Article

Defining Test Cases and Test Methods

Add test cases and test methods to a test target to confirm that your code performs as expected.

Overview

Add tests to your Xcode project by writing one or more test methods, each of which verifies a specific aspect of your code. Group related test methods into test cases, each of which is a subclass of `XCTestCase`.

To add tests to your project:

- Create a new subclass of `XCTestCase` within a test target.
- Add one or more test methods to the test case.
- Add one or more test assertions to each test method.

A test method is an instance method on an `XCTestCase` subclass, with no parameters, no return value, and a name that begins with the lowercase word *test*. Test methods are automatically detected by the XCTest framework in Xcode.

Listing 1. Example test case and test method

Swift | Objective-C

```
class TableValidationTests: XCTestCase {
    /// Tests that a new table instance has zero rows and columns.
    func testEmptyTableRowAndColumnCount() {
        let table = Table()
        XCTAssertEqual(table.rowCount, 0, "Row count was not zero.")
        XCTAssertEqual(table.columnCount, 0, "Column count was not zero.")
```

```
    }
}
```

This example defines an XCTestCase subclass, TableValidationTests, with a single test method, testEmptyTableRowAndColumnCount(). This test method creates a new instance of a class called Table, and checks that its rowCount and columnCount properties are both equal to 0 after initialization.

> **Tip**
>
> Test case and test method names are used in Xcode's test navigator and integration reports to group and identify tests.
>
> To help clarify the organization of your tests, give each test case a name that summarizes the tests within it, such as TableValidationTests, NetworkReachabilityTests, or JSONParsingTests.
>
> To help identify failing tests, give each test method a name that makes it clear what is tested by that method, such as testEmptyTableRowAndColumnCount(), testUnreachableURLAccessThrowsAnError(), or testUserJSONFeedParsing().

Asserting Test Conditions

You can check (or *assert*) conditions inside test methods to make sure that your code is behaving as expected. Use the XCTAssert family of functions to check for Boolean conditions, nil or non-nil values, expected values, errors, or thrown exceptions.

For example, Listing 1 above uses the XCTAssertEqual macro to assert that two integers have the same value.

See Also

Test cases and test methods

`class` XCTestCase

The primary class for defining test cases, test methods, and performance tests.

`class` XCTest

An abstract base class for creating, managing, and executing tests.