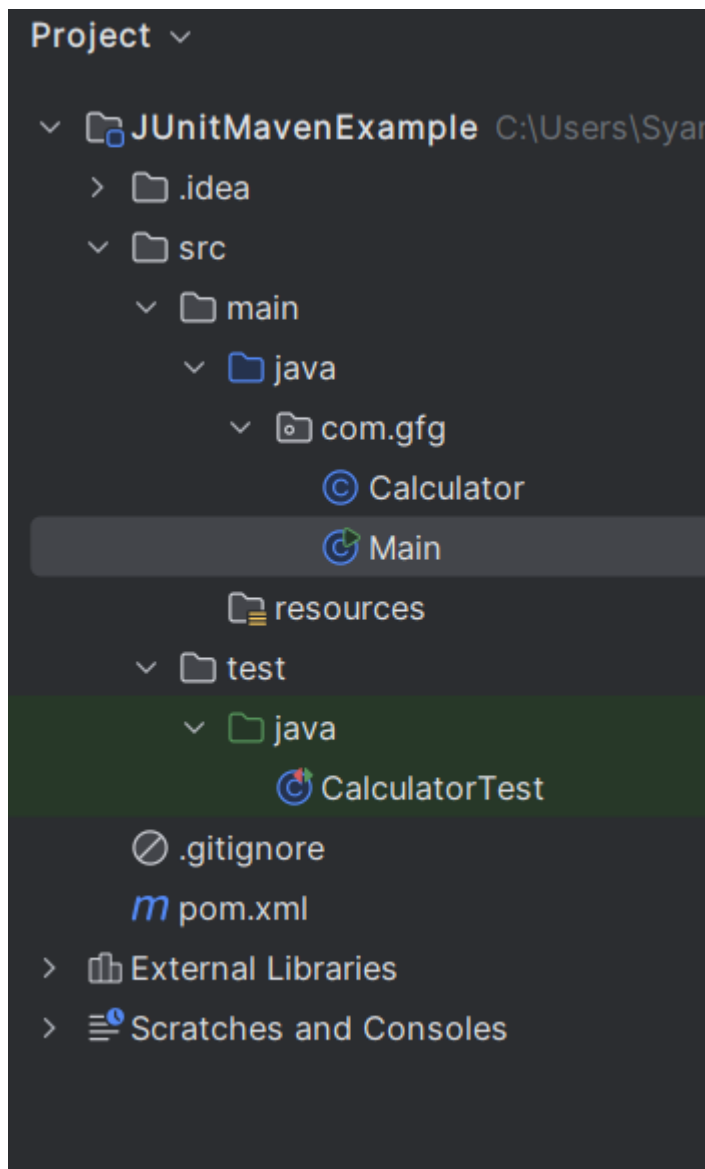
Cree un nuevo proyecto Maven con IntelliJ IDEA. Elija las siguientes opciones:

- **Nombre** :JUnitMavenExample
- **Sistema de compilación** : Maven
- Haga clic en el botón **Crear** .



Estructura del proyecto

Una vez realizada con éxito la creación del proyecto, la estructura de la carpeta se verá como la siguiente imagen:



Paso 2: Agregar dependencias apom.xml

Abra el pom.xml archivo y agregue la dependencia JUnit, que es necesaria para ejecutar las pruebas JUnit.

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<project xmlns="https://maven.apache.org/POM/4.0.0"
```

```
    xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
```

```
    xsi:schemaLocation="https://maven.apache.org/POM/4.0.0
https://maven.apache.org/xsd/maven-4.0.0.xsd">
```

```
    <modelVersion>4.0.0</modelVersion>
```

```
    <groupId>com.gfg</groupId>
```

```
    <artifactId>JUnitMavenExample</artifactId>
```

```
<version>1.0-SNAPSHOT</version>
```

```
<properties>
```

```
  <maven.compiler.source>17</maven.compiler.source>
```

```
  <maven.compiler.target>17</maven.compiler.target>
```

```
  <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
```

```
</properties>
```

```
<dependencies>
```

```
  <!-- JUnit dependency for testing -->
```

```
  <dependency>
```

```
    <groupId>junit</groupId>
```

```
    <artifactId>junit</artifactId>
```

```
    <version>4.13.2</version>
```

```
    <scope>test</scope> <!-- Indicates that this dependency is only required for the  
test phase -->
```

```
  </dependency>
```

```
</dependencies>
```

```
<build>
```

```
  <plugins>
```

```
    <plugin>
```

```
      <groupId>org.apache.maven.plugins</groupId>
```

```
      <artifactId>maven-surefire-plugin</artifactId>
```

```
      <version>2.22.2</version>
```

```
    </plugin>
```

```
  </plugins>
```

```
</build>
```

```
</project>
```

La dependencia JUnit se incluye en el alcance **de la prueba** , lo que significa que solo se utiliza durante la fase de prueba. Esto maven-surefire-plugin garantiza que las pruebas se ejecuten durante el proceso de compilación.

Paso 3: Crear la clase Calculadora

Crea la Calculator clase, que tendrá métodos aritméticos básicos: add() y subtract().

Calculadora.java

```
package com.gfg;
```

```
public class Calculator {  
  
    // Method to add two numbers  
    public int add(int a, int b) {  
        return a + b; // Returns the sum of a and b  
    }  
  
    // Method to subtract one number from another  
    public int subtract(int a, int b) {  
        return a - b; // Returns the difference of a and b  
    }  
}
```

- **add(int a, int b)** : este método suma los dos números y devuelve el resultado.
- **restar(int a, int b)** : este método resta el segundo número del primer número.

Paso 4: Crear la clase principal

Cree la Main clase para realizar las operaciones aritméticas básicas y demostrar el uso de la Calculator clase.

Principal.java

```
package com.gfg;
```

```
//TIP To <b>Run</b> code, press <shortcut actionId="Run"/> or  
// click the <icon src="AllIcons.Actions.Execute"/> icon in the gutter.
```

```
public class Main {  
  
    public static void main(String[] args) {  
        Calculator calculator = new Calculator();  
  
        // Example of adding two numbers
```

```

    int sum = calculator.add(5, 3);

    System.out.println("Sum of 5 and 3 is: " + sum);

    // Example of subtracting two numbers

    int difference = calculator.subtract(5, 3);

    System.out.println("Difference of 5 and 3 is: " + difference);

}
}

```

El main método permite la ejecución independiente de la clase. Demuestra el uso de los métodos add y subtract mediante la creación de una instancia de la Calculator clase y la impresión de los resultados en la consola.

Paso 5: Crear la clase de prueba JUnit

Cree la CalculatorTest clase que contiene los métodos de prueba para verificar la funcionalidad de la Calculator clase.

CalculadoraTest.java

```

import com.gfg.Calculator;

import org.junit.Test;

import static org.junit.Assert.assertEquals;

public class CalculatorTest {

    // Instantiate the Calculator object

    Calculator calculator = new Calculator();

    // Test for the add method

    @Test

    public void testAdd() {

        int result = calculator.add(2, 3); // Calls the add method

        assertEquals(5, result); // Verifies if 2 + 3 equals 5

    }

    // Test for the subtract method

    @Test

```

```

public void testSubtract(){

    int result = calculator.subtract(5, 3); // Calls the subtract method

    assertEquals(2, result); // Verifies if 5 - 3 equals 2


}
}

```

- **testAdd()** : comprueba si el add() método de Calculator devuelve la suma correcta.
- **testSubtract()** : verifica la salida del subtract() método.
- assertEquals(expected, actual): Afirma que los valores esperados y reales son los mismos.

Paso 6: Ejecutar la aplicación

Una vez completado el proyecto, ejecute la Main clase y debería ver el siguiente resultado:



```

Run Main x
"C:\Program Files\Java\jdk-17\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2024.1.4\bin\ -Dfile.encoding=UTF-8 -classpath "C:\Users\Syam\OneDrive\Desktop\Spring Projects\JUnit
Sum of 5 and 3 is: 8
Difference of 5 and 3 is: 2
Process finished with exit code 0

```

Paso 7: Ejecutar las pruebas con Maven

Para ejecutar las pruebas, ejecute el siguiente comando Maven en el directorio del proyecto:

prueba mvn

Este comando compila el código y ejecuta los casos de prueba ubicados en el src/test/javadirectorio.

Resultados de la prueba:

```

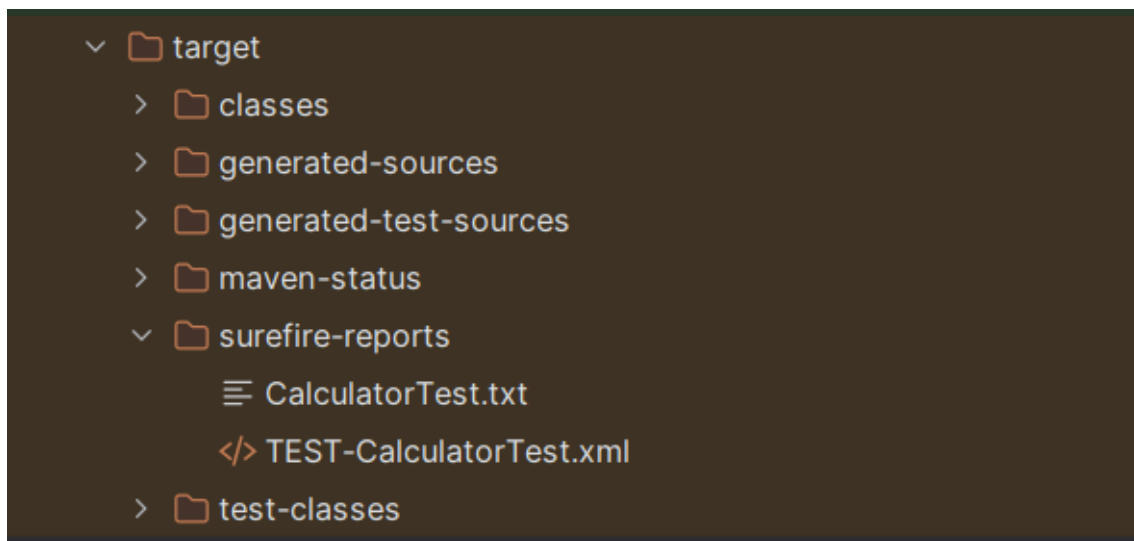
[INFO] -----
[INFO] T E S T S
[INFO] -----
[INFO] Running CalculatorTest
[INFO] Tests run: 2, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.068 s - in CalculatorTest
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 2, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 4.399 s
[INFO] Finished at: 2024-10-02T20:46:10+05:30
[INFO] -----
PS C:\Users\Syam\OneDrive\Desktop\Spring Projects\JUnitMavenExample>

```

Paso 8: Informe de prueba generado

El complemento Surefire de Maven genera automáticamente un informe de prueba en el target/surefire-reports/directorio.

- **Archivo de informe Surefire** : target/surefire-reports/CalculatorTest.txt
este archivo contiene un registro detallado de la ejecución de la prueba, incluido el estado de aprobación/rechazo y los seguimientos de la pila para cualquier prueba fallida.



Este proyecto de ejemplo demuestra cómo configurar el proyecto Maven para trabajar con JUnit, incluyendo la creación de la clase Calculadora simple con métodos aritméticos y la escritura de la clase de prueba JUnit correspondiente para verificar el comportamiento de la clase Calculadora, la ejecución de las pruebas usando el comando Maven (mvn test) y la visualización de los informes generados.