Swift Interoperability

Objective-C + Swift = □

Swift Interoperability

Swift is designed to work seamlessly with Objective-C.

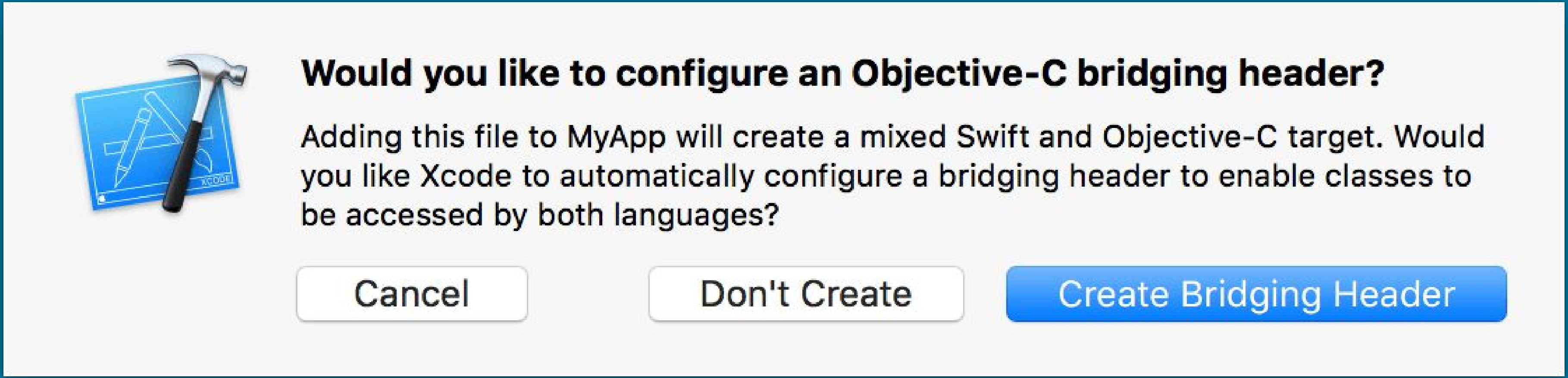
Interoperability is the ability to interface between Swift and Objective-C in either direction, letting you access and use pieces of code written in one language in a file of the other language.

Access Objective-C in Swift

If you want to access an Objective-C object in Swift, you must use a bridging header file.

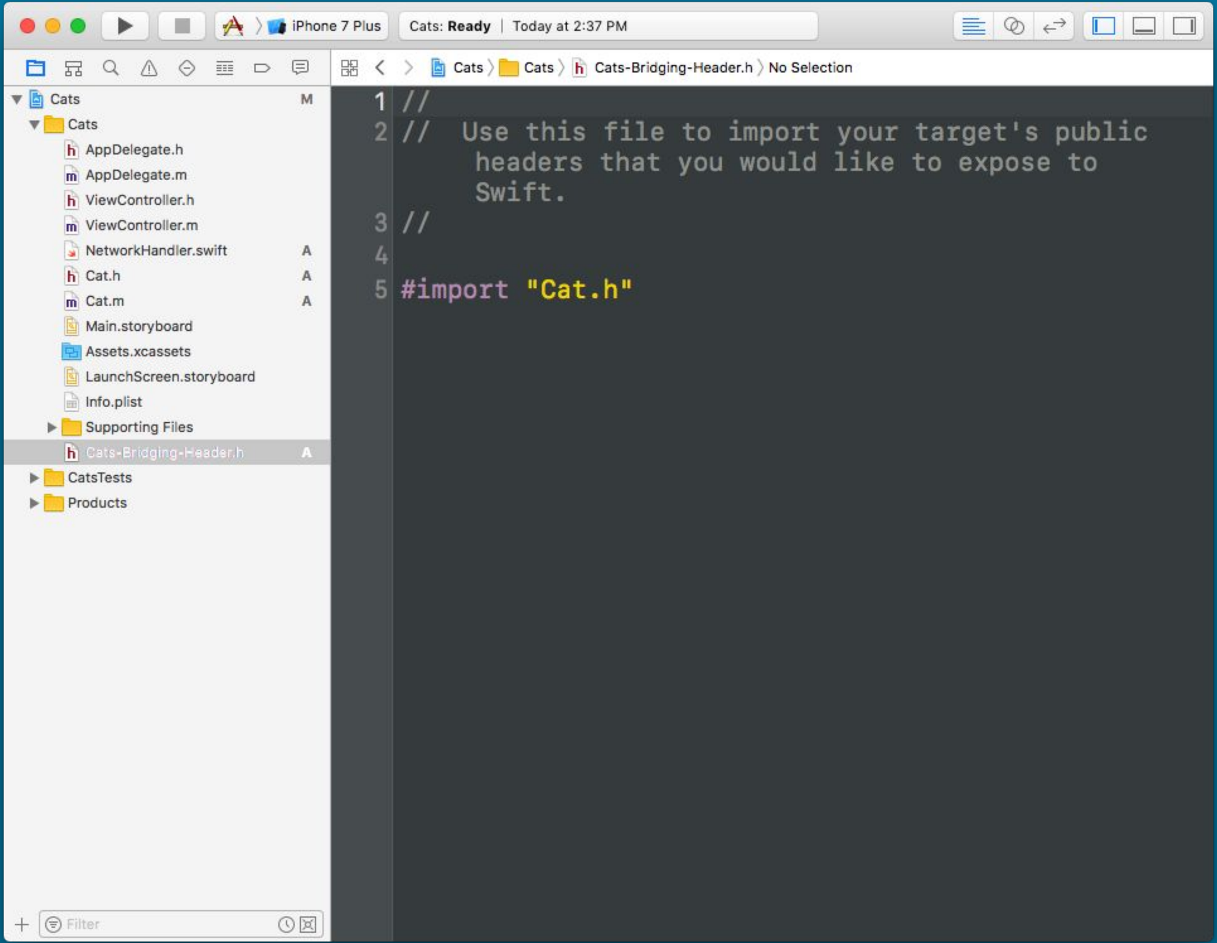
You can create this header file yourself, or let Xcode do it.

The first time you try to mix languages in a project, Xcode will ask you if you want to create a bridging header.



Would you like to configure an Objective-C bridging header?

Adding this file to MyApp will create a mixed Swift and Objective-C target. Would you like Xcode to automatically configure a bridging header to enable classes to be accessed by both languages?



```
//
//  Use this file to import your target's public headers that you would like to expose to Swift.
//

#import "Cat.h"
```

Access Swift in Objective-C

If you want to access a Swift file in Objective-C, you must import your project's Swift target header file.

This is a file that Xcode will generate and manage for us so we will never see it or modify it.

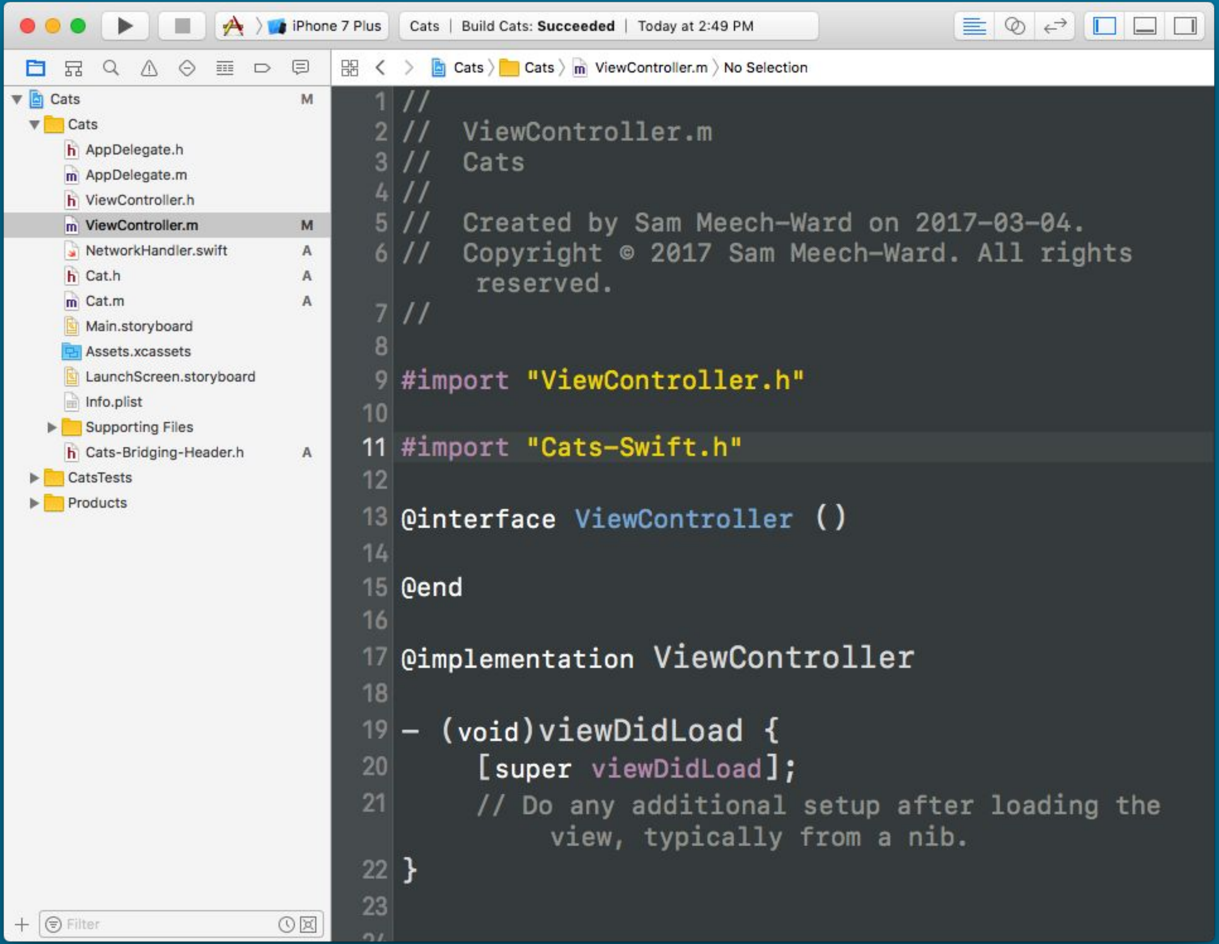
The file you need to import is {Your Project Name}-Swift.h

Access Swift in Objective-C

All you have to do is import this file into your Objective-C .m file

```
#import "Cats-Swift.h"
```

To avoid cyclical references, don’t import Swift code into an Objective-C header (.h) file, always import into an implementation (.m) file.



```
//
//  ViewController.m
//  Cats
//
//  Created by Sam Meech-Ward on 2017-03-04.
//  Copyright © 2017 Sam Meech-Ward. All rights
    reserved.
//

#import "ViewController.h"

#import "Cats-Swift.h"

@interface ViewController ()

@end

@implementation ViewController

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

Demo

- Add a Swift Object to an Objective-C Project.
- Use the Swift Object from Objective-C.

LIGHTHOUSE LABS

Nullability

In Objective-C, we work with pointers to objects that could be NULL (nil) at any time.

In Swift we use optional types to indicate NULL. So only optionals can be set to nil.

How does Objective-C tell swift if something is optional or not?

LIGHTHOUSE LABS

Nullability

In Objective-C, if we want to explicitly specify if a property or parameter can be NULL or not, we have to use the following annotations:

Function parameter: _Nonnull, _Nullable
Property: nonnull, nullable

Nullability

By default, any property that doesn’t specify its nullability will be a force unwrapped optional.

```objc
NS_ASSUME_NONNULL_BEGIN

@interface Person : NSObject

@property (nonatomic, copy) NSString *name;
@property (nonatomic, copy, nullable) NSString *address;
@property (nonatomic, copy, nullable) NSArray<Cat *> *pets;

- (instancetype)initWithName:(NSString *)name address:(NSString * _Nullable)address
NS_DESIGNATED_INITIALIZER;

@end

NS_ASSUME_NONNULL_END
```

```swift
class Person : NSObject {

    var name: String
    var address: String?
    var pets: [Cat]?

    init(name: String, address: String?)
}
```

Demo

- Use an Objective-C Object from Swift.

Subclassing Across Languages

You can’t subclass a Swift object in Objective-C. The other way is fine though.

Git Tips

- Keep your storyboards small.
- Don’t work on the same storyboard at the same time.
- Commit, push, & merge often.