UI Testing in Xcode

Session 406

Wil Turner Developer Tools
Brooke Callahan Developer Tools
Overview
Overview

UI testing
Overview

UI testing

• Find and interact with UI elements
Overview

UI testing

• Find and interact with UI elements
• Validate UI properties and state
Overview

UI testing
- Find and interact with UI elements
- Validate UI properties and state

UI recording
Overview

UI testing
- Find and interact with UI elements
- Validate UI properties and state

UI recording

Test reports
Core Technologies
Core Technologies

XCTest
Core Technologies

XCTest + Accessibility
XCTest
Xcode’s testing framework
XCTest

Xcode’s testing framework

Test case subclasses
XCTest
Xcode’s testing framework

Test case subclasses
Test methods
XCTest

Xcode’s testing framework

Test case subclasses
Test methods
Assertions
XCTest

Xcode’s testing framework

Test case subclasses
Test methods
Assertions
Integrated with Xcode
XCTest

Xcode’s testing framework

Test case subclasses
Test methods
Assertions
Integrated with Xcode
CI via Xcode Server and xcodebuild
XCTest
Xcode’s testing framework

Test case subclasses
Test methods
Assertions
Integrated with Xcode
CI via Xcode Server and xcodebuild
Swift and Objective-C
<table>
<thead>
<tr>
<th></th>
<th>Unit</th>
<th>UI</th>
</tr>
</thead>
<tbody>
<tr>
<td>Correctness</td>
<td>✔</td>
<td></td>
</tr>
<tr>
<td>Performance</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
# Testing Matrix

<table>
<thead>
<tr>
<th></th>
<th>Unit</th>
<th>UI</th>
</tr>
</thead>
<tbody>
<tr>
<td>Correctness</td>
<td>✔</td>
<td></td>
</tr>
<tr>
<td>Performance</td>
<td>✔</td>
<td></td>
</tr>
</tbody>
</table>
# Testing Matrix

<table>
<thead>
<tr>
<th></th>
<th>Unit</th>
<th>UI</th>
</tr>
</thead>
<tbody>
<tr>
<td>Correctness</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Performance</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>
Core Technologies

XCTest + Accessibility
Accessibility

Rich semantic data about UI
Accessibility

Rich semantic data about UI
UIKit and AppKit integration
Accessibility

Rich semantic data about UI
UIKit and AppKit integration
APIs for fine tuning
Accessibility

Rich semantic data about UI
UIKit and AppKit integration
APIs for fine tuning
UI tests interact with the app the way a user does
Requirements
UI testing depends on new OS features
Requirements

UI testing depends on new OS features

- iOS 9
Requirements

UI testing depends on new OS features
• iOS 9
• OS X 10.11
Requirements

UI testing depends on new OS features

• iOS 9
• OS X 10.11

Privacy protection
Requirements

UI testing depends on new OS features

• iOS 9
• OS X 10.11

Privacy protection

• iOS devices
Requirements

UI testing depends on new OS features
• iOS 9
• OS X 10.11

Privacy protection
• iOS devices
  - Enabled for development
Requirements

UI testing depends on new OS features
• iOS 9
• OS X 10.11

Privacy protection
• iOS devices
  - Enabled for development
  - Connected to a trusted host running Xcode
Requirements

UI testing depends on new OS features

- iOS 9
- OS X 10.11

Privacy protection

- iOS devices
  - Enabled for development
  - Connected to a trusted host running Xcode
- OS X must grant permission to Xcode Helper
Requirements

UI testing depends on new OS features
- iOS 9
- OS X 10.11

Privacy protection
- iOS devices
  - Enabled for development
  - Connected to a trusted host running Xcode
- OS X must grant permission to Xcode Helper
  - Prompted on first run
Getting Started
Getting Started

Xcode target type
Getting Started

Xcode target type
Getting Started

Xcode target type
APIs
Getting Started

Xcode target type
APIs
UI recording
UI Testing Xcode Targets
UI Testing Xcode Targets

UI tests have special requirements
UI Testing Xcode Targets

UI tests have special requirements

- Execute in a separate process
UI Testing Xcode Targets

UI tests have special requirements

• Execute in a separate process
• Permission to use Accessibility
UI Testing Xcode Targets

UI tests have special requirements

- Execute in a separate process
- Permission to use Accessibility

New Xcode target templates
UI Testing Xcode Targets

UI tests have special requirements
- Execute in a separate process
- Permission to use Accessibility

New Xcode target templates
- Cocoa Touch UI Testing Bundle (iOS)
- Cocoa UI Testing Bundle (OS X)
UI Testing Xcode Targets

UI tests have special requirements
• Execute in a separate process
• Permission to use Accessibility

New Xcode target templates
• Cocoa Touch UI Testing Bundle (iOS)
• Cocoa UI Testing Bundle (OS X)

“Target to be Tested” setting
APIs
APIs

Three new classes
APIs

Three new classes

• XCUIApplication
APIs

Three new classes

- XCUIApplication
- XCUIElement
APIs

Three new classes

- XCUIApplication
- XCUIElement
- XCUIElementQuery
UI Recording
UI Recording

Interact with your app
UI Recording

Interact with your app
Recording generates the code
UI Recording

Interact with your app
Recording generates the code
• Create new tests
UI Recording

Interact with your app

Recording generates the code

- Create new tests
- Expand existing tests
Demo

Getting started with UI testing
What Did You See?
What Did You See?

Adding a UI testing target
What Did You See?

Adding a UI testing target
Using recording
What Did You See?

Adding a UI testing target

Using recording

• Finding UI elements
What Did You See?

Adding a UI testing target

Using recording

• Finding UI elements
• Synthesizing user events
What Did You See?

Adding a UI testing target
Using recording
  • Finding UI elements
  • Synthesizing user events
Adding validation with XCTAssert
UI Testing API
UI Testing API
UI Testing API

XCUIApplication

XCUlApplication
UI Testing API

- XCUIApplication
- XCUIElement
UI Testing API

- XCUIApplication
- XCUIElement
- XCUIElementQuery
Example
Testing the Add button

// application:
let app = XCUIApplication()
app.launch()

// element and query:
let addButton = app.buttons[“Add”]
addButton.tap()

// assertion:
XCTAssertEqual(app.tables.cells.count, 1)
Example
Testing the Add button

// application:
let app = XCUIApplication()
app.launch()

// element and query:
let addButton = app.buttons[“Add”]
addButton.tap()

// assertion:
XCTAssertEqual(app.tables.cells.count, 1)
Example

Testing the Add button

// application:
let app = XCUIApplication()
app.launch()

// element and query:
let addButton = app.buttons[“Add”]
addButton.tap()

// assertion:
XCTAssertEqual(app.tables.cells.count, 1)
Example

Testing the Add button

```swift
// application:
let app = XCUIApplication()
app.launch()

// element and query:
let addButton = app.buttons[“Add”]
addButton.tap()

// assertion:
XCTAssertEqual(app.tables.cells.count, 1)
```
Example

Testing the Add button

```
// application:
let app = XCUIApplication()
app.launch()

// element and query:
let addButton = app.buttons["Add"]
addButton.tap()

// assertion:
XCTAssertEqual(app.tables.cells.count, 1)
```
Example

Testing the Add button

// application:
let app = XCUIApplication()
app.launch()

// element and query:
let addButton = app.buttons[“Add”]
addButton.tap()

// assertion:
XCTAssertEqual(app.tables.cells.count, 1)
Example

Testing the Add button

```swift
// application:
let app = XCUIApplication()
app.launch()

// element and query:
let addButton = app.buttons["Add"]
addButton.tap()

// assertion:
XCTAssertEqual(app.tables.cells.count, 1)
```
Example

Testing the Add button

```swift
// application:
let app = XCUIApplication()
app.launch()

// element and query:
let addButton = app.buttons["Add"]
addButton.tap()

// assertion:
XCTAssertEqual(app.tables.cells.count, 1)
```
Example

Testing the Add button

// application:
let app = XCUIApplication()
app.launch()

// element and query:
let addButton = app.buttons[“Add”]
addButton.tap()

// assertion:
XCTAssertEqual(app.tables.cells.count, 1)
XCUIApplication

Proxy for the tested application
XCUIApplication

Proxy for the tested application

- Tests run in a separate process
XCUIApplication

Proxy for the tested application

- Tests run in a separate process

Launch
XCUIApplication

Proxy for the tested application

- Tests run in a separate process

Launch

- Always spawns a new process
XCUIApplication

Proxy for the tested application

• Tests run in a separate process

Launch

• Always spawns a new process
• Implicitly terminates any preexisting instance
XCUIApplication

Proxy for the tested application
- Tests run in a separate process

Launch
- Always spawns a new process
- Implicitly terminates any preexisting instance

Starting point for finding elements
XCUIElement
XCUIElement

Proxy for elements in application
XCUIElement

Proxy for elements in application
Types
XCUIElement

Proxy for elements in application

Types

• Button, Cell, Window, etc.
XCUIElement

Proxy for elements in application

Types

• Button, Cell, Window, etc.

Identifiers
Proxy for elements in application

Types

• Button, Cell, Window, etc.

Identifiers

• Accessibility identifier, label, title, etc.
XCUIElement

Proxy for elements in application

Types
• Button, Cell, Window, etc.

Identifiers
• Accessibility identifier, label, title, etc.

Most elements are found by combining type and identifier
Element Hierarchy
Element Hierarchy

Application is the root of a tree of elements
Element Hierarchy

Application is the root of a tree of elements
Element Hierarchy

Application is the root of a tree of elements
Element Hierarchy

Application is the root of a tree of elements
Element Hierarchy

Application is the root of a tree of elements
Element Hierarchy

Application is the root of a tree of elements
Used by queries with type and identifiers
Element Uniqueness
Element Uniqueness

Every XCUIElement is backed by a query
Element Uniqueness

Every XCUIElement is backed by a query
Query must resolve to exactly one match
Element Uniqueness

Every XCUIElement is backed by a query
Query must resolve to exactly one match
• No matches or multiple matches cause test failure
Element Uniqueness

Every XCTestElement is backed by a query

Query must resolve to exactly one match

• No matches or multiple matches cause test failure
• Failure raised when element resolves query
Element Uniqueness

Every XCUITarget is backed by a query

Query must resolve to exactly one match

• No matches or multiple matches cause test failure
• Failure raised when element resolves query

Exception
Element Uniqueness

Every XCUIElement is backed by a query

Query must resolve to exactly one match

- No matches or multiple matches cause test failure
- Failure raised when element resolves query

Exception

- `exists` property
Event Synthesis
Event Synthesis

Simulate user interaction on elements
Event Synthesis

Simulate user interaction on elements
APIs are platform-specific
Event Synthesis

Simulate user interaction on elements

APIs are platform-specific

button.click() // OS X
Event Synthesis

Simulate user interaction on elements

APIs are platform-specific

button.click() // OS X
button.tap() // iOS
Event Synthesis

Simulate user interaction on elements

APIs are platform-specific

button.click() // OS X
button.tap() // iOS
textField.typeText(“Hello, World!”) // iOS & OS X
XCUIElementQuery
API for specifying elements
XCUIElementQuery
API for specifying elements

Queries resolve to collections of accessible elements
XCUUIElemeentQuery
API for specifying elements

Queries resolve to collections of accessible elements

• Number of matches: count
XCUIElementQuery
API for specifying elements

Queries resolve to collections of accessible elements

- Number of matches: `count`
- Specify by identifier: subscripting
XCUIElementQuery

API for specifying elements

Queries resolve to collections of accessible elements

- Number of matches: `count`
- Specify by identifier: subscripting
- Specify by index: `elementAtIndex()`
How do queries work?
XCUIElementQuery
How do queries work?

Relationships
XCUUIElementQuery

How do queries work?

Relationships
Filtering
Expressing relationships

Application
  \- Navigation Bar
    \- Title Label
      \- “Add” Button
  \- View
    \- Table
      \- Cell
        \- “Groceries” Label
      \- Cell
        \- “Tech Toys” Label
      \- Cell
        \- “Today” Label
XCUIElementQuery

Expressing relationships

Descendants

```
<table>
<thead>
<tr>
<th>Lister</th>
</tr>
</thead>
</table>
| Groceries
| Tech Toys
| Today |
```

```
<table>
<thead>
<tr>
<th>Application</th>
</tr>
</thead>
<tbody>
<tr>
<td>Navigation Bar</td>
</tr>
<tr>
<td>Title Label</td>
</tr>
<tr>
<td>&quot;Add&quot; Button</td>
</tr>
</tbody>
</table>

```
<table>
<thead>
<tr>
<th>View</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table</td>
</tr>
</tbody>
</table>

```
<table>
<thead>
<tr>
<th>Cell</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Groceries&quot; Label</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Cell</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Tech Toys&quot; Label</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Cell</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Today&quot; Label</td>
</tr>
</tbody>
</table>
```
XCUIElementTypeQuery

Expressing relationships

Descendants

Children
XCUIElementQuery

Expressing relationships

Descendants

Children
XCUIElementQuery

Expressing relationships

Descendants
Children
Containment
XCUIClientQuery

Expressing relationships

Descendants
Children
Containment

Application
  Navigation Bar
    Title Label
    “Add” Button
  View
    Table
      Cell
        “Groceries” Label
      Cell
        “Tech Toys” Label
      Cell
        “Today” Label
XCUIElementQuery
Filtering
XCUUIElementQuery

Filtering

Element type

• Button, table, menu, etc.
XCUUIElementQuery

Filtering

Element type

• Button, table, menu, etc.

Identifiers

• Accessibility identifier, label, title, etc.
Filtering

Element type
• Button, table, menu, etc.

Identifiers
• Accessibility identifier, label, title, etc.

Predicates
• Value, partial matching, etc.
Combining Relationships and Filtering

descendantsMatchingType()
Combining Relationships and Filtering
descendantsMatchingType()

Most common query
Combining Relationships and Filtering

descendantsMatchingType()

Most common query

let allButtons = app.descendantsMatchingType(.Button)
Combining Relationships and Filtering
descendantsMatchingType()

Most common query

let allButtons = app.descendantsMatchingType(.Button)

let allCellsInTable = table.descendantsMatchingType(.Cell)
Combining Relationships and Filtering
descendantsMatchingType()

Most common query

```swift
let allButtons = app.descendantsMatchingType(.Button)

let allCellsInTable = table.descendantsMatchingType(.Cell)

let allMenuItemsInMenu = menu.descendantsMatchingType(.MenuItem)
```
Combining Relationships and Filtering
descendantsMatchingType()

So common, we provide convenience API for each type

let allButtons = app.descendantsMatchingType(.Button)

let allCellsInTable = table.descendantsMatchingType(.Cell)

let allMenuItemsInMenu = menu.descendantsMatchingType(.MenuItem)
Combining Relationships and Filtering
descendantsMatchingType()

So common, we provide convenience API for each type

```swift
let allButtons = app.descendantsMatchingType(.Button)
let allCellsInTable = table.descendantsMatchingType(.Cell)
let allMenuItemsInMenu = menu.descendantsMatchingType(.MenuItem)
```
Combining Relationships and Filtering

descendantsMatchingType()

So common, we provide convenience API for each type

```swift
let allButtons = app.buttons

let allCellsInTable = table.descendantsMatchingType(.Cell)

let allMenuItemsInMenu = menu.descendantsMatchingType(.MenuItem)
```
Combining Relationships and Filtering
descendantsMatchingType()

So common, we provide convenience API for each type

```swift
let allButtons = app.buttons

let allCellsInTable = table.descendantsMatchingType(.Cell)

let allMenuItemsInMenu = menu.descendantsMatchingType(.MenuItem)
```
Combining Relationships and Filtering
descendantsMatchingType()

So common, we provide convenience API for each type

let allButtons = app.buttons

let allCellsInTable = table.cells

let allMenuItemsInMenu = menu.descendantsMatchingType(.MenuItem)
Combining Relationships and Filtering

descendantsMatchingType()

So common, we provide convenience API for each type

```swift
let allButtons = app.buttons

let allCellsInTable = table.cells

let allMenuItemsInMenu = menu.descendantsMatchingType(.MenuItem)
```
Combining Relationships and Filtering
descendantsMatchingType()

So common, we provide convenience API for each type

let allButtons = app.buttons

let allCellsInTable = table.cells

let allMenuItemsInMenu = menu.menuItems
Combining Relationships and Filtering

childrenMatchingType()
Combining Relationships and Filtering

`childrenMatchingType()`

Differentiates between any descendant and a direct child relationship.
Combining Relationships and Filtering

callrenMatchingType()

Differentiates between any descendant and a direct child relationship

let allButtons = app.buttons // descendantsMatchingType(.Button)
Combining Relationships and Filtering

childrenMatchingType()

Differentiates between any descendant and a direct child relationship

```swift
let allButtons = app.buttons // descendantsMatchingType(.Button)

let childButtons = navBar.childrenMatchingType(.Button)
```
Combining Relationships and Filtering

containingType()
Combining Relationships and Filtering

containingType()

Find elements by describing their descendants
Combining Relationships and Filtering

containingType()

Find elements by describing their descendants

```
<table>
<thead>
<tr>
<th>Table</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cell</td>
</tr>
<tr>
<td>Cell</td>
</tr>
<tr>
<td>“Groceries” Label</td>
</tr>
<tr>
<td>Cell</td>
</tr>
<tr>
<td>“Tech Toys” Label</td>
</tr>
<tr>
<td>Cell</td>
</tr>
<tr>
<td>“Today” Label</td>
</tr>
</tbody>
</table>
```
Combining Relationships and Filtering

containingType()

Find elements by describing their descendants
Combining Relationships and Filtering

containingType()

Find elements by describing their descendants

let cellQuery = cells.containingType(.StaticText, identifier:"Groceries")
Combining Relationships and Filtering

Find elements by describing their descendants

let cellQuery = cells.containingType(.StaticText, identifier: "Groceries")

Predicate variant also available
XCUICollectionViewQuery
Combining relationships and filtering

descendantsMatchingType()
childrenMatchingType()
containingType()
Combining Queries
Combining Queries

Queries can be “chained” together
Combining Queries

Queries can be “chained” together

Output of each query is the input of the next query
Combining Queries

Queries can be “chained” together
Output of each query is the input of the next query
Combining Queries

Queries can be “chained” together
Output of each query is the input of the next query

```
let labelsInTable = app
```
Combining Queries

Queries can be “chained” together
Output of each query is the input of the next query

let `labelsInTable = app.tables`
Combining Queries

Queries can be “chained” together
Output of each query is the input of the next query

let labelsInTable = app.tables.staticTexts
Getting Elements from Queries
## Getting Elements from Queries

<table>
<thead>
<tr>
<th>Subscripting</th>
<th><code>table.staticTexts[“Groceries”]</code></th>
</tr>
</thead>
</table>

This section discusses methods for accessing specific elements from queries, with a particular focus on subscripting to retrieve values from a `table` object's `staticTexts` array.
## Getting Elements from Queries

<table>
<thead>
<tr>
<th>Method</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subscripting</td>
<td><code>table.staticTexts[&quot;Groceries&quot;]</code></td>
</tr>
<tr>
<td>Index</td>
<td><code>table.staticTexts.elementAtIndex(0)</code></td>
</tr>
</tbody>
</table>
## Getting Elements from Queries

<table>
<thead>
<tr>
<th>Subscripting</th>
<th><code>table.staticTexts[“Groceries”]</code></th>
</tr>
</thead>
<tbody>
<tr>
<td>Index</td>
<td><code>table.staticTexts.elementAtIndex(0)</code></td>
</tr>
<tr>
<td>Unique</td>
<td><code>app.navigationBar.element</code></td>
</tr>
</tbody>
</table>
Evaluating Queries
Evaluating Queries

Queries are evaluated on demand
Evaluating Queries

Queries are evaluated on demand

XCUInitial
• Synthesizing events
• Reading property values
Evaluating Queries

Queries are evaluated on demand

XCUIElement
  • Synthesizing events
  • Reading property values

XCUIElementQuery
  • Getting number of matches (.count)
  • Getting all matches (.allElementsBoundByAccessibilityElement)
Evaluating Queries

Queries are evaluated on demand

**XCUIElement**
- Synthesizing events
- Reading property values

**XCUIElementQuery**
- Getting number of matches (.count)
- Getting all matches (.allElementsBoundByAccessibilityElement)

Re-evaluated when UI changes
Queries and Elements

Similar to URLs
Queries and Elements

Similar to URLs

Creating a URL does not fetch a resource
Queries and Elements

Similar to URLs

Creating a URL does not fetch a resource

- URL could be invalid, error raised when requested
Queries and Elements
Similar to URLs

Creating a URL does not fetch a resource
• URL could be invalid, error raised when requested

Queries and elements
Queries and Elements

Similar to URLs

Creating a URL does not fetch a resource

- URL could be invalid, error raised when requested

Queries and elements

- Just a specification for accessible elements in the tested application
Queries and Elements

Similar to URLs

Creating a URL does not fetch a resource
• URL could be invalid, error raised when requested

Queries and elements
• Just a specification for accessible elements in the tested application
• Not resolved until needed
API Recap

XCUIApplication

XCUIElement

XCUIElementQuery
Accessibility and UI Testing
Accessibility and UI Testing
Accessibility and UI Testing

Accessibility data makes UI testing possible
Accessibility and UI Testing

Accessibility data makes UI testing possible
Accessibility and UI Testing

Debugging tips
Accessibility and UI Testing

Debugging tips

Not accessible
Accessibility and UI Testing

Debugging tips

Not accessible

• Custom view subclasses
Accessibility and UI Testing

Debugging tips

Not accessible

- Custom view subclasses
- Layers, sprites, and other graphics objects
Accessibility and UI Testing

Debugging tips

Not accessible
• Custom view subclasses
• Layers, sprites, and other graphics objects

Poor accessibility data
Accessibility and UI Testing

Debugging tips

Not accessible
• Custom view subclasses
• Layers, sprites, and other graphics objects

Poor accessibility data

Tools
Accessibility and UI Testing

Debugging tips

Not accessible
• Custom view subclasses
• Layers, sprites, and other graphics objects

Poor accessibility data

Tools
• UI recording
Accessibility and UI Testing

Debugging tips

Not accessible
• Custom view subclasses
• Layers, sprites, and other graphics objects

Poor accessibility data

Tools
• UI recording
• Accessibility inspectors
Accessibility and UI Testing

Improving data
Accessibility and UI Testing

Improving data

Interface Builder inspector
Accessibility and UI Testing

Improving data

Interface Builder inspector
Accessibility and UI Testing

Improving data

Interface Builder inspector

API
Accessibility and UI Testing

Improving data

Interface Builder inspector

API

- UIAccessibility (iOS)
- NSAccessibility (OS X)
Demo
What Did You See?
What Did You See?

Advanced UI testing
What Did You See?

Advanced UI testing
Correcting queries
What Did You See?

Advanced UI testing
Correcting queries
Looping over elements
What Did You See?

Advanced UI testing
Correcting queries
Looping over elements
Improving accessibility
Test Reports

UI Refresh
# Test Reports

Show results for all tests

<table>
<thead>
<tr>
<th>Tests</th>
<th>Status</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>RuleSystemTests → DemoBotsUnitTests (iOS)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>testDoEventsOnFail()</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>testDoEventsOnFail()</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>testGoodRule()</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>testLevelStateSnapshot()</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>testPercentageHighRule()</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>testPercentageLowRule()</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>testThreatsAggMediumRule()</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>testThreatsAggFarRule()</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>testPlayerBotMediumRule()</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>testPlayerBotFarRule()</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>testThreats()</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>testGoodBotMediumRule()</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>testGoodBotFarRule()</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>SceneTests → DemoBotsUnitTests (iOS)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>testLoadScene()</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>testGameplayConfiguration()</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>testLevelConfiguration()</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>testScoreAIICoding()</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>testSceneMetadata()</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>testSceneResourcesLoading()</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>testScoreOverlay()</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>StackTests → DemoBotsUnitTests (iOS)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>testBeamCasingState()</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>testBeamFrangibleState()</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>testBeamIndeterminateState()</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>testBeamPerfectState()</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>testScoreOverlayState()</td>
<td>✓</td>
<td></td>
</tr>
</tbody>
</table>
Test Reports

Show results for all tests

- Pass/fail
Test Reports

Show results for all tests

- Pass/fail
- Failure reason
Test Reports

Show results for all tests

• Pass/fail
• Failure reason
• Performance metrics
Test Reports

Show results for all tests
- Pass/fail
- Failure reason
- Performance metrics

Same UI in Xcode and in Xcode Server
Test Reports

Show results for all tests
  • Pass/fail
  • Failure reason
  • Performance metrics

Same UI in Xcode and in Xcode Server
Per-device results for Xcode Server
Test Reports

UI testing additions
Test Reports

UI testing additions

New data
Test Reports

UI testing additions

New data

Screenshots
Test Reports

UI testing additions

New data
Screenshots
Nested activities
Test Reports

Nested activities

testPlayerNameChange()

- Wait for app to idle (Start)
- Tap the "Options" Button (1.00s)
- Tap the "PlayerName" TextField (3.00s)
- Type 'HAL 9000' into the "PlayerName" TextField (6.00s)
  - Wait for app to idle (3.00s)
  - Find the "PlayerName" TextField (3.00s)
  - Dispatch the event (5.00s)
  - Wait for app to idle (6.00s)
- Type " " into the "PlayerName" TextField (7.00s)
  - Wait for app to idle (6.00s)
  - Find the "PlayerName" TextField (6.00s)
  - Dispatch the event (6.00s)
  - Wait for app to idle (7.00s)
  - Find the "PlayerName" TextField (7.00s)
  - Tap the "Done." Button (8.00s)

Assertion Failure: failed - Expected player name to successfully change, value is still "HAL 9000"
# Test Reports

**Nested activities**

UI testing APIs have several steps:

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Wait for app to idle</td>
</tr>
<tr>
<td>2</td>
<td>Tap the &quot;Options&quot; Button (1.00s)</td>
</tr>
<tr>
<td>3</td>
<td>Tap the &quot;PlayerName&quot; TextField (3.00s)</td>
</tr>
<tr>
<td>4</td>
<td>Type &quot;HAL 9000&quot; into the &quot;PlayerName&quot; TextField (6.00s)</td>
</tr>
<tr>
<td>5</td>
<td>Wait for app to idle (3.00s)</td>
</tr>
<tr>
<td>6</td>
<td>Find the &quot;PlayerName&quot; TextField (3.00s)</td>
</tr>
<tr>
<td>7</td>
<td>Dispatch the event (5.00s)</td>
</tr>
<tr>
<td>8</td>
<td>Wait for app to idle (6.00s)</td>
</tr>
<tr>
<td>9</td>
<td>Type &quot; &quot; into the &quot;PlayerName&quot; TextField (7.00s)</td>
</tr>
<tr>
<td>10</td>
<td>Wait for app to idle (6.00s)</td>
</tr>
<tr>
<td>11</td>
<td>Find the &quot;PlayerName&quot; TextField (6.00s)</td>
</tr>
<tr>
<td>12</td>
<td>Dispatch the event (6.00s)</td>
</tr>
<tr>
<td>13</td>
<td>Wait for app to idle (7.00s)</td>
</tr>
<tr>
<td>14</td>
<td>Find the &quot;PlayerName&quot; TextField (7.00s)</td>
</tr>
<tr>
<td>15</td>
<td>Tap the &quot;Done.&quot; Button (8.00s)</td>
</tr>
</tbody>
</table>

**Assertion Failure:** failed - Expected player name to successfully change, value is still "HAL 9000"
Test Reports

Nested activities

UI testing APIs have several steps

Typing into a textfield

```
@(obsessive) testPlayerNameChange()

  Wait for app to idle (Start)
  ▶ Tap the "Options" Button (1.00s)
  ▶ Tap the "PlayerName" TextField (3.00s)
  ▶ Type 'HAL 9000' into the "PlayerName" TextField (6.00s)
    Wait for app to idle (3.00s)
    Find the "PlayerName" TextField (3.00s)
    Dispatch the event (5.00s)
    Wait for app to idle (6.00s)
  ▶ Type '" into the "PlayerName" TextField (7.00s)
    Wait for app to idle (6.00s)
    Find the "PlayerName" TextField (6.00s)
    Dispatch the event (6.00s)
    Wait for app to idle (7.00s)
    Find the "PlayerName" TextField (7.00s)
  ▶ Tap the "Done." Button (8.00s)

Assertion Failure: failed - Expected player name to successfully change, value is still "HAL 9000"
```
Test Reports

Nested activities

UI testing APIs have several steps
Typing into a textfield
• Wait for the app to idle

```
testPlayerNameChange()
  Wait for app to idle (Start)
  ▶ Tap the "Options" Button (1.00s)
  ▶ Tap the "FieldName" TextField (3.00s)
  ▶ Type 'HAL 9000' into the "FieldName" TextField (6.00s)
    Wait for app to idle (3.00s)
    Find the "FieldName" TextField (3.00s)
    Dispatch the event (5.00s)
    Wait for app to idle (6.00s)
  ▶ Type " " into the "FieldName" TextField (7.00s)
    Wait for app to idle (6.00s)
    Find the "FieldName" TextField (6.00s)
    Dispatch the event (6.00s)
    Wait for app to idle (7.00s)
    Find the "FieldName" TextField (7.00s)
    ▶ Tap the "Done." Button (8.00s)

Assertion Failure: failed - Expected player name to successfully change, value is still "HAL 9000"
```
Test Reports

Nested activities

UI testing APIs have several steps:

Typing into a textfield:
- Wait for the app to idle
- Evaluate the textfield query

```
testPlayerNameChange()
  Wait for app to idle (Start)
  ▶ Tap the "Options" Button (1.00s)
  ▶ Tap the "PlayerName" TextField (3.00s)
  ▶ Type 'HAL 9000' into the "PlayerName" TextField (6.00s)
    Wait for app to idle (3.00s)
    Find the "PlayerName" TextField (3.00s)
    Dispatch the event (5.00s)
    Wait for app to idle (6.00s)
  ▶ Type " into the "PlayerName" TextField (7.00s)
    Wait for app to idle (6.00s)
    Find the "PlayerName" TextField (6.00s)
    Dispatch the event (6.00s)
    Wait for app to idle (7.00s)
    Find the "PlayerName" TextField (7.00s)
  ▶ Tap the "Done." Button (8.00s)

Assertion Failure: failed - Expected player name to successfully change, value is still "HAL 9000"
```
UI testing APIs have several steps

Typing into a textfield

- Wait for the app to idle
- Evaluate the textfield query
- Synthesize the text input

Assertion Failure: failed - Expected player name to successfully change, value is still "HAL 9000"
Test Reports

Nested activities

UI testing APIs have several steps:

1. Typing into a textfield
2. Wait for the app to idle
3. Evaluate the textfield query
4. Synthesize the text input
5. Wait for the app to idle

Assertion Failure: failed - Expected player name to successfully change, value is still "HAL 9000"
UI testing APIs have several steps:

1. Typing into a textfield:
   - Wait for the app to idle
   - Evaluate the textfield query
   - Synthesize the text input
   - Wait for the app to idle

2. QuickLook for screenshots

Test Reports

Nested activities

```
- testPlayerNameChange()
  - Wait for app to idle (Start)
  - Tap the "Options" Button (1.00s)
  - Tap the "PlayerName" TextField (3.00s)
  - Type "HAL 9000" into the "PlayerName" TextField (6.00s)
    - Wait for app to idle (3.00s)
    - Find the "PlayerName" TextField (3.00s)
    - Dispatch the event (5.00s)
  - Wait for app to idle (6.00s)
  - Type " " into the "PlayerName" TextField (7.00s)
    - Wait for app to idle (6.00s)
    - Find the "PlayerName" TextField (6.00s)
    - Dispatch the event (6.00s)
    - Wait for app to idle (7.00s)
    - Find the "PlayerName" TextField (7.00s)
  - Tap the "Done." Button (8.00s)

Assertion Failure: failed - Expected player name to successfully change, value is still "HAL 9000"
```
Test Reports

Nested activities

UI testing APIs have several steps:

• Typing into a textfield
• Wait for the app to idle
• Evaluate the textfield query
• Synthesize the text input
• Wait for the app to idle

QuickLook for screenshots

[Image: Game Options screen with player name set to HAL 9000]
When to Use UI Testing
Using UI Testing
Using UI Testing

Complements unit testing
Using UI Testing

Complements unit testing

Unit testing more precisely pinpoints failures
Using UI Testing

Complements unit testing

Unit testing more precisely pinpoints failures

UI testing covers broader aspects of functionality
Using UI Testing

Complements unit testing
Unit testing more precisely pinpoints failures
UI testing covers broader aspects of functionality
Find the right blend of UI tests and unit tests for your project
Candidates for UI Testing
Candidates for UI Testing

Demo sequences
Candidates for UI Testing

Demo sequences
Common workflows
Candidates for UI Testing

Demo sequences
Common workflows
Custom views
Candidates for UI Testing

Demo sequences
Common workflows
Custom views
Document creation, saving, and opening
Summary
Summary

UI testing
Summary

UI testing

- Find and interact with UI elements
Summary

UI testing

• Find and interact with UI elements
• Validate UI properties and state
Summary

UI testing
- Find and interact with UI elements
- Validate UI properties and state

UI recording
Summary

UI testing
- Find and interact with UI elements
- Validate UI properties and state

UI recording

Test reports
More Information

Testing in Xcode Documentation
http://developer.apple.com/testing

Accessibility for Developers Documentation
http://developer.apple.com/accessibility

Apple Developer Forums
http://developer.apple.com/forums

Stefan Lesser
Developer Tools Evangelist
slesser@apple.com
# Related Sessions

<table>
<thead>
<tr>
<th>Session</th>
<th>Location</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>iOS Accessibility</td>
<td>Pacific Heights</td>
<td>Tuesday 9:00 AM</td>
</tr>
<tr>
<td>Continuous Integration and Code Coverage in Xcode</td>
<td>Presidio</td>
<td>Thursday 10:00 AM</td>
</tr>
<tr>
<td>Testing and Continuous Integration</td>
<td>Developer Tools Lab B</td>
<td>Wednesday 1:30 PM</td>
</tr>
<tr>
<td>-----------------------------------</td>
<td>------------------------</td>
<td>-------------------</td>
</tr>
<tr>
<td>Testing and Continuous Integration</td>
<td>Developer Tools Lab B</td>
<td>Thursday 1:30 PM</td>
</tr>
</tbody>
</table>