JUnit: Java 程式測試框架

JUnit 是一個流行的 Java 測試框架。它允許開發者編寫和運行可重複的自動化測試，這有助於確保代碼按預期運行。以下是如何使用 JUnit 進行測試的基本指南：

1. 在項目中設置 JUnit

- **Maven**：在 `pom.xml` 文件中添加 JUnit 依賴。

```
<dependency>
    <groupId>junit</groupId>
    <artifactId>junit</artifactId>
    <version>4.13.2</version>
    <scope>test</scope>
</dependency>
```

- **Gradle**：在 `build.gradle` 文件中添加 JUnit 依賴。

```
testImplementation 'junit:junit:4.13.2'
```

2. 編寫測試類

為測試創建一個新的 Java 類。最好將測試類命名為正在測試的類，並附加“Test”後綴。

```
import org.junit.Test;
import static org.junit.Assert.assertEquals;

public class MyClassTest {

    @Test
    public void testAddition() {
        MyClass myClass = new MyClass();
        int result = myClass.add(2, 3);
        assertEquals(5, result);
    }
}
```

3. 注釋

- `@Test`：表示該方法是測試方法。
- `@Before`：在每個測試方法之前運行。適合設置。

- `@After`：在每個測試方法之後運行。適合清理。
- `@BeforeClass`：在類中的任何測試方法之前運行一次。
- `@AfterClass`：在類中的所有測試方法之後運行一次。

4. 斷言

斷言用於驗證代碼按預期運行。

- `assertEquals(expected, actual)`：檢查兩個值是否相等。
- `assertTrue(condition)`：檢查條件是否為真。
- `assertFalse(condition)`：檢查條件是否為假。
- `assertNull(object)`：檢查對象是否為空。
- `assertNotNull(object)`：檢查對象是否不為空。

5. 運行您的測試

您可以使用 IDE 的內建工具或通過命令行使用 Maven 或 Gradle 運行測試。

- **Maven**：`mvn test`
- **Gradle**：`gradle test`

範例

以下是一個簡單的類及其相應測試類的範例：

```
// MyClass.java
public class MyClass {
    public int add(int a, int b) {
        return a + b;
    }
}

// MyClassTest.java
import org.junit.Test;
import static org.junit.Assert.assertEquals;

public class MyClassTest {

    @Test
    public void testAddition() {
```

```
        MyClass myClass = new MyClass();
        int result = myClass.add(2, 3);
        assertEquals(5, result);
    }
}
```

額外提示

- **測試獨立性**：每個測試應獨立於其他測試。避免測試之間的依賴。
- **測試覆蓋率**：力求高測試覆蓋率，但要專注於有意義的測試，而不是僅僅增加覆蓋率百分比。
- **模擬**：使用模擬框架如 Mockito 將正在測試的類與其依賴隔離。

按照這些步驟，您可以有效地使用 JUnit 為 Java 應用編寫和運行測試。