



# Test Automatisé

---

Présentée par :  
Emna Ouni



# Sommaire



1

**INTRODUCTION CUCUMBER**

2

**TDD**

3

**BDD**

4

**STRUCTURE SCENARIO BDD**

5

**CONCEPTE CUCUMBER**

6

**FONCTIONALITES ET METHODES**

7

**PAGE FACTORY**

8

**JUNIT**

9

**TESTNG**



## TestNG

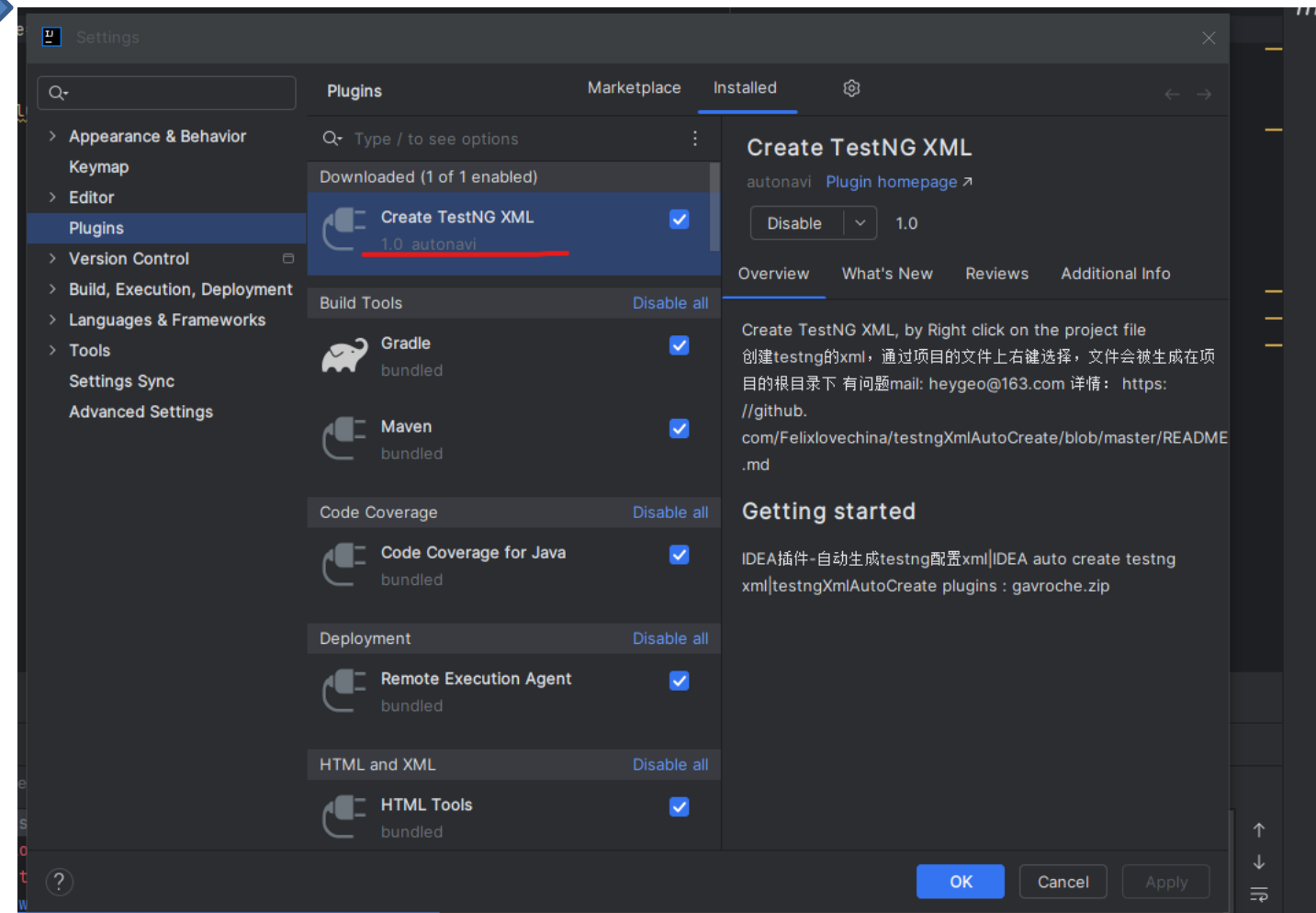
**TestNG** est un framework de test basé sur Java qui est conçu pour faciliter l'écriture et l'exécution de tests unitaires et d'intégration. Il est largement utilisé pour les tests automatisés, en particulier avec des applications développées en Java. TestNG offre des fonctionnalités avancées comme la gestion des dépendances, le regroupement des tests, la parallélisation,

# Implémenté TestNG

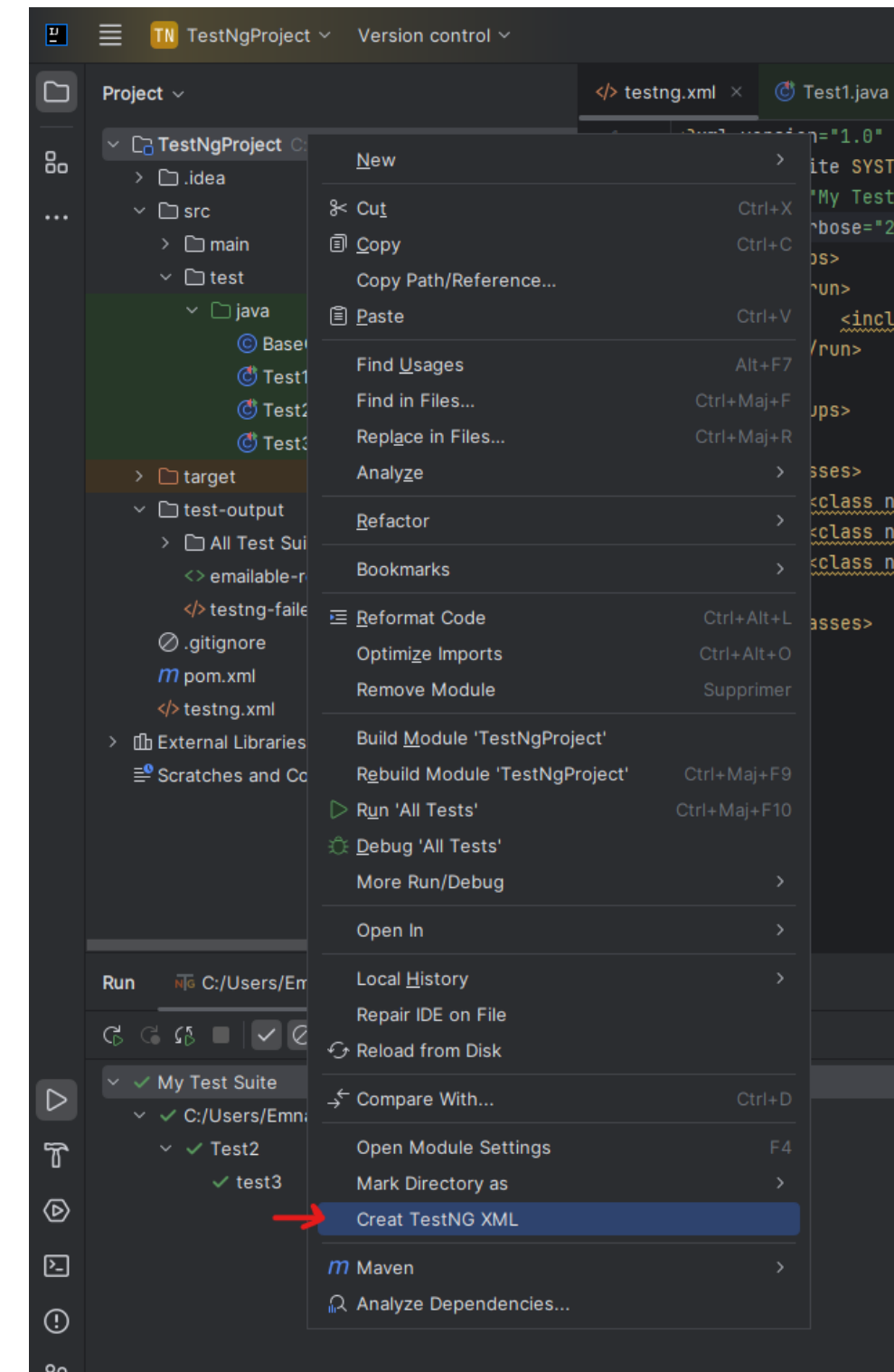
```
16      <dependencies>
17          <dependency>
18              <groupId>org.seleniumhq.selenium</groupId>
19              <artifactId>selenium-java</artifactId>
20              <version>4.22.0</version>
21          </dependency>
22          <!-- https://mvnrepository.com/artifact/org.testng/testng -->
23          <dependency>
24              <groupId>org.testng</groupId>
25              <artifactId>testng</artifactId>
26              <version>7.10.2</version>
27              <scope>test</scope>
28          </dependency>
29      </dependencies>
30
```

# Création testng.xml

1



2



## Annotations principales de TestNG :

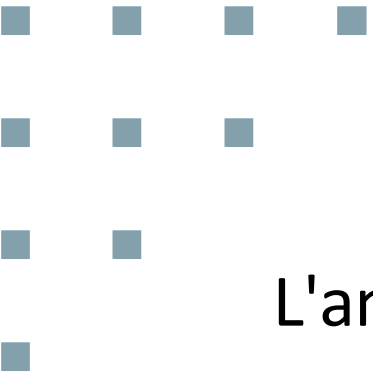
**TestNG** offre plusieurs annotations qui vous permettent de structurer et de contrôler vos tests de manière efficace.

### @Test

- Utilisée pour marquer une méthode comme un test à exécuter.
- Options : enabled, timeout, priority, dependsOnMethods, etc.

```
@Test (enabled = false)
public void Test() {
    driver.get("https://www.saucedemo.com/");
    String title = driver.getTitle();
    String expected = "Swag Labs";
    Assert.assertEquals(expected, title);
}
```





L'annotation **timeout** est utilisée pour définir une limite de temps d'exécution pour un test. Si le test prend plus de temps que le délai spécifié pour s'exécuter, il sera considéré comme échoué.

```
@Test (timeout = 3000)
public void Test1() {
    driver.get("https://www.saucedemo.com/");
    String title = driver.getTitle();
    String expected = "Swag Labs";
    Assert.assertEquals(expected, title);
}
```



Dans TestNG, pour désactiver un test, vous pouvez utiliser l'attribut **enabled=false** dans l'annotation `@Test`. Cela permet d'ignorer un test sans le supprimer du code.

```
@Test(enabled = false)
public void Test1() {
    driver.get("https://www.saucedemo.com/");
    String title = driver.getTitle();
    String expected = "Swag Labs";
    Assert.assertEquals(expected, title);
}
```



L'attribut **priority** est utilisé pour définir l'ordre d'exécution des tests dans une classe de tests. Les tests avec des valeurs de priorité inférieures seront exécutés avant ceux avec des valeurs plus élevées.

```
@Test(priority = 1)
public void test1() {
    System.out.println("first_test");
}

@Test(priority = 2)
public void test2() {
    System.out.println("second_test");
}

@Test(priority = 0)
public void test3() {
    System.out.println("third_test");
}

@Test
public void test4() {
    System.out.println("last_test");
}
```

✓ Default Suite	11 ms	✓ Tests passed: 4 of 4 tests – 11 ms
✓ Testng1	11 ms	third_test
✓ TESTNG	11 ms	last_test
✓ test3	8 ms	first_test
✓ test4	1 ms	second_test
✓ test1	1 ms	
✓ test2	1 ms	=====

L'attribut **dependsOnMethods** permet de spécifier que l'exécution d'un test dépend de la réussite d'un ou plusieurs autres tests. Si le test dont dépend un autre échoue, le test dépendant sera ignoré.

```
@Test(priority = 1)
public void test1() {
    System.out.println("first_test");
}

@Test(priority = 2)
public void test2() {
    System.out.println("second_test");
}

@Test(priority = 0)
public void test3() {
    System.out.println("third_test");
}

@Test
public void test4() {
    System.out.println("last_test");
}

@Test (dependsOnMethods = {"test3"})
public void test5(){
    System.out.println("test fivee");
}
```

✓ Testng1	11 ms	third_test
✓ TESTTNG	11 ms	last_test
✓ test3	7 ms	test fivee
✓ test4	1 ms	first_test
✓ test5	1 ms	second_test
✓ test1	1 ms	
✓ test2	1 ms	=====
		Default Suite

L'annotation **@Groups** est utilisée pour regrouper les tests et les exécuter par groupe.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE suite SYSTEM "http://testng.org/testng-1.0.dtd">
<suite name="My Test Suite">
  <test verbose="2" preserve-order="true" name="C:/Users/Emna/IdeaProjects/TestNgProject">
    <groups>
      <run>
        <include name="regression"></include>
      </run>
    </groups>
    <classes>
      <class name="Test1"></class>
      <class name="Test2"></class>
      <class name="Test3"></class>
    </classes>
  </test>
</suite>
```

Run C:/Users/Emna/IdeaProjects/TestNgProject/testng.xml

✓ My Test Suite 10 ms

✓ C:/Users/Emna/IdeaProjects/TestNgProject 10 ms

✓ Test2 10 ms

✓ test3 0 ms

SLF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder"

SLF4J: Defaulting to no-operation (NOP) logger implementation

SLF4J: See <http://www.slf4j.org/codes.html#StaticLoggerBinder>

Before Groups

Methode3\_Groupe2

After Groups

My Test Suite

Total tests run: 1, Passes: 1, Failures: 0, Skips: 0

Process finished with exit code 0

```
import ...

public class Test2 {
  @BeforeGroups("regression")
  public void beforeGroups() { System.out.println("Before Groups"); }

  @Test(groups = "Groupe1")
  public void test1() {
    System.out.println("Methode1_Groupe1");
  }

  @Test(groups = "Groupe1")
  public void test2() {
    System.out.println("Methode2_Groupe1");
  }

  @Test(groups = {"regression"})
  public void test3() {
    System.out.println("Methode3_Groupe2");
  }

  @AfterGroups("regression")
  public void afterGroups() {
    System.out.println("After Groups");
  }
}
```



■ ■ ■ ■

■ ■ ■

■ ■

■

L'annotation **@DataProvider** dans TestNG permet de fournir des données à vos tests sous forme de tableau. Cela vous permet d'exécuter le même test avec plusieurs ensembles de données, facilitant ainsi les tests paramétrés.

```
@DataProvider(name = "Login")
public Object[][] Login() {
    return new Object[][]{
        {"standard_user", "secret_sauce"}, // Cas 1 : Connexion réussie
        {"AMEL", "secret_sauce"}, // Cas 2 : Mot de passe incorrect
        {"AMEL", "secret_sauce"}, // Cas 3 : Nom d'utilisateur incorrect
        {"", "secret_sauce"}, // Cas 4 : Nom d'utilisateur vide
        {"AMEL", ""}, // Cas 5 : Mot de passe vide
    };
}
```

```
@Test(dataProvider = "Login")
public void Test2(String Username, String Password) {
    driver.get("https://www.saucedemo.com/");
    String title = driver.getTitle();
    String expected = "Swag Labs";
    Assert.assertEquals(expected, title);
    WebElement username = driver.findElement(By.xpath(xpathExpression: "//*[@id='user-name']"));
    username.sendKeys(Username);
    WebElement password = driver.findElement(By.xpath(xpathExpression: "//*[@id='password']"));
    password.sendKeys(Password);
    WebElement login = driver.findElement(By.xpath(xpathExpression: "//*[@id='login-button']"));
    login.click();
}
```

# Création rapport Testng

