



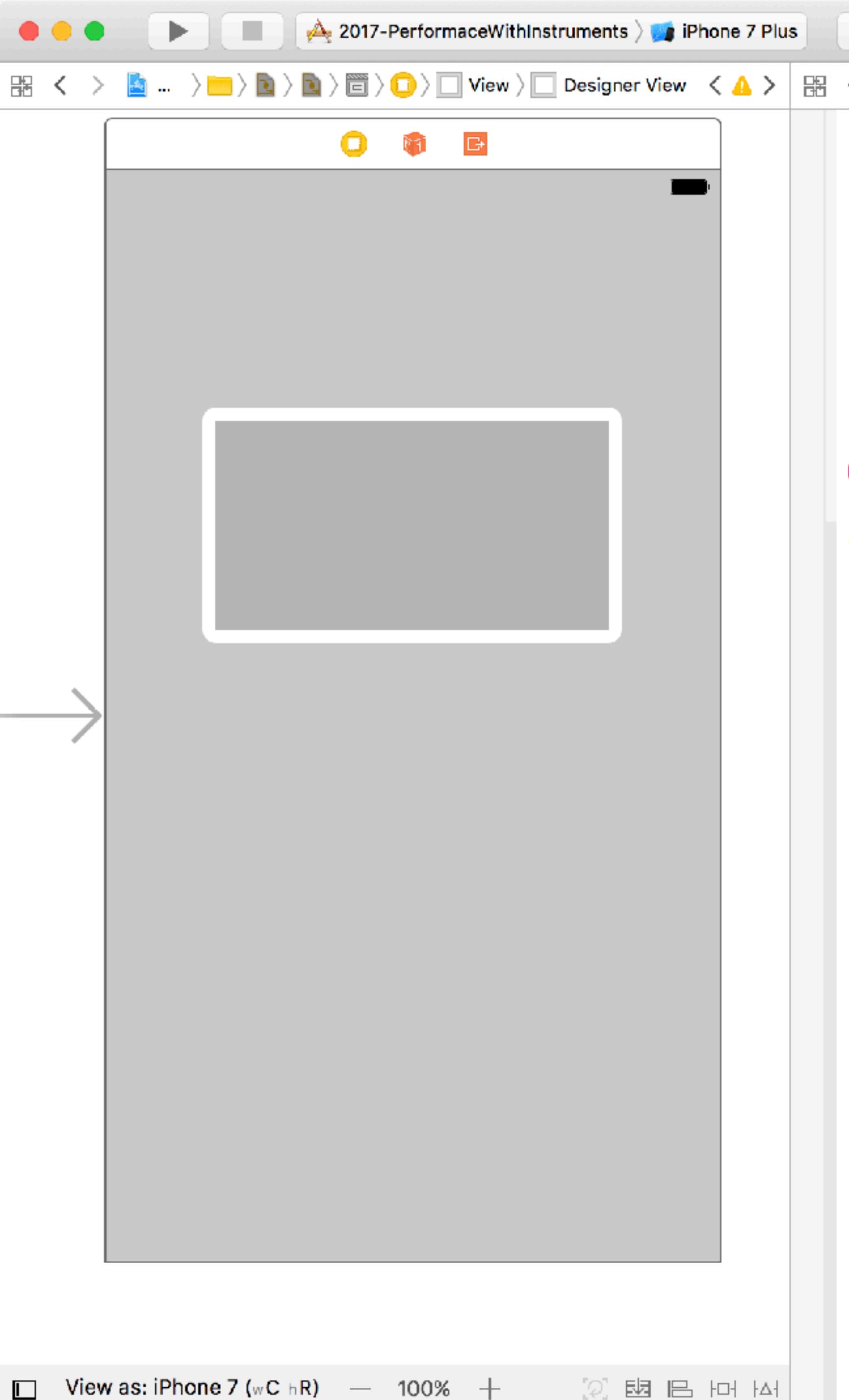
ADVANCED iOS APPLICATION DEVELOPMENT

MPCS 51032 • SPRING 2020 • SESSION 8

**CUSTOM DESIGNED
VIEWS**

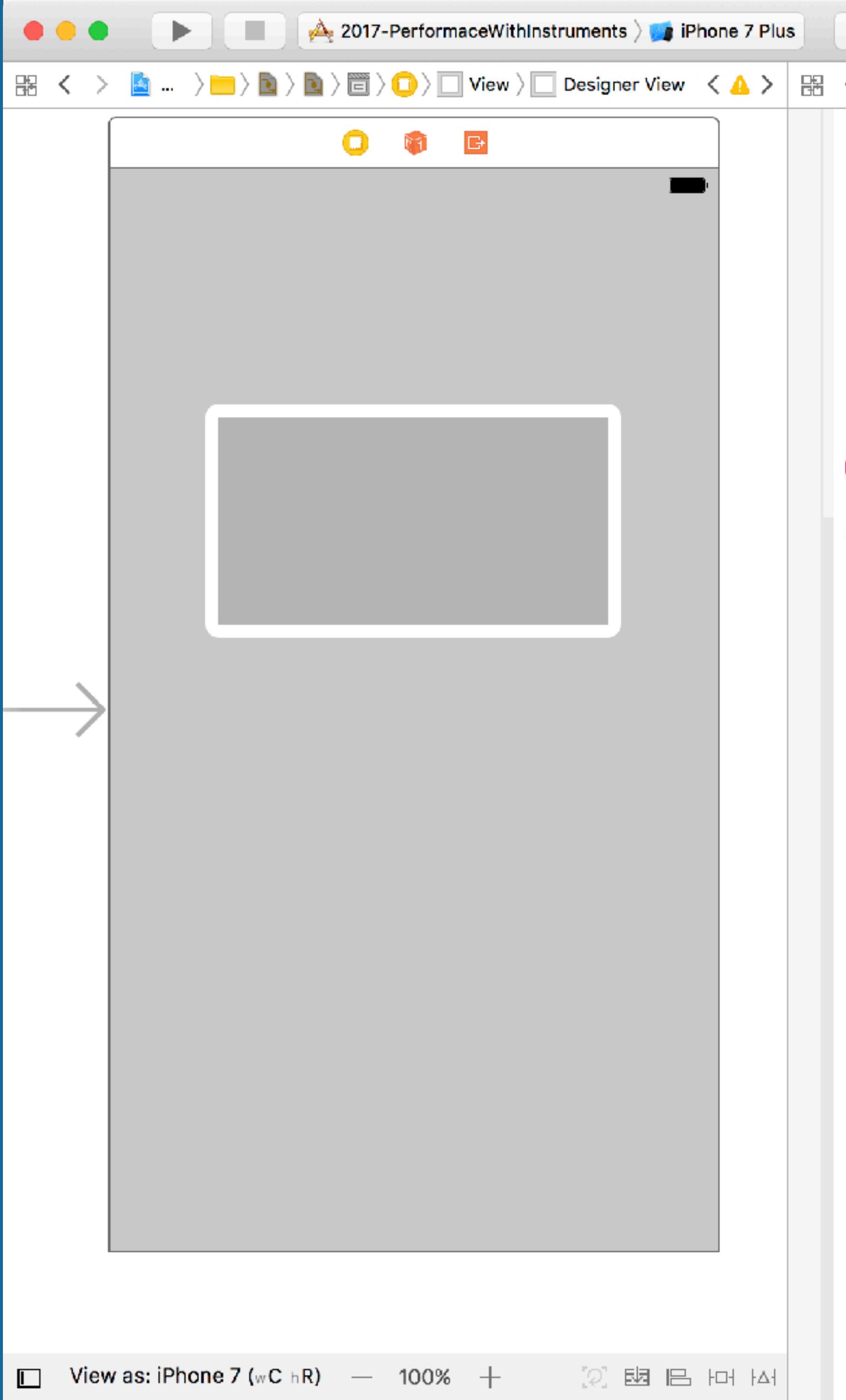
CUSTOM DESIGNED VIEWS

- Interface builder declarations that expose your custom attributes in Interface Builder
- Allows custom classes to be rendered in real time in Interface Builders without running



CUSTOM DESIGNED VIEWS

- `@IBInspectable` - adds interface to interface builder for custom views
- `@IBDesignable` - renders custom views in interface builder at design time



CUSTOM DESIGNED VIEWS

The screenshot shows the Xcode interface with the following details:

- Top Bar:** Shows the project name "2017-PerformaceWithInstruments", target "iPhone 7 Plus", and status "Finished running - Profiling 2017-PerformaceWithInstrument.. 2".
- Left Navigator:** Displays file paths: 2...ts > 2... > 2... > 2... > View > Designer View.
- Editor Area:** Shows the code for `DesignerView.swift`. The code includes comments about the file creation date (5/20/17) and copyright information, followed by the standard `import UIKit` statement and the class definition `class DesignerView: UIView { }`.
- Right Sidebar (Identity Inspector):** Contains settings for the selected view:
 - View:** Content Mode: Scale To Fill, Semantic: Unspecified, Tag: 0.
 - Interaction:** User Interaction Enabled (checked), Multiple Touch (unchecked).
 - Alpha:** 1.
 - Background:** Light Gray Color.
 - Tint:** Default.
 - Drawing:** Opaque (checked), Hidden (unchecked), Clears Graphics Context (checked), Clip To Bounds (unchecked), Autoresize Subviews (checked).
 - Stretching:** X: 0, Y: 0.
 - Width:** 1, **Height:** 1.
 - Installed:** Checked.

CUSTOM DESIGNED VIEWS

The screenshot shows the Xcode interface with a custom designed view in the storyboard. A blue arrow points from the storyboard preview to the code editor.

Storyboard Preview: A light gray square placeholder representing the custom view.

Code Editor:

```
//  
// Created by T. Andrew Binkowski on 5/20/17.  
// Copyright © 2017 T. Andrew Binkowski. All rights reserved.  
  
import UIKit  
  
@IBDesignable  
  
class DesignerView: UIView {  
    @IBInspectable var color: UIColor = UIColor.clear {  
        didSet {  
            self.backgroundColor = color  
        }  
    }  
}
```

Attributes Inspector:

- Content Mode: Scale To Fill
- Semantic: Unspecified
- Tag: 0
- Interaction:
 - User Interaction Enabled
 - Multiple Touch
- Alpha: 1
- Background: Light Gray Color
- Tint: Default
- Drawing:
 - Opaque
 - Hidden
 - Clears Graphics Context
 - Clip To Bounds
 - Autoresizes Subviews
- Stretching:
 - X: 0
 - Y: 0
 - Width: 1
 - Height: 1
- Installed:

CUSTOM DESIGNED VIEWS

SUBTITLE

- Valid types to be inspectable
- They do not have to be visible

- Int
- CGFloat
- Double
- String
- Bool
- CGPoint
- CGSize
- CGRect
- UIColor
- UIImage

CUSTOM DESIGNED VIEWS

The screenshot shows the Xcode interface with the following details:

- Project Bar:** 2017-PerformanceWithInstruments > iPhone 7 Plus
- Status Bar:** 2017-PerformanceWithInstruments | Build Succeeded
- Editor Area:** DesignerView.swift file open.
- Code Preview:** A preview window on the left shows a gray square with rounded corners and a thin white border.
- Inspector:** On the right, the "Designer View" tab is selected, showing properties for the view:
 - Color: Light Gray Color
 - Border Color: Default
 - Border Width: 0
 - Corner Radius: 0
- View Properties:** Content Mode: Scale To Fill, Semantic: Unspecified, Tag: 0, Interaction: User Interaction Enabled (checked), Alpha: 1, Background: Light Gray Color, Tint: Default, Drawing: Opaque (checked), Clear Graphics Context: checked, Clip To Bounds: unchecked, Autoresize Subviews: checked, Stretching: X: 0, Y: 0, Width: 1, Height: 1, Installed: checked.
- Code Content:**

```
// DesignerView.swift
// 2017-PerformanceWithInstruments
//
// Created by T. Andrew Binkowski on 5/20/17.
// Copyright © 2017 T. Andrew Binkowski. All rights reserved.

import UIKit

@IBDesignable

class DesignerView: UIView {
    @IBInspectable var color: UIColor = UIColor.clear {
        didSet {
            self.backgroundColor = color
        }
    }

    @IBInspectable var borderColor: UIColor = UIColor.white {
        didSet {
            self.layer.borderColor = borderColor.cgColor
        }
    }

    @IBInspectable var borderWidth: CGFloat = 1.0 {
        didSet {
            self.layer.borderWidth = borderWidth
        }
    }

    @IBInspectable
    public var cornerRadius: CGFloat = 2.0 {
        didSet {
            self.layer.cornerRadius = cornerRadius
        }
    }
}
```

CUSTOM DESIGNED VIEWS

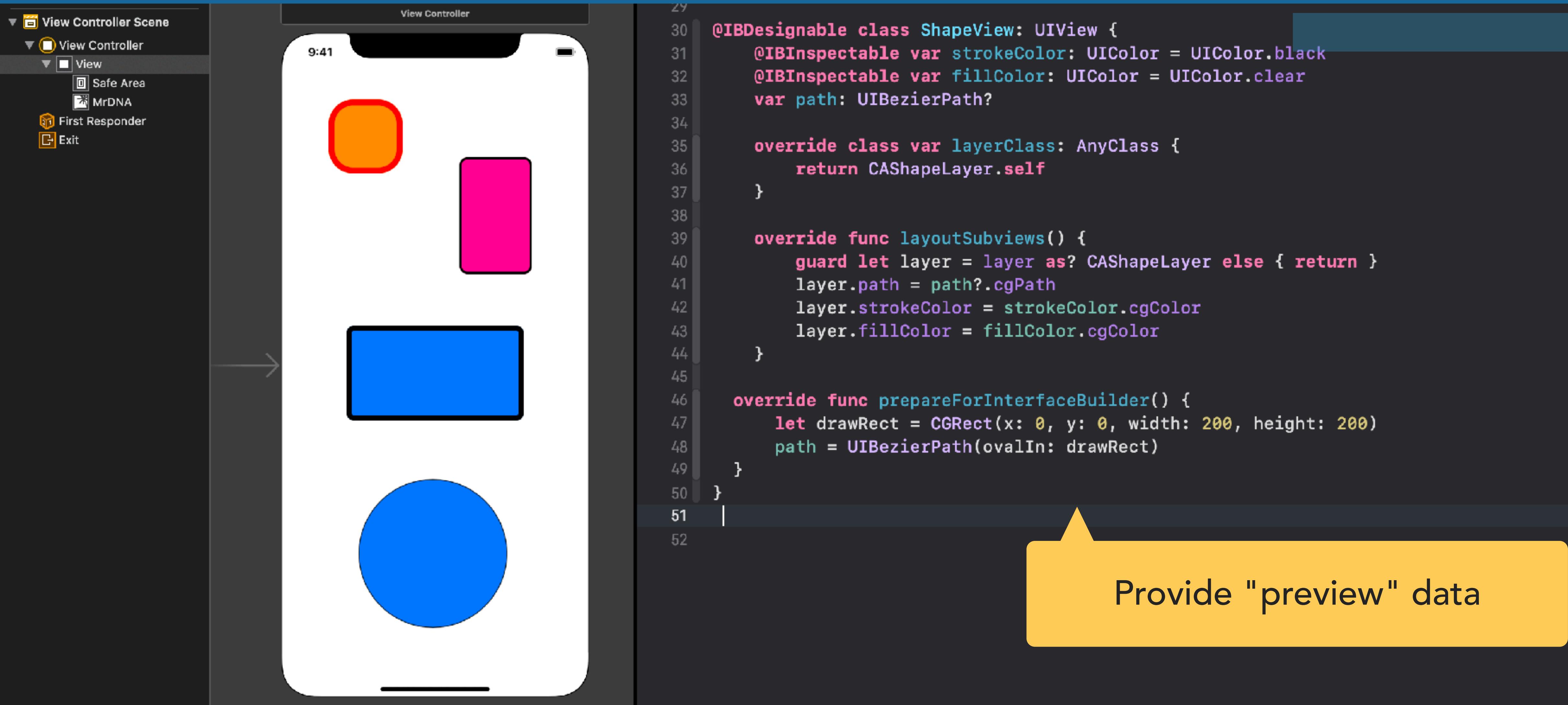
Inspectable as Bool

The screenshot shows the Xcode interface with a project named "2018-session-7-designable-inspectable". The storyboard on the left displays a view controller containing a circular image view with a cartoon character. The code editor in the center shows the implementation of a custom class, CircleImageView, which inherits from UIImageView and adds rounded corner functionality via a @IBInspectable var rounded: Bool property. The right side of the screen is occupied by the Xcode Inspector, specifically the "View" inspector, which is highlighted with a yellow box. This inspector shows various properties for the selected view, including "Rounded" set to "On", "Image" set to "MrDNA", and "Content Mode" set to "Aspect Fit". A callout arrow points from the text "Inspectable as Bool" at the top right towards the "Rounded" property in the inspector.

```
5 // Created by T. Andrew Binkowski on 5/26/20.
6 // Copyright © 2020 T. Andrew Binkowski. All rights reserved.
7 //
8
9 import UIKit
10
11 @IBDesignable
12 class CircleImageView: UIImageView {
13
14     @IBInspectable public var rounded: Bool = false {
15         didSet {
16             if rounded {
17                 self.layer.cornerRadius = 125
18             } else {
19                 self.layer.cornerRadius = 0
20             }
21             self.setNeedsDisplay()
22         }
23     }
24 }
25
```

Circle Image View
Rounded On
Image MrDNA
Highlighted Highlighted Image
State Highlighted
Accessibility Adjusts Image Size
Symbol Configuration
Configuration Unspecified
Scale Unspecified
Weight Unspecified
View
Content Mode Aspect Fit
Semantic Unspecified
Tag 0
Interaction User Interaction Enabled
Multiple Touch
Alpha 1
Background Default
Tint Default
Drawing Opaque
Hidden
Clears Graphics Context
Clip to Bounds
Autoresize Subviews
Stretching 0 0
X 1 Y 1

CUSTOM DESIGNED VIEWS



The image shows a screenshot of the Xcode interface. On the left, the Project Navigator displays a file named "View Controller Scene". The main area shows a storyboard with a single view controller containing a view with three subviews: a red rounded rectangle, a pink rectangle, and a blue rectangle. A large blue circle is also present at the bottom. The status bar indicates it's 9:41. On the right, the code editor shows the implementation of a custom view class, "ShapeView". The code includes properties for stroke and fill colors, a layer class (CAShapeLayer), and methods for layout and preparation. A yellow callout bubble with the text "Provide 'preview' data" points to the code.

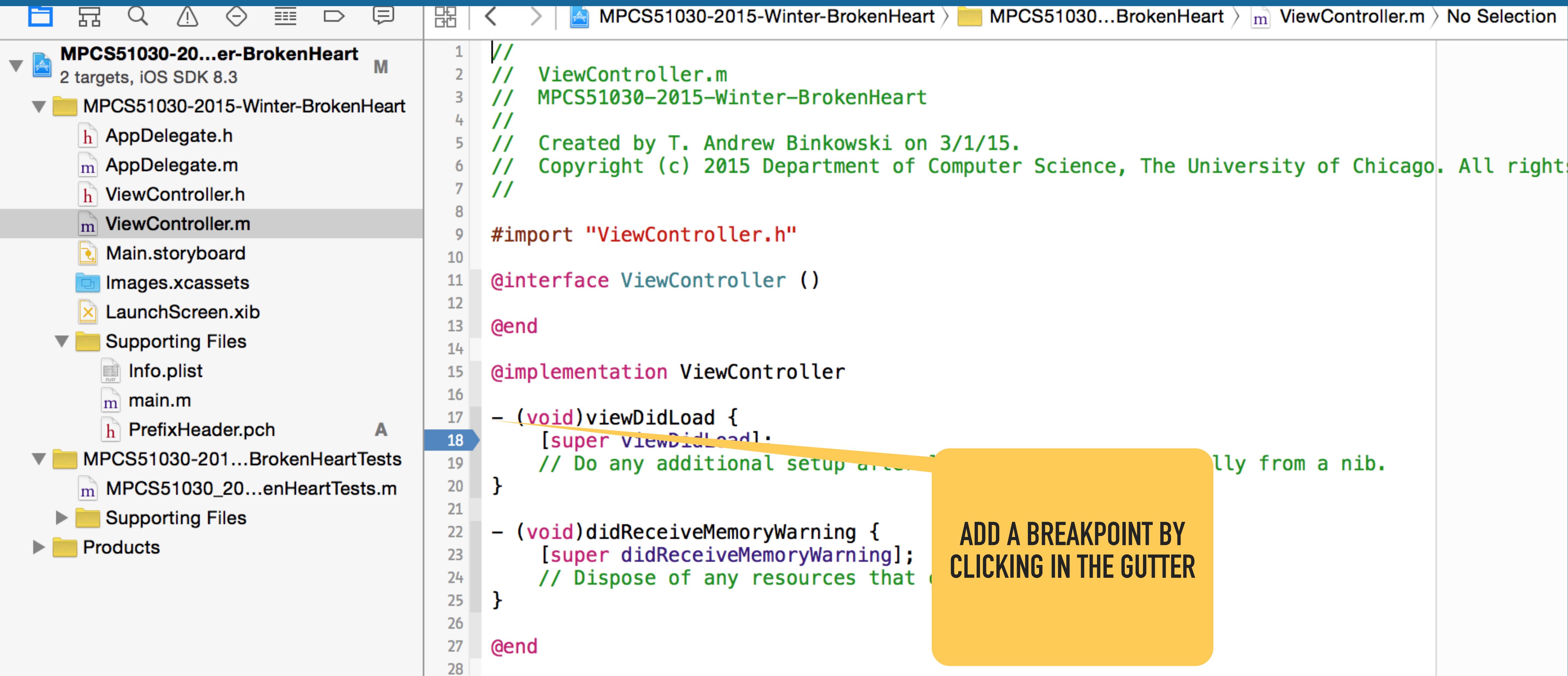
```
29
30 @IBDesignable class ShapeView: UIView {
31     @IBInspectable var strokeColor: UIColor = UIColor.black
32     @IBInspectable var fillColor: UIColor = UIColor.clear
33     var path: UIBezierPath?
34
35     override class var layerClass: AnyClass {
36         return CAShapeLayer.self
37     }
38
39     override func layoutSubviews() {
40         guard let layer = layer as? CAShapeLayer else { return }
41         layer.path = path?.cgPath
42         layer.strokeColor = strokeColor.cgColor
43         layer.fillColor = fillColor.cgColor
44     }
45
46     override func prepareForInterfaceBuilder() {
47         let drawRect = CGRect(x: 0, y: 0, width: 200, height: 200)
48         path = UIBezierPath(ovalIn: drawRect)
49     }
50 }
51
52
```

Provide "preview" data

DEBUGGING



DEBUGGING



The screenshot shows the Xcode interface with the following details:

- Project Navigator:** Shows the project structure under "MPC51030-20...er-BrokenHeart". The "ViewController.m" file is selected.
- Editor:** Displays the "ViewController.m" code. A blue arrow points from the gutter at line 18 to a yellow callout bubble.
- Code:**

```
// ViewController.m
// MPC51030-2015-Winter-BrokenHeart
//
// Created by T. Andrew Binkowski on 3/1/15.
// Copyright (c) 2015 Department of Computer Science, The University of Chicago. All rights reserved.

#import "ViewController.h"

@interface ViewController : UIViewController

@end

@implementation ViewController

- (void)viewDidLoad {
    [super viewDidLoad];
    // Do any additional setup after loading the view.
}

- (void)didReceiveMemoryWarning {
    [super didReceiveMemoryWarning];
    // Dispose of any resources that can be recreated.
}

@end
```

A yellow callout bubble with the text "ADD A BREAKPOINT BY CLICKING IN THE GUTTER" is positioned over the gutter at line 18.

DEBUGGING



The screenshot shows the Xcode interface with the following details:

- Project Navigator:** Shows the project structure for "MPCS51030-20...er-BrokenHeart". The "ViewController.m" file is selected.
- File Navigator:** Shows files like AppDelegate.h, AppDelegate.m, ViewController.h, ViewController.m, Main.storyboard, Images.xcassets, LaunchScreen.xib, Info.plist, main.m, and PrefixHeader.pch.
- Editor:** Displays the content of ViewController.m. The code includes comments about the file being created by T. Andrew Binkowski on 3/1/15 and copyright information for the University of Chicago.
- Breakpoint Context Menu:** A context menu is open at line 18, listing the following options:
 - Edit Breakpoint...
 - Disable Breakpoint
 - Delete Breakpoint
 - Reveal in Breakpoint Navigator
- Toolbar:** Standard Xcode toolbar icons for file operations.
- Navigation Bar:** Shows the current project path: MPCS51030-2015-Winter-BrokenHeart > MPCS51030...BrokenHeart > ViewController.m > No Selection.

DEBUGGING



The screenshot shows the Xcode interface with the following details:

- Project Navigator:** Shows the project structure for "MPCS51030-20...er-BrokenHeart".
- File Navigator:** Shows files like AppDelegate.h, AppDelegate.m, ViewController.h, ViewController.m, Main.storyboard, Images.xcassets, LaunchScreen.xib, Info.plist, main.m, and PrefixHeader.pch.
- Editor:** Displays the content of ViewController.m. The code includes comments about the file being created by T. Andrew Binkowski on 3/1/15 and copyright information for the University of Chicago.
- Breakpoint Context Menu:** A context menu is open at line 18, listing the following options:
 - Edit Breakpoint...
 - Disable Breakpoint
 - Delete Breakpoint
 - Reveal in Breakpoint Navigator

DEBUGGING



Jenny Tank <jennytank@gmail.com>

In Project Ignoring Case

Name AppDelegate.swift
Type Default - Swift Source
Location Relative to Group
Full Path /Users/tabinkowski/Google Drive/g-Teaching/uchicago.mobi/uchicago.mobi-courses/mpcs51030/mpcs51030-2018-winter/mpcs51030-demonstrations/session10-splishsplash-privacy/session10-splishsplash/AppDelegate.swift

```
12 class AppDelegate: UIResponder, UIApplicationDelegate {  
13  
14     var window: UIWindow?  
15     var launchFromTerminated = false  
16  
17     func application(_ application: UIApplication,  
18                         didFinishLaunchingWithOptions launchOptions: [UIApplicationLaunchOptionsKey: Any]?) -> Bool {  
19         self.launchFromTerminated = true  
20         return true  
21     }  
22  
23     func applicationWillEnterBackground(_ application: UIApplication) {  
24         print("Entering background")  
25         showSplashScreen(autoDismiss: false, label: "••")  
26     }  
27  
28     func applicationWillBecomeActive(_ application: UIApplication) {  
29         print("Did become active")  
30         if launchFromTerminated {  
31             showSplashScreen(autoDismiss: false, label: "Splash")  
32             launchFromTerminated = false  
33         }  
34     }  
35  
36 }
```

Session: session10-splishsplash | Thread 1 | 0 AppDelegate.applicationDidBecomeActive(_)

▶ A application = (UIApplication) 0x00007fd945901580
▶ A self = (session10_splishsplash.AppDelegate) 0x000060800003a500

(lldb) p launchFromTerminated
(Bool) \$R2 = true
(lldb) po launchFromTerminated
true
(lldb)

No Matches

DEBUGGING



Remove by dragging or deleting

```
12 class AppDelegate: UIResponder, UIApplicationDelegate {
13
14     var window: UIWindow?
15     var launchFromTerminated = false
16
17     func application(application: UIApplication,
18                      didFinishLaunchingWithOptions launchOptions: [UIApplicationLaunchOptionsKey: Any]?) -> Bool {
19         launchFromTerminated = true
20
21         return true
22     }
23
24     func applicationWillEnterBackground(_ application: UIApplication) {
25         print("Entering background")
26         showSplashScreen(autoDismiss: false, label: "00")
27     }
28
29     func applicationDidBecomeActive(_ application: UIApplication) {
30         print("Did become active")
31         if launchFromTerminated {
32             showSplashScreen(autoDismiss: false, label: "Splash")
33             launchFromTerminated = false
34         }
35     }
36 }
```

Thread 1: breakpoint 2.1

application = (UIApplication) 0x00007fd945901580
self = (session10_splishsplash.AppDelegate) 0x000060800003a500

(lldb) p launchFromTerminated
(Bool) \$R2 = true
(lldb) po launchFromTerminated
true

(lldb)

No Matches

VARIABLES AND CONSOLE

VARIABLES VIEW

CPU 0%
Memory 17.1 MB
Disk Zero KB/s
Network Zero KB/s

Thread 1 Queue: com.apple.main-thread (serial)

- 0 -[ViewController viewDidAppear:]
- 1 -[UIViewController _setViewApp...
- 12 UIApplicationMain
- 13 main
- 14 start

Thread 2 Queue: com.apple.libdispatch-mana...

Thread 3

Thread 4 Queue: FBSSerialQueue (serial)

Thread 5

```
46 self.hearts = [NSMutableArray arrayWithCapacity:255];
47 for (int i=0; i<255; i++) {
48     CGRect frame = CGRectMake(i+5, i+5, 20, 20);
49
50     // Create a heart object and store the frame and color
51     Heart *heart = [[Heart alloc] initWithFrame:frame];
52     heart.backgroundColor = RGB(i, 0, 0);
53     heart.tag = i;
54     [self.hearts addObject:heart];
55 }
56
57 - (void)viewDidAppear:(BOOL)animated
58 {
59     // Add the hearts
60     for (Heart *heart in self.hearts) {
61         [self.view addSubview:heart];
62     }
63 }
64
65 }
66
67
68 @end
69
```

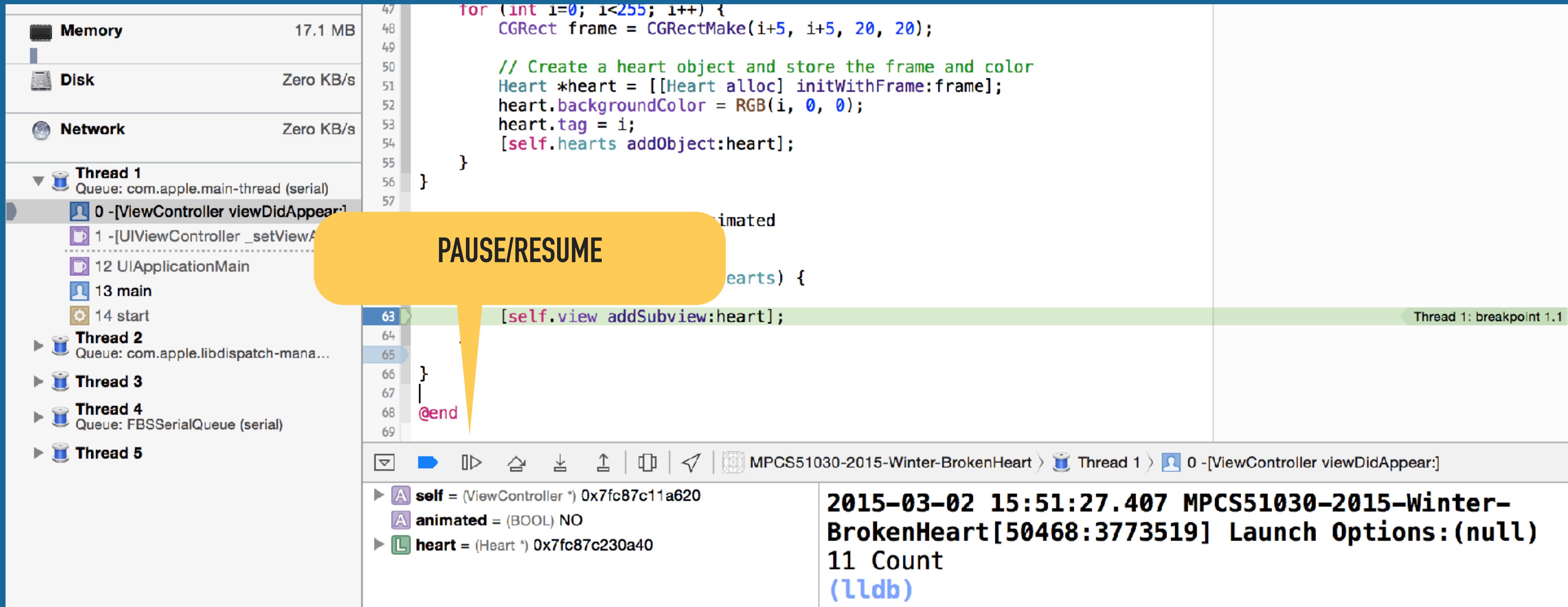
Variable view

MPCS51030-2015-Winter-BrokenHeart > Thread 1 > 0 -[ViewController viewDidAppear:]

► A self = (ViewController *) 0x7fc87c11a620	2015-03-02 15:51:27.407 MPCS51030-2015-Winter-
► A animated = (BOOL) NO	BrokenHeart[50468:3773519] Launch Options:(null)
► L heart = (Heart *) 0x7fc87c230a40	11 Count

(lldb)

VARIABLES VIEW



Memory 17.1 MB
Disk Zero KB/s
Network Zero KB/s

Thread 1
Queue: com.apple.main-thread (serial)
0 -[ViewController viewDidAppear:]
1 -[UIViewController _setViewAnimated:
12 UIApplicationMain
13 main
14 start

Thread 2
Queue: com.apple.libdispatch-man...

Thread 3

Thread 4
Queue: FBSSerialQueue (serial)

Thread 5

```
for (int i=0; i<255; i++) {  
    CGRect frame = CGRectMake(i+5, i+5, 20, 20);  
  
    // Create a heart object and store the frame and color  
    Heart *heart = [[Heart alloc] initWithFrame:frame];  
    heart.backgroundColor = RGB(i, 0, 0);  
    heart.tag = i;  
    [self.hearts addObject:heart];  
}  
  
[self.view addSubview:heart];  
}
```

PAUSE/RESUME

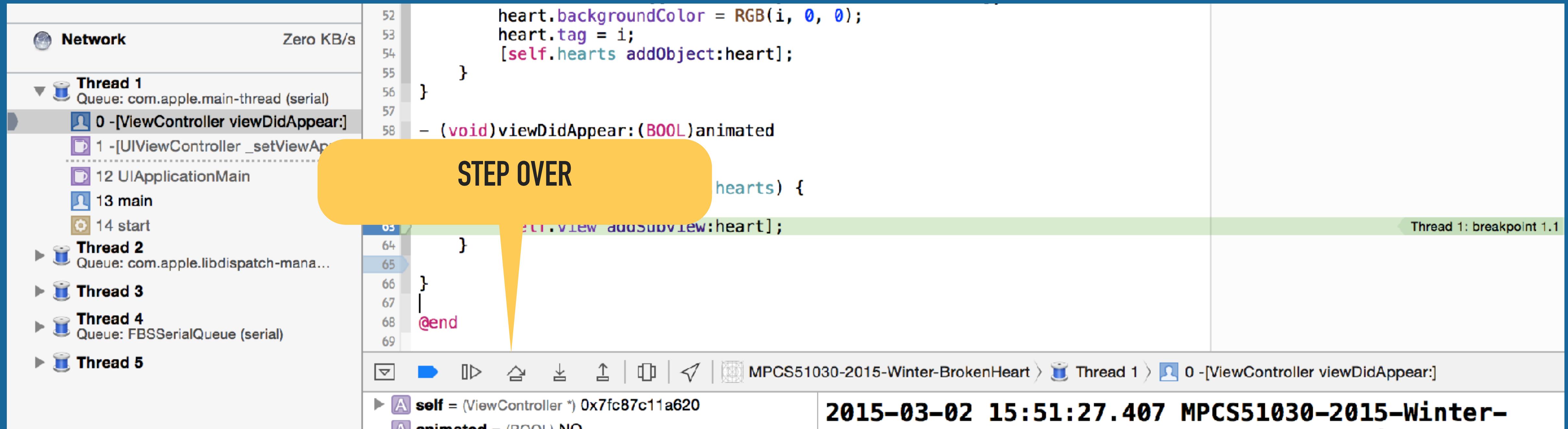
MPCS51030-2015-Winter-BrokenHeart > Thread 1 > 0 -[ViewController viewDidAppear:]

▶ A self = (ViewController *) 0x7fc87c11a620
▶ A animated = (BOOL) NO
▶ L heart = (Heart *) 0x7fc87c230a40

2015-03-02 15:51:27.407 MPKS51030-2015-Winter-BrokenHeart[50468:3773519] Launch Options:(null) 11 Count (lldb)

- Pause or resume execution of code

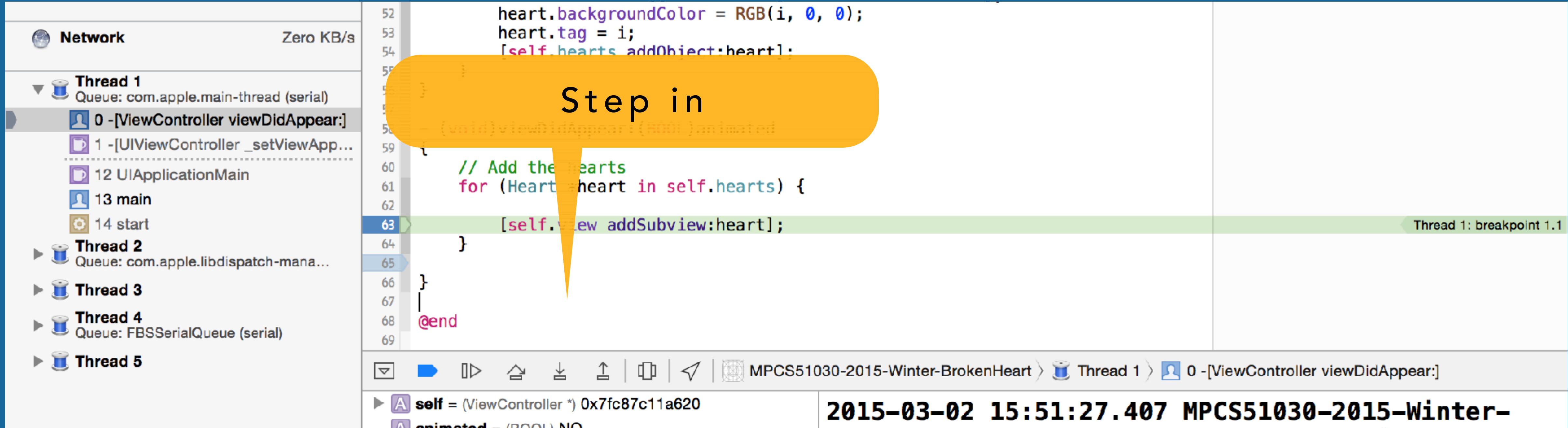
VARIABLES VIEW



The screenshot shows the Xcode debugger interface. On the left, the Network status is "Zero KB/s". Thread 1 is selected, showing the stack trace: 0 -[ViewController viewWillAppear:], 1 -[UIViewController _setViewAp..., 12 UIApplicationMain, 13 main, 14 start. Thread 2, 3, 4, and 5 are also listed. The main area shows the code for `viewWillAppear:`:
52 heart.backgroundColor = RGB(i, 0, 0);
53 heart.tag = i;
54 [self.hearts addObject:heart];
55 }
56 }
57 – (void)viewWillAppear:(BOOL)animated
58
59 for (Heart *heart in self.hearts) {
60 heart.backgroundColor = RGB(i, 0, 0);
61 heart.tag = i;
62 [heart.view addSubview:heart];
63 }
64 }
65 }
66 }
67 @end
68
69
A yellow callout bubble with the text "STEP OVER" is positioned over the line of code at line 59. The status bar at the bottom right indicates "Thread 1: breakpoint 1.1". The bottom navigation bar includes icons for step over, step into, step out, and back.

- Step over
 - Execute the current line of code
 - If the statement is a function call, the function will be executed
 - Debugger will stop at the statement after the function call

VARIABLES VIEW



The screenshot shows the Xcode debugger interface. On the left, the Threads tab displays five threads: Network (Zero KB/s), Thread 1 (com.apple.main-thread (serial)), Thread 2 (com.apple.libdispatch-man...), Thread 3, Thread 4 (FBSSerialQueue (serial)), and Thread 5. Thread 1 is selected, showing the stack trace: 0 -[ViewController viewDidAppear:] → 1 -[UIViewController _setViewApp...]. The main area shows the code for Thread 1:

```
52     heart.backgroundColor = RGB(i, 0, 0);
53     heart.tag = i;
54     [self.hearts addObject:heart];
55 }
56 - (void)viewDidAppear:(BOOL)animated
57 {
58     // Add the hearts
59     for (Heart *heart in self.hearts) {
60         [self.view addSubview:heart];
61     }
62 }
63 }
64 }
65 }
66 }
67 }
68 @end
69 }
```

A yellow callout bubble with the text "Step in" is positioned over the line of code at line 63. The line 63 is highlighted with a green background. A blue arrow points from the bottom of the callout bubble towards the line 63. The status bar at the bottom right indicates "Thread 1: breakpoint 1.1". The bottom status bar also shows the date and time: "2015-03-02 15:51:27.407 MPCS51030-2015-Winter-".

- Step in
 - Execute the current line of code
 - If it is a routine jump to its first line
 - The debugger will stop at the first statement inside the function

VARIABLES VIEW

The screenshot shows the Xcode debugger interface. On the left, the Thread Navigator lists threads: Network (Zero KB/s), Thread 1 (com.apple.main-thread (serial)), Thread 2 (com.apple.libdispatch-man...), Thread 3, Thread 4 (FBSSerialQueue (serial)), and Thread 5. Thread 1 is selected, showing the stack trace: 0 -[ViewController viewDidAppear:] (highlighted), 1 -[UIViewController _setViewApp...], 12 UIApplicationMain, 13 main, 14 start. The main area shows the code for `viewDidAppear:`:

```
52     heart.backgroundColor = RGB(i, 0, 0);
53     heart.tag = i;
54     [self.hearts addObject:heart];
55 }
56 }
57 -(void)viewDidAppear:(BOOL)animated
58 {
59 // Add the hearts
60 for (Heart *heart in self.hearts) {
61     [self.view addSubview:heart];
62 }
63 }
64 }
65 }
66 }
67 }
68 @end
69 }
```

A yellow callout bubble with the text "Step in" points to the line `[self.view addSubview:heart];`. The status bar at the bottom right indicates "Thread 1: breakpoint 1.1". The bottom status bar also shows the date and time: "2015-03-02 15:51:27.407 MPC51030-2015-Winter-".

- Step into and Step over are indistinguishable if the present statement is not a function call

VARIABLES VIEW

The screenshot shows the Xcode debugger interface. On the left is the Stack View, listing threads and their stack frames. Thread 1 is active, showing the execution path from `viewDidAppear:` down to `[self.view addSubview:heart];`. A yellow callout bubble with the text "Step out" points to this line. The main area is the Code View, displaying the source code with line numbers 52 through 69. A green bar at the bottom indicates a breakpoint at line 63. The bottom right shows the date and time: 2015-03-02 15:51:27.407. The bottom status bar displays variable values: `self = (ViewController *) 0x7fc87c11a620` and `animated = (BOOL) NO`.

```
52     heart.backgroundColor = RGB(i, 0, 0);
53     heart.tag = i;
54     [self.hearts addObject:heart];
55 }
56 }
57
58 - (void)viewDidAppear:(BOOL)animated
59 {
60     // Add the hearts
61     for (Heart *heart in self.hearts) {
62         [self.view addSubview:heart];
63     }
64 }
65 }
66 }
67 }
68 @end
69 }
```

Thread 1: breakpoint 1.1

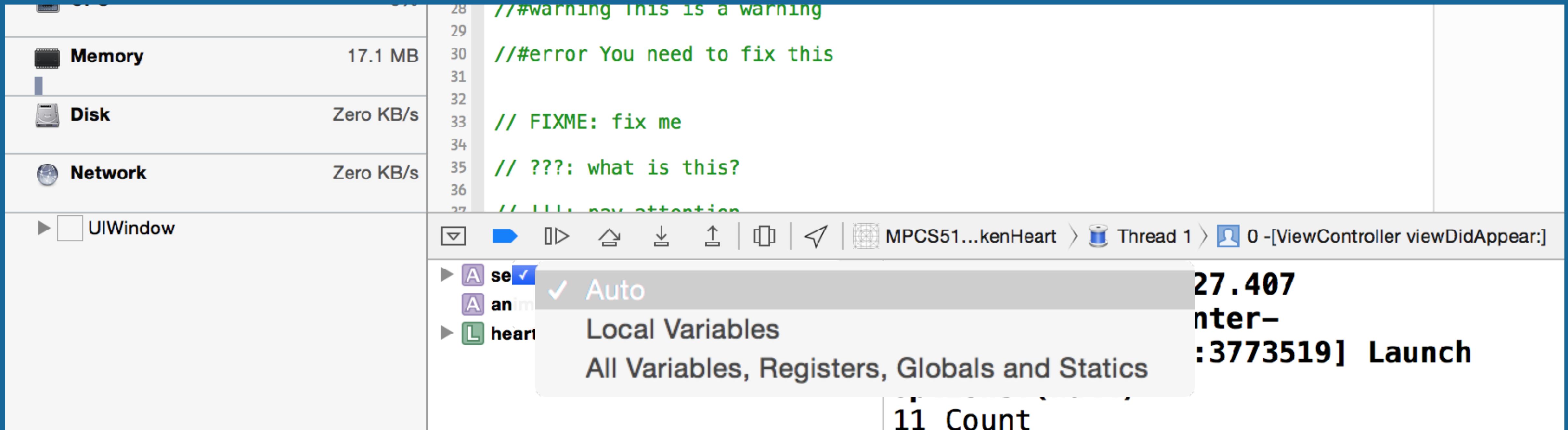
MPCS51030-2015-Winter-BrokenHeart > Thread 1 > 0 -[ViewController viewDidAppear:]

A self = (ViewController *) 0x7fc87c11a620
A animated = (BOOL) NO

2015-03-02 15:51:27.407 MPCS51030-2015-Winter-

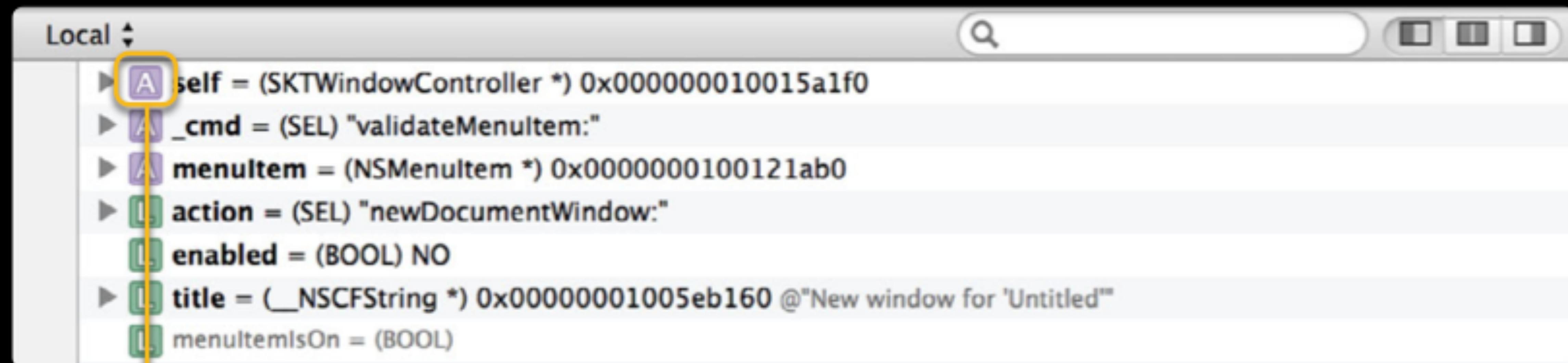
- “Step out” of a jumped-to routine
 - Complete the current routine
 - Step to the next routine or back to the calling routine
 - Debugger will stop at the statement after the function call.

VARIABLES VIEW



- Viewing Mode
 - Auto - variables around the line of code you're paused at
 - Local - all variables in local scope
 - All - all variables including globals and registers

VARIABLES VIEW



Variable Kind

L Local Variable

A Argument

S Static Variable

V Global Variable

R Register

V Instance Variable

E Expression

VARIABLES VIEW

The screenshot shows the Xcode Variables View window. At the top, there is a toolbar with various icons. Below the toolbar, a dropdown menu is set to "Auto". The main area displays four variables:

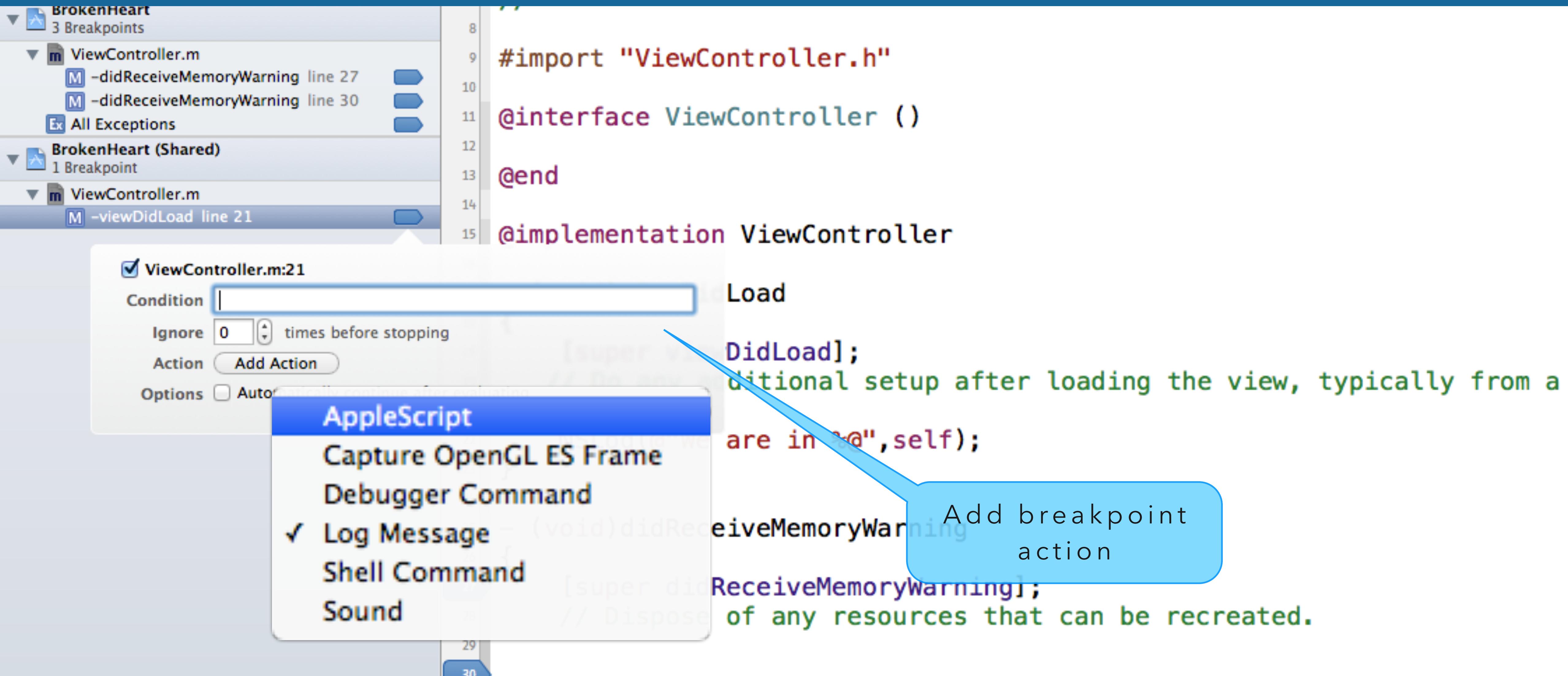
- A variable named `self` of type `ViewController *`, with a value of `0x071c6900`. This row has a yellow callout pointing to the "VARIABLE NAME" label.
- A variable named `i` of type `(int)`, with a value of `0`.
- A variable named `_debugFlag` of type `NSString *`, with a value of `@"DebugFlagOn"`.
- A variable named `_reds` of type `NSMutableArray *`, with a value of `@"0 objects"`.

Below the list of variables, there are four yellow callouts with labels:

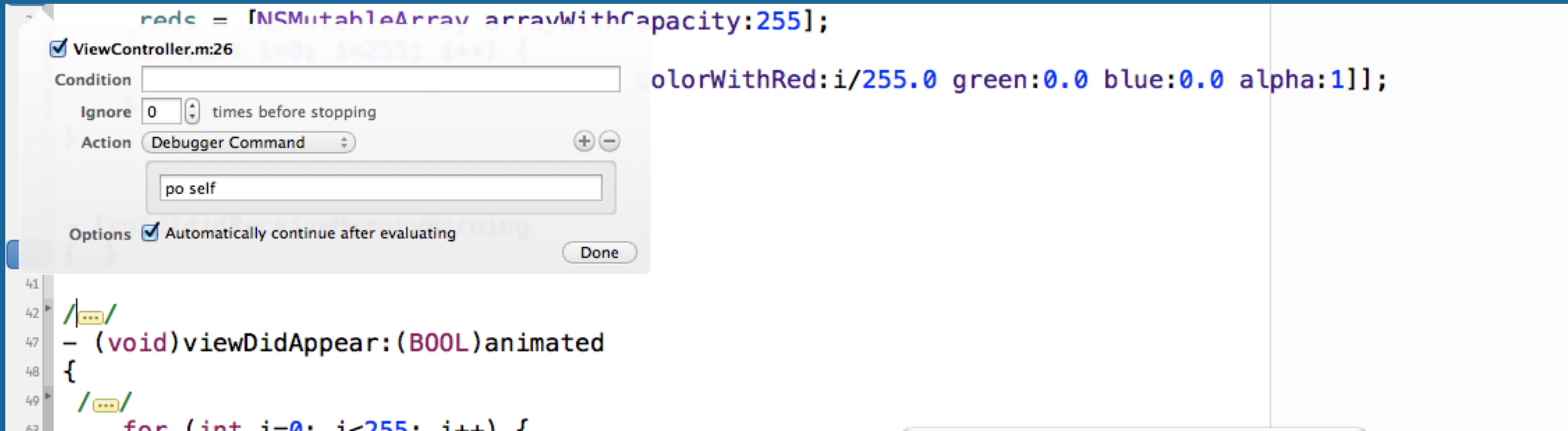
- "VARIABLE NAME" points to the name column of the first variable row.
- "VARIABLE VALUE" points to the value column of the first variable row.
- "TYPE" points to the type column of the first variable row.
- "VARIABLE SUMMARY" points to the summary column of the first variable row.

BREAKPOINTS ACTIONS

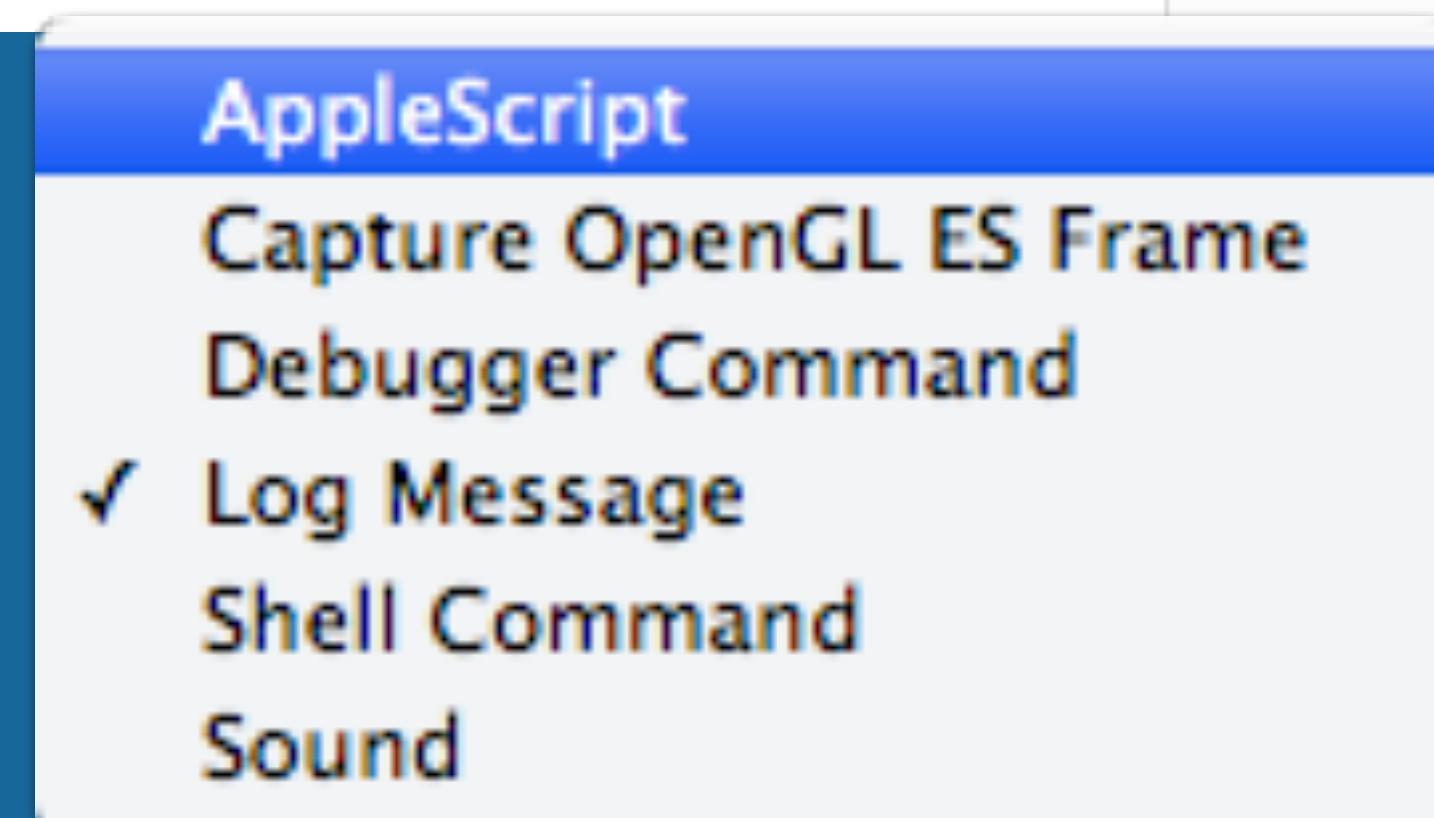
BREAKPOINT ACTIONS



BREAKPOINT ACTIONS



- Add multiple actions per breakpoint
- Print object description
 - `po self`



BREAKPOINT ACTIONS DEBUGGER COMMANDS

The screenshot shows two code snippets in Xcode with their respective breakpoint action configurations.

Top Snippet (ViewController.m:26):

```
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
```

Breakpoint Action (ViewController.m:26):

- Condition:
- Ignore: 0 times before stopping
- Action: Debugger Command
- Options: Automatically continue after evaluating

Bottom Snippet (ViewController.m:42):

```
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
```

Breakpoint Action (ViewController.m:42):

- Condition:
- Ignore: 0 times before stopping
- Action: Debugger Command
- Options: Automatically continue after evaluating

A blue callout bubble points from the "po self" button in the first breakpoint's action configuration to the "po self" command in the second snippet's code, illustrating its use.

BREAKPOINT ACTIONS

- Trying to avoid a littering of log statements

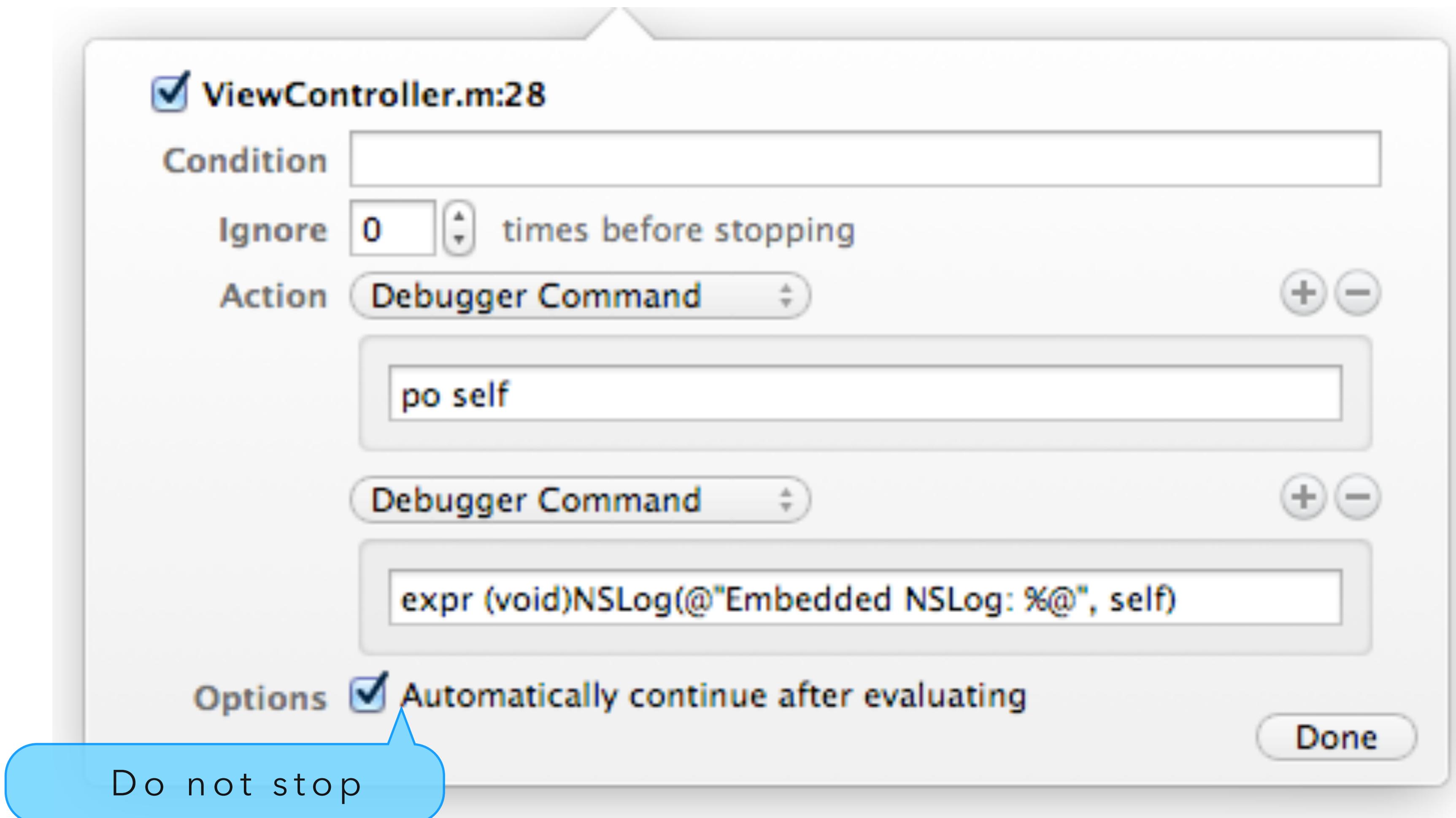
```
print("We are in \$(self)")
```



- Use breakpoints in Xcode to create embedded NSLogs

```
expr (void)NSLog(@"%@", @"Embedded NSLog: %@", self)
```

BREAKPOINT ACTIONS DEBUGGER COMMANDS



BREAKPOINT ACTIONS

The screenshot shows the Xcode interface with a code editor and a breakpoint editor overlay. The code editor displays the following code:

```
22 - (void)viewDidLoad
23 {
24     [super viewDidLoad];
25
26     reds = [NSMutableArray arrayWithCapacity:255];
27
28     for (int i=0; i<255; i++) {
29         reds[i] = [[UIColor alloc] initWithRed:i/255.0 green:0.0 blue:0.0 alpha:1];
30     }
31 }
```

The breakpoint at line 26 is selected, indicated by a blue arrow on the left. The breakpoint editor shows the following configuration:

- Condition:** Empty text field.
- Ignore:** 0 times before stopping.
- Action:** Debugger Command (selected).
 - po self** (Action 1)
 - Debugger Command (disabled)** (Action 2)
 - expr (void) NSLog(@"%@", self)** (Action 3)
- Options:** Automatically continue after evaluating.

Two blue callout bubbles point to the second and third debugger commands in the action list, both labeled "Embedded NSLog".

BREAKPOINT ACTIONS

ViewController.m:36

Condition `(BOOL)[self.debugFlag isEqualToString:@"DebugFlagOn"]`

Ignore `0` times before stopping

Action `Log Message` + -

`%B : i=@i@ (hit %H times)`

Log message to console `@exp@ = expression`
 Speak message `%B = breakpoint name`
`%H = breakpoint hit count`

Options Automatically continue after evaluating Done

BREAKPOINT ACTIONS

NSString flag

ViewController.m:36

Condition `(BOOL)[self.debugFlag isEqualToString:@"DebugFlagOn"]`

Ignore times before stopping

Action Log Message



`%B : i=@i@ (hit %H times)`

Log message to console

`@exp@ = expression`

Speak message

`%B = breakpoint name`

`%H = breakpoint hit count`

Options Automatically continue after evaluating

Done

BREAKPOINT ACTIONS CONDITION

The screenshot shows a Xcode interface with a code editor at the top and a breakpoint configuration dialog below it.

Code Editor:

```
18 //  
23 - (void)viewDidLoad  
{  
25     [super viewDidLoad];  
26     _debugFlag = @"DebugFlagOff";  
27  
28     _reds = [NSMutableArray arrayWithCapacity:255];  
29     for (int i=0; i<255; i++) {  
30         [self.reds addObject:[UIColor colorWithRed:i/255.0 green:0.0 blue:0.0 alpha:1]];  
31     }  
32 }
```

Breakpoint Configuration Dialog:

- Condition:** (BOOL)[self.debugFlag isEqualToString:@"DebugFlagOn"]
- Ignore:** 0 times before stopping
- Action:** Log Message
 - %B : i=@i@ (hit %H times)
 - Log message to console
 - Speak message
 - @exp@ = expression
 - %B = breakpoint name
 - %H = breakpoint hit count
- Options:** Automatically continue after evaluating

Annotations:

- A callout bubble points to the `_debugFlag` variable in the code with the text "NSString flag".
- A callout bubble points to the `Condition` field with the text "Test value".
- A callout bubble points to the "Not run" option in the Action dropdown with the text "Not run".

BREAKPOINT ACTIONS CONDITION

The screenshot shows the Xcode debugger interface with a breakpoint condition dialog open. The code in the background is:

```
18 //  
23 - (void)viewDidLoad  
{  
25     [super viewDidLoad];  
26     _debugFlag = @"DebugFlagOn";  
27  
28     _reds = [NSMutableArray arrayWithCapacity:255];  
29     for (int i=0; i<255; i++) {  
30         [self.reds addObject:[UIColor colorWithRed:i/255.0 green:0.0 blue:0.0 alpha:1]];  
31     }  
32 }
```

The breakpoint condition dialog has the following settings:

- Condition:** (BOOL)[self.debugFlag isEqualToString:@"DebugFlagOn"]
- Ignore:** 0 times before stopping
- Action:** Log Message

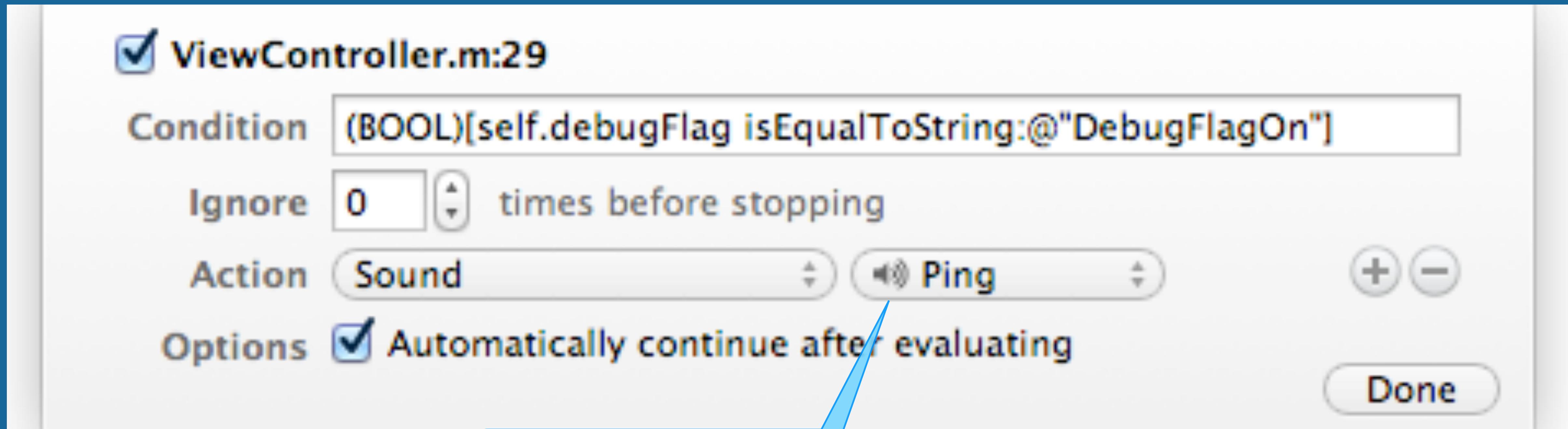
A callout bubble points to the condition field with the text "NSString flag". Another callout bubble points to the ignore field with the text "Test value". A third callout bubble points to the action field with the text "Run".

At the bottom of the dialog, there is a note about the format string: "%B : i=@i@ (hit %H times)" with options: "Log message to console" (selected) and "Speak message". It also defines abbreviations: "@exp@" = expression, "%B" = breakpoint name, and "%H" = breakpoint hit count.

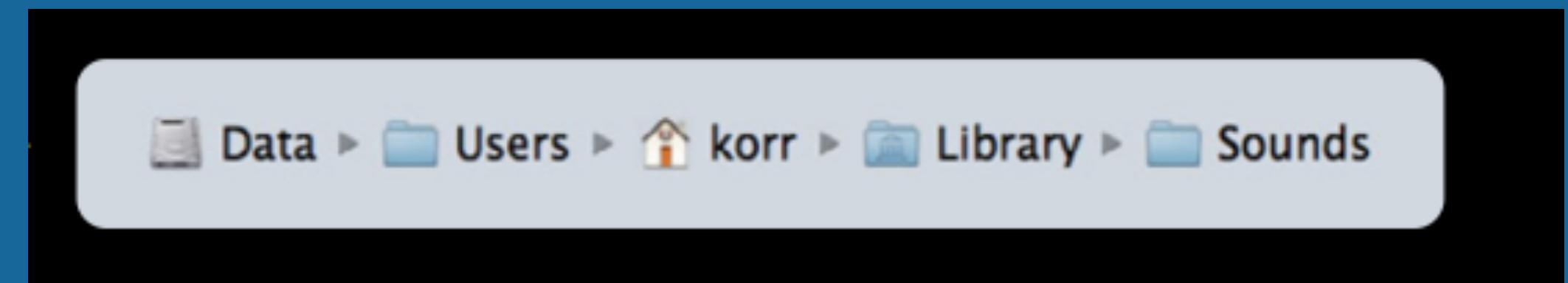
Options at the bottom include "Automatically continue after evaluating" (checked) and a "Done" button.

The bottom of the screen shows the Xcode toolbar and the "BrokenHeart" project name.

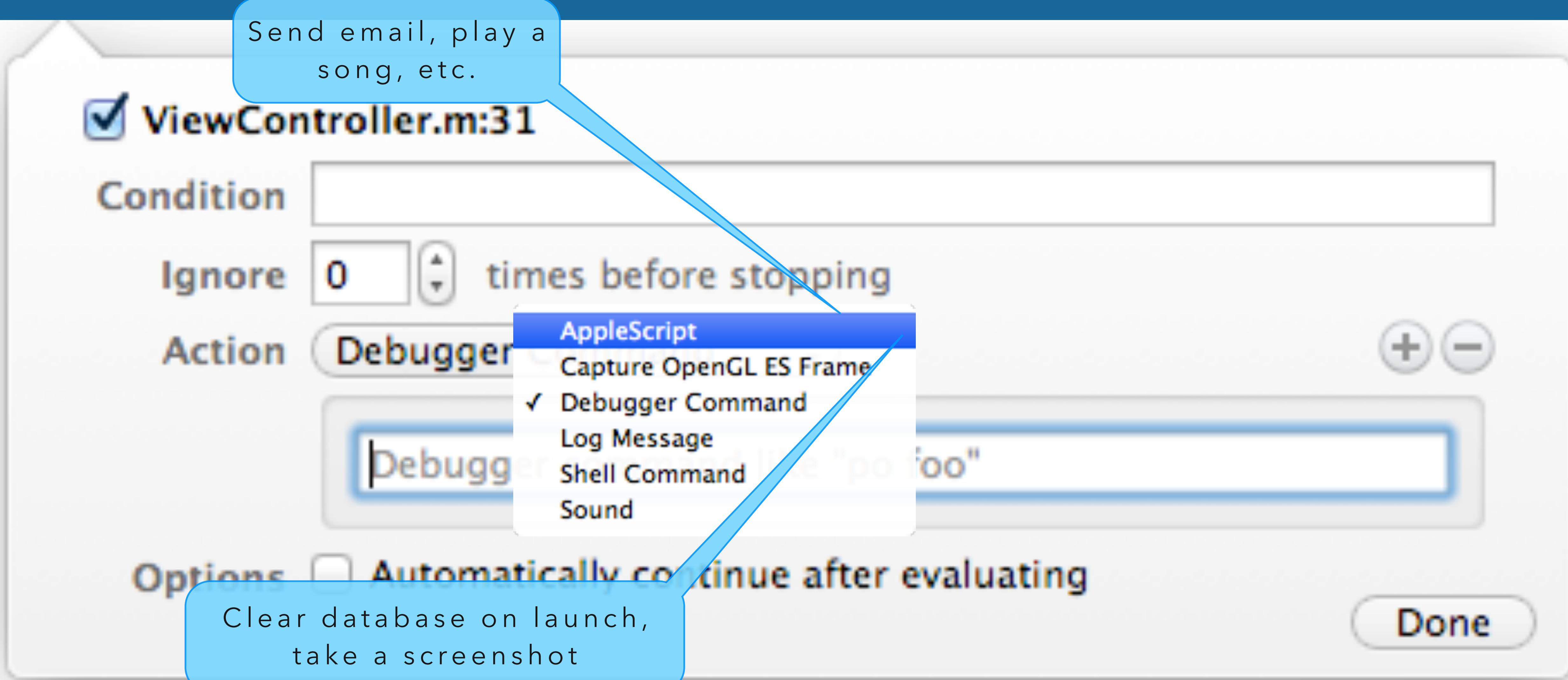
BREAKPOINT ACTIONS SOUNDS



- Sound library



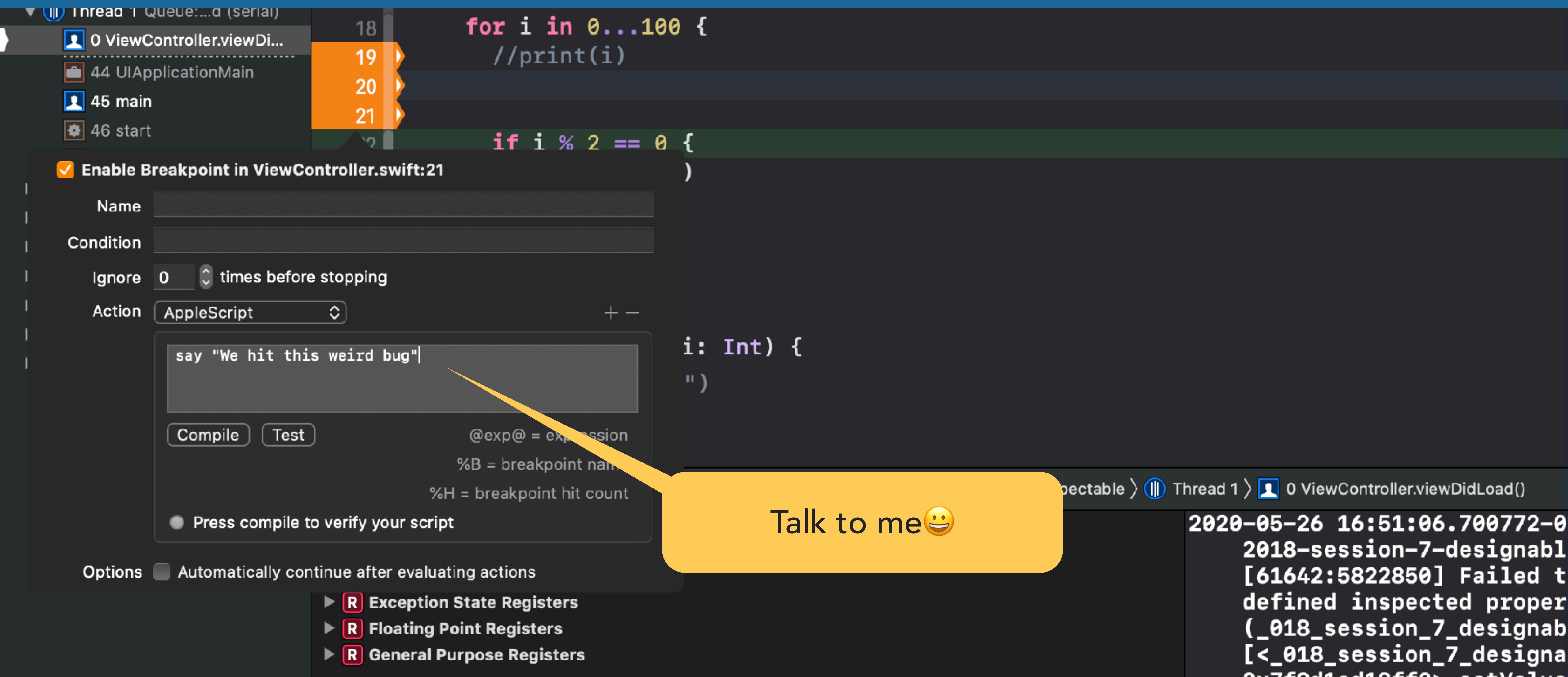
BREAKPOINT ACTIONS APPLE SCRIPT



BREAKPOINT ACTIONS APPLE SCRIPT

```
tell application "Mail"
    set myMessage to make new outgoing message with properties
        {visible:false, subject:"Test Failed", content:"The test failed"}
    tell myMessage
        make new to recipient at end of to recipients with properties
            {name:"Ken Orr", address:"orr@apple.com"}
        myMessage send
    end tell
end tell
```

BREAKPOINT ACTIONS



LOGGING WITH THE DEBUGGER

LOGGING

The screenshot shows the Xcode interface with the following details:

- Project Structure:** Shows a folder named "DataSourceDelegates" containing several Swift files like TableDataSourceDelegate.swift, SimpleTableDataSourceDelegate.swift, etc.
- MasterViewController.swift:** The current file being edited. It contains code related to split view controllers and table data sources.
- Breakpoint Editor:** A floating window showing a breakpoint for line 45. The condition is empty, and the action is set to "Log Message".
 - Action Details:** "in View Did Load" is selected. The "Log message to console" option is checked, while "Speak message" is unselected.
 - Log Message Options:** Includes @exp@ = expression, %B = breakpoint name, and %H = breakpoint hit count.
- Callout Bubble:** An orange callout bubble points from the "Log message" text in the breakpoint editor towards the word "Log message" in the list of code completion suggestions on the right.
- Suggestions List:** A sidebar on the right lists various Swift language features with their descriptions:
 - MasterViewController.swift**: Full Path /Users/tbinkowski/Google Drive/g-Teaching/Splitter/
 - Swift Subclass**: Define a Swift subclass.
 - Switch Statement**: Execute different sections of code when an expression has one of several const...
 - Test Method**: Add a test case method to a test class.
 - Union Declaration**: Declare a new union type, where all fields overlap at the same memory location.
 - While Statement**: Execute code while a condition is true.
 - Objective-C -initWithFrame:** Method - Initialize a custom view subclass.
 - Appledoc @name Declaration**
 - 2-Table Auto Resizing**
 - My Function Declaration**

LOGGING

The screenshot shows the Xcode interface with the following details:

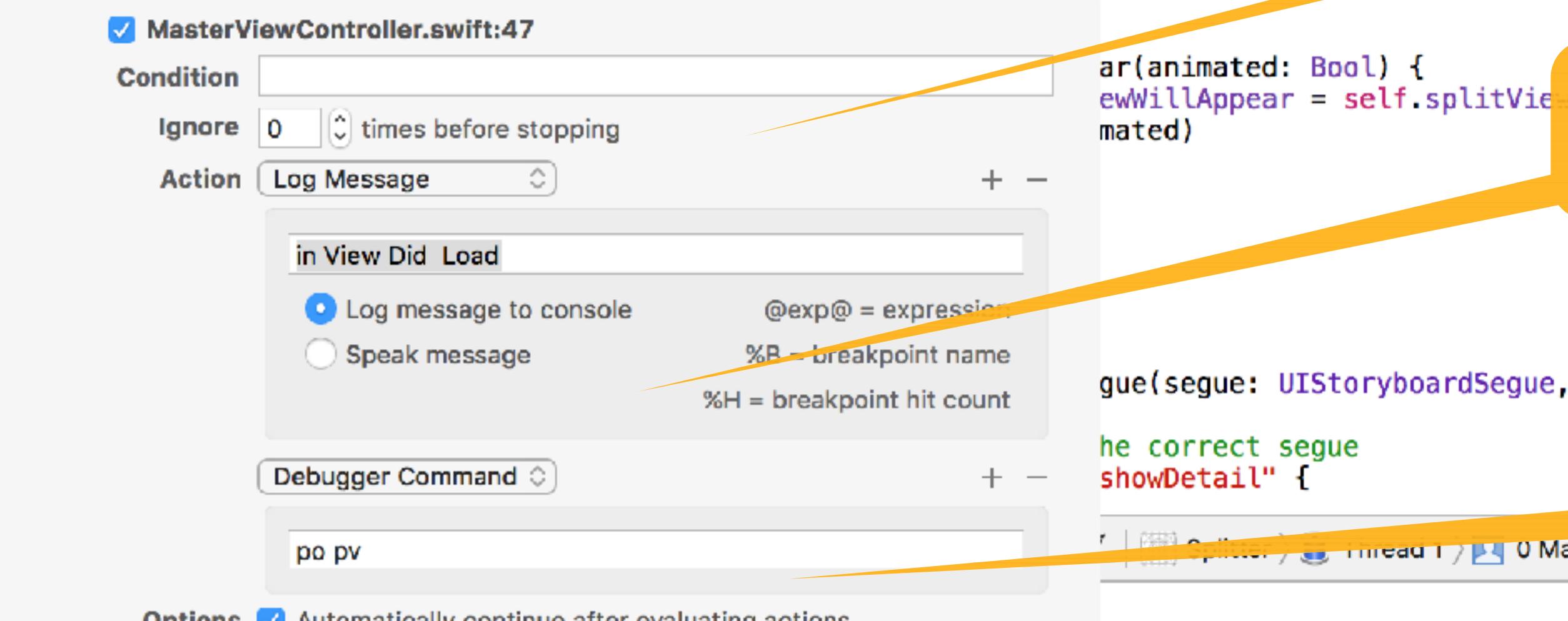
- Left Sidebar:** Shows system status (DISK, Network) and a list of threads:
 - Thread 1 Queue: co...read (serial):
 - 0 MasterViewController.viewDidLoad
 - 34 UIApplicationMain
 - 35 main
 - 36 start
 - Thread 2 Queue: co...ager (serial)
 - Thread 3
 - Thread 4
 - Thread 5
- Right Side:** A code editor window displaying Swift code. The code handles view loading and delegates for a split view controller. A yellow callout bubble points from the bottom right towards the code, containing the text "Log message".
- Bottom Status Bar:** Shows "Thread 1: breakpoint 1.1".

```
28
29 override func viewDidLoad() {
30     super.viewDidLoad()
31
32     // Split view controller stuff
33     if let split = self.splitViewController {
34         let controllers = split.viewControllers
35         self.detailViewController = (controllers[controllers.count-1] as! UINavigationController).topViewController as?
36 DetailViewController
37     }
38
39     // Set the table source and delegate object
40     self.tableView.delegate = tableDataSourceDelegate
41     self.tableView.dataSource = tableDataSourceDelegate
42
43     if let split = self.splitViewController {
44         let controllers = split.viewControllers
45         self.detailViewController = (controllers[controllers.count-1] as! UINavigationController).topViewController as?
46 DetailViewController
47
48     let pv = ProgressView()
49     split.view.addSubview(pv)
50 }
```

Log message

po - "print object"

Don't break



LOGGING

The screenshot shows a Xcode interface with a code editor on the left and a sidebar of code snippets on the right.

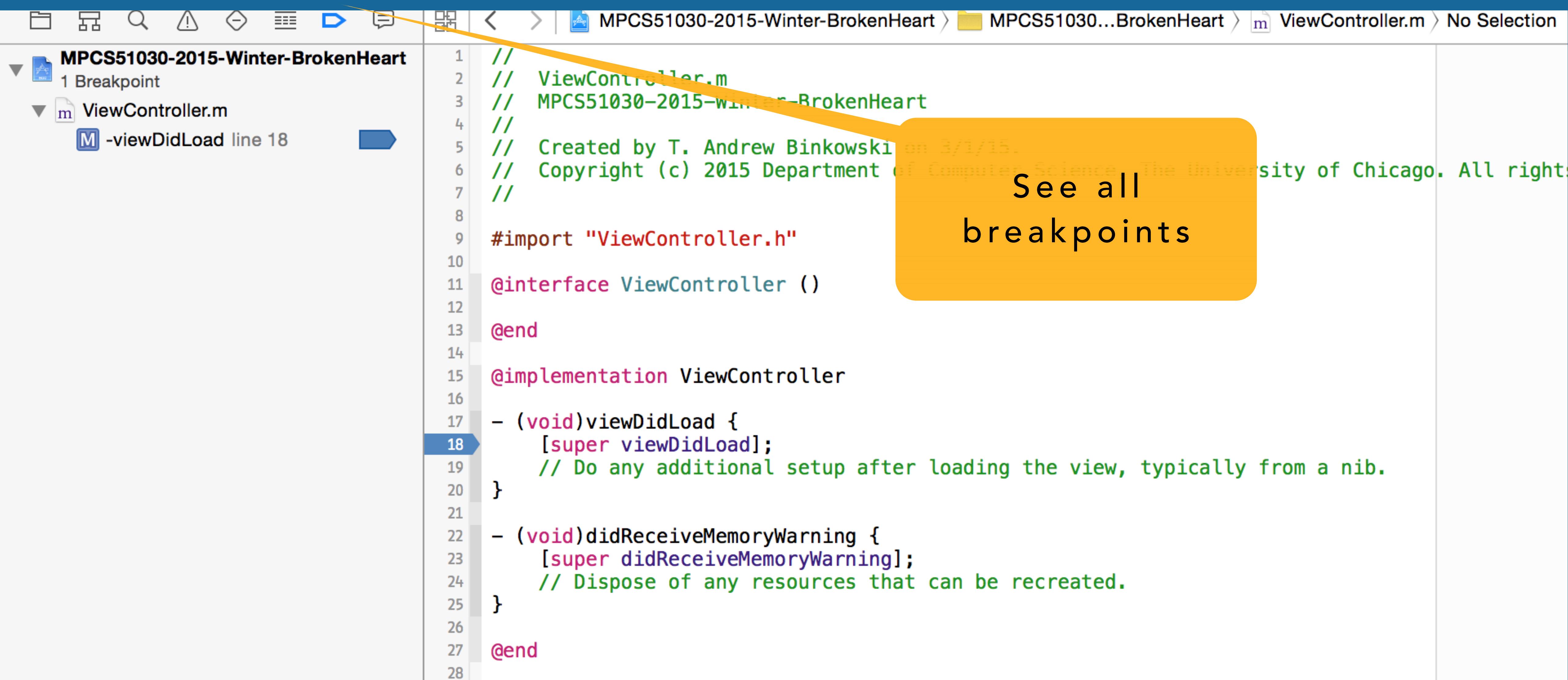
Code Editor:

```
28 override func viewDidLoad() {
29     super.viewDidLoad()
30
31     // Split view controller stuff
32     if let split = self.splitViewController {
33         let controllers = split.viewControllers
34         self.detailViewController = (controllers[controllers.count-1] as! UINavigationController).topViewController as?
35 DetailViewController
36     }
37
38     // Set the table source and delegate object
39     self.tableView.delegate = tableViewDataSourceDelegate
40     self.tableView.dataSource = tableViewDataSourceDelegate
41
42     if let split = self.splitViewController {
43         let controllers = split.viewControllers
44         self.detailViewController = (controllers[controllers.count-1] as! UINavigationController).topViewController as?
45 DetailViewController
46
47     let pv = ProgressView()
48     split.view.addSubview(pv)
49 }
50
51 override func viewDidAppear(animated: Bool) {
52     self.clearsSelectionOnViewWillAppear = self.splitViewController!.collapsed
53     super.viewDidAppear(animated)
54 }
55
56 //
57 // MARK: - Segues
58 //
59
60 override func prepareForSegue(segue: UIStoryboardSegue, sender: AnyObject?) {
61
62     // Verify we are using the correct segue
63     if segue.identifier == "showDetail" {
64 }
```

Code Snippets Sidebar:

- Swift Subclass** - Define a Swift subclass.
- Switch Statement** - Execute different sections of code when an expression has one of several constant values.
- Test Method** - Add a test case method to a test class.
- Union Declaration** - Declare a new union type, where all fields overlap at the same memory location.
- While Statement** - Execute code while a condition is true.
- Objective-C -initWithFrame:** Method - Initialize a custom view subclass.
- Appledoc @name Declaration**
- 2-Table Auto Resizing**
- My Function Declaration**

LOGGING



A screenshot of the Xcode IDE interface. The title bar shows the project name "MPCS51030-2015-Winter-BrokenHeart" and the file "ViewController.m". The left sidebar shows the project structure with "ViewController.m" selected, and a blue arrow points to a breakpoint at line 18. The main editor area displays the code for ViewController.m, including a copyright notice and several methods. A yellow callout bubble with the text "See all breakpoints" is positioned over the breakpoint icon in the sidebar.

```
// ViewController.m
// MPCS51030-2015-Winter-BrokenHeart
//
// Created by T. Andrew Binkowski on 3/1/15.
// Copyright (c) 2015 Department of Computer Science, The University of Chicago. All rights reserved.

#import "ViewController.h"

@interface ViewController : UIViewController

- (void)viewDidLoad {
    [super viewDidLoad];
    // Do any additional setup after loading the view, typically from a nib.
}

- (void)didReceiveMemoryWarning {
    [super didReceiveMemoryWarning];
    // Dispose of any resources that can be recreated.
}

@end
```

LOGGING

The screenshot shows the Xcode interface with the following details:

- Memory:** 16.6 MB
- Disk:** Zero KB/s
- Network:** Zero KB/s

The code editor displays the following Objective-C code:

```
36 // !!!: pay attention
37 // TODO: Do this message
38
39 - (void)viewDidLoad {
40     [super viewDidLoad];
41
42     self.hearts = [NSMutableArray arrayWithCapacity:255];
43     for (int i=0; i<255; i++) {
44         CGRect frame = CGRectMake(i+5, i+5, 20, 20);
45
46         // Create a heart object and store the frame and color
47         Heart *heart = [[Heart alloc] initWithFrame:frame];
48         heart.backgroundColor = RGB(i, 0, 0);
49         heart.tag = i;
50         [self.hearts addObject:heart];
51     }
52 }
53
54 - (void)viewDidAppear:(BOOL)animated
55 {
56     // Add the hearts
57     for (Heart *heart in self.hearts) {
58
59         [self.view addSubview:heart];
60     }
61 }
```

A yellow callout bubble with the word "Skip" is pointing to the code between lines 50 and 53.

The bottom left shows the file is ViewController.m at line 63, with a condition set to ignore 10 times before stopping. The bottom right shows the project name is MPCS51030-2015-Winter-BrokenHeart.

LOGGING

The screenshot shows the Xcode interface with the following details:

- Top Bar:** Shows the project name "MPCS51030-2015-Winter-BrokenHeart", file path "ViewController.m", and method "-viewDidAppear:".
- Left Sidebar:** Displays system monitoring for the application "MPCS51030-2015-Winter-BrokenHeart" (PID 46799, Running). It includes sections for CPU (0%), Memory (16.6 MB), Disk (Zero KB/s), and Network (Zero KB/s).
- Code Editor:** Shows the source code for `ViewController.m`. Lines 31 through 44 are visible, containing various TODO and FIXME comments:

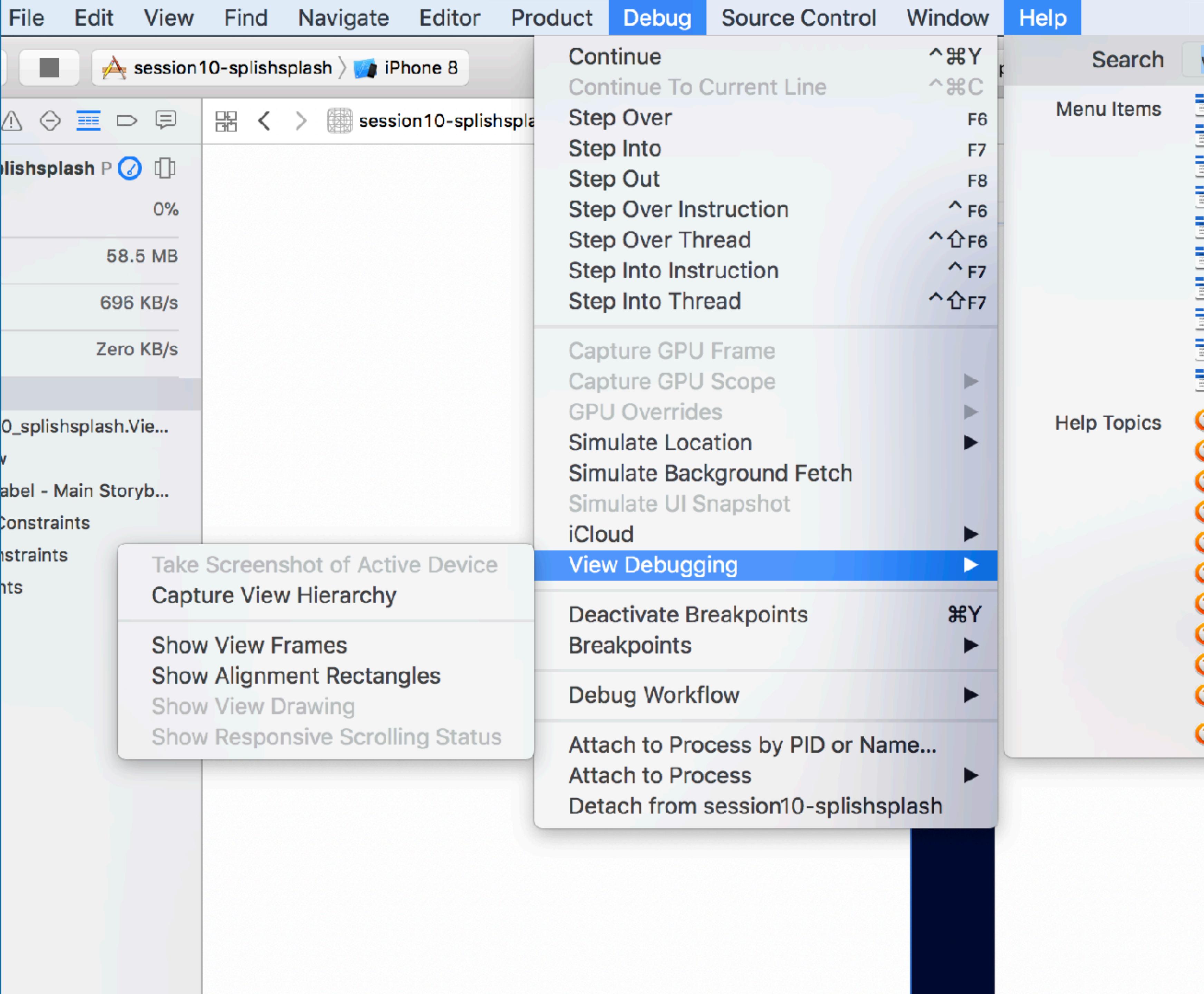
```
31
32
33 // FIXME: fix me
34
35 // ????: what is this?
36
37 // !!!: pay attention
38
39 // TODO: Do this message
40
41
42 - (void)viewDidLoad {
43     [super viewDidLoad];
44 }
```
- Log Output:** Below the code editor, the log output window displays the following entries:

```
@viewDidLoad
11 Count
12 Count
13 Count
14 Count
15 Count
16 Count
17 Count
18 Count
19 Count
20 Count
21 Count
```

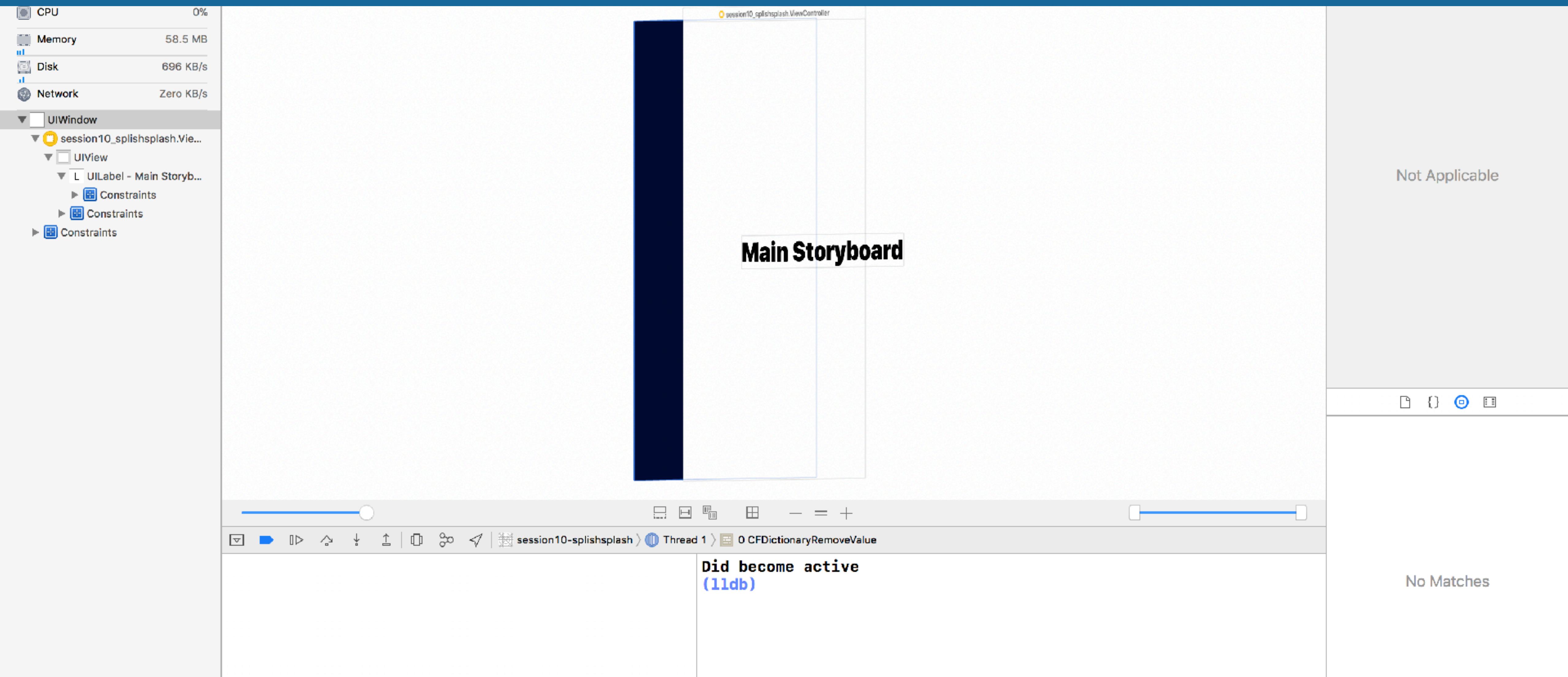
RUNTIME DEBUGGING

RUNTIME DEBUGGING

- Add a breakpoint
- When stopped, capture the view hierarchy



RUNTIME DEBUGGING



RUNTIME DEBUGGING

session10-splishsplash iPhone 8

Running session10-splishsplash on iPhone 8

session10-splishsplash P 0%
CPU 0%
Memory 58.5 MB
Disk 696 KB/s
Network Zero KB/s

session10-splishsplash (3)
 AppDelegate (1)
 0x60000002fc00
 SplashViewController (1)
 0x7f8cee407be0
 ViewController (1)
 0x7f8cf0801ce0
AccessibilitySettingsLoader (3)
 AssistiveTouchHelper (1)
 0x6000000041c0
 SpeakTypingManager (1)
 0x608000004790
 ZoomServicesUI (1)
 0x608000004100
BaseBoard (1)
 AXBaseBoardGlue (1)
 0x608000004d60
DebugHierarchyFoundation (1)
 DBGTargetHub (1)
dyld_sim (107)
iTunesStoreFramework (1)
libdispatch.dylib (113)
libcucore.A.dylib (89)
libobjc.A.dylib (3)
libprotobuf.dylib (22)

session10-splishsplash > AppDelegate > 0x60000002fc00 > No Selection

Not Applicable

malloc<64> +48 bytes AppDelegate

MEMORY GRAPH

session10-splishsplash > Thread 1 > 0 CFDictionaryRemoveValue
Did become active (lldb)

No Matches

The screenshot illustrates the Xcode Memory Graph tool, which visualizes the state of memory in a running application. The left sidebar provides system-level metrics: CPU at 0%, Memory at 58.5 MB, Disk at 696 KB/s, and Network at Zero KB/s. The main pane shows a hierarchical tree of memory allocations for the 'session10-splishsplash' process. A specific allocation for 'AppDelegate' at address 0x60000002fc00 is selected. A tooltip shows that 48 bytes were allocated from a 'malloc<64>' block. A yellow callout box highlights the 'MEMORY GRAPH' feature. The bottom status bar indicates the current thread is Thread 1 and the last event was '0 CFDictionaryRemoveValue'.

RUNTIME DEBUGGING

The screenshot shows the Xcode Memory Profiler interface. On the left, a sidebar lists various system components and processes. In the center, three main charts provide usage information.

- Percentage Used:** A gauge chart showing 13% usage. The scale ranges from 0 to 800.
- Usage Comparison:** A donut chart showing the distribution of memory usage. The segments are: session10-splishsplash (13%), Other Processes (356%), and Free (431%).
- Usage over Time:** A histogram showing memory usage over a duration of 5 min 32 sec. The y-axis represents percentage (0%, 13%) and the x-axis represents time (0s, 130s). The peak usage is at 13% for the first few seconds.

Threads: A table listing threads and their types:

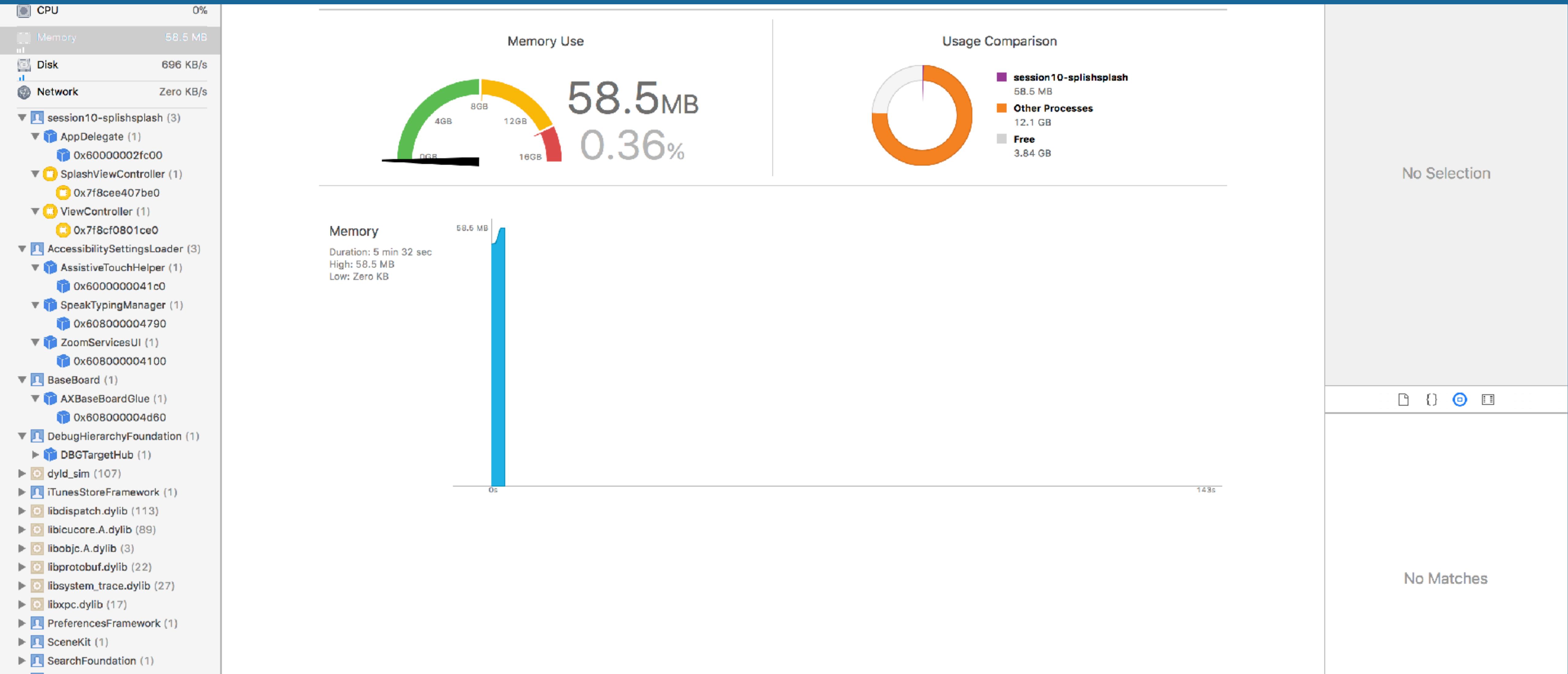
Thread 1	User Interactive
Thread 2	User Interactive
Thread 3	QoS Unavailable
Thread 4	QoS Unavailable
Thread 5	QoS Unavailable
Thread 6	User Initiated
com.apple.ui...h-thread (7)	User Interactive

Navigation: On the right side, there are navigation icons for back, forward, search, and other file operations. Below the search bar, it says "No Selection". At the bottom right, it says "No Matches".

Left Sidebar (List of Components/Processes):

- CPU: 0%
- Memory: 58.5 MB
- Disk: 696 KB/s
- Network: Zero KB/s
- session10-splishsplash (3)
 - AppDelegate (1)
 - 0x60000002fc00
 - SplashViewController (1)
 - 0x7f8cee407be0
 - ViewController (1)
 - 0x7f8cf0801ce0
- AccessibilitySettingsLoader (3)
 - AssistiveTouchHelper (1)
 - 0x6000000041c0
 - SpeakTypingManager (1)
 - 0x608000004790
 - ZoomServicesUI (1)
 - 0x608000004100
- BaseBoard (1)
 - AXBaseBoardGlue (1)
 - 0x608000004d60
- DebugHierarchyFoundation (1)
 - DBGTargetHub (1)
- dyld_sim (107)
- iTunesStoreFramework (1)
- libdispatch.dylib (113)
- libcucore.A.dylib (89)
- libobjc.A.dylib (3)
- libprotobuf.dylib (22)
- libsystem_trace.dylib (27)
- libxpc.dylib (17)
- PreferencesFramework (1)
- SceneKit (1)
- SearchFoundation (1)

RUNTIME DEBUGGING



RUNTIME DEBUGGING

CPU 0%

Memory 58.5 MB

Disk 696 KB/s

Network Zero KB/s

session10-splishsplash (3)

AccessibilitySettingsLoader (3)

BaseBoard (1)

AXBaseBoardGlue (1)
0x608000004d60

DebugHierarchyFoundation (1)

DBGTargetHub (1)

dyld_sim (107)

iTunesStoreFramework (1)

libdispatch.dylib (113)

libcucore.A.dylib (89)

libobjc.A.dylib (3)

libprotobuf.dylib (22)

libsystem_trace.dylib (27)

libxpc.dylib (17)

PreferencesFramework (1)

SceneKit (1)

SearchFoundation (1)

UIKit (1)

UIKit (162)

Foundation (1387)

CoreFoundation (16839) !

CoreFoundation (5)

CoreData (5)

ColorSync (10)

CoreGraphics (124)

CoreText (17)

ImageIO (19)

QuartzCore (57)

Accelerate / vImage / libC... (4)

Receiving

0.0 KB/s Per Second

0.0 KB Total

Sending

0.0 KB/s Per Second

0.0 KB Total

No Selection

Receiving and Sending Rates

0.0 KB/s

0.0 KB/s

0

1

Received Sent

Active Connections

Protocol	Local Address	Local Port	Remote Address	Remote Port	Interface	State	Bytes In	Δ Bytes In	Bytes Out	Δ Bytes O
No Active Connections										

No Matches

STATIC TABLES & FORMS

FORM

- Static Table Demo

The screenshot shows the Xcode interface with the following components:

- Main.storyboard:** Displays a static table view controller titled "USER". It contains three sections: "Name" (with a text field placeholder "First, Last Name"), "Email" (with a text field placeholder "test@test.com"), and "Find out about new products?" (with a UISwitch that is turned on).
- FormTableViewController.swift:** The Swift code defines a `FormTableViewController` class that inherits from `UITableViewController`. It includes outlets for `firstNameField`, `emailField`, `dataTextView`, and `contactUser`. The `viewDidLoad` method sets delegates for the text fields and configures the keyboard dismiss mode. An `IBAction` for the switch handles the "newProductSwitch" event.
- Output Window:** Shows log messages related to keyboard dismissal and background tasks.

```
// FormTableViewController.swift
// Form-ula
//
// Created by T. Andrew Binkowski on 3/5/20.
// Copyright © 2020 T. Andrew Binkowski. All rights reserved.

import UIKit

class FormTableViewController: UITableViewController {

    @IBOutlet weak var firstNameField: UITextField!
    @IBOutlet weak var emailField: UITextField!
    @IBOutlet weak var dataTextView: UITextView!
    @IBOutlet weak var contactUser: UISwitch!

    override func viewDidLoad() {
        super.viewDidLoad()
        self.firstNameField.delegate = self
        self.emailField.delegate = self
        self.tableView.keyboardDismissMode = .onDrag
        // Automatically show/hide keyboard
        //firstNameField.becomeFirstResponder()
    }

    @IBAction func newProductSwitch(_ sender: UISwitch) {
        print("switch: \(sender.isOn) #2")
    }
}

// Dismiss Keyboard #2
```

```
Did End Editing: test
First:
2020-03-06 12:31:50.888405-0600 Form-ula[3292:11691264]
[Snapshotting] Snapshotting a view (0x7f889e0bba00,
UIKeyboardImpl) that is not in a visible window requires
afterScreenUpdates:YES.
2020-03-06 12:31:52.379056-0600 Form-ula[3292:11691264] Can't end
BackgroundTask: no background task exists with identifier 1
(0x1), or it may have already been ended. Break in
UIApplicationEndBackgroundTaskError() to debug.
```

A FORM

FORM

- Form
 - UITableView Controller
 - Static table
 - UIControls

The screenshot shows the Xcode interface with the following components:

- Top Bar:** Shows "iPhone 11", "Running Form-ula on iPhone 11", and a warning icon.
- Project Navigator:** Shows the project structure with "Main.storyboard" selected.
- Document Outline:** Shows the hierarchy of the storyboard, including "Form Table View Contr...", "User", "Data", and various UI elements like "Name", "Email", and a "Table View".
- Preview View:** Displays the iPhone 11 simulator screen with a form titled "USER". It contains fields for "Name" (placeholder "First, Last Name") and "Email" (placeholder "test@test.com"). Below these is a switch labeled "Find out about new products?" which is turned on. A large text area labeled "DATA" contains placeholder text: "Lorem ipsum dolor sit er elit lamet, consectetur cillum adipisicing pecu, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum. Nam liber te conscient to factor tum poen legum odioque civiuda."
- Code Editor:** Shows the "FormTableViewController.swift" file with the following code:

```
// FormTableViewController.swift
// Form-ula
//
// Created by T. Andrew Binkowski on 3/5/20.
// Copyright © 2020 T. Andrew Binkowski. All rights reserved.

import UIKit

class FormTableViewController: UITableViewController {

    @IBOutlet weak var firstNameField: UITextField!
    @IBOutlet weak var emailField: UITextField!
    @IBOutlet weak var dataTextView: UITextView!
    @IBOutlet weak var contactUser: UISwitch!

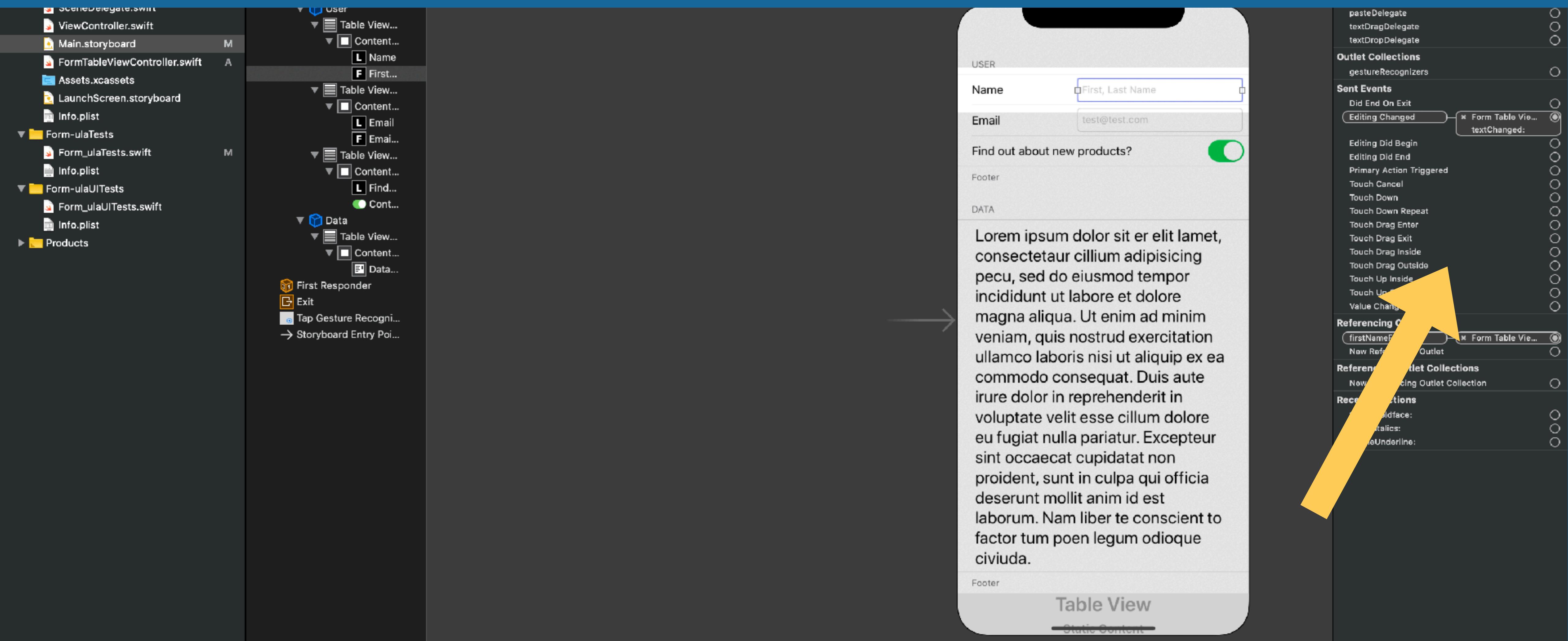
    override func viewDidLoad() {
        super.viewDidLoad()
        self.firstNameField.delegate = self
        self.emailField.delegate = self
        self.tableView.keyboardDismissMode = .onDrag
        // Automatically show/hide keyboard
        //firstNameField.becomeFirstResponder()
    }

    @IBAction func newProductSwitch(_ sender: UISwitch) {
        print("switch: \(sender.isOn) #2")
    }

    // Dismiss Keyboard #2
}
```
- Output Navigator:** Shows log entries:

```
Did End Editing: test
First:
2020-03-06 12:31:50.888405-0600 Form-ula[3292:11691264]
[Snapshotting] Snapshotting a view (0x7f889e0bba00,
UIKeyboardImpl) that is not in a visible window requires
afterScreenUpdates:YES.
2020-03-06 12:31:52.379056-0600 Form-ula[3292:11691264] Can't end
BackgroundTask: no background task exists with identifier 1
(0x1), or it may have already been ended. Break in
UIApplicationEndBackgroundTaskError() to debug.
```

FORM



FORM

SUBTITLE

- Keyboard can obscure the fields
 - Resize the view by listening for notifications

Find out about new products?

Footer

DATA

Lore ipsum dolor sit er elit lamet,
consectetaur cillum adipisicing
pecu, sed do eiusmod tempor
incididunt ut labore et dolore
magna aliqua. Ut enim ad minim
veniam, quis nostrud exercitation
ullamco laboris nisi ut aliquip ex ea

[UIKeyboardWillShowNotification](#)
[UIKeyboardDidShowNotification](#)
[UIKeyboardWillHideNotification](#)
[UIKeyboardDidHideNotification](#)

FORM

UIKeyboardWillShowNotification
UIKeyboardDidShowNotification
UIKeyboardWillHideNotification
UIKeyboardDidHideNotification

```
let center = NotificationCenter.default

center.addObserver(self,
    selector: selector(keyboardWillBeShown(note:)),
name: Notification.Name.UIKeyboardWillShow, object: nil)

center.addObserver(self,
    selector: selector(keyboardWillBeHidden(note:)),
name: Notification.Name.UIKeyboardWillHide, object: nil)
```

FORM

Change the view

```
func keyboardWillBeShown(note: Notification) {  
    let userInfo = note.userInfo  
    let keyboardFrame = userInfo?  
        [UIKeyboardFrameEndUserInfoKey] as! CGRect  
    let contentInset = UIEdgeInsetsMake(0.0, 0.0,  
        keyboardFrame.height, 0.0)  
    scrollView.contentInset = contentInset  
    scrollView.scrollIndicatorInsets = contentInset  
    scrollView.scrollRectToVisible(textField.frame, animated: true)  
}
```

FORM

SUBTITLE

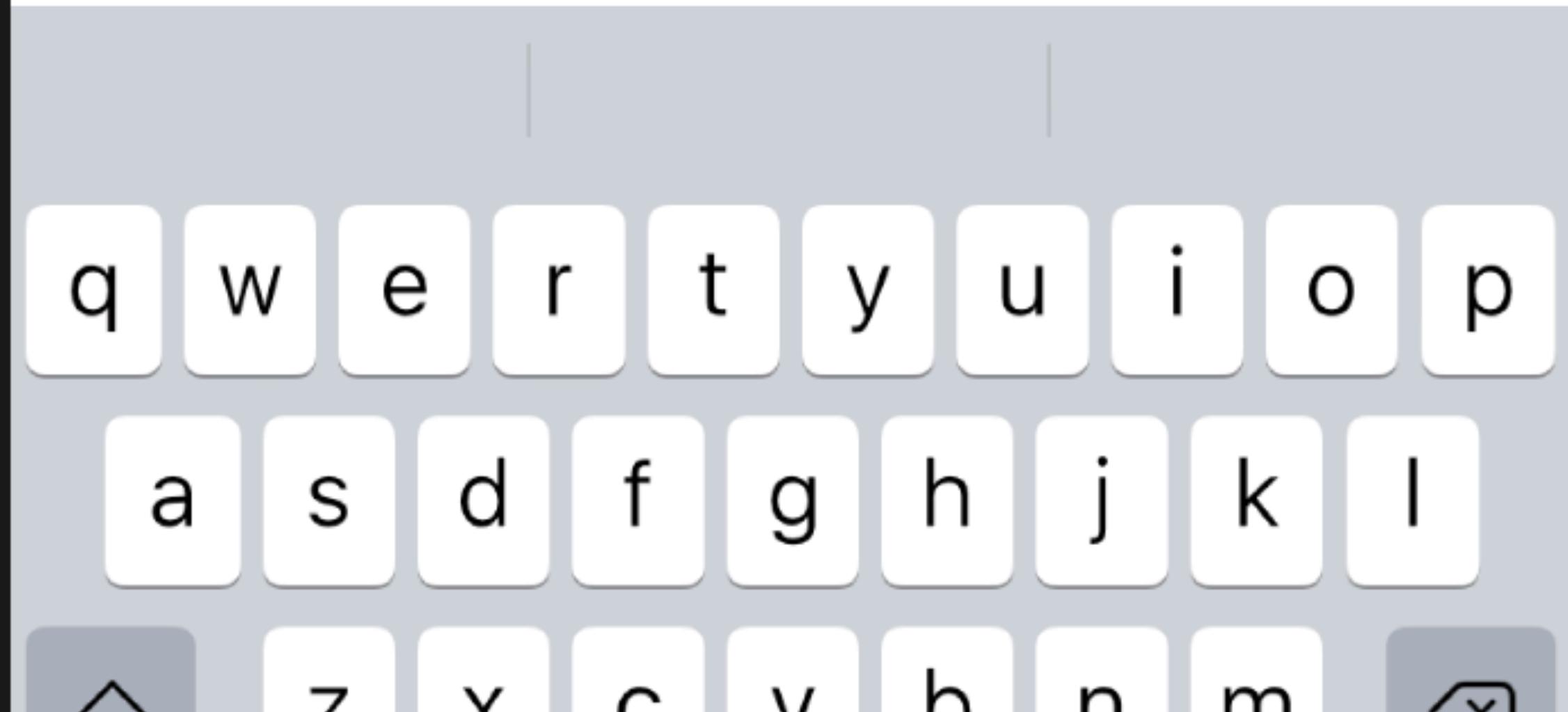
- Keyboard can obscure the fields
 - UITableViewController automatically adjusts

Find out about new products?

Footer

DATA

Lorem ipsum dolor sit er elit lamet,
consectetaur cillum adipisicing
pecu, sed do eiusmod tempor
incididunt ut labore et dolore
magna aliqua. Ut enim ad minim
veniam, quis nostrud exercitation
ullamco laboris nisi ut aliquip ex ea
commodo consequat. Duis aute





ADVANCED iOS APPLICATION DEVELOPMENT

MPCS 51032 • SPRING 2020 • SESSION 8