Testing in Xcode 5

Session 409

Mike Swingler
Xcode Engineer

Why Test?

Why Test?

- Catch crashes

Why Test?

- Catch crashes
- Catch logic errors

Why Test?

- Catch crashes
- Catch logic errors
- Help you write your code

Why Test?

- Catch crashes
- Catch logic errors
- Help you write your code
- Catch regressions

Why Test?

- Catch crashes
- Catch logic errors
- Help you write your code
- Catch regressions
- Cover more configurations

Why Test?

- Catch crashes
- Catch logic errors
- Help you write your code
- Catch regressions
- Cover more configurations
- Cover everyone, all the time

Overview

Overview

- What is a unit test?

Overview

- What is a unit test?
- Introducing XCTest

Overview

- What is a unit test?
- Introducing XCTest
- Writing, running and debugging tests

Overview

- What is a unit test?
- Introducing XCTest
- Writing, running and debugging tests
- Continuous integration

Overview

- What is a unit test?
- Introducing XCTest
- Writing, running and debugging tests
- Continuous integration
- Advanced setups

Unit Testing

What Is a Unit Test?

- Tests **ONE** thing
  - Single “unit” of functionality
  - Pass/Fail
  - Small, fast, isolated

What is a Unit Test?

- Tests **ONE** thing
  - Single “unit” of functionality
  - Pass/Fail
  - Small, fast, isolated
- Unit tests don’t cover
  - Performance

What is a Unit Test?

- Tests **ONE** thing
  - Single “unit” of functionality
  - Pass/Fail
  - Small, fast, isolated
- Unit tests don’t cover
  - Performance
  - UI interaction

What is a Unit Test?

- Tests **ONE** thing
  - Single “unit” of functionality
  - Pass/Fail
  - Small, fast, isolated
- Unit tests don’t cover
  - Performance
  - UI interaction
  - Whole system integration

Your App

Where to start testing

Your App

Where to start testing

Database

View

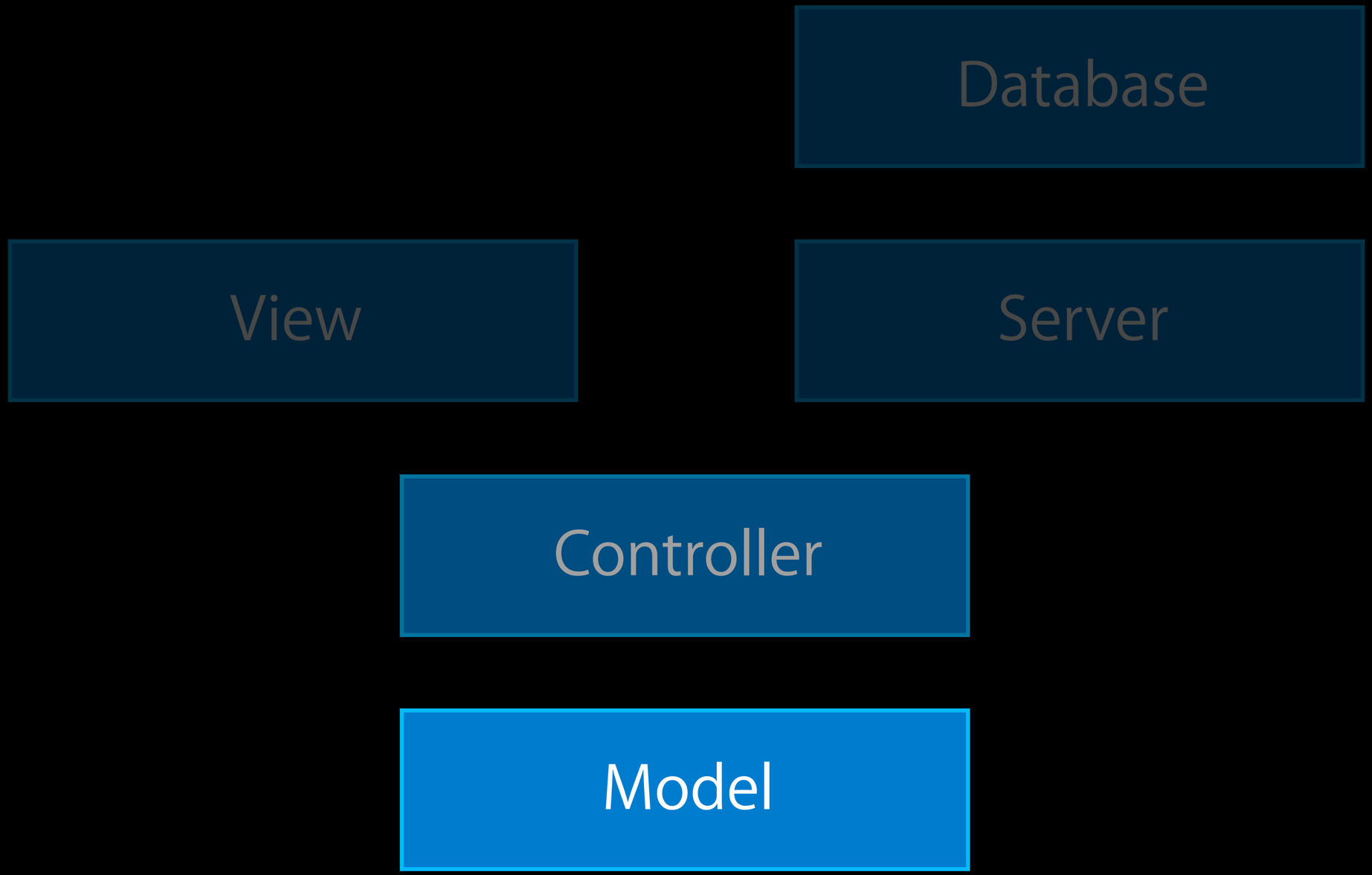
Server

Controller

Model

Your App

Where to start testing



Introducing XCTest

XCTest.framework

- iOS and OS X

XCTest.framework

- iOS and OS X
- Requires Xcode 5

XCTest.framework

- iOS and OS X
- Requires Xcode 5
- Derived from OCUnit
  - Modernized
  - Migration tool

XCTest.framework

- iOS and OS X
- Requires Xcode 5
- Derived from OCUnit
  - Modernized
  - Migration tool
- Builds .xctest bundles

XCTest.framework

NEW

- iOS and OS X
- Requires Xcode 5
- Derived from OCUnit
  - Modernized
  - Migration tool
- Builds .xctest bundles
- Test runner injected in your app

XCTest.framework

- iOS and OS X
- Requires Xcode 5
- Derived from OCUnit
  - Modernized
  - Migration tool
- Builds .xctest bundles
- Test runner injected in your app
- Test rig loads libraries and test bundles

Test Code

Test Code

```
@interface ExampleTests : XCTestCase
@end

@implementation ExampleTests

- (void) testExample {
    XCTAssertTrue(2 + 2 == 4);
}

@end
```

Test Code

- Subclass `XCTestCase`

```
@interface ExampleTests : XCTestCase
@end

@implementation ExampleTests

- (void) testExample {
    XCTAssertTrue(2 + 2 == 4);
}

@end
```

Test Code

- Subclass `XCTestCase`
- test = method

```
@interface ExampleTests : XCTestCase
@end

@implementation ExampleTests

- (void) testExample {
    XCTAssertTrue(2 + 2 == 4);
}

@end
```

Test Code

- Subclass `XCTestCase`
- test = method
- Prefixed “`test`”

```
@interface ExampleTests : XCTestCase
@end

@implementation ExampleTests

- (void) testExample {
    XCTAssertTrue(2 + 2 == 4);
}

@end
```

Test Code

- Subclass `XCTestCase`
- test = method
- Prefixed "`test`"
- No arguments, returns void

```
@interface ExampleTests : XCTestCase
@end

@implementation ExampleTests

- (void) testExample {
    XCTAssertTrue(2 + 2 == 4);
}

@end
```

Test Code

- Subclass `XCTestCase`
- test = method
- Prefixed "`test`"
- No arguments, returns void
- Makes assertions

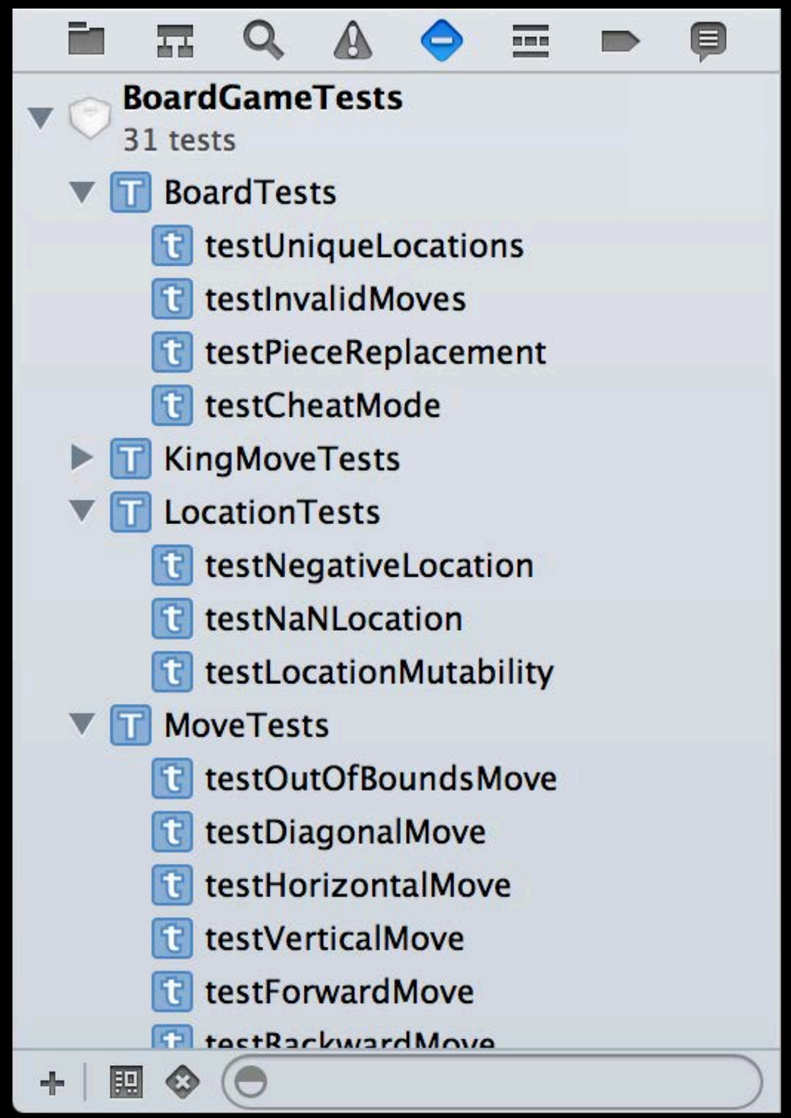
```
@interface ExampleTests : XCTestCase
@end

@implementation ExampleTests

- (void) testExample {
    XCTAssertTrue(2 + 2 == 4);
}

@end
```

Test Code

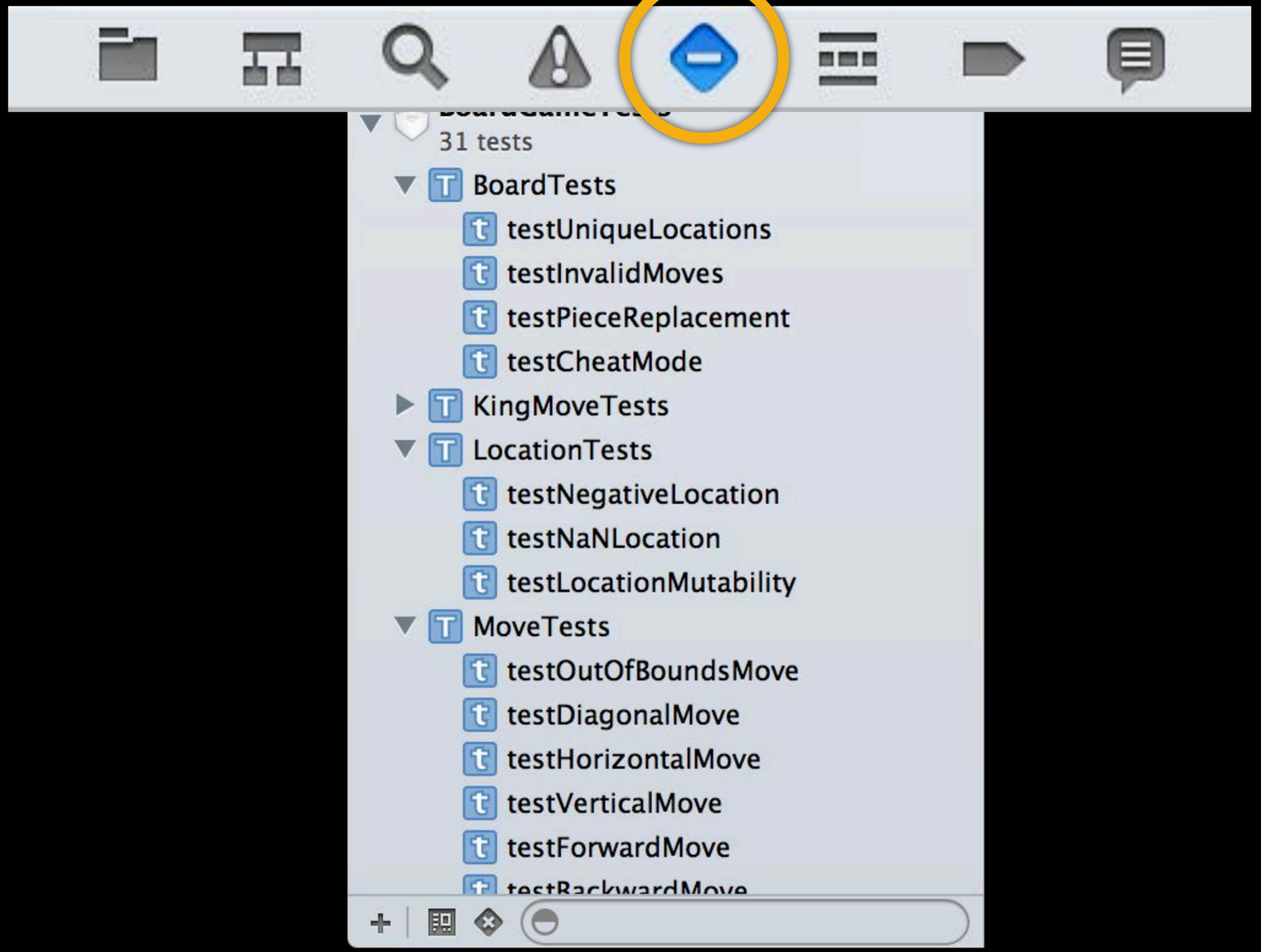
- Subclass `XCTestCase`
- test = method
- Prefixed "`test`"
- No arguments, returns void
- Makes assertions
- Built into test target

```
@interface ExampleTests : XCTestCase
@end

@implementation ExampleTests

- (void) testExample {
    XCTAssertTrue(2 + 2 == 4);
}

@end
```

Test Navigator

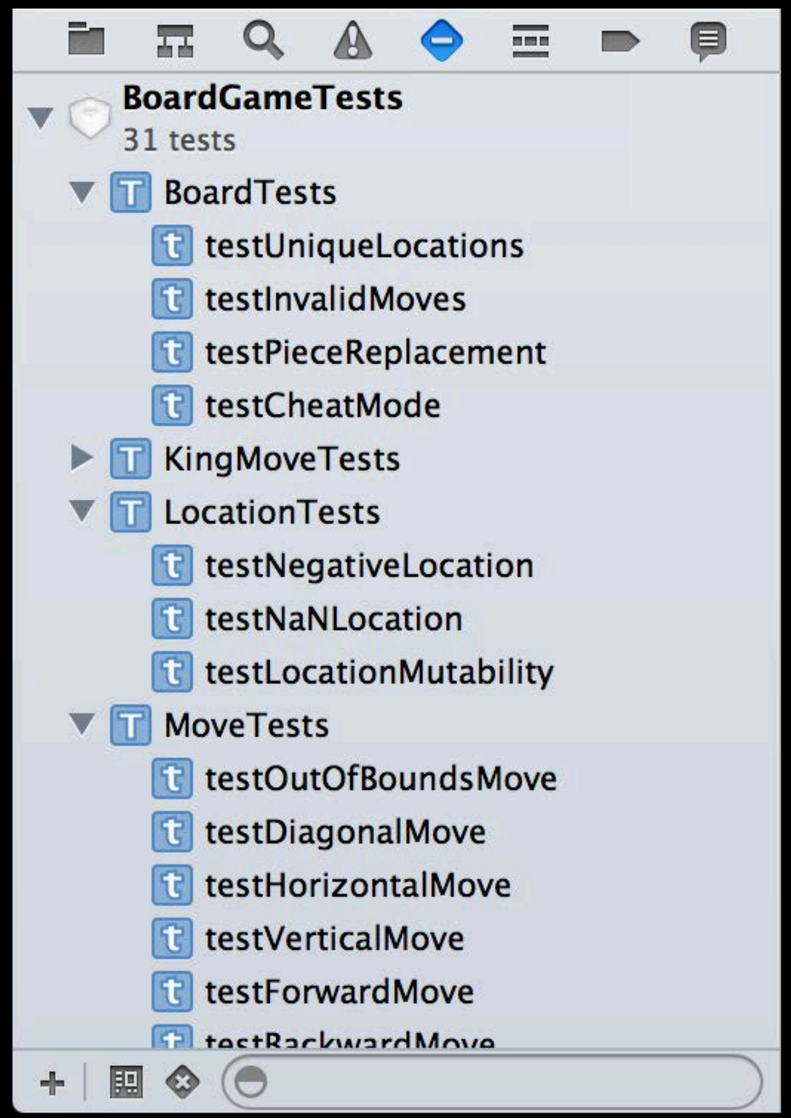


Test Navigator



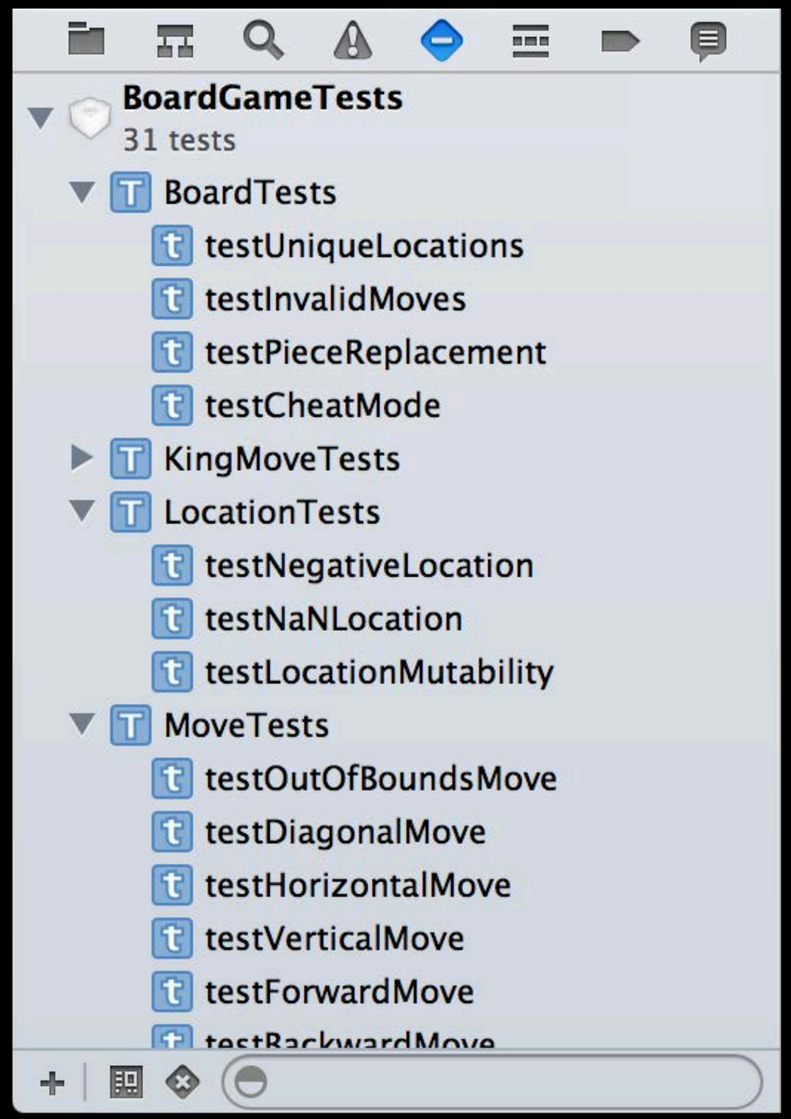
Test Navigator

- Test bundle targets
  - Test classes
    - Test methods



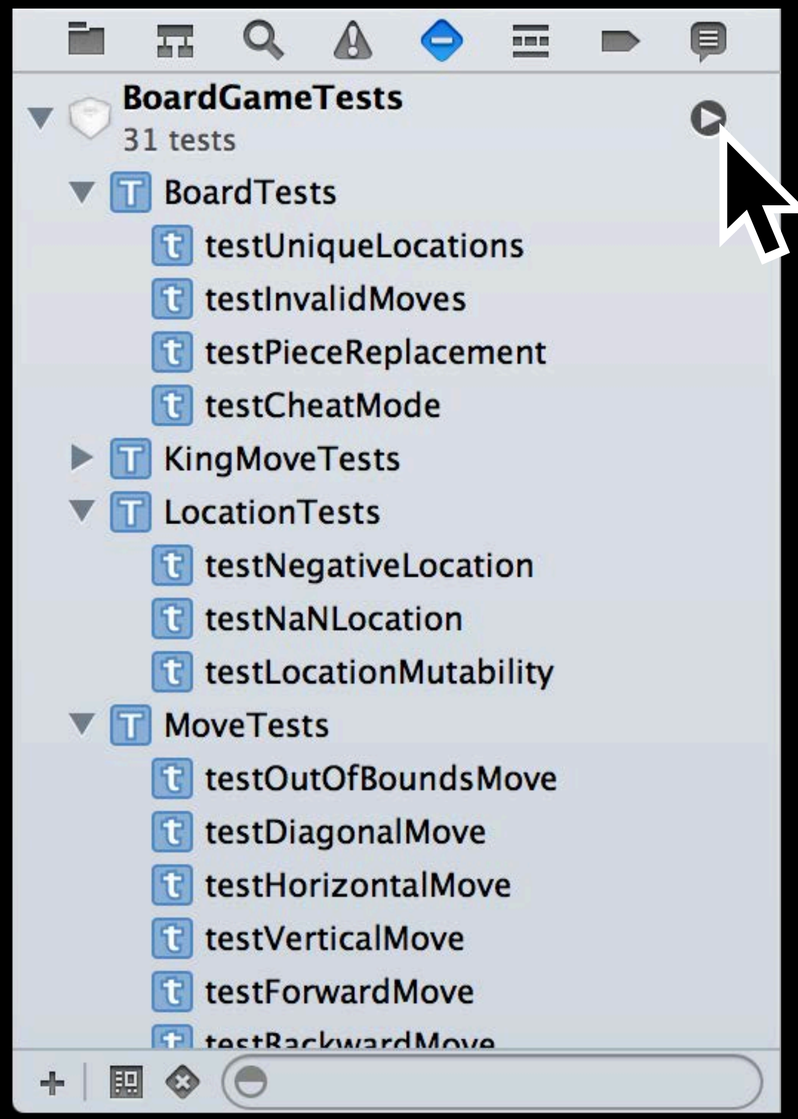
Test Navigator

- Test bundle targets
  - Test classes
    - Test methods
- Click to run



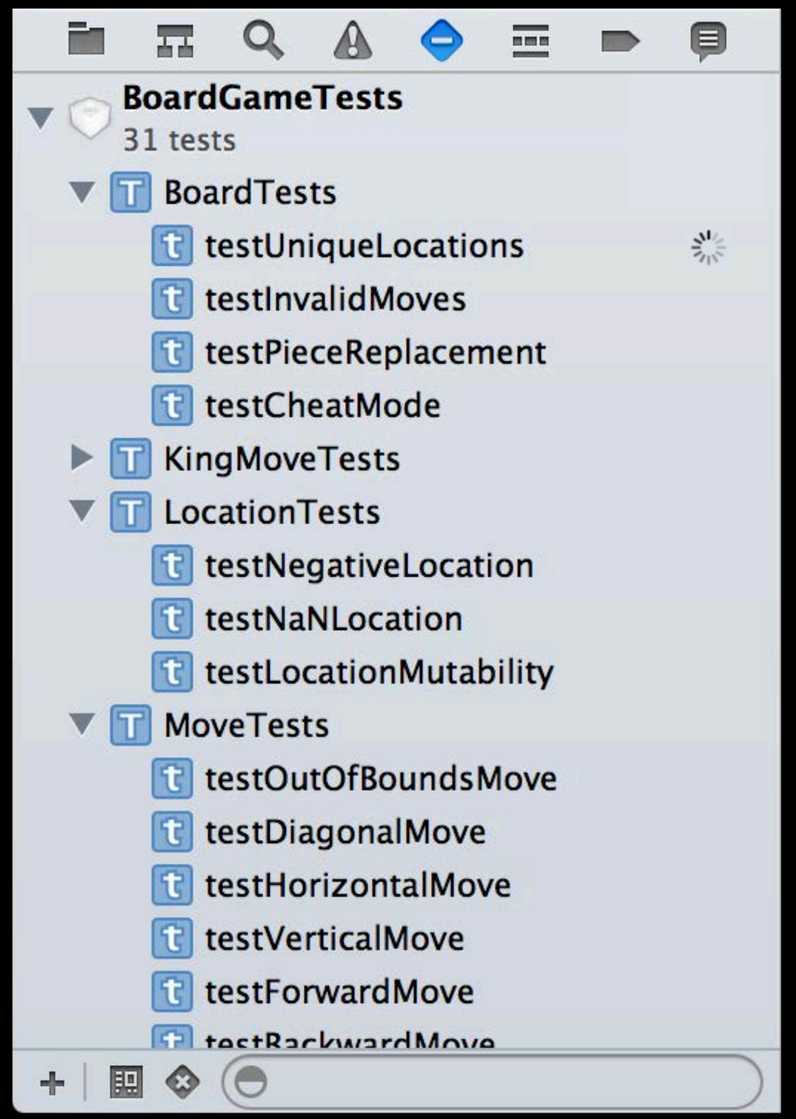
Test Navigator

- Test bundle targets
  - Test classes
    - Test methods
- Click to run



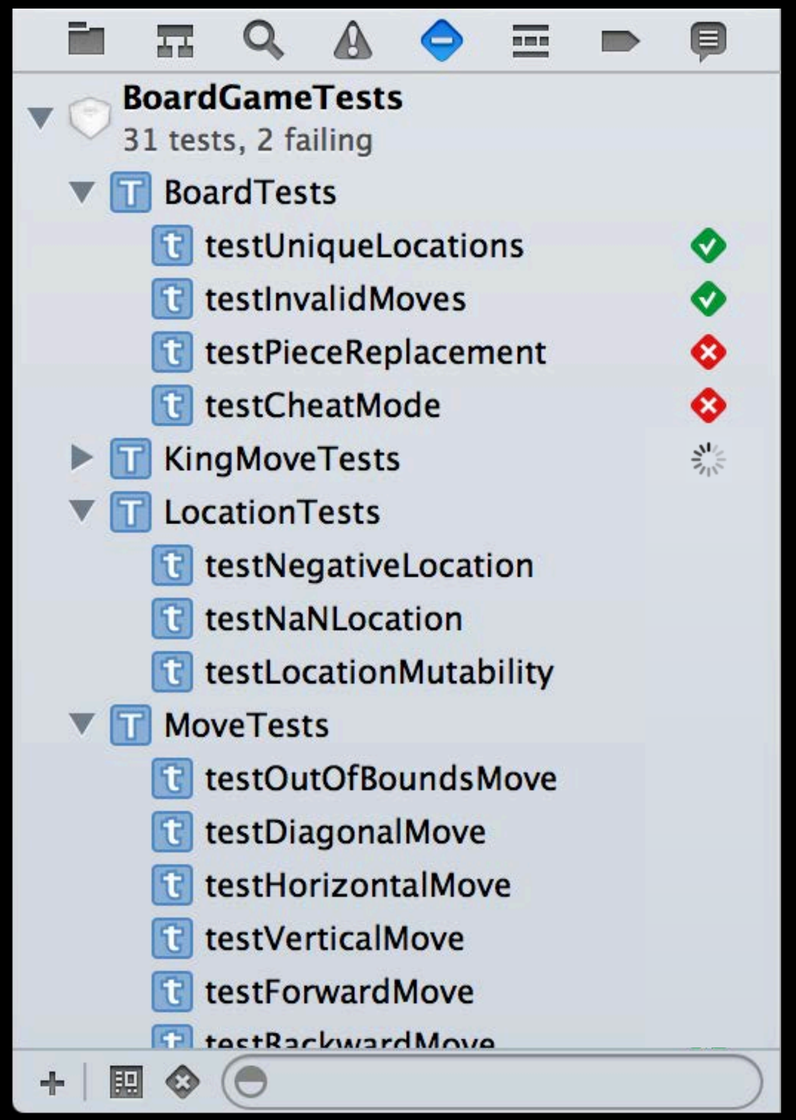
Test Navigator

- Test bundle targets
  - Test classes
    - Test methods
- Click to run



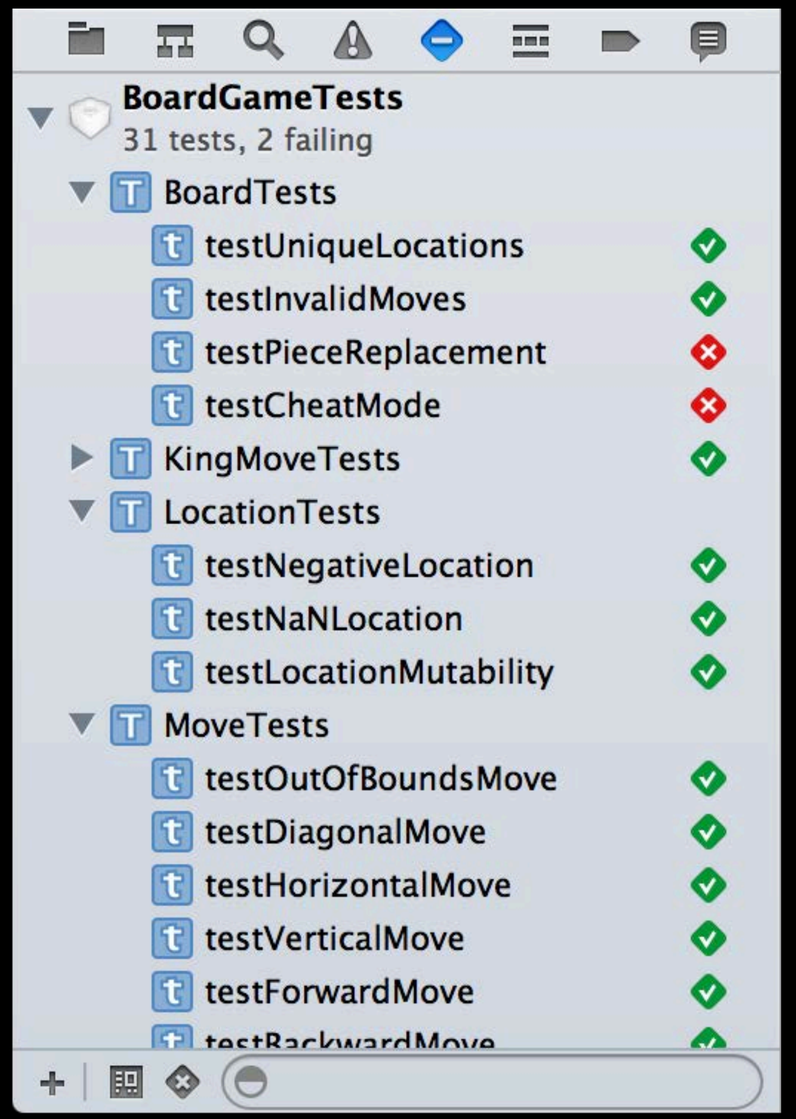
Test Navigator

- Test bundle targets
  - Test classes
    - Test methods
- Click to run
- Results inline



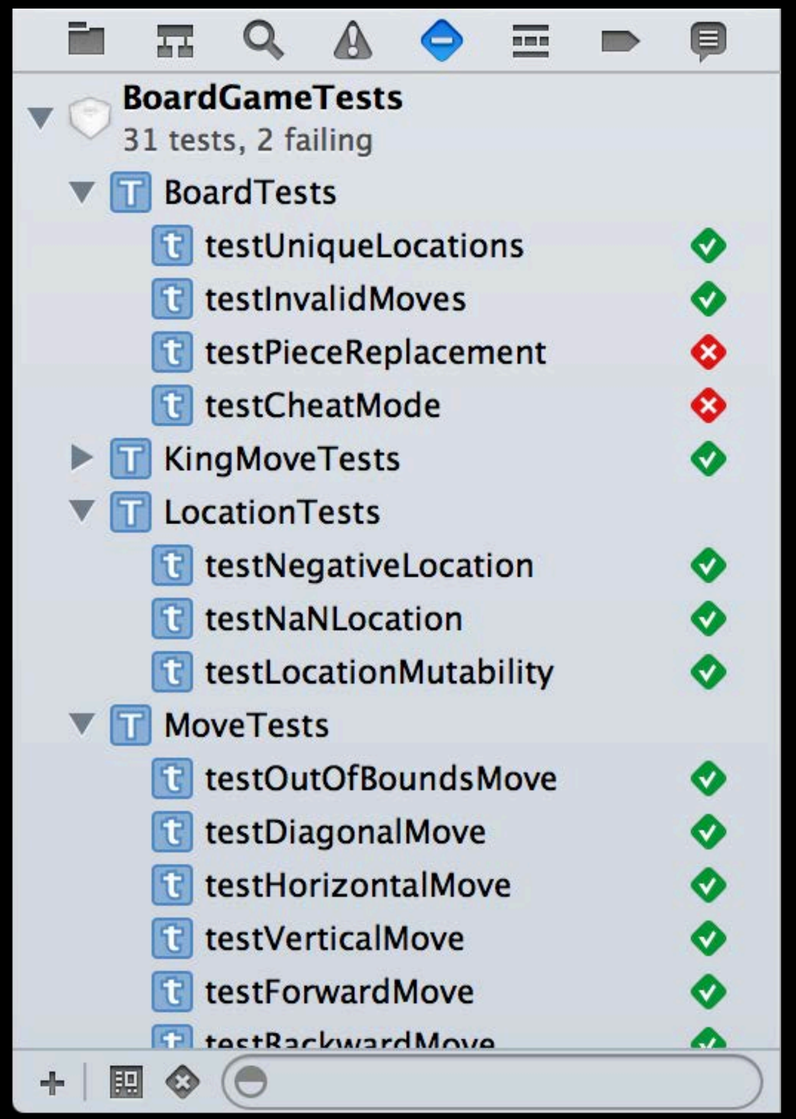
Test Navigator

- Test bundle targets
  - Test classes
    - Test methods
- Click to run
- Results inline



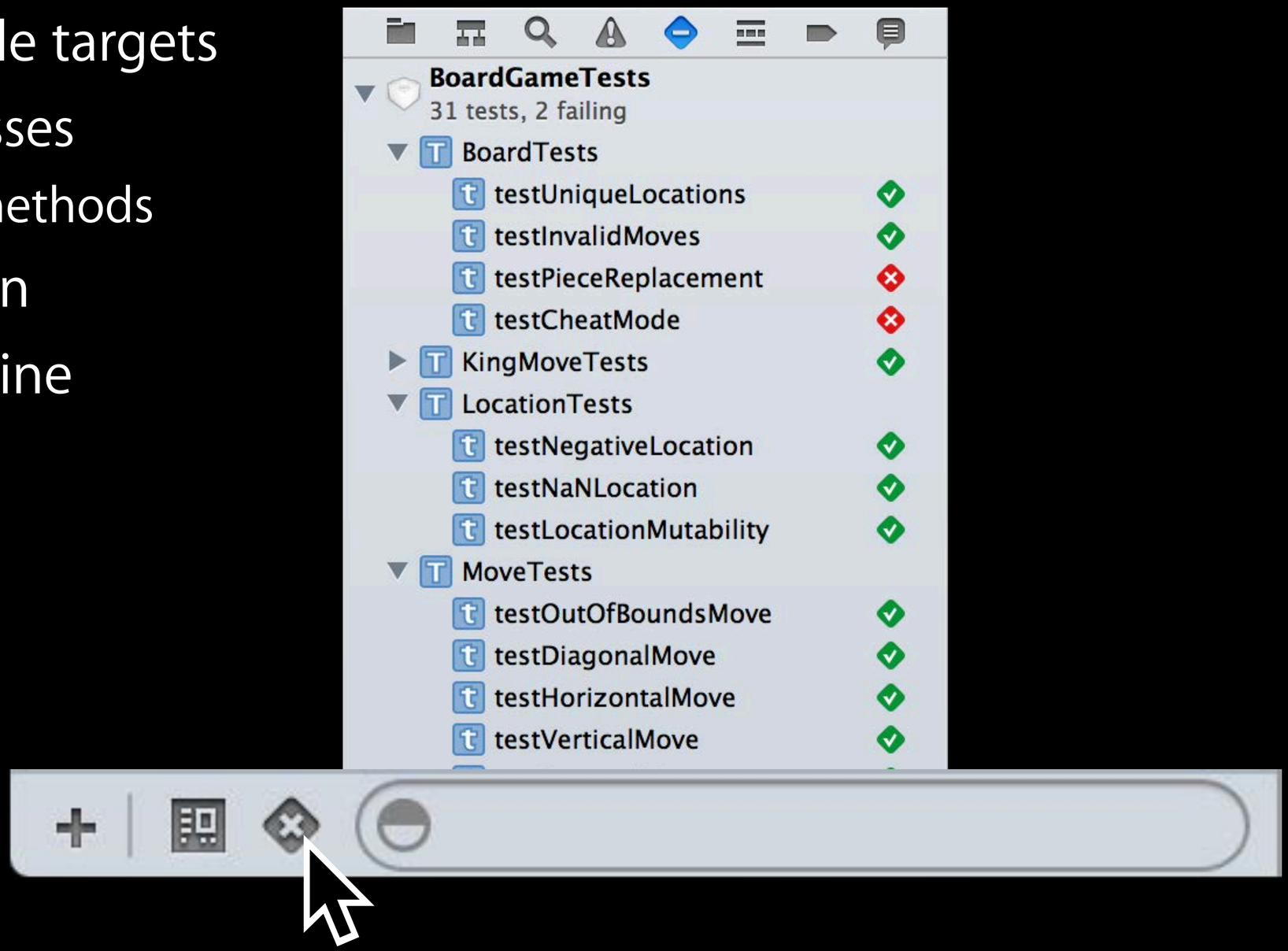
Test Navigator

- Test bundle targets
  - Test classes
    - Test methods
- Click to run
- Results inline



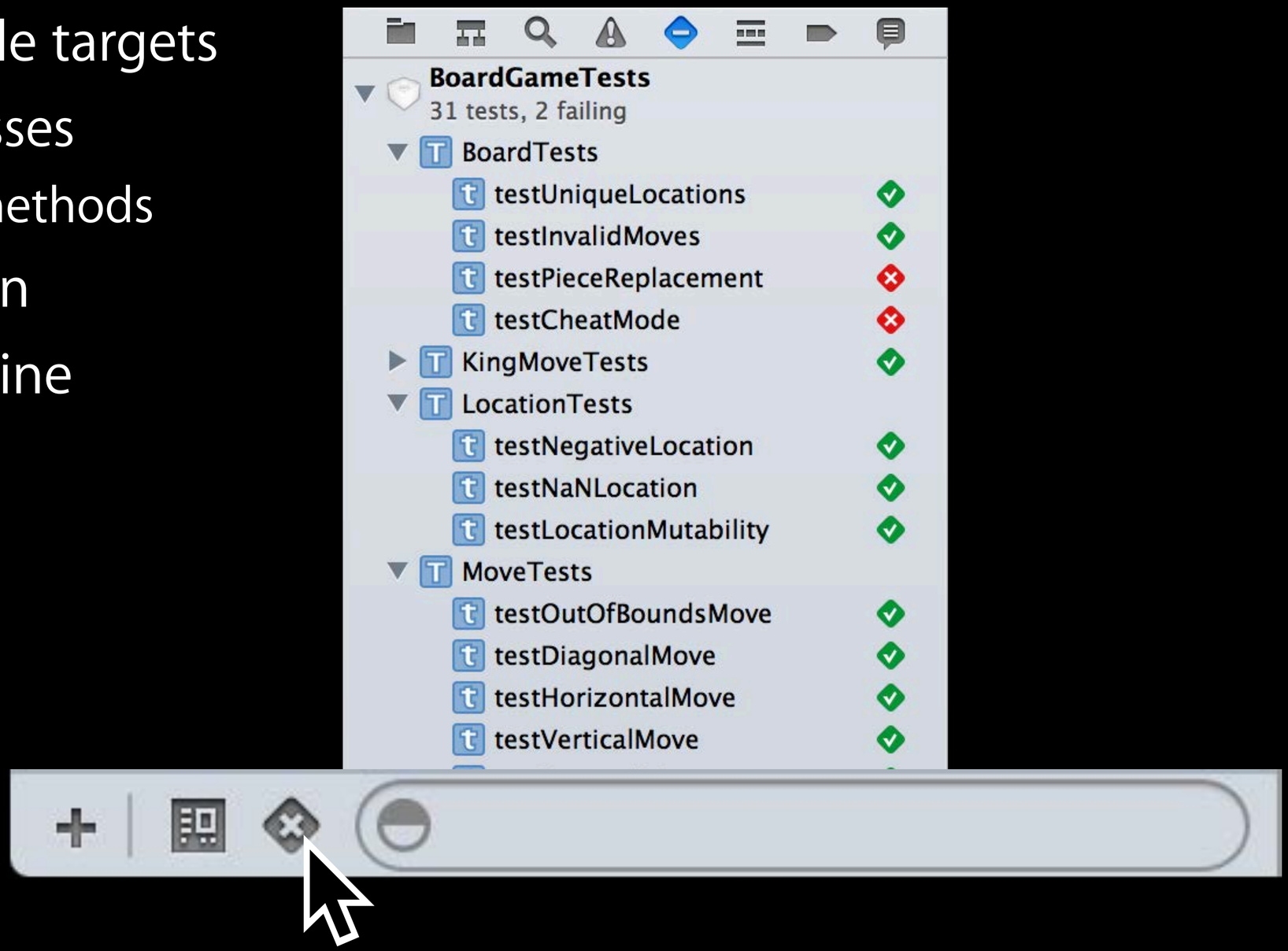
Test Navigator

- Test bundle targets
  - Test classes
    - Test methods
- Click to run
- Results inline



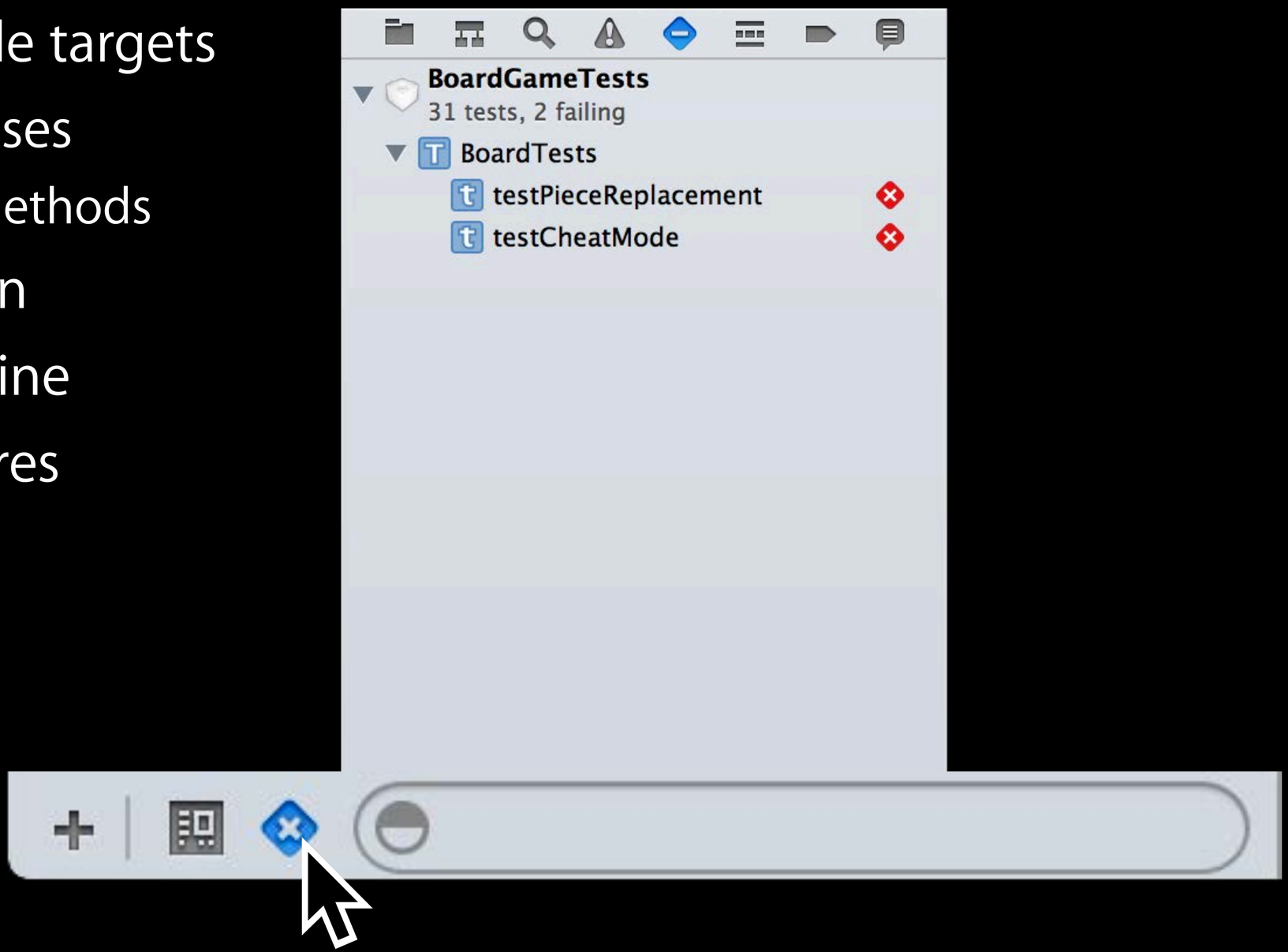
Test Navigator

- Test bundle targets
  - Test classes
    - Test methods
- Click to run
- Results inline



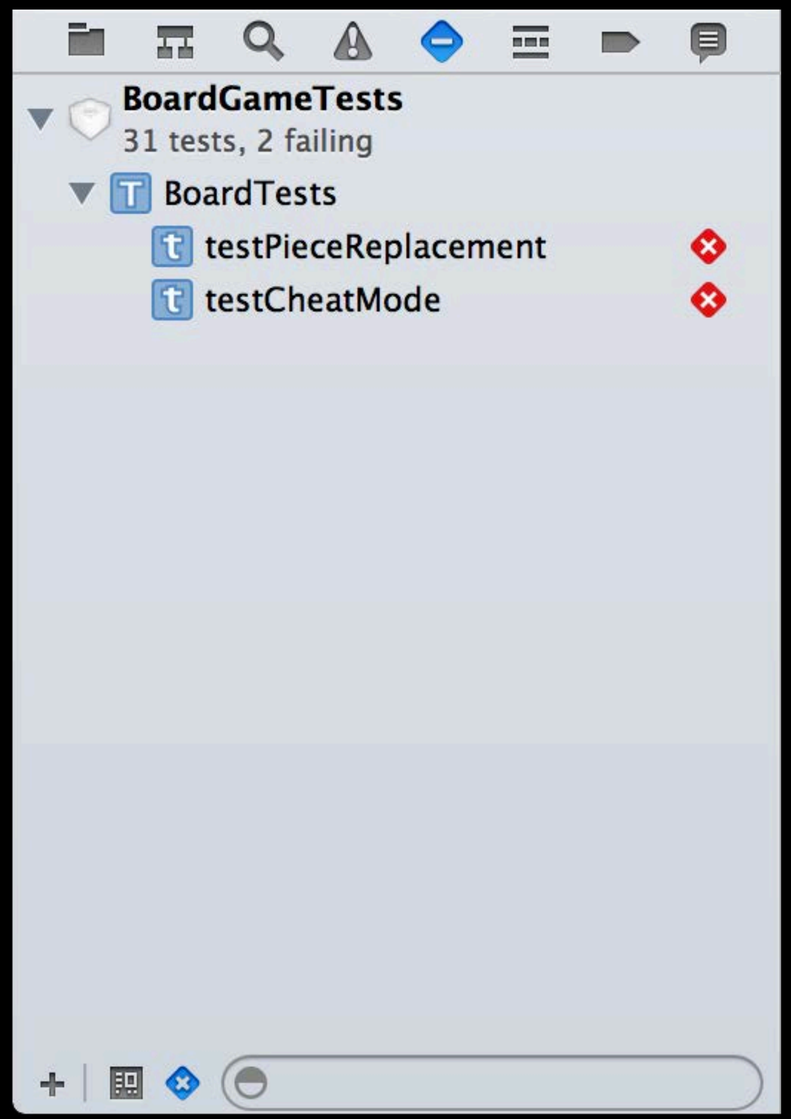
Test Navigator

- Test bundle targets
  - Test classes
    - Test methods
- Click to run
- Results inline
- Filter failures



Test Navigator

- Test bundle targets
  - Test classes
    - Test methods
- Click to run
- Results inline
- Filter failures



Editor Test Indicators


```
32
33  - (void)testVerticalMove {
34      Location *invalidLocation = [[Location alloc] initWithX:0 andY:0];
35      XCTAssertFalse([self.ruleChecker validateLocation:invalidLocation forPiece:self.piece]);
36
37      Location *location = [[Location alloc] initWithX:0 andY:1];
38      BOOL verticalMoveWasValid = [_ruleChecker validateLocation:location forPiece:_piece];
39      XCTAssertFalse(verticalMoveWasValid, @"Vertical move succeeded!");
40  }
41
42  - (void)testOutOfBoundsMove {
43      // negative location
44      Location *location = [[Location alloc] initWithX:-1 andY:-1];
45      BOOL negativeMoveWasValid = [_ruleChecker validateLocation:location forPiece:_piece];
46      XCTAssertFalse(negativeMoveWasValid, @"Negative bounds move succeeded");
47
48      // too big x
49      location = [[Location alloc] initWithX:9 andY:0];
```

Editor Test Indicators


```
32
33
34
35  void)testVerticalMove {
36      Location *invalidLocation = [[Location alloc] initWithX:0 andY:0];
        XCTAssertFalse([self.ruleChecker validateLocation:invalidLocation forPiece:self.piece]);
37
        Location *location = [[Location alloc] initWithX:0 andY:1];
        BOOL verticalMoveWasValid = [_ruleChecker validateLocation:location forPiece:_piece];
38      XCTAssertFalse(verticalMoveWasValid, @"Vertical move succeeded!");
39
    void)testOutOfBoundsMove {
        // negative location
40      Location *location = [[Location alloc] initWithX:-1 andY:-1];
        BOOL negativeMoveWasValid = [_ruleChecker validateLocation:location forPiece:_piece];
41      XCTAssertFalse(negativeMoveWasValid, @"Negative bounds move succeeded");
42      // too big x
        location = [[Location alloc] initWithX:9 andY:0];
43
```

Editor Test Indicators

NEW

```
32
33
34
35  void)testVerticalMove {
36      Location *invalidLocation = [[Location alloc] initWithX:0 andY:0];
        XCTAssertFalse([self.ruleChecker validateLocation:invalidLocation forPiece:self.piece]);
37
        Location *location = [[Location alloc] initWithX:0 andY:1];
        BOOL verticalMoveWasValid = [_ruleChecker validateLocation:location forPiece:_piece];
38      XCTAssertFalse(verticalMoveWasValid, @"Vertical move succeeded!");
39
    void)testOutOfBoundsMove {
        // negative location
40      Location *location = [[Location alloc] initWithX:-1 andY:-1];
        BOOL negativeMoveWasValid = [_ruleChecker validateLocation:location forPiece:_piece];
41      XCTAssertFalse(negativeMoveWasValid, @"Negative bounds move succeeded");
42      // too big x
        location = [[Location alloc] initWithX:9 andY:0];
```

Editor Test Indicators

NEW

```
32
33
34
35  void)testVerticalMove {
36      Location *invalidLocation = [[Location alloc] initWithX:0 andY:0];
        XCTAssertFalse([self.ruleChecker validateLocation:invalidLocation forPiece:self.piece]);
37
        Location *location = [[Location alloc] initWithX:0 andY:1];
        BOOL verticalMoveWasValid = [_ruleChecker validateLocation:location forPiece:_piece];
38      XCTAssertFalse(verticalMoveWasValid, @"Vertical move succeeded!");
39
    void)testOutOfBoundsMove {
        // negative location
40      Location *location = [[Location alloc] initWithX:-1 andY:-1];
        BOOL negativeMoveWasValid = [_ruleChecker validateLocation:location forPiece:_piece];
41      XCTAssertFalse(negativeMoveWasValid, @"Negative bounds move succeeded");
                "negativeMoveWasValid" should be false. Negative bounds move succeeded
        // too big x
42      location = [[Location alloc] initWithX:9 andY:0];
```

Demo

Making your first unit test in Xcode 5

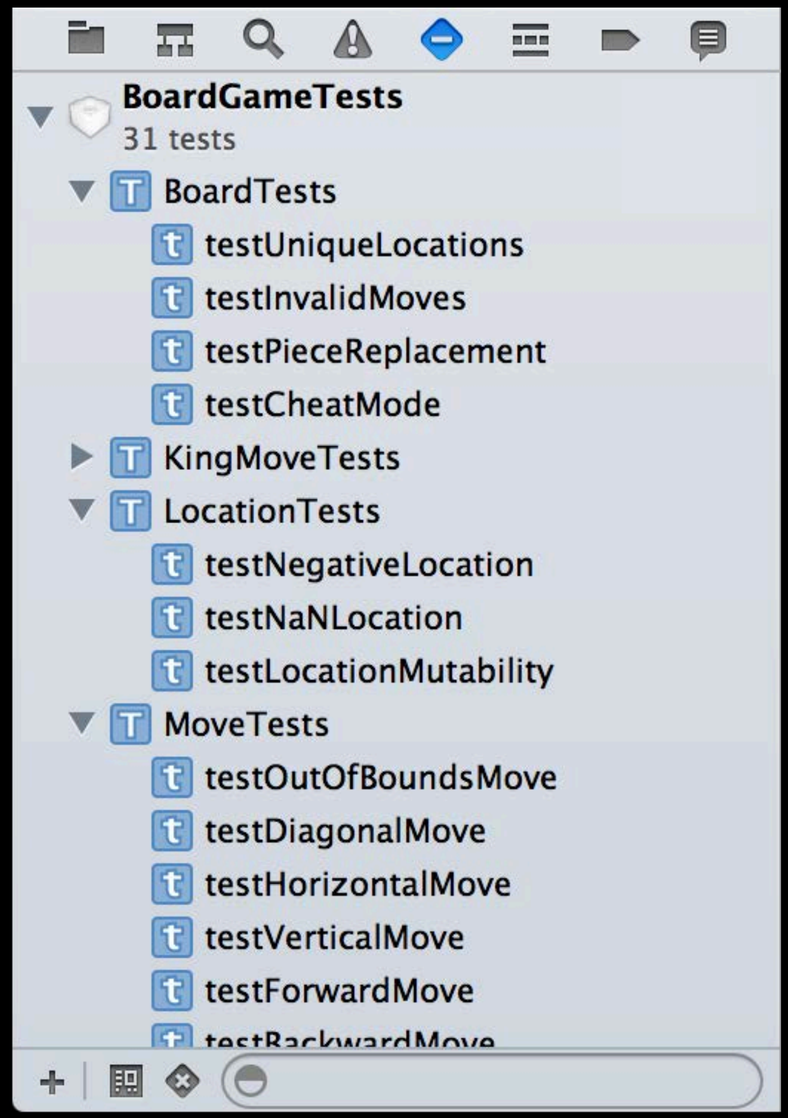
Mike Swingler
Xcode Engineer

Overview

Making your first unit test in Xcode 5

NEW

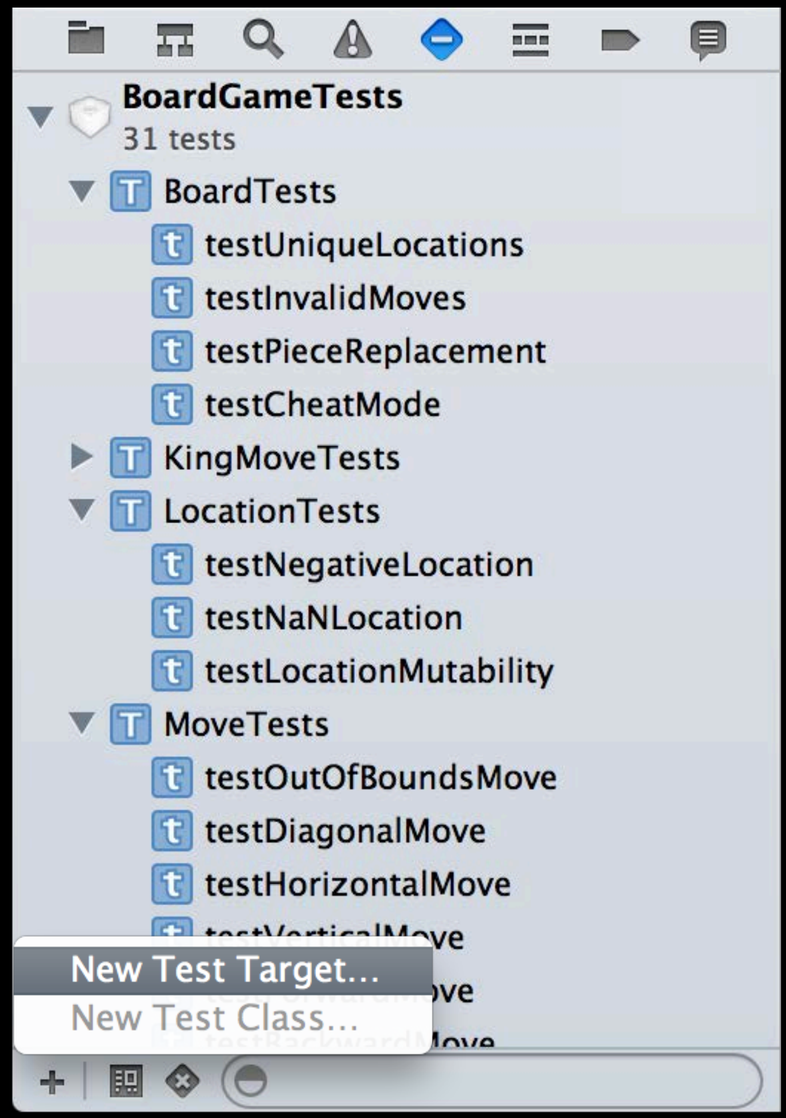
- Creating a test target is easy
- Add a new test by adding a method
- Easy to run tests
  - Test navigator
  - Editor



Overview

Making your first unit test in Xcode 5

- Creating a test target is easy
- Add a new test by adding a method
- Easy to run tests
  - Test navigator
  - Editor

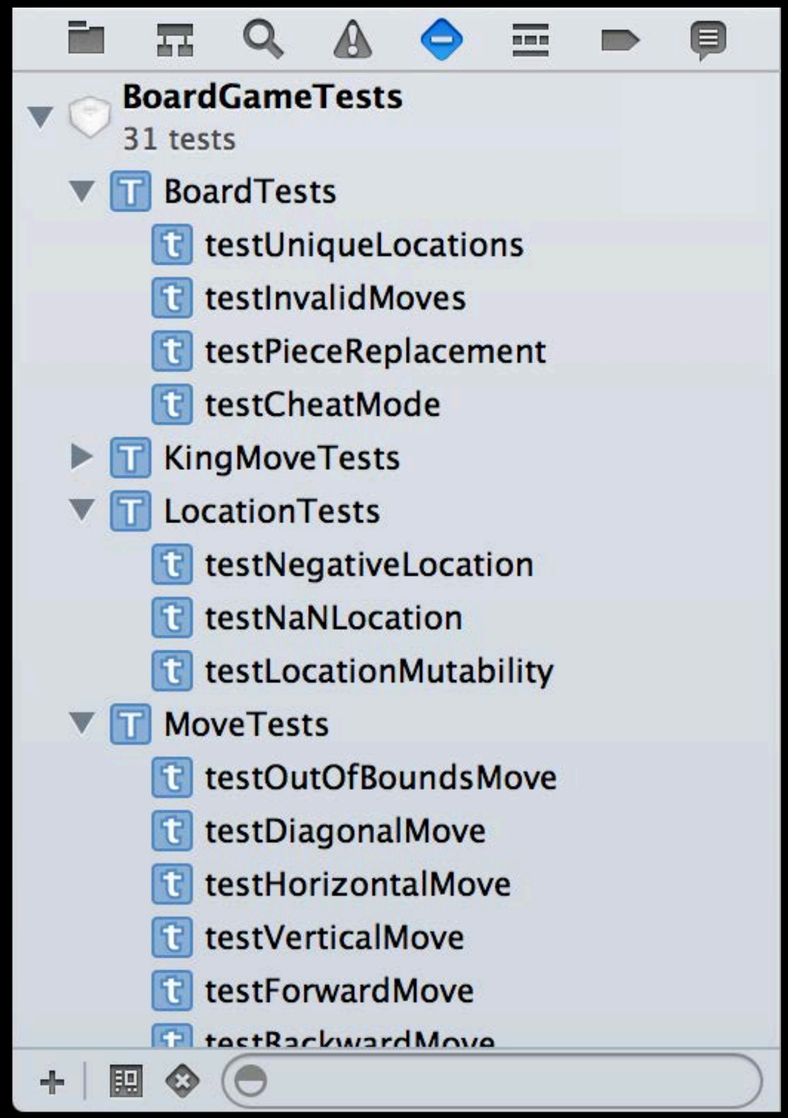


Overview

Making your first unit test in Xcode 5

NEW

- Creating a test target is easy
- Add a new test by adding a method
- Easy to run tests
  - Test navigator
  - Editor

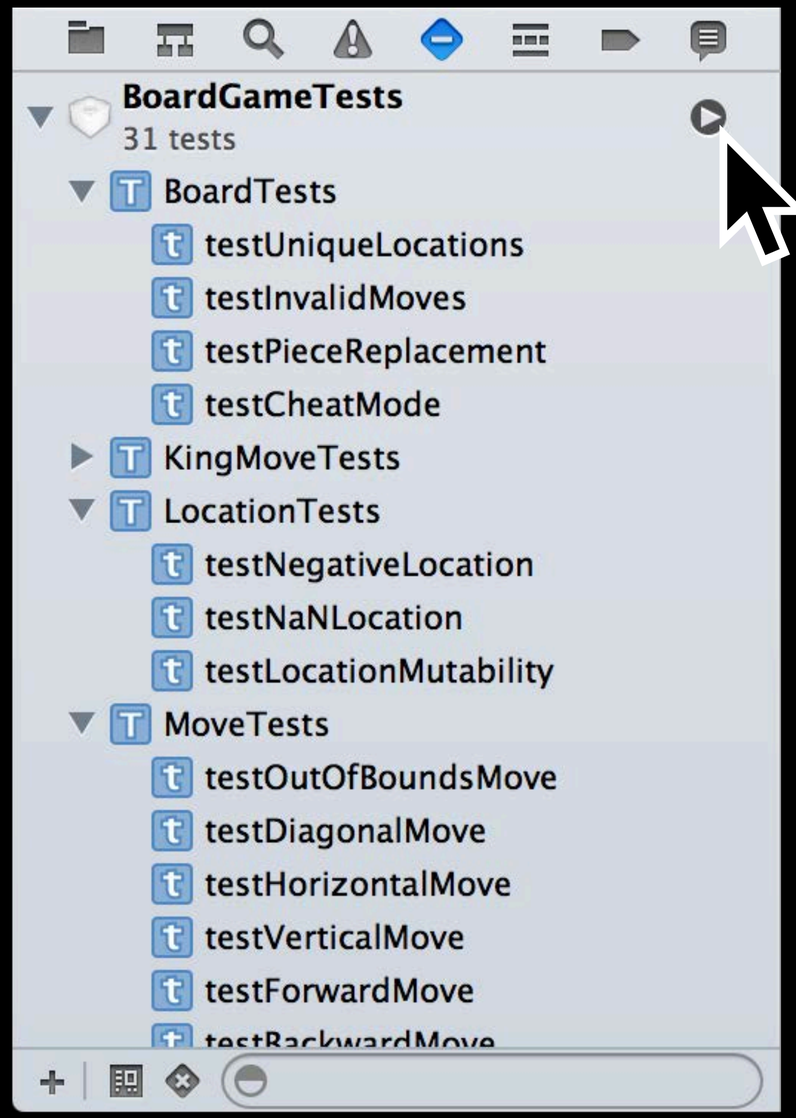


Overview

Making your first unit test in Xcode 5

NEW

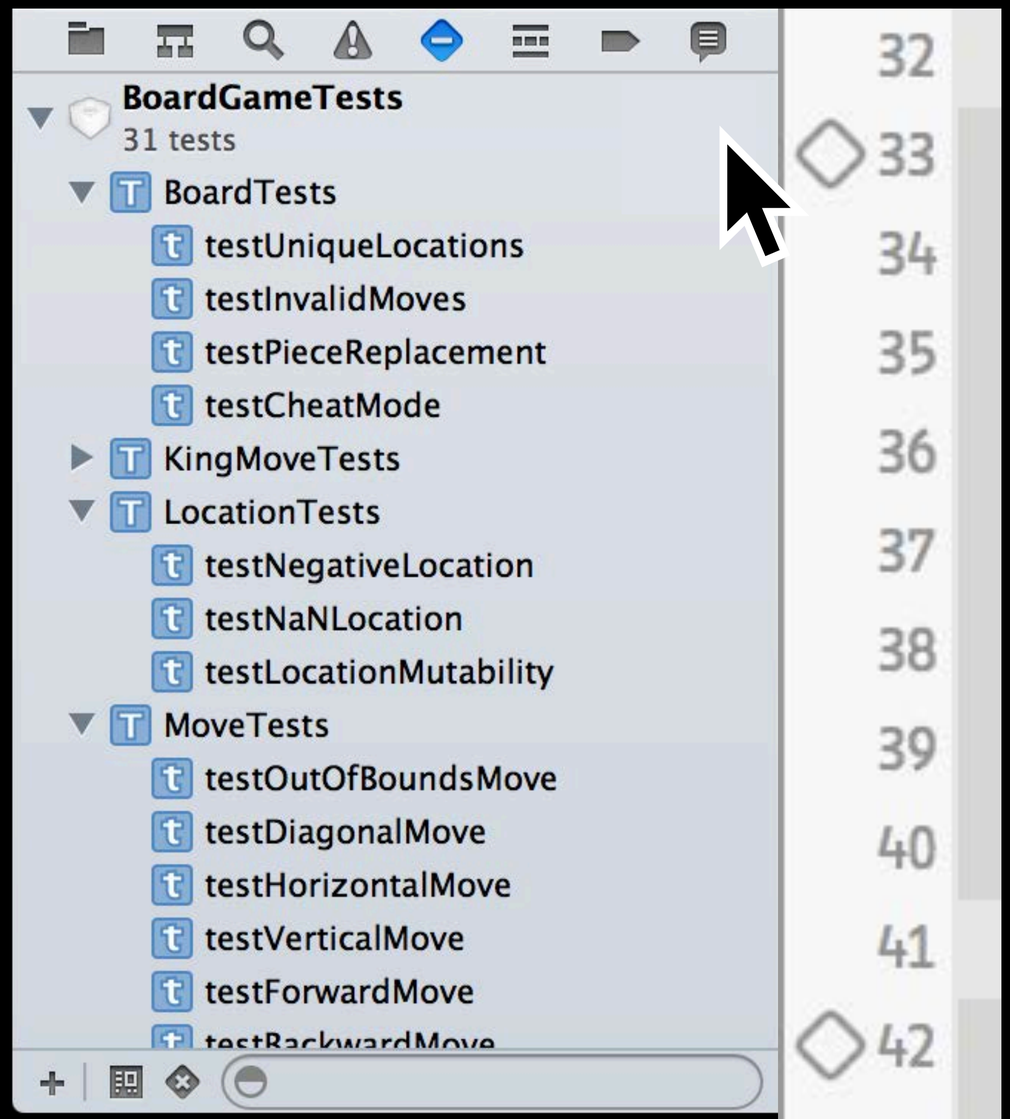
- Creating a test target is easy
- Add a new test by adding a method
- Easy to run tests
  - Test navigator
  - Editor



Overview

Making your first unit test in Xcode 5

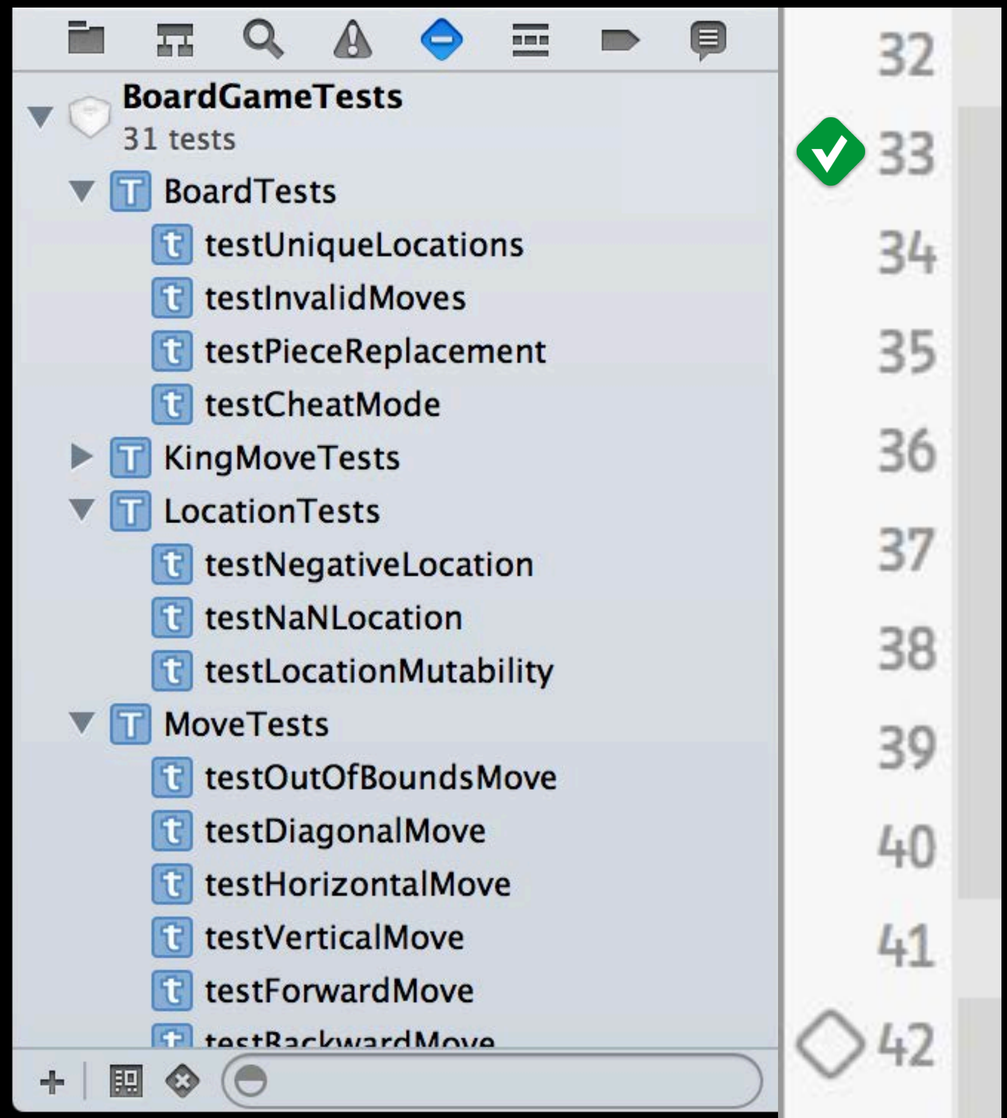
- Creating a test target is easy
- Add a new test by adding a method
- Easy to run tests
  - Test navigator
  - Editor



Overview

Making your first unit test in Xcode 5

- Creating a test target is easy
- Add a new test by adding a method
- Easy to run tests
  - Test navigator
  - Editor



Assert the Expected

XCTestAssertions.h

Assert the Expected

XCTestAssertions.h

XCTAssertThrowsSpecific

XCTAssertTrueNoThrow

XCTAssertNil

XCTAssertNoThrowSpecific

XCTAssertNotNil

XCTAssertEqualObjects

XCTAssertEqualsWithAccuracy

XCTFail

XCTAssertNoThrow

XCTAssertTrue

XCTAssertThrowsSpecificNamed

XCTAssertFalse

XCTAssertThrows

XCTAssertNoThrowSpecificNamed

XCTAssertEquals

XCTAssertFalseNoThrow

Assert the Expected

XCTestAssertions.h

XCTAssertThrowsSpecific

XCTAssertTrueNoThrow

XCTAssertNil

XCTAssertNoThrowSpecific

XCTAssertNotNil

XCTAssertEqualObjects

XCTAssertEqualsWithAccuracy

XCTFail

XCTAssertNoThrow

XCTAssertTrue

XCTAssertThrowsSpecificNamed

XCTAssertFalse

XCTAssertThrows

XCTAssertNoThrowSpecificNamed

XCTAssertEquals

XCTAssertFalseNoThrow

Assert the Expected

XCTestAssertions.h

XCTAssertThrowsSpecific

XCTAssertTrueNoThrow

XCTAssertNil

XCTAssertNoThrowSpecific

XCTAssertNotNil

XCTAssertEqualObjects

XCTAssertEqualsWithAccuracy

XCTFail

XCTAssertNoThrow

XCTAssertTrue

XCTAssertThrowsSpecificNamed

XCTAssertFalse

XCTAssertThrows

XCTAssertNoThrowSpecificNamed

XCTAssertEquals

XCTAssertFalseNoThrow

Assert the Expected

XCTestAssertions.h

XCTAssertThrowsSpecific

XCTAssertTrueNoThrow

XCTAssertNil

XCTAssertNoThrowSpecific

XCTAssertNotNil

XCTAssertEqualObjects

XCTAssertEqualsWithAccuracy

XCTFail

XCTAssertNoThrow

XCTAssertTrue

XCTAssertFalse

XCTAssertThrowsSpecificNamed

XCTAssertThrows

XCTAssertNoThrowSpecificNamed

XCTAssertEquals

XCTAssertFalseNoThrow

Assert the Expected

XCTestAssertions.h

XCTAssertThrowsSpecific

XCTAssertTrueNoThrow

XCTAssertNil

XCTAssertNoThrowSpecific

XCTAssertNotNil

XCTAssertEqualObjects

XCTAssertEqualsWithAccuracy

XCTFail

XCTAssertNoThrow

XCTAssertTrue

XCTAssertThrowsSpecificNamed

XCTAssertFalse

XCTAssertThrows

XCTAssertNoThrowSpecificNamed

XCTAssertEquals

XCTAssertFalseNoThrow

Expect the Unexpected

Expect the Unexpected

- Expected success

Expect the Unexpected

- Expected success
  - Can come first

Expect the Unexpected

- Expected success
  - Can come first
- Regressions

Expect the Unexpected

- Expected success
  - Can come first
- Regressions
- Expected failure

Expect the Unexpected

- Expected success
  - Can come first
- Regressions
- Expected failure
  - Overflow, ∞, NaN

Expect the Unexpected

- Expected success
  - Can come first
- Regressions
- Expected failure
  - Overflow, ∞, NaN
  - Nil

Expect the Unexpected

- Expected success
  - Can come first
- Regressions
- Expected failure
  - Overflow, ∞, NaN
  - Nil
  - Empty collections

Expect the Unexpected

- Expected success
  - Can come first
- Regressions
- Expected failure
  - Overflow, ∞, NaN
  - Nil
  - Empty collections
  - Unexpected types in collections

Expect the Unexpected

- Expected success
  - Can come first
- Regressions
- Expected failure
  - Overflow, ∞, NaN
  - Nil
  - Empty collections
  - Unexpected types in collections
  - NSError

Set Up a Test

`-setUp`

Set Up a Test

`-setUp`

- Runs before every test method

Set Up a Test

`-setUp`

- Runs before every test method
- Create “shim” objects

Set Up a Test

`-setUp`

- Runs before every test method
- Create "shim" objects
- Load data from .xctest bundle
  - `[NSBundle bundleForClass:[MyTestClass class]]`

Set Up a Test

`-setUp`

- Runs before every test method
- Create “shim” objects
- Load data from .xctest bundle
  - `[NSBundle bundleForClass:[MyTestClass class]]`
- …anything you need to setup “the world”

Set Up a Test

`-setUp`

- Runs before every test method
- Create "shim" objects
- Load data from .xctest bundle
  - `[NSBundle bundleForClass:[MyTestClass class]]`
- …anything you need to setup "the world"
- Use `-tearDown` to perform any cleanup

How XCTest Works

Test class and methods

```
@interface ExampleTests : XCTestCase
@implementation ExampleTests

+ (void) setUp { /* Class set-up. */ }

+ (void) tearDown { /* Class tear-down. */ }

- (void) setUp { /* Test set-up. */ }

- (void) tearDown { /* Test tear-down. */ }

- (void) testExamplePassing {
    XCTAssertTrue(YES);
}

- (void) testExampleFailing {
    XCTAssertTrue(NO);
}
```

How XCTest Works

Test class and methods

```
@interface ExampleTests : XCTestCase
@implementation ExampleTests

+ (void) setUp { /* Class set-up. */ }

+ (void) tearDown { /* Class tear-down. */ }

- (void) setUp { /* Test set-up. */ }

- (void) tearDown { /* Test tear-down. */ }

- (void) testExamplePassing {
    XCTAssertTrue(YES);
}

- (void) testExampleFailing {
    XCTAssertTrue(NO);
}
```

How XCTest Works

Test class and methods

```
@interface ExampleTests : XCTestCase
@implementation ExampleTests

+ (void) setUp { /* Class set-up. */ }

+ (void) tearDown { /* Class tear-down. */ }

- (void) setUp { /* Test set-up. */ }

- (void) tearDown { /* Test tear-down. */ }

- (void) testExamplePassing {
    XCTAssertTrue(YES);
}

- (void) testExampleFailing {
    XCTAssertTrue(NO);
}
```

How XCTest Works

Test class and methods

```
@interface ExampleTests : XCTestCase
@implementation ExampleTests

+ (void) setUp { /* Class set-up. */ }

+ (void) tearDown { /* Class tear-down. */ }

- (void) setUp { /* Test set-up. */ }

- (void) tearDown { /* Test tear-down. */ }

- (void) testExamplePassing {
    XCTAssertTrue(YES);
}

- (void) testExampleFailing {
    XCTAssertTrue(NO);
}
```

How XCTest Works

Test class and methods

```
@interface ExampleTests : XCTestCase
@implementation ExampleTests

+ (void) setUp { /* Class set-up. */ }

+ (void) tearDown { /* Class tear-down. */ }

- (void) setUp { /* Test set-up. */ }

- (void) tearDown { /* Test tear-down. */ }

- (void) testExamplePassing {
    XCTAssertTrue(YES);
}

- (void) testExampleFailing {
    XCTAssertTrue(NO);
}
```

How XCTest Works

Test class and methods

```
@interface ExampleTests : XCTestCase
@implementation ExampleTests

+ (void) setUp { /* Class set-up. */ }

+ (void) tearDown { /* Class tear-down. */ }

- (void) setUp { /* Test set-up. */ }

- (void) tearDown { /* Test tear-down. */ }

- (void) testExamplePassing {
    XCTAssertTrue(YES);
}

- (void) testExampleFailing {
    XCTAssertTrue(NO);
}
```

How XCTest Works

Test class and methods

```
@interface ExampleTests : XCTestCase
@implementation ExampleTests

+ (void) setUp { /* Class set-up. */ }

+ (void) tearDown { /* Class tear-down. */ }

- (void) setUp { /* Test set-up. */ }

- (void) tearDown { /* Test tear-down. */ }

- (void) testExamplePassing {
    XCTAssertTrue(YES);
}

- (void) testExampleFailing {
    XCTAssertTrue(NO);
}
```

How XCTest Works

Test class and methods

```
@interface ExampleTests : XCTestCase
@implementation ExampleTests

+ (void) setUp { /* Class set-up. */ }

+ (void) tearDown { /* Class tear-down. */ }

- (void) setUp { /* Test set-up. */ }

- (void) tearDown { /* Test tear-down. */ }

- (void) testExamplePassing {
    XCTAssertTrue(YES);
}

- (void) testExampleFailing {
    XCTAssertTrue(NO);
}
```

How XCTest Works

Test class and methods

```
@interface ExampleTests : XCTestCase
@implementation ExampleTests

+ (void) setUp { /* Class set-up. */ }

+ (void) tearDown { /* Class tear-down. */ }

- (void) setUp { /* Test set-up. */ }

- (void) tearDown { /* Test tear-down. */ }

- (void) testExamplePassing {
    XCTAssertTrue(YES);
}

- (void) testExampleFailing {
    XCTAssertTrue(NO);
}
```

How XCTest Works

Test class and methods

```
@interface ExampleTests : XCTestCase
@implementation ExampleTests

+ (void) setUp { /* Class set-up. */ }

+ (void) tearDown { /* Class tear-down. */ }

- (void) setUp { /* Test set-up. */ }

- (void) tearDown { /* Test tear-down. */ }

- (void) testExamplePassing {
    XCTAssertTrue(YES);
}

- (void) testExampleFailing {
    XCTAssertTrue(NO);
}
```

How XCTest Works

Test class and methods

```
@interface ExampleTests : XCTestCase
@implementation ExampleTests

+ (void) setUp { /* Class set-up. */ }

+ (void) tearDown { /* Class tear-down. */ }

- (void) setUp { /* Test set-up. */ }

- (void) tearDown { /* Test tear-down. */ }

- (void) testExamplePassing {
    XCTAssertTrue(YES);
}

- (void) testExampleFailing {
    XCTAssertTrue(NO);
}
```

OCUnit

OCUnit

- Co-exists with XCTest

OCUnit

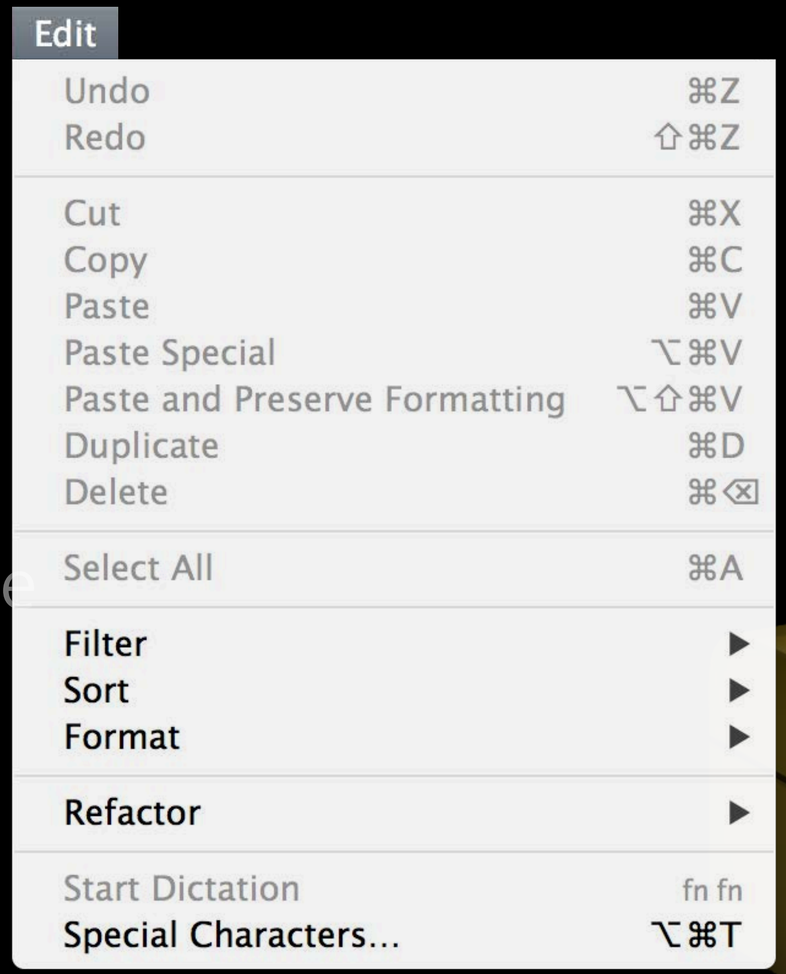
- Co-exists with XCTest
- You may still need it
  - iOS 6

OCUnit

- Co-exists with XCTest
- You may still need it
  - iOS 6
- Migration tool available

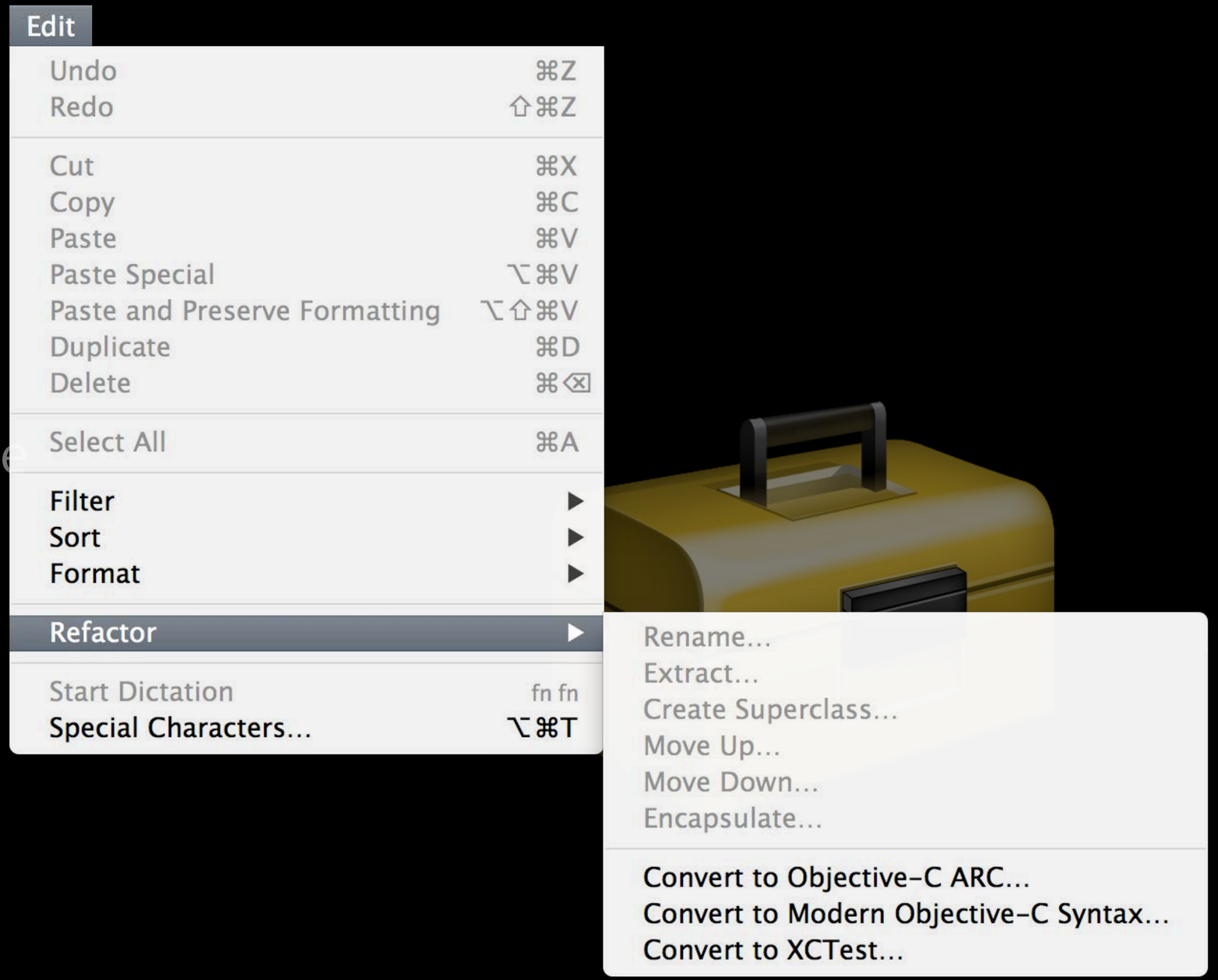
OCUnit

- Co-exists with XCTest
- You may still need it
  - iOS 6
- Migration tool available



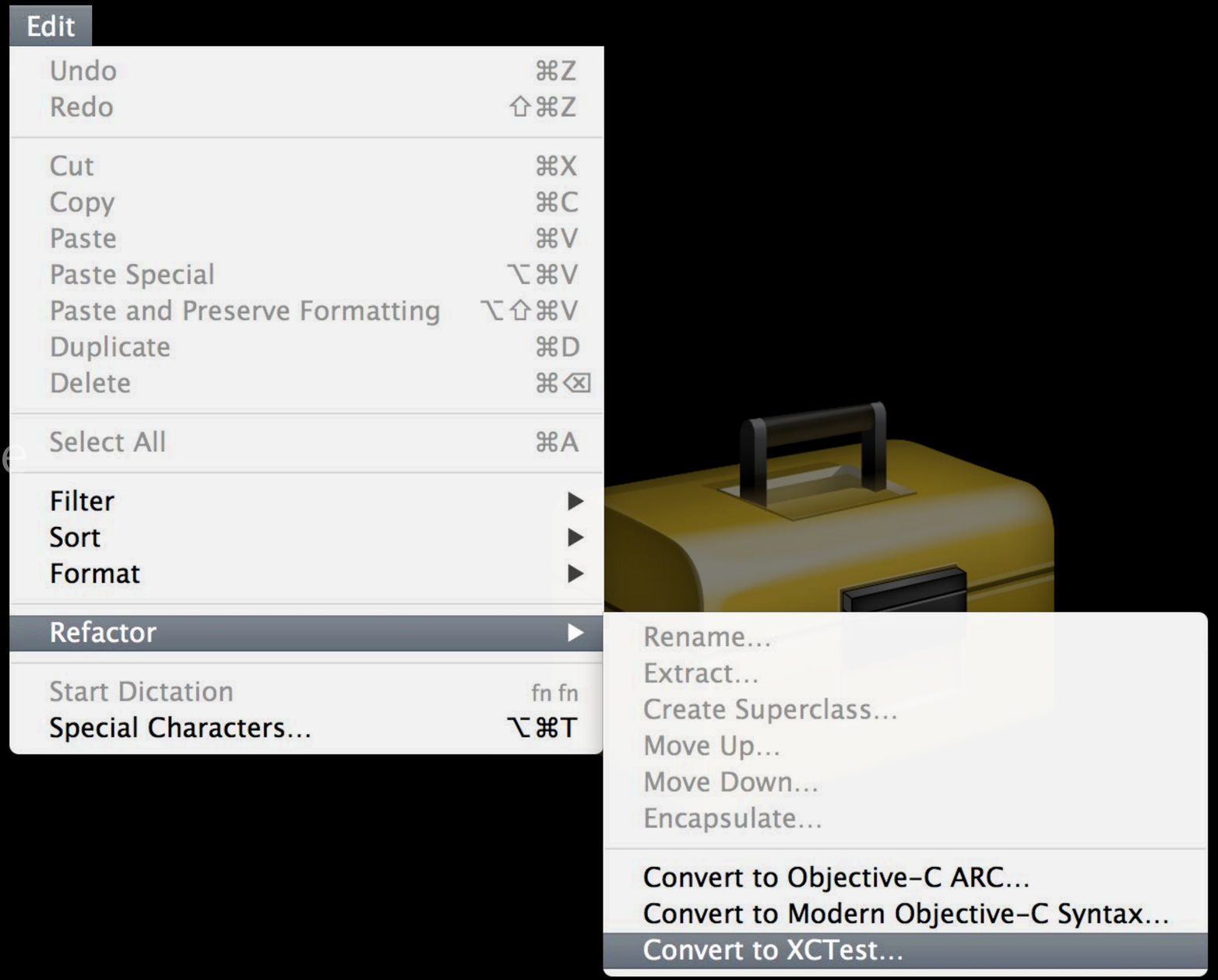
OCUnit

- Co-exists with XCTest
- You may still need it
  - iOS 6
- Migration tool available



OCUnit

- Co-exists with XCTest
- You may still need it
  - iOS 6
- Migration tool available



Debugging Tests

Demo

Debugging tests

Bino George
Xcode Engineer

Overview

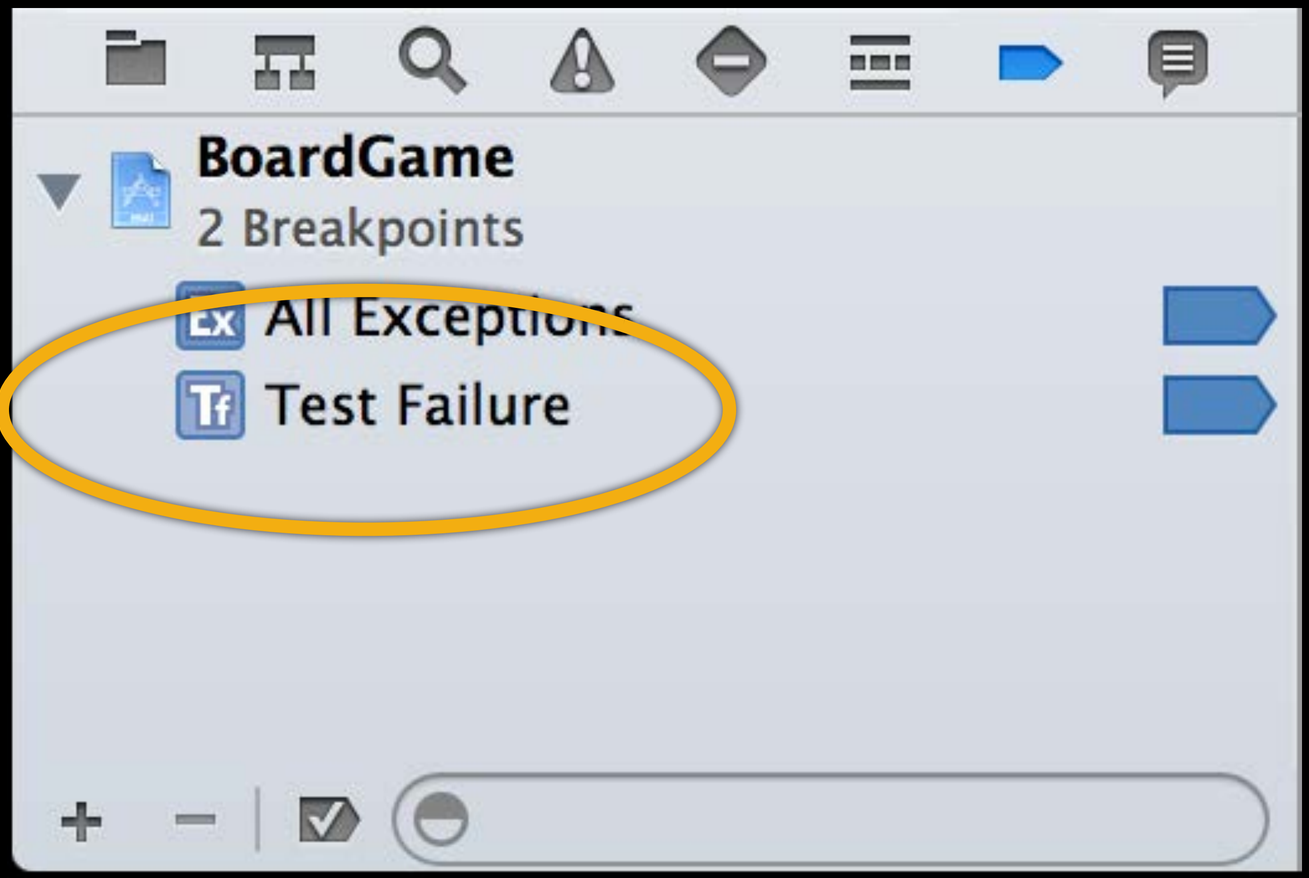
New test debugging UI

- Test breakpoint

Overview

New test debugging UI

- Test breakpoint



Overview

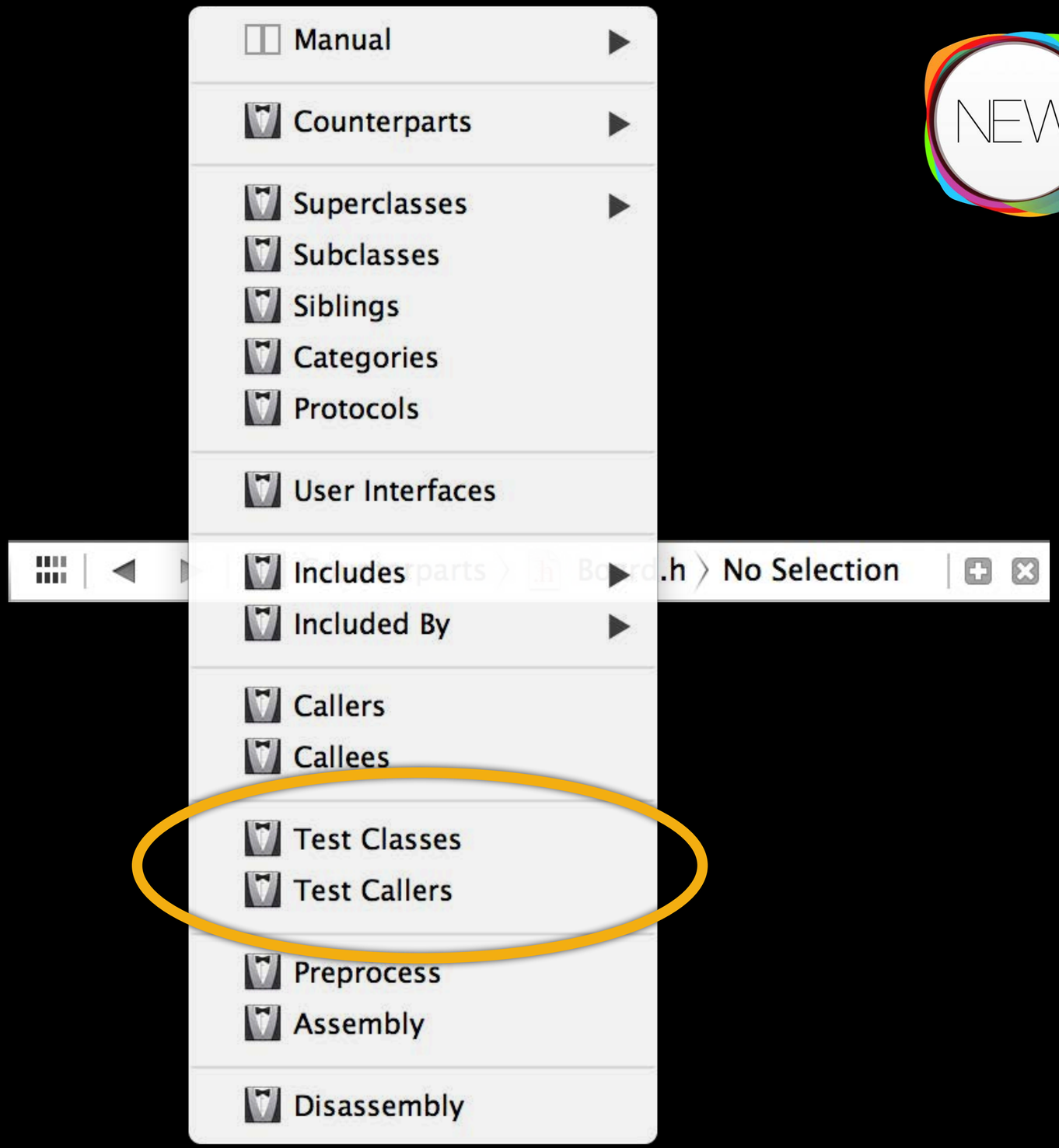
New test debugging UI

- Test breakpoint
- Test assistant categories

Overview

New test debugging UI

- Test breakpoint
- Test assistant categories



Overview

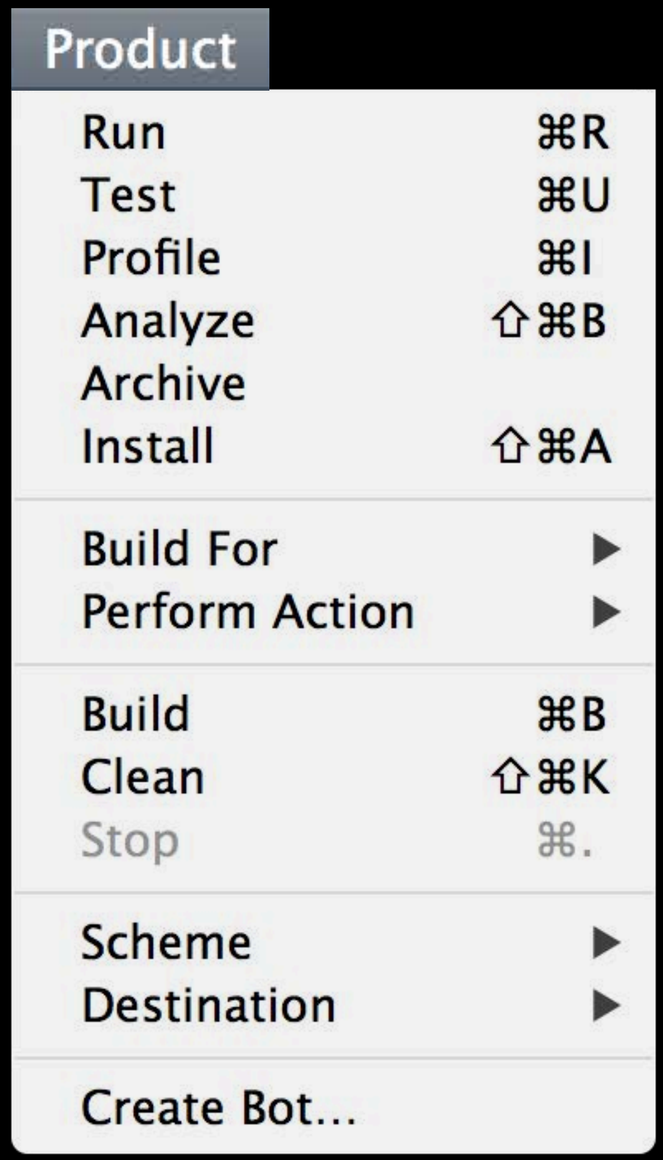
New test debugging UI

- Test breakpoint
- Test assistant categories
- Test Again command

Overview

New test debugging UI

- Test breakpoint
- Test assistant categories
- Test Again command

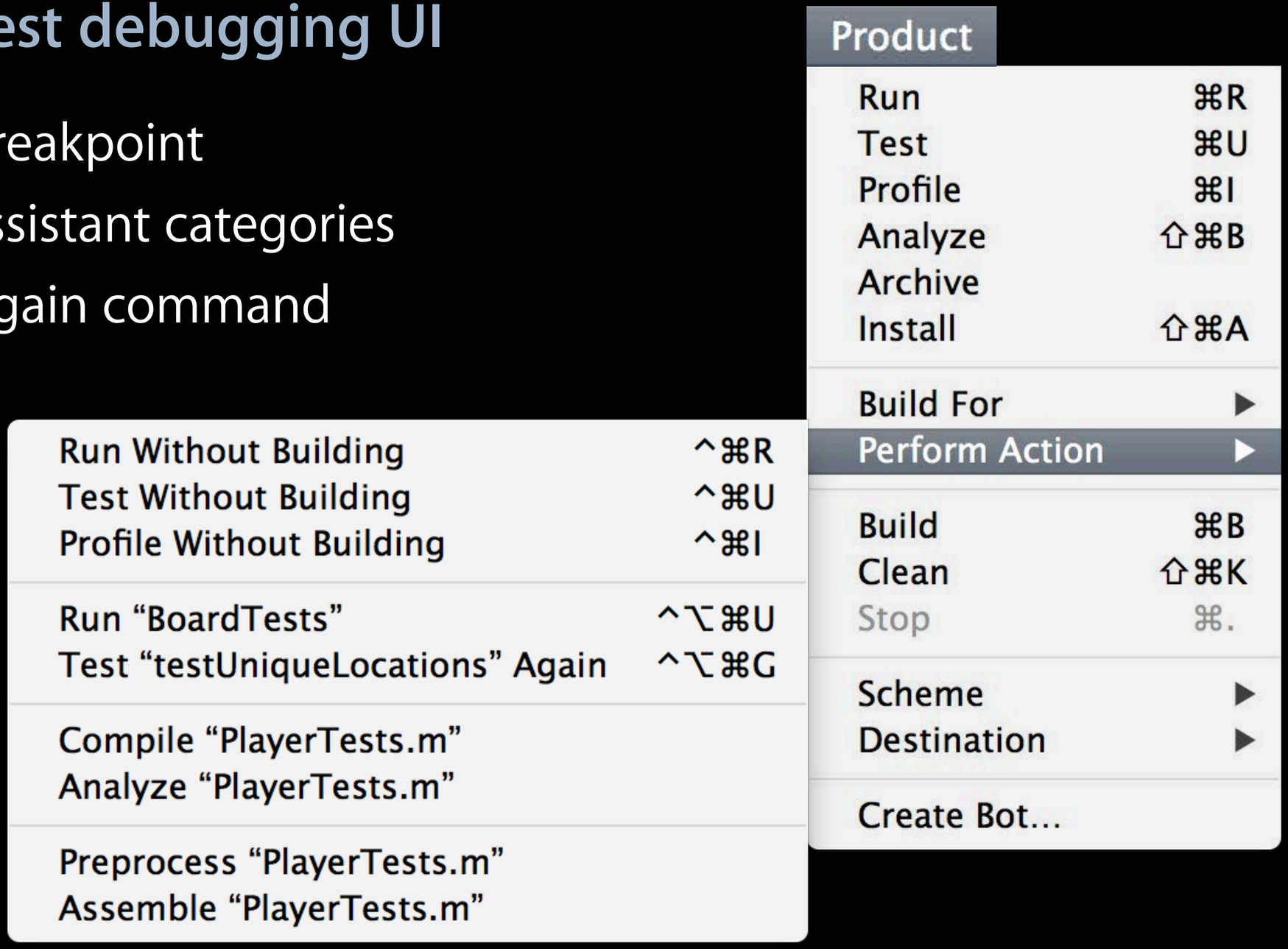


NEW

Overview

New test debugging UI

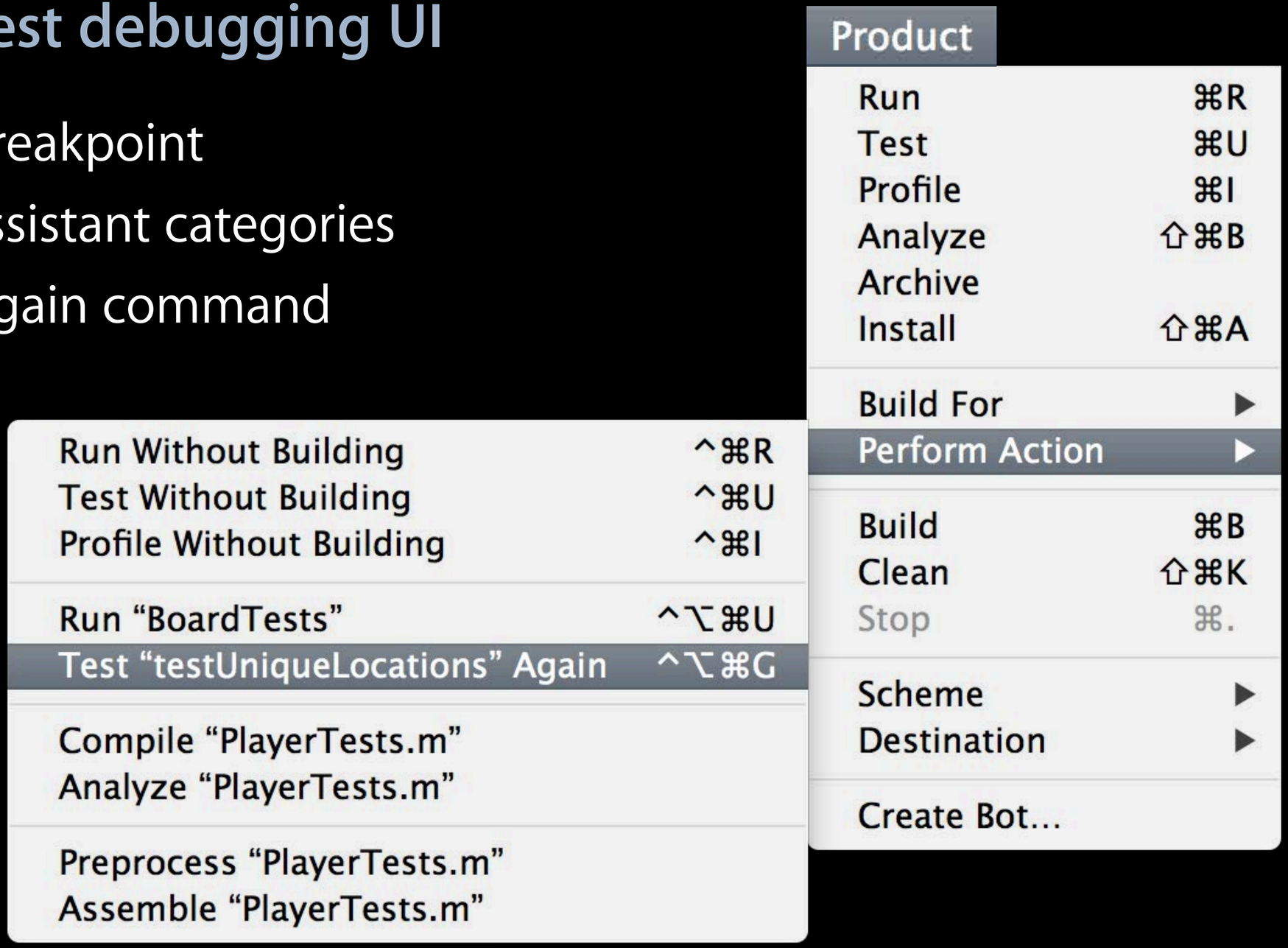
- Test breakpoint
- Test assistant categories
- Test Again command



Overview

New test debugging UI

- Test breakpoint
- Test assistant categories
- Test Again command



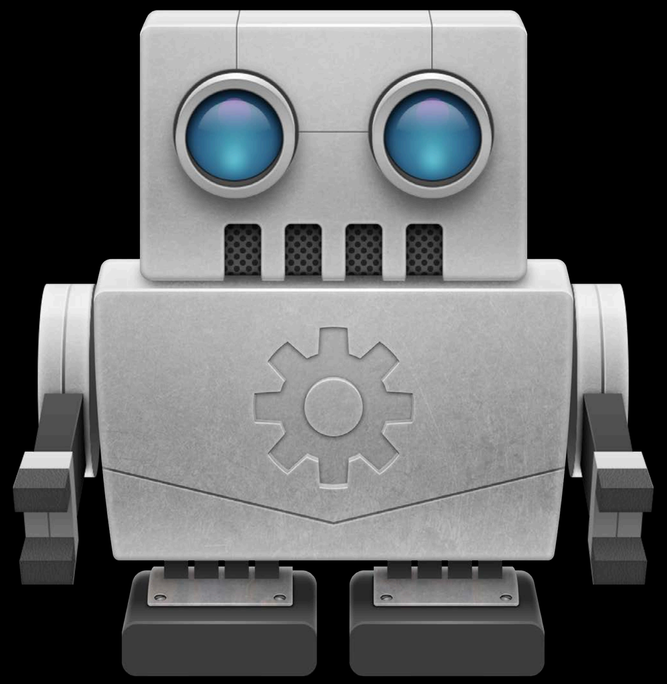
Continuous Integration and Testing

Testing Using OS X Server

NEW

Testing Using OS X Server

- Xcode service Bots perform integrations
  - Every time your commit code
  - On the hour



Testing Using OS X Server

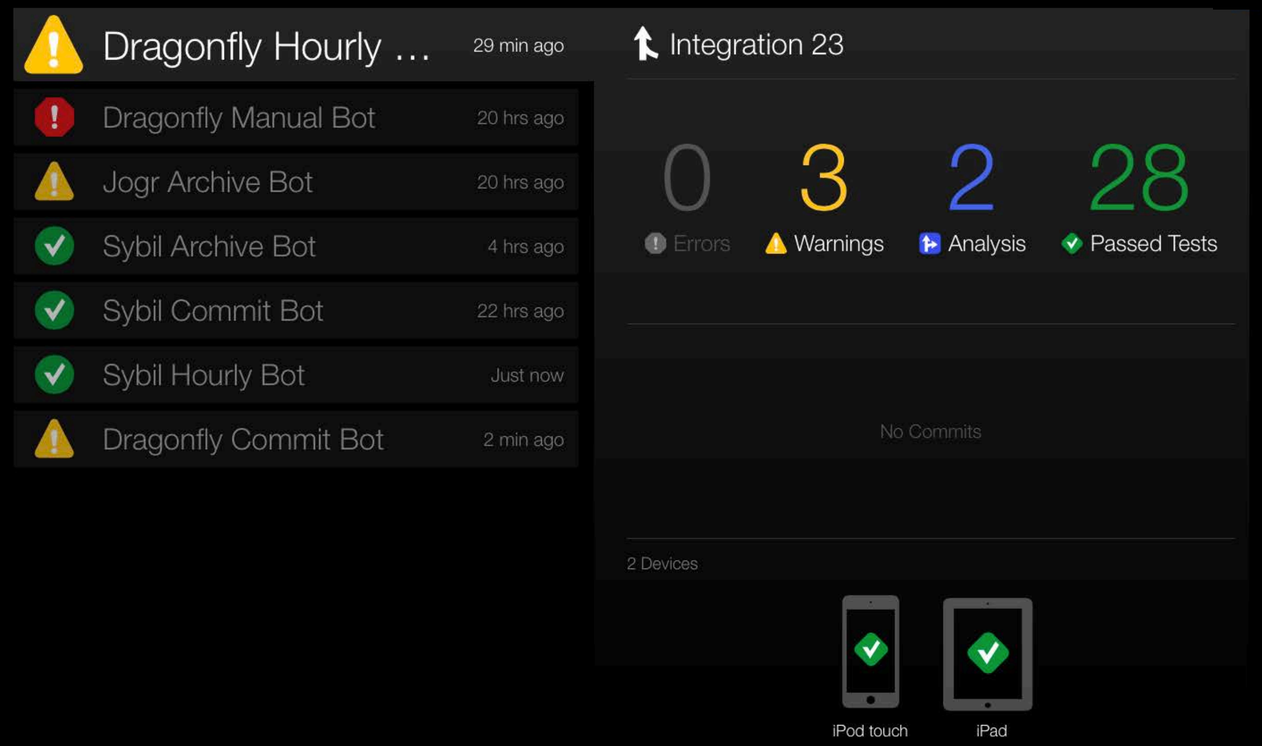
- Xcode service bots perform integrations
  - Every time your commit code
  - On the hour
- Create a Scheme
  - Bots run shared schemes

Testing Using OS X Server

- Xcode service bots perform integrations
  - Every time your commit code
  - On the hour
- Create a Scheme
  - Bots run shared schemes

Testing Using OS X Server

- Xcode service bots perform integrations
  - Every time your commit code
  - On the hour
- Create a Scheme
  - Bots run shared schemes
- Brings results to you



Many Configurations

Many Configurations

NEW

- iOS devices

Many Configurations

NEW

- iOS devices
- iOS simulator

Many Configurations

- iOS devices
- iOS simulator
- Different OS versions

Many Configurations

- iOS devices
- iOS simulator
- Different OS versions
- OS X

Demo

Continuous integration with OS X Server

Overview

Continuous integration with OS X Server

Overview

Continuous integration with OS X Server

- Setup shared schemes for bot

Overview

Continuous integration with OS X Server

- Setup shared schemes for bot
- Covers multiple configurations
  - Devices, OS versions, simulators

Overview

Continuous integration with OS X Server

- Setup shared schemes for bot
- Covers multiple configurations
  - Devices, OS versions, simulators
- Bot and test results summary

Command-line Testing

Command-line Testing


```
> xcodebuild test
```

Command-line Testing


```
> xcodebuild test
-scheme MyLibrary
```

Command-line Testing


```
> xcodebuild test
-scheme MyLibrary
-destination 'platform=OS X,arch=x86_64'
```

Command-line Testing


```
> xcodebuild test
-scheme MyLibrary
-destination 'platform=OS X,arch=x86_64'
-destination 'platform=iOS,name=My Development iPod Touch’
```

Command-line Testing


```
> xcodebuild test
-scheme MyLibrary
-destination 'platform=OS X,arch=x86_64'
-destination 'platform=iOS,name=My Development iPod Touch’
-destination 'platform=iOS Simulator,name=iPhone'
```

Command-line Testing


```
> xcodebuild test
-scheme MyLibrary
-destination 'platform=OS X,arch=x86_64'
-destination 'platform=iOS,name=My Development iPod Touch’
-destination 'platform=iOS Simulator,name=iPhone,OS=6.1'
```

Wrap Up

Wrap Up

- What is a unit test?

Wrap Up

- What is a unit test?
- Introducing XCTest

Wrap Up

- What is a unit test?
- Introducing XCTest
- Writing, running and debugging tests

Wrap Up

- What is a unit test?
- Introducing XCTest
- Writing, running and debugging tests
  - Test Navigator

Wrap Up

- What is a unit test?
- Introducing XCTest
- Writing, running and debugging tests
  - Test Navigator
  - Editor indicators

Wrap Up

- What is a unit test?
- Introducing XCTest
- Writing, running and debugging tests
  - Test Navigator
  - Editor indicators
  - Test failure breakpoint

Wrap Up

- What is a unit test?
- Introducing XCTest
- Writing, running and debugging tests
  - Test Navigator
  - Editor indicators
  - Test failure breakpoint
  - Assistant categories

Wrap Up

- What is a unit test?
- Introducing XCTest
- Writing, running and debugging tests
  - Test Navigator
  - Editor indicators
  - Test failure breakpoint
  - Assistant categories
  - Test Again command

Wrap Up

- What is a unit test?
- Introducing XCTest
- Writing, running and debugging tests
  - Test Navigator
  - Editor indicators
  - Test failure breakpoint
  - Assistant categories
  - Test Again command
- Continuous integration, advanced setups

Wrap Up

- What is a unit test?
- Introducing XCTest
- Writing, running and debugging tests
  - Test Navigator
  - Editor indicators
  - Test failure breakpoint
  - Assistant categories
  - Test Again command
- Continuous integration, advanced setups
- Testing helps you write better, high quality apps

More Information

Dave DeLong
Developer Tools Evangelist
delong@apple.com

Xcode Documentation
http://developer.apple.com/

Apple Developer Forums
http://devforums.apple.com

Related Sessions

Continuous Integration with Xcode 5	Presidio Tuesday 3:15PM

Related Labs

Xcode and Continuous Integration Lab	Tools Lab A Thursday 9:00AM
Tools Lab	Tools Lab Ongoing

WWDC2013