Assignment 4

TestNG

TestNG is a testing framework for the Java programming language created by Cédric Beust and inspired by JUnit and NUnit. The design goal of TestNG is to cover a wider range of test categories: unit, functional, end-to-end, integration, etc., with more powerful and easy-to-use functionalities.

TestNG is inspired from JUnit and NUnit but introducing some new functionalities that make it more powerful and easier to use, such as:

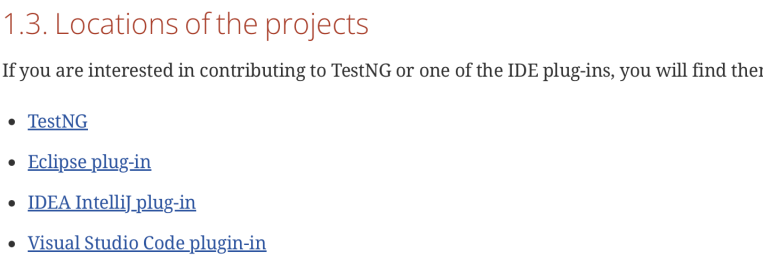
- Annotations.
- Run your tests in arbitrarily big thread pools with various policies available (all methods in their own thread, one thread per test class, etc…).
- Test that your code is multithread safe.
- Flexible test configuration.
- Support for data-driven testing (with @DataProvider).
- Support for parameters.
- Powerful execution model (no more TestSuite).
- Supported by a variety of tools and plug-ins (Eclipse, IDEA, Maven, etc…).
- Embeds BeanShell for further flexibility.
- Default JDK functions for runtime and logging (no dependencies).
- Dependent methods for application server testing.

TestNG makes test creation, execution and reporting efficient

It uses annotations like:

- **@BeforeMethod**: This will be executed before every @test annotated method.
- **@AfterMethod:** This will be executed after every @test annotated method.
- **@BeforeClass:** This will be executed before first @Test method execution. It will be executed one only time throughout the test case.
- **@AfterClass:** This will be executed after all test methods in the current class have been run
- **@BeforeTest:** This will be executed before the first @Test annotated method. It can be executed multiple times before the test case.
- **@AfterTest**: A method with this annotation will be executed when all @Test annotated methods complete the execution of those classes inside the <test> tag in the TestNG.xml file.
- **@BeforeSuite**: It will run only once, before all tests in the suite are executed.
- **@AfterSuite:** A method with this annotation will run once after the execution of all tests in the suite is complete.
- **@BeforeGroups**: This method will run before the first test run of that specific group.
- **@AfterGroups**: This method will run after all test methods of that group complete their execution.

Install TestNG to your Eclipse (or any IDE that you are using). Go to https://testng.org/ and follow the link that is given there in section 1.3. Locations of the projects.



You will be directed to the github repository. Follow the steps of installation.

Then add the dependency for Maven from https://mvnrepository.com/artifact/org.testng/testng/7.10.2

Test Log

Test logging and test reporting are an integral part of the Software Testing Life Cycle. A test log is a vital component of test reporting because it provides the detailed information needed to support the conclusions and insights presented in the test report.

Test logging refers to the process of documenting the details of the testing process, including the test case execution results, the environment configuration, and the issues encountered during testing with proper RCA.

The prime purpose of test logging is to keep a record of the testing process so that the testing team can review the results and determine whether the software under test meets the expected quality standards. The focus is to enable post execution diagnosis of failures and defects in the software.

Test management tools often provide built-in capabilities to generate test logs automatically or enable testers to export the log information in a specific format.

The test log file serves as a valuable source of information for test reporting, tracking progress, identifying trends, and analyzing the overall quality of the software under test. It can also facilitate collaboration among team members and provide a historical record for future reference or audits.

Advantages of Test Log

*Traceability***:** All the test activities and results can be tracked with a test log. This helps to ensure that all the required test activities were performed and completed for a particular iteration as a proof to show the required audience whenever it is demanded. Also, any issues identified during testing can be accurately traced back to their source.

Defect tracking: Issues found during testing can be tracked with details such as severity, priority and status. This helps the QA team to prioritise the issues which need early resolution before release.

*Documentation***:** Test artifacts and deliverables are the fundamental part of the Software Testing Lifecycle. A test log consists of all the necessary test execution details which can be used as a formal record. It also helps to provide a historical record of the testing process which can be referred to in future.

Transparency between teams: A test log facilitates communication between QA members as well as with the development and product team. For example- Test logs can be uploaded on Confluence or Sharepoint and can serve as a common platform which can be referred by any team to view test logs.

Test analysis and improvement: The historical test log data can be reviewed by the testing team to identify trends (w.r.t issues), patterns, and areas for improvement. This helps to enhance the testing process and upgrade the quality of software/ application under test.

Components of Log4j

The Log4j logging framework comprises the following components:

- Logger
- Appenders
- Layout

Logger

The function of the logger in Log4j is basically storing and capturing all the necessary logging information that will be generated using the framework.

To truly understand its functioning, let's dig a little deeper and discuss the logger class, and log level methods. The loggers also decide which priority is going to be captured.

Logger Class – To fully use the logger, create an instance for a logger class where all the generic methods will be at the user's disposal, required to use Log4j.

Log Levels – These are the methods that will be used to print the log messages. There are primarily only a few log levels that are used in a script.

ALL **–** This level will prioritize and include everything in the logs.

ERROR **–** This level will show messages that inform users about error events that may not stop the application.

WARN **–** This level will show information regarding warnings, that may not stop the execution but may still cause problems.

DEBUG **–** This level will log debugging information.

INFO **–** This level will log the progress of the application.

FATAL **–** This will print information critical to the system that may even crash the application.

Appenders

The appender basically grabs information from the logger and writes log messages to a file or any other storage location. The following are some of the appenders one can use for Log4j:

FileAppender – This will append the log messages to a file.

RollingFileAppender – It will perform the same function as FileAppender, but users will be able to specify the maximum file size. Once the limit is exceeded, the appender will create another file to write the messages.

DailyRollingFileAppender – It specifies the frequency by which the messages are to be written to the file.

ConsoleAppender – In this, the appender will simply write the log messages in the console.

Layout

The layout is where the format in which log messages will appear is decided. There are several layouts one can use for log messages:

Pattern Layout – The user must specify a conversion pattern based on which the logs will be displayed. Otherwise, it takes the default conversion pattern in case of "no pattern specified".

HTML Layout – In this layout, the format of logs will be in the form of an HTML table.

XML Layout – This will show the logs in an XML format.

Extent Report

Extent Reports is an open-source reporting library useful for test automation. It can be easily integrated with major testing frameworks like JUnit, NUnit, TestNG, etc. These reports are HTML documents that depict results as pie charts. They also allow the generation of custom logs, snapshots, and other customized details.

Once an automated test script runs successfully, testers need to generate a test execution report. While TestNG does provide a default report, they do not provide the details.

Extent Reports in Selenium contain two major, frequently used classes:

- *ExtentReports* class
- *ExtentTest* class

The ExtentReports class generates HTML reports based on a path specified by the tester. Based on the Boolean flag, the existing report has to be overwritten or a new report must be generated. 'True' is the default value, meaning that all existing data will be overwritten.

The ExtentTest class logs test steps onto the previously generated HTML report.

Both classes can be used with the following built-in methods:

- *startTest*: Executes preconditions of a test case
- *endTest*: Executes postconditions of a test case
- *Log*: Logs the status of each test step onto the HTML report being generated
- *Flush*: Erases any previous data on a relevant report and creates a whole new report

A Test Status can be indicated by the following values:

- *PASS*
- *FAIL*
- *SKIP*
- *INFO*

Tasks:

1) **Implement TestNG annotations (30 pts)**

Use any website by your choice. Your code should perform at least three actions on that website. Use the **TestNG annotations**. Provide the program code and the resulting screenshots in your report. Additionally, provide the screenshot of the *index.html* file from the "*test-output*" folder of your project, go to different links from that file and make screenshots of some meaningful results.

2) **Configure and implement logging (30 pts)**

You can use **Log4j** for Java, **logging module** for Python, or other techniques. Implement some actions on the chosen website. Then insert the information about the success of these actions into a log file with date and time. Again, provide the code of your program, the screenshots of the resulting web pages and the log file.

3) **Implement Extent Report and Screenshots (40 pts)**

Implement the Extent Report (either with Basic or TestNG usage) to some web-site, performing at least one action on it with the Screenshots feature. Provide the code of the program, the figures of the resulting web-pages of the chosen web-site and the figures of the extent.html file with the screenshots taken by Extent Report.