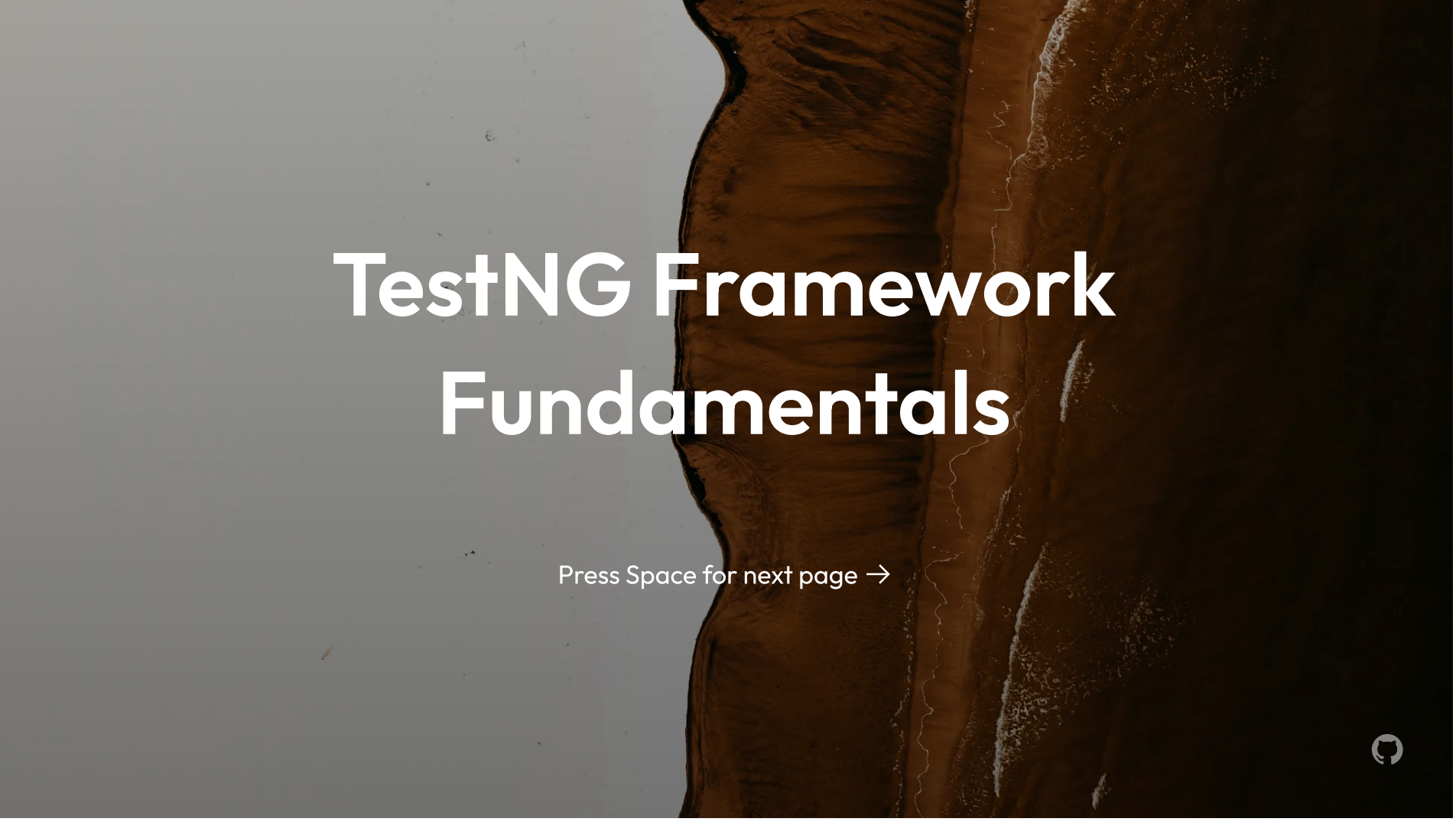

TestNG Framework
Fundamentals
Press Space for next page →

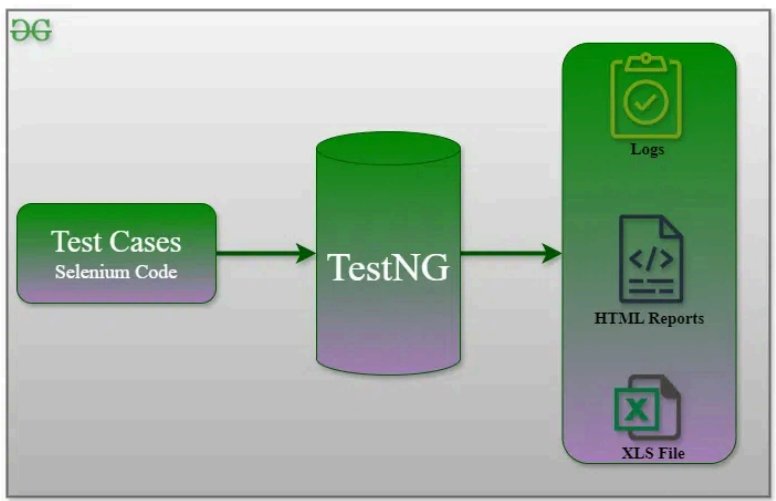
Agenda

1. What is TestNG?
2. Why TestNG?
3. TestNG Annotations
4. TestNG Annotations hierarchy
5. Code Example: TestNG Annotations
6. Dependent Test
7. Code Example: Dependent Test - Method
8. Always Run Test
9. Exception Handling in TestNG
10. Priority in TestNG

What is TestNG?

> TestNG(Test Next Generation) is a testing framework inspired from JUnit and NUnit but introducing some new functionalities that make it more powerful and easier to use.

Official Website: testng.org



Why TestNG?

- TestNG is designed to cover all categories of tests: unit, functional, end-to-end, integration, etc.
- **Advanced annotations:** More powerful and flexible annotations.
- **Grouping and Prioritizing:** TestNG allows you to group test cases and prioritize
- **Dependency Testing:** TestNG allows you to define dependencies between test methods.
- **Data-Driven Testing:** TestNG allows you to run the same test case with different data sets.
- **Parallel Testing:** TestNG allows you to run test cases in parallel.
- **Reporting:** TestNG provides detailed reports of test execution.

TestNG Annotations

- **@BeforeSuite:**
- **@AfterSuite:**
- **@BeforeTest:**
- **@AfterTest:**
- **@BeforeGroups:**
- **@AfterGroups:**
- **@BeforeClass:**
- **@AfterClass:**
- **@BeforeMethod:**
- **@AfterMethod:**

Java Annotations Syntax

- **Marker Annotations:** `@Test`
- **Single-Value Annotations:** `@Test(priority = 1)`
- **Multi-Value Annotations:** `@Test(priority = 1, enabled = true)`

TestNG Annotations hierarchy

```
BeforeSuite
'   BeforeTest
'   '   BeforeClass
'   '   '   BeforeMethod
'   '   '   '   myTestMethod3
'   '   '   AfterMethod
'   '   '   BeforeMethod
'   '   '   '   myTestMethod4
'   '   '   AfterMethod
'   '   AfterClass
'   AfterTest
'   BeforeTest
'   '   BeforeClass
'   '   '   BeforeMethod
'   '   '   '   myTestMethod1
'   '   '   AfterMethod
'   '   '   BeforeMethod
'   '   '   '   myTestMethod2
'   '   '   AfterMethod
'   '   AfterClass
'   '   BeforeClass
'   '   '   BeforeMethod
'   '   '   '   myTestMethod3
'   '   '   AfterMethod
'   '   '   BeforeMethod
'   '   '   '   myTestMethod4
'   '   '   AfterMethod
'   '   AfterClass
'   AfterTest
AfterSuite
```

Code Example: TestNG Annotations

```
class TestNGAnnotations {
    @BeforeSuite
    public void beforeSuite() {
        System.out.println("Before Suite");
    }
    @BeforeTest
    public void beforeTest() {
        System.out.println("Before Test");
    }
    @BeforeClass
    public void beforeClass() {
        System.out.println("Before Class");
    }
    @BeforeMethod
    public void beforeMethod() {
        System.out.println("Before Method");
    }
    @Test
    public void test1() {
        System.out.println("Test 1");
    }
    @Test
    public void test2() {
        System.out.println("Test 2");
    }
    @AfterMethod
    public void afterMethod() {
```

```
28          System.out.println("After Method");
29      }
30      @AfterClass
31      public void afterClass() {
32          System.out.println("After Class");
33      }
34      @AfterTest
35      public void afterTest() {
36          System.out.println("After Test");
37      }
38      @AfterSuite
39      public void afterSuite() {
40          System.out.println("After Suite");
41      }
42  }
```

Dependent Test

- `@Test(dependsOnMethods = {"methodName"} )`
- `@Test(dependsOnGroups = {"groupName"} )`

Code Example: Dependent Test - Method

```
class DependentTest {
    @Test
    public void login() {
        System.out.println("Login");
    }
    @Test(dependsOnMethods = {"login"})
    public void search() {
        System.out.println("Search");
    }
    @Test(dependsOnMethods = {"search"})
    public void logout() {
        System.out.println("Logout");
    }
}
```

Dependent Test - Group

```
class DependentTest {
    @Test(groups = {"login"})
    public void login() {
        System.out.println("Login");
    }
    @Test(groups = {"search"}, dependsOnGroups = {"l
    public void search() {
        System.out.println("Search");
    }
    @Test(dependsOnGroups = {"search"})
    public void logout() {
        System.out.println("Logout");
    }
}
```

Always Run Test

> If set to true, this test method will always be run even if it depends on a method that failed.

- **Syntax:** `@Test(alwaysRun = true)`

```java
class AlwaysRunTest {
    @Test
    public void login() {
        System.out.println("Login");
    }
    @Test(dependsOnMethods = {"login"})
    public void search() {
        int a = 10/0;
    }
    @Test(dependsOnMethods = {"search"}, alwaysRun = true)
    public void logout() {
        System.out.println("Logout");
    }
}
```

Exception Handling in TestNG

- `@Test(expectedExceptions = {Exception.class})`

```
class ExceptionHandling {
    @Test(expectedExceptions = {ArithmeticException.class})
    public void test() {
        int a = 10/0;
    }
}
```

Priority in TestNG

> The priority for this test method. Lower priorities will be scheduled first.

- **Syntax:** `@Test(priority = 1)`

```java
class PriorityTest {
    @Test(priority = 1)
    public void test1() {
        System.out.println("Test 1");
    }
    @Test(priority = 2)
    public void test2() {
        System.out.println("Test 2");
    }
}
```

Thank you ❤️

qa-june-2024-automation-with-java-slides