Framework

XCTest

Create and run unit tests, performance tests, and UI tests for your Xcode project.

xcode 5.0+

Overview

Use the XCTest framework to write unit tests for your Xcode projects that integrate seamlessly with Xcode's testing workflow.

Tests assert that certain conditions are satisfied during code execution, and record test failures (with optional messages) if those conditions aren't satisfied. Tests can also measure the performance of blocks of code to check for performance regressions. Use XCTest in combination with XCUIAutomation to interact with an application's UI and validate user interaction flows. For more information, see Recording UI automation for testing.

Tip

Xcode 16 and later includes Swift Testing, a framework for writing unit tests that takes advantage of the powerful capabilities of the Swift programming language. Consider using Swift Testing for new unit test development and migrating existing tests as described in Migrating a test from XCTest. A test target can contain tests using both Swift Testing and XCTest, however don't mix API from the two frameworks in the same test. Continue to use XCTest for user interface tests and Performance Tests.

Topics

Test cases and test methods

Defining Test Cases and Test Methods

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

`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.

Test assertions

Boolean Assertions

Test a condition that generates a true or false result.

Nil and Non-Nil Assertions

Check whether a test condition has, or doesn’t have, a value.

Equality and Inequality Assertions

Check whether two values are equal or unequal.

Comparable Value Assertions

Compare two values to determine whether one is larger or smaller than the other.

Error Assertions

Check whether a function call throws, or doesn’t throw, an error.

NSException Assertions

Check whether a function call throws, or doesn’t throw, an exception.

Unconditional Test Failures

Generate a failure immediately and unconditionally.

Expected Failures

Anticipate known test failures to prevent failing tests from affecting your workflows.

Methods for Skipping Tests

Skip tests when meeting specified conditions.

Asynchronous tests

Asynchronous Tests and Expectations

Verify that asynchronous code behaves as expected.

UI tests

XCUIAutomation

Replicate sequences of interactions and make sure that your app's user interface behaves as intended.

Performance tests

Performance Tests

Gather metrics while running your code, and report a failure if the metrics become significantly worse than a baseline value.

Activities and attachments

Activities and Attachments

Split long tests into substeps with activities, and attach output data like files and screenshots.

Test execution

Test Execution and Observation

Observe, introspect, and customize the test execution flow.

Deprecated

Deprecated Symbols

These symbols are deprecated and are no longer recommended.

Variables

```
var XCT_UI_TESTING_AVAILABLE: Int32
```

Functions

```
func XCTAssertNoThrow<T>(@autoclosure () throws -> T, @autoclosure () -
> String, file: StaticString, line: UInt)
```

Asserts that an expression doesn’t throw an error.