122 TestNG interview questions to ask your applicants

Questions

1. What is TestNG and why do we use it?

2. Can you explain the basic structure of a TestNG test class?

3. What are TestNG annotations? Give some examples.

4. How do you run a simple TestNG test?

5. What is the purpose of the @BeforeTest and @AfterTest annotations?

6. How can you group tests in TestNG?

7. What is the use of the testng.xml file?

8. How do you pass parameters to a TestNG test?

9. What is dataProvider in TestNG and where do we use it?

10. How do you handle exceptions in TestNG tests?

11. How to skip a test case in TestNG?

12. What are assertions in TestNG? Give some examples.

13. How do you generate reports in TestNG?

14. What is the difference between @BeforeMethod and @BeforeClass?

15. Can you tell me something about parallel test execution in TestNG?

16. How do you set priority to test methods in TestNG?

17. What is the role of 'suite' in TestNG?

18. What do you understand by the term 'listeners' in TestNG?

19. How do you configure logging in TestNG?

20. How to make a test dependent on another test in TestNG?

21. What is the use of ITestContext in TestNG?

22. Can you explain the difference between 'hard assert' and 'soft assert'?

23. What are some advantages of using TestNG over JUnit?

24. How do you configure a timeout for a test case?

25. What is the purpose of using groups in testng.xml?

26. Explain the use of the 'enabled' attribute in @Test annotation.

27. What is IRetryAnalyzer and how is it useful?

28. What is the difference between @Parameters and DataProvider annotations?

29. What is TestNG, in super simple terms, and why do we use it for testing our code, like checking if your toys work properly?

30. Imagine each test is a tiny puzzle. How does TestNG help us put those puzzles in order and run them automatically?

31. Can you describe a simple test you've written using TestNG, like making sure a calculator app adds numbers correctly?

32. What's an annotation in TestNG? Think of it as a sticky note that tells TestNG what to do with your test. Give an example.

33. What is @Test annotation? Give an example where it is used in a program

34. What is @BeforeMethod annotation? Give an example where it is used in a program

35. What is @AfterMethod annotation? Give an example where it is used in a program

36. What is @BeforeClass annotation? Give an example where it is used in a program

37. What is @AfterClass annotation? Give an example where it is used in a program

38. What is @BeforeTest annotation? Give an example where it is used in a program

39. What is @AfterTest annotation? Give an example where it is used in a program

40. What is @BeforeSuite annotation? Give an example where it is used in a program

41. What is @AfterSuite annotation? Give an example where it is used in a program

42. How do you check if a test in TestNG passes or fails, like knowing if you got all the answers right on a quiz?

43. What's the difference between 'Assert.assertEquals' and 'Assert.assertTrue' in TestNG? It's like checking if something is exactly what you expect or just generally true.

44. If a test fails, how can you tell TestNG to try running it again? It's like getting a second chance to solve a puzzle.

45. What is the use of testng.xml?

46. How can you tell TestNG to skip a test, like deciding not to play a game right now?

47. Can you group tests together in TestNG, like putting all your dinosaur toys in one box and your car toys in another?

48. How do you pass information to your tests in TestNG, like giving instructions to a robot?

49. Have you ever used TestNG to test a website? What did you check, like making sure all the buttons work?

50. What's one thing you find a little tricky about TestNG, and how are you trying to learn more about it?

51. How do you handle exceptions within TestNG tests, and what are the best practices for doing so?

52. Explain the concept of 'Data Provider' in TestNG. How do you parameterize your tests using it, and what are its advantages?

53. What is the significance of the dependsOnMethods attribute in TestNG, and how can it be used to manage test dependencies?

54. Describe how you would use TestNG to perform cross-browser testing. What configurations and setup are required?

55. How can you configure TestNG to run tests in parallel? What are the different parallel modes available, and when would you use each one?

56. Explain the use of listeners in TestNG. How can you create custom listeners, and what events can they listen to?

57. How does TestNG support reporting? What built-in reporters are available, and how can you integrate custom reporting mechanisms?

58. Describe the concept of 'retryAnalyzer' in TestNG. How do you implement it, and when would you use it to re-run failed tests?

59. How would you use TestNG to test REST APIs? What libraries or tools might you integrate with TestNG for API testing?

60. Explain how you can group tests in TestNG. How are these groups defined and used for test execution?

61. How can you prioritize test execution in TestNG? What attributes or mechanisms are available for setting test priorities?

62. What are the differences between @BeforeSuite, @BeforeTest, @BeforeClass, and @BeforeMethod annotations in TestNG, and when would you use each?

63. How do you handle configuration failures in TestNG? What are the different ways to control test execution when a configuration method fails?

64. Explain the use of ITestContext and ITestResult interfaces in TestNG. How can you retrieve information about the test execution context and results?

65. How can you integrate TestNG with a build tool like Maven or Gradle? What configurations are necessary to run TestNG tests as part of the build process?

66. Describe how you would use TestNG to perform data-driven testing. What are the different ways to provide test data to your tests?

67. Explain the purpose of the invocationCount and threadPoolSize attributes in the @Test annotation. How can they be used to control test execution?

68. How can you conditionally enable or disable tests in TestNG? What attributes or mechanisms are available for controlling test execution based on certain conditions?

69. Describe how you would use TestNG to perform exception testing. How can you verify that a test method throws a specific exception?

70. Explain how to use and configure the suiteXmlFile parameter in TestNG. How do you specify which test suites to run?

71. How do you pass parameters from the command line to your TestNG tests? Provide an example.

72. What is the use of the alwaysRun attribute in @Before and @After annotations? Explain with a scenario.

73. How does TestNG's dependency injection mechanism work, and what are its advantages in a complex test suite?

74. Explain the concept of 'grouping' in TestNG and how you can leverage it to manage and execute specific test scenarios efficiently.

75. Describe how you would implement parallel test execution using TestNG, including considerations for thread safety and resource contention.

76. What are TestNG Listeners, and can you provide an example of a scenario where you would use a custom listener?

77. How does TestNG handle exceptions during test execution, and what options do you have for controlling exception behavior?

78. Explain the differences between using dataProvider in the same class vs. a separate class, highlighting the advantages and disadvantages.

79. Describe how you would integrate TestNG with a CI/CD pipeline, including generating reports and handling test failures.

80. What are the different ways to configure TestNG, and how do you decide which approach is best for a given project?

81. How can you use TestNG to perform data-driven testing with different data sources, such as CSV files or databases?

82. Explain the concept of 'retry analyzers' in TestNG and how you would implement one to handle flaky tests.

83. How do you handle asynchronous operations or multithreaded code in your TestNG tests to ensure accurate and reliable results?

84. Describe a situation where you would use TestNG's factory feature, and how it differs from using a dataProvider.

85. What are some strategies for optimizing TestNG test execution speed, especially when dealing with a large number of tests?

86. How can you use TestNG to test REST APIs, including handling different request methods and validating responses?

87. Explain how to use TestNG's IAnnotationTransformer to modify test annotations dynamically at runtime.

88. Describe how you would create a custom TestNG reporter to generate specific types of reports or integrate with external tools.

89. How can you ensure that your TestNG tests are isolated from each other, preventing interference and maintaining test independence?

90. Explain the trade-offs between using TestNG's assert methods and using a dedicated assertion library like AssertJ or Hamcrest.

91. How would you design a TestNG test suite to thoroughly test a complex business process involving multiple modules and dependencies?

92. Describe how to configure and use the preserve-order attribute in TestNG's XML suite file.

93. How can you use TestNG to verify the performance of your application, including measuring execution time and identifying bottlenecks?

94. Explain how to implement a soft assertion mechanism in TestNG, where tests continue execution even if an assertion fails.

95. Describe how to use TestNG's parameter tag in the XML suite file to pass configuration values to your tests.

96. How can you use TestNG to test database interactions, including verifying data integrity and handling database transactions?

97. Explain how to handle timeouts in TestNG tests, preventing them from running indefinitely and consuming resources.

98. Describe how to use TestNG's invocationCount and threadPoolSize attributes to perform load testing.

99. How can you use TestNG to test web applications, including interacting with web elements and verifying UI behavior?

100. Explain how to implement a test suite that runs different sets of tests based on the environment (e.g., development, staging, production).

101. Describe how to use TestNG's dependsOnMethods attribute to create complex test dependencies and ensure proper test execution order, along with the risks involved.

102. Explain your approach to debugging TestNG tests, particularly when dealing with parallel execution or complex dependencies.