# Automation Frameworks

Automation bootcamp - day 2

## TestNG - Execution flow

@Test : runs the annotated method and reports its result. This is where the verifications are made.

@BeforeTest: runs the annotated method before each test method. Here should be all instantiation code that needs to be refreshed before executing each test.

@AfterTest: runs the annotated method after each test method. Usually used to free resources that are no longer needed after executing each test.

@BeforeClass: runs the annotated method before instantiating the test class. Here we should instantiate anything that is needed across all tests.

@AfterClass: runs the annotated method after executing all the test class. Here we free the resources that will no longer be used after executing all tests.

**IMPORTANTE:** importar las annotations de org.testng.annotations y no las de junit!

# TestNG - Maven dependency

https://mvnrepository.com/artifact/org.testng/testng

```
<dependency>
   <groupId>org.testng</groupId>
   <artifactId>testng</artifactId>
   <version>6.14.2</version>
   <scope>test</scope>
</dependency>
```

# TestNG - Exercise 1

Write a test class that contains a print line with the name of the annotation that is used.

The test class should contain each of the previously mentioned annotations.

Expected result:

```
Before class method

Before test method

Test 1 method

After test method

After class method
```

## TestNG - Data Providers

A data provider is a data matrix containing rows of test parameters. The test takes each row of parameters and runs, so the tests are repeated n times, being n the amount of rows.

This enables data driven test, where test data is not hardcoded.

A data driven test executes verifications on different inputs and verifies that the outputs are correct for each input set.

## TestNG - Data Providers

```
@DataProvider(name = "Authentication")

  public static Object[][] credentials() {

        return new Object[][] { { "testuser_1", "Test@123" }, { "testuser_1", "Test@123" }};

  }



@Test(dataProvider = "Authentication")

  public void test(String sUsername, String sPassword) {
        // test code
  }
```

## TestNG - Exercise 2

Write a new class with the same structure as Exercise 1 but write the string to be printed in the test method should come from a data provider parameter.  Expected result:

```
Before class method
Before test method
FIRST TEST STRING
After test method
Before test method
SECOND TEST STRING
After test method
After class method
```

## Assert

Assertions are checkpoints during a test execution where we can verify that some condition is met.

If the condition fails, the test will be failed and stop, if it passes, the test will continue.

Examples:

```
Assert.assertTrue(a == b, “Value A and B are the same”);

Assert.assertFalse(a == b, “Value A and B are different”);
```

# TestNG - Exercise 3

Write a test class that contains two test methods, one failing and one passing.