

■ TOP 10 MOST IMPORTANT TESTNG INTERVIEW QUESTIONS

A Must-Read Guide for SDET, QA & Automation Engineers

These 10 TestNG questions are the most commonly asked in automation interviews.

This guide explains each concept in a clear, thorough, and interview-oriented manner so you can understand the “why” behind each TestNG feature—not just the definition.

Perfect for anyone preparing for SDET, QA, and Automation roles!

- **Pro Tip:** Review this before every interview—TestNG is the backbone of automation frameworks.
- **Career Tip:** Mastering TestNG = stronger frameworks + cleaner automation + better confidence.

Prepared by: [Ajay Varma](#)
LinkedIn: <https://www.linkedin.com/in/ajay-varma-profile>

1. What is TestNG?

Explanation:

TestNG (Test Next Generation) is a powerful testing framework inspired by JUnit and NUnit. Unlike JUnit, TestNG supports:

- ✓ Parallel Test Execution
- ✓ Test Grouping
- ✓ Dependency Testing
- ✓ Data-driven Testing using DataProviders
- ✓ HTML Reporting

Because of these advanced features, almost all modern Selenium automation frameworks rely heavily on TestNG.

2. What are the most commonly used TestNG annotations?

Explanation:

Annotations in TestNG control the entire flow of test execution. They define *when* and *how* different setup, cleanup, and test steps should run.

List of key annotations:

- ✓ @BeforeSuite — Runs once before the entire test suite
- ✓ @BeforeTest — Runs before *test tag* in XML
- ✓ @BeforeClass — Runs before any method in the class
- ✓ @BeforeMethod — Runs before every @Test method
- ✓ @Test — The actual test case
- ✓ @AfterMethod — Runs after every @Test
- ✓ @AfterClass — Runs after all methods in the class
- ✓ @AfterTest — Runs after test tag execution
- ✓ @AfterSuite — Runs once after the entire suite execution

These provide complete lifecycle control for any automation framework.

Example:

```
@BeforeSuite  
public void setupSuite(){}  
  
@BeforeClass  
public void setupClass(){}  
  
@BeforeMethod  
public void setupMethod(){}  
  
@Test  
public void testCase(){}  
  
@AfterMethod  
public void tearDown(){}  

```

3. What is the use of @Test annotation?

Explanation:

The @Test annotation marks a method as a test case in TestNG. Without @Test, the framework will not execute the method.

Why it's important:

- ✓ Enables setting priority
- ✓ Supports parameters
- ✓ Allows retry logic
- ✓ Enables grouping
- ✓ Supports enabling/disabling tests

Example:

```
@Test  
public void testLogin(){  
    System.out.println("Login test executed");  
}
```

4. How do you prioritize TestNG test cases?

Explanation:

Priority defines the order in which tests run. Lower number = higher priority.

Importance:

- ✓ Useful when some tests must run first (e.g., login)
- ✓ Makes test flows predictable
- ✓ Helps build structured test execution

Example:

```
@Test(priority=1)
public void openBrowser(){}  
  
@Test(priority=2)
public void login(){}
}
```

5. How do you skip a test case in TestNG?

Explanation:

You can skip a test using **enabled=false**. This is helpful when:

- ✓ A test is under development
- ✓ A feature is incomplete
- ✓ You want temporary selective execution

Example:

```
@Test(enabled=false)
public void paymentTest(){ }
```

6. What is DataProvider in TestNG?

Explanation:

DataProvider allows running the same test method multiple times with different data sets. It is the backbone of data-driven testing.

Why it's important:

- ✓ Eliminates duplicate code for similar tests
- ✓ Supports multiple input sets
- ✓ Integrates with Excel/CSV/Databases

Example:

```
@DataProvider(name="loginData")
public Object[][][] getData(){
    return new Object[][][]{
        {"user1", "pass1"},
        {"user2", "pass2"}
    };
}

@Test(dataProvider="loginData")
public void loginTest(String user, String pass){}
```

7. How do you run tests in parallel?

Explanation:

Parallel execution reduces the total test execution time.

Used when:

- ✓ Running tests on multiple browsers
- ✓ Running tests on Selenium Grid
- ✓ CI/CD executions

Example:

```
<suite parallel="tests" thread-count="3">
```

8. What is test grouping in TestNG?

Explanation:

Grouping helps categorize tests logically.

Examples of groups:

- ✓ smoke
- ✓ regression
- ✓ sanity
- ✓ api

You can run a specific group using XML.

Example:

```
@Test(groups={"smoke"})
public void testSearch(){}
```

9. How do you generate TestNG reports?

Explanation:

TestNG automatically generates:

- ✓ test-output/index.html (Execution Report)
- ✓ test-output/emailable-report.html (Email-friendly report)

SDETs often extend reporting using:

- ✓ Extent Reports
- ✓ Allure Reports
- ✓ Custom listeners

10. What is dependsOnMethods?

Explanation:

Helps create controlled execution flow where one test depends on another.

Useful when:

- ✓ A test cannot run unless a previous test passes
- ✓ Building real workflow automation (login → add item → checkout)

Example:

```
@Test  
public void login(){}  
  
@Test(dependsOnMethods={"login"})  
public void checkout(){}
```