11Q) what are different annotations in testng and junit

The annotation supports are implemented in both JUnit 4 and TestNG look similar.

|  |  |  |
| --- | --- | --- |
| Feature | JUnit 4 | TestNG |
| test annotation | @Test | @Test |
| run before all tests in this suite have run | — | @BeforeSuite |
| run after all tests in this suite have run | — | @AfterSuite |
| run before the test | — | @BeforeTest |
| run after the test | — | @AfterTest |
| run before the first test method that belongs to any of these groups is invoked | — | @BeforeGroups |
| run after the last test method that belongs to any of these groups is invoked | — | @AfterGroups |
| run before the first test method in the current class is invoked | @BeforeClass | @BeforeClass |
| run after all the test methods in the current class have been run | @AfterClass | @AfterClass |
| run before each test method | @Before | @BeforeMethod |
| run after each test method | @After | @AfterMethod |
| ignore test | @ignore | @Test(enbale=false) |
| expected exception | @Test(expected = ArithmeticException.class) | @Test(expectedExceptions = ArithmeticException.class) |
| timeout | @Test(timeout = 1000) | @Test(timeout = 1000) |

The main annotation differences between JUnit4 and TestNG are

1. In JUnit 4, we have to declare “@BeforeClass” and “@AfterClass” method as static method. TestNG is more flexible in method declaration, it does not have this constraints.

2. 3 additional setUp/tearDown level: suite and group (@Before/AfterSuite, @Before/AfterTest, @Before/AfterGroup). See more

12Q) what is group and suite and parallel execution in testing?

Parallelism or multi-threading in software terms is defined as the ability of the software, operating system, or program to execute multiple parts or sub-components of another program simultaneously. TestNG allows the tests to run in parallel or multi-threaded mode. This means that based on the test suite configuration, different threads are started simultaneously and the test methods are executed in them. This gives a user a lot of advantages over normal execution, mainly reduction in execution time and ability to verify a multi-threaded code.

13Q) how to decide which test cases needs to be automated?

Identify the parameters on which you will base your test case as a candidate for automation.

As of now I am identifying the below parameters, you can have your own parameters depending on your application.

* Test case executed with different set of data
* Test case executed with different browser
* Test case executed with different environment
* Test case executed with complex business logic
* Test case executed with different set of users
* Test case Involves large amount of data
* Test case has any dependency
* Test case requires Special data

Step 2:

Break each application into modules. For each module, analyze and try to identify the test cases which should be automated based on the parameters. This list will vary for projects to projects and can also be enhanced to suite your needs:

In a similar way, for all modules, this list can be used to identify the automation candidate test cases.

Step 3:

Consolidate and group the number of test cases for each module

Step 4:

Once you have identified all the granular level details, you can present them in the below way. We are now progressing to calculate the ROI.

We should also take into account the below attributes which forms the basis of deterring the ROI:

* Purchasing and licensing cost of the tool
* Time to develop the scripts
* Time to maintain the scripts.
* Time to analyze the results manually and automatically
* Time and cost to train the resources.
* Management overheads.