Application

谷方明
fmgu2002@sina.com

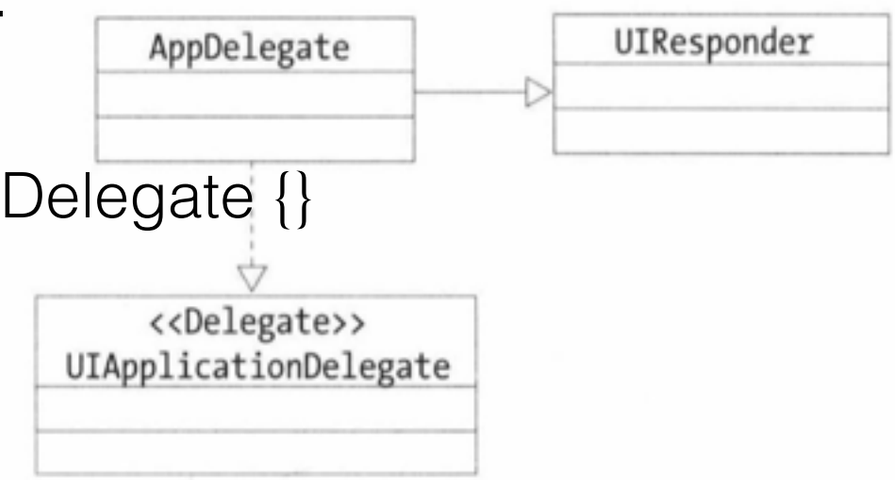
UIApplication

- Every iOS app has exactly one instance of **UIApplication** .
- When an app is launched, the system calls the UIApplicationMain(_:_:_:_:) function; among its other tasks, this function creates a Singleton **UIApplication** object. Thereafter you access the object by calling the shared class method.
- A major role of your app's application object is to **handle the initial routing of incoming user events**. It dispatches action messages forwarded to it by control objects (instances of the UIControl class) to appropriate target objects. The application object maintains a list of open windows (UIWindow objects) and through those can retrieve any of the app's UIView objects.

AppDelegate

- The **UIApplication** class defines a delegate that conforms to the UIApplicationDelegate protocol and must implement some of the protocol's methods.
- The application object informs the delegate of significant runtime events—for example, app launch, low-memory warnings, and app termination—giving it an opportunity to respond appropriately.
- AppDelegate.swift:

  class AppDelegate: UIResponder, UIApplicationDelegate {}



UIResponder类具有处理响应事件的能力；UIApplicationDelegate协议能够委托应用程序对象，用于响应应用程序的生命周期。

Delegate VS SubClass

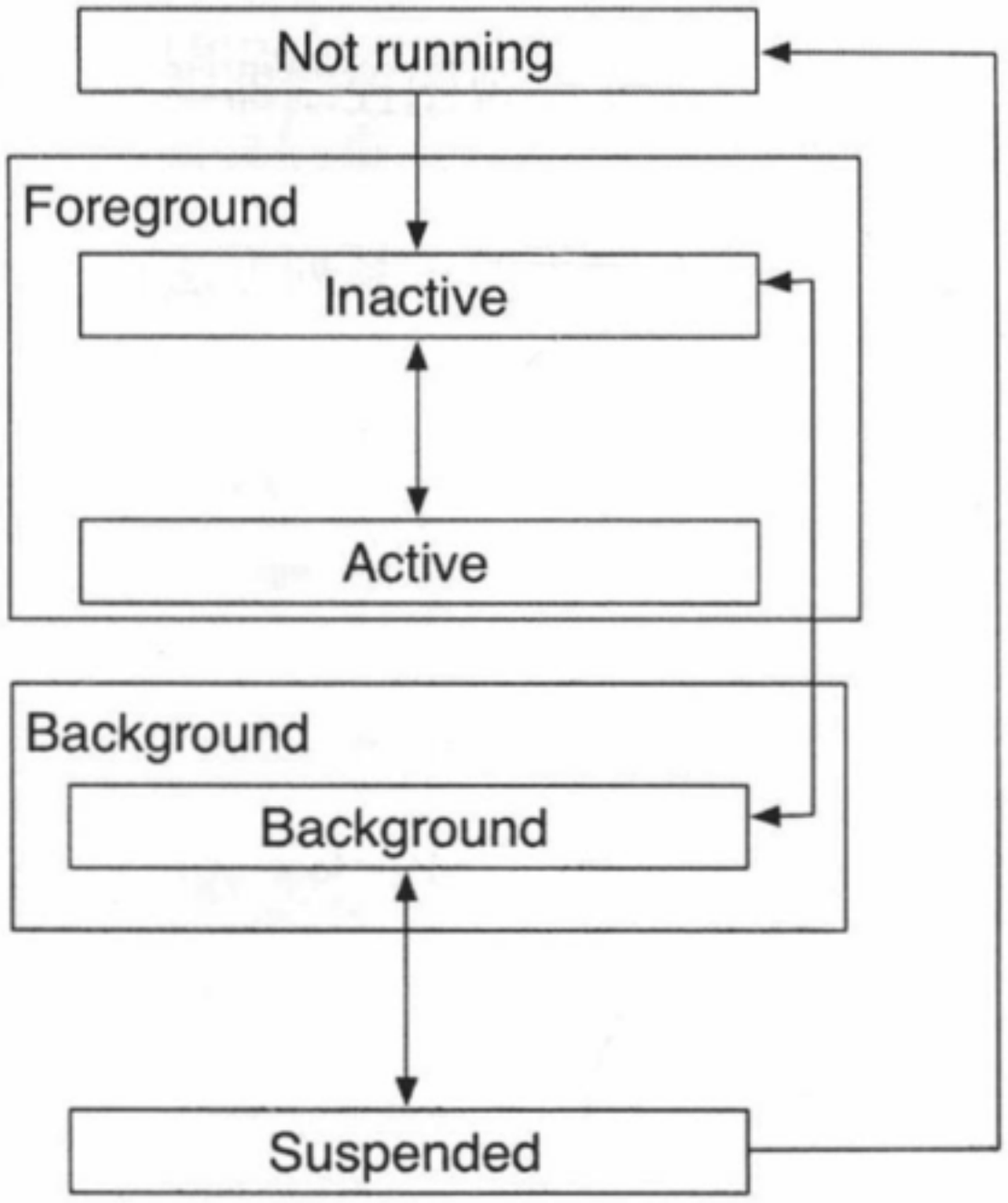
- Generally, you use an app delegate to manage interactions between the system and the app.
  - delegate是负责为其它对象处理特定任务的对象，Cocoa Touch 广泛的使用delegate
  - 优点：不需要了解类的任何内部机制

- If your app must handle incoming events before the system does—a very rare situation—you can implement a custom event or action dispatching mechanism. To do this, subclass **UIApplication** and override the sendEvent(_:) and/or the sendAction(_:to:from:for:) methods.

iOS应用的5种状态

- Not running(非运行状态)
  - 应用没运行或被系统终止
- Inactive(前台非活动状态)
  - 应用正在进入前台状态，但是不能接受事件
- Active(前台活动状态)
  - 应用进入前台状态，能接受事件
- Background(后台状态)
  - 应用进入后台。如果有可执行代码，就执行；如果没有执行代码或执行代码执行完毕，就要进入挂起状态
- Suspended(挂起状态)
  - 应用挂起后，不能执行代码；如果内存不够，应用会被终止

iOS应用生命周期(状态跃迁)



生命周期对应的方法

方法	说明
application:didFinishLaunchingWithOptions	应用启动进行初始化时会调用。实例化根视图控制器
applicationDidBecomeActive	应用进入前台并处于活动时会调用。可以恢复UI（例如游戏状态）
applicationWillResignActive	应用从活动状态进入非活动状态时会调用。可以保存UI（例如游戏状态）
applicationDidEnterBackground	应用进入后台时会调用。可以保存用户数据，释放一些资源。
applicationWillEnterForeground	应用进入前台但还没有处于活动状态时会调用。可以恢复用户数据
applicationWillTerminate	应用被终止时调用。但内存清除除外。

生命周期测试

- 使用HelloWorld或空应用

```
    func application(_ application: UIApplication, didFinishLaunchingWithOptions
launchOptions: [UIApplicationLaunchOptionsKey: Any]?) -> Bool {
    print(“application:didFinishLaunchingWithOptions”)//print(#function)
        return true
    }
    func applicationWillResignActive(_ application: UIApplication) {
        print("applicationWillResignActive")
    }
    func applicationDidEnterBackground(_ application: UIApplication) {
        print("applicationDidEnterBackground")
    }

    func applicationWillEnterForeground(_ application: UIApplication) {
        print("applicationWillEnterForeground")
    }
    func applicationDidBecomeActive(_ application: UIApplication) {
        print("applicationDidBecomeActive")
    }
    func applicationWillTerminate(_ application: UIApplication) {
        print("applicationWillTerminate")
    }
```

测试任务1

- 应用启动
  - Xcode编译执行 或 第一次启动
  - Not running -> Inactive -> Active
- 应用退出
  - Simulator -> Hardware -> Home
  - Active -> Inactive -> Background
- 应用挂起重新运行
  - Simulator -> 点击应用图标
  - Background-> Inactive -> Active

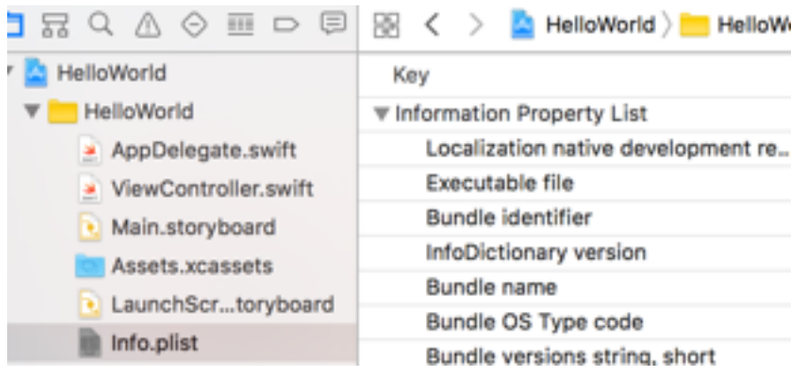
测试任务2

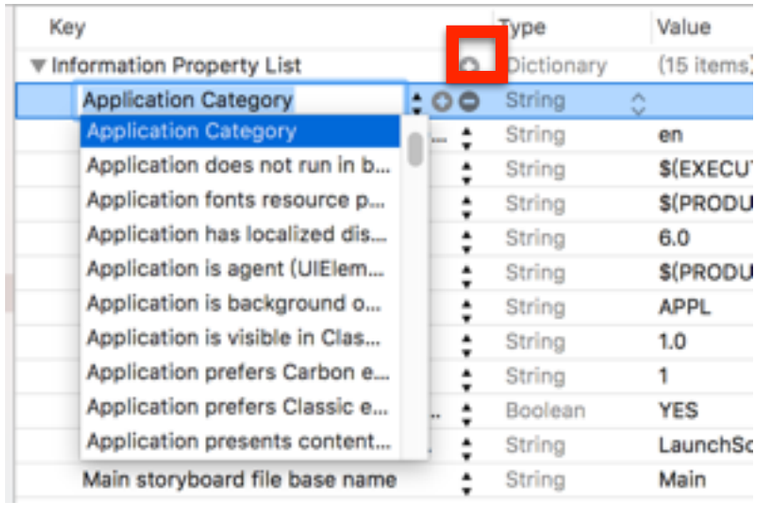
- Application does not run in background

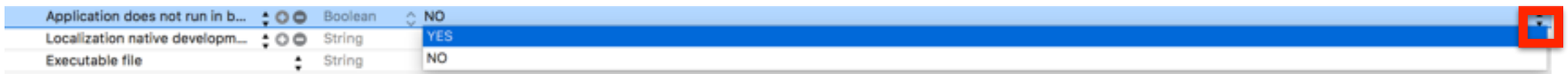
- 应用退出
  - Simulator -> Hardware -> Home
  - Active -> Inactive -> Background->Suspended->Not running

ps： 项目运行属性设置

- Application does not run in background
  - 选择 Infolplist
  - 添加属性
  - 选择Key
  - 设置Value







应用的三种构建方式

- 故事板文件
- XIB文件
- 纯代码

故事板构建

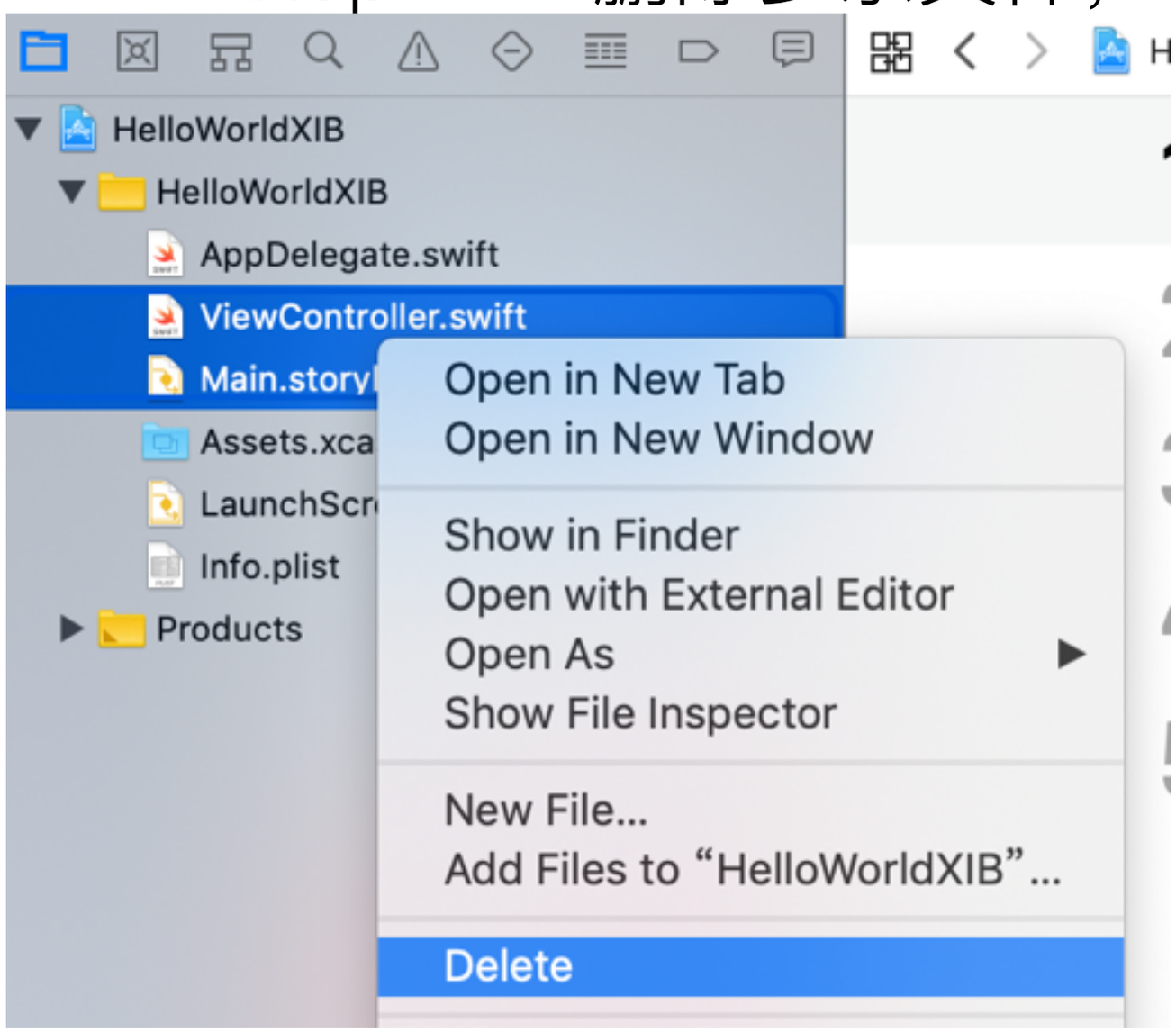
- Main.storyboard,称为“故事板”文件
- 故事板的概念源于电影行业和动画行业，也称分镜头
- 故事板文件本质上是一个XML文件，可以用来描述应用中有哪些场景（Scene）、场景中有哪些视图元素，它们的布局、事件处理以及场景间如何跳转（转场，segue）

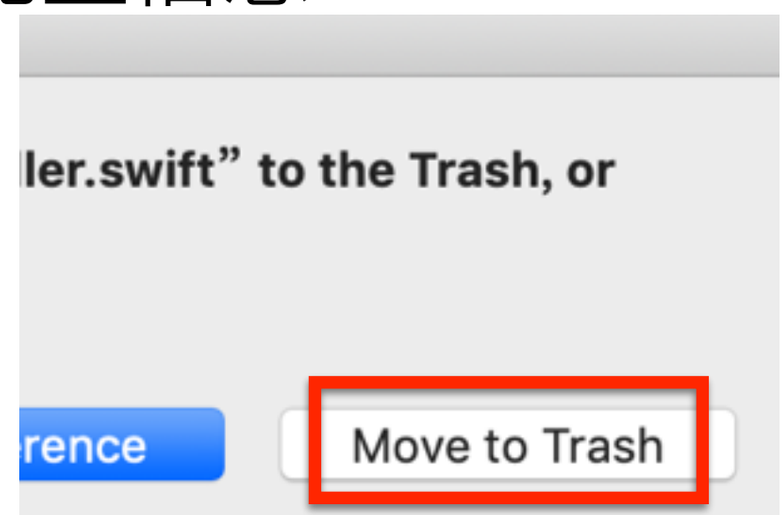
XIB文件构建

- 在一些老版本Xcode创建的工程中，经常会看到XIB文件。
- 在苹果的官方资料中会看到NIB。最初苹果的界面是使用NIB文件构建的，后来由于文件格式采用了XML格式，于是更名为XIB

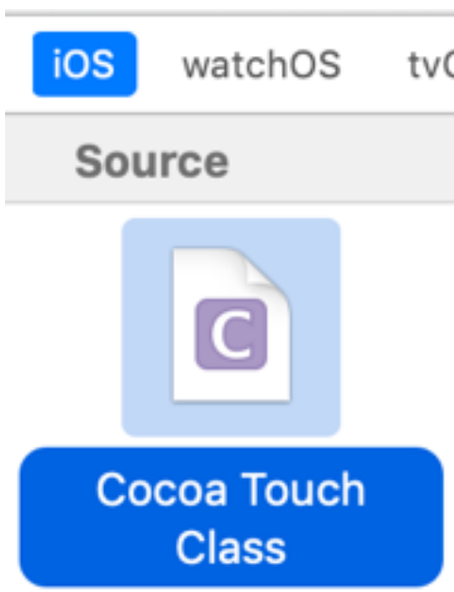
XIB文件重构HelloWorld

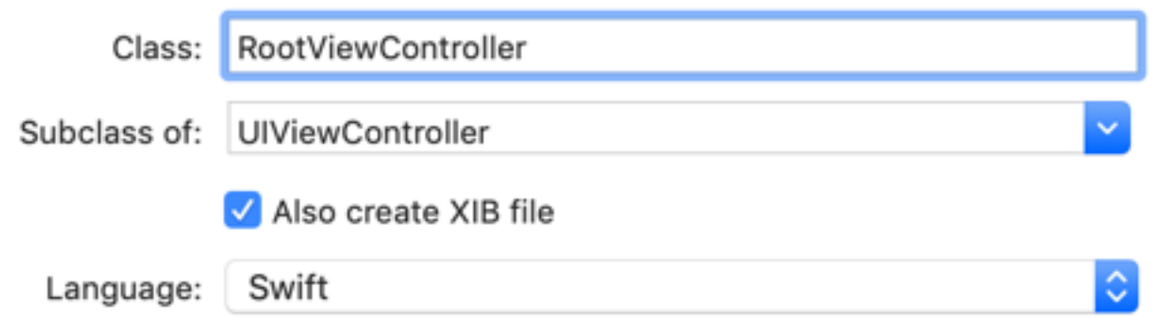
- Step 0： 新建工程，命名HelloWorldXIB
- Step 1： 删除多余文件，调整配置信息





- **Step 2： 添加根视图控制器**
  - 新建文件



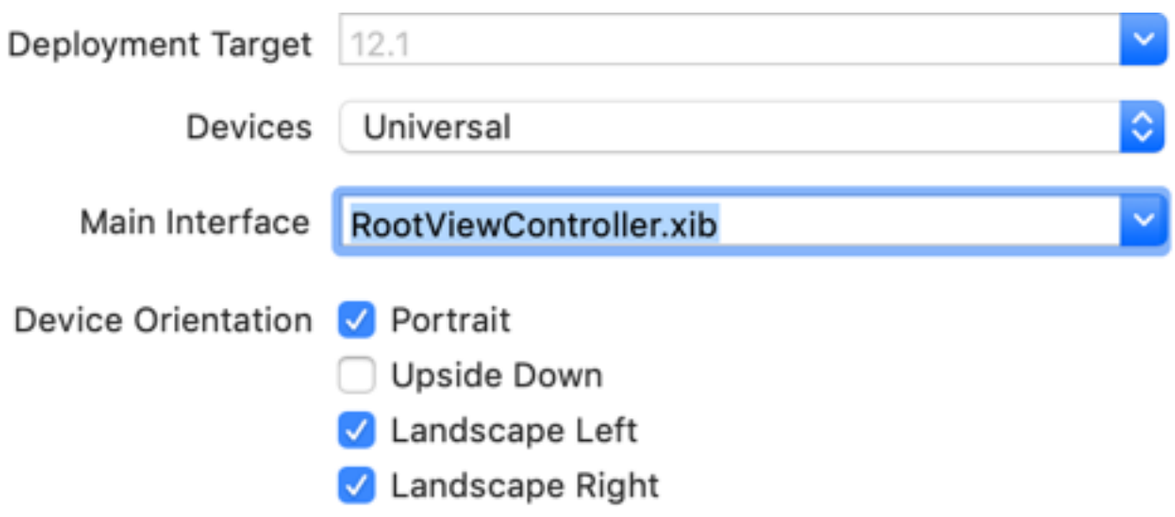


  - 使用RootViewController.XIB构建界面

- **Step 3： 在AppDelegate添加代码**

```
  func application(_ application: UIApplication,
didFinishLaunchingWithOptions launchOptions:
[UIApplicationLaunchOptionsKey: Any]?) -> Bool {

        self.window = UIWindow(frame:
UIScreen.main.bounds)
        self.window?.rootViewController =
RootViewController(nibName: "RootViewController",
bundle: nil)
        self.window?.makeKeyAndVisible()

        return true
    }
```

- **Step 4： 调整项目属性**
  - General



  - Info.plist: 删掉“Main nib file base name” (Xcode 10.1)

XIB文件 VS 故事板

- 一个工程可以有多个XIB文件，一个XIB文件对应一个视图控制器。每个XIB文件只能描述单个界面。

- 一个工程里只有一个主故事板文件。故事板文件能描述多个界面及其转场

纯代码构建

- 代码是万能的，通过代码完全可以构建应用界面。
- 但是，调试起来非常麻烦。每次修改和调整，都要重新运行查看。不是所见即所得。

纯代码重构HelloWorld

- 前面和XIB方式类似
  - Step 0: 创建项目，命名HelloWorldCode
  - Step 1: 删除ViewController和Main.storyboard
  - Step 2: 创建根视图控制器，不要选“Also create XIB file”

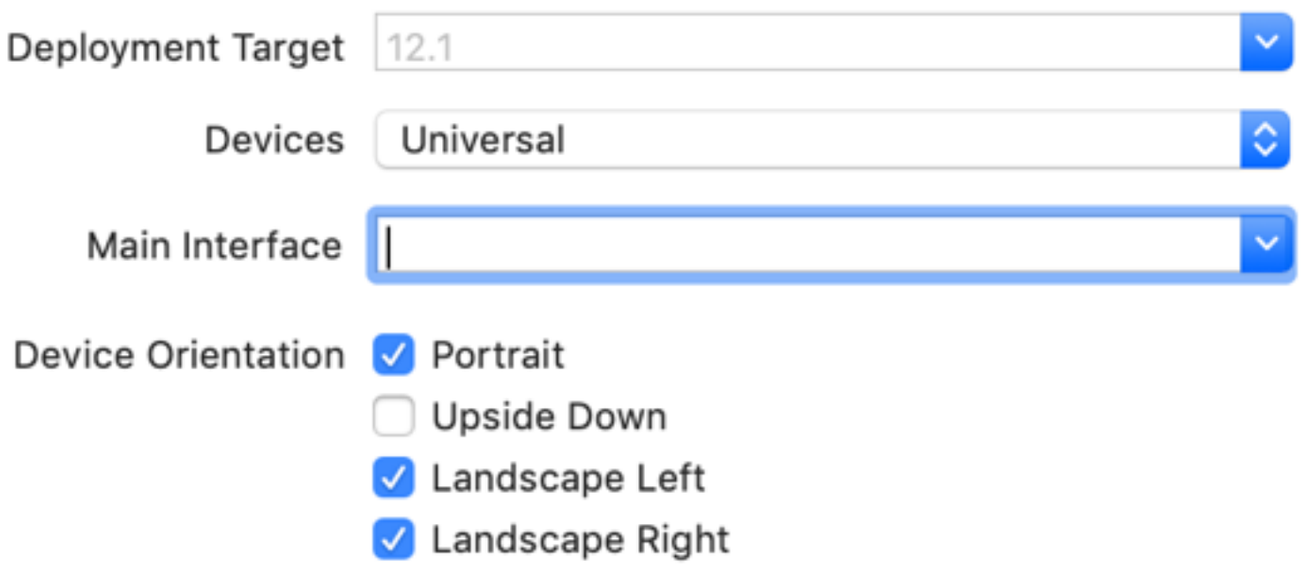
- **Step 3： 在AppDelegate添加代码**

```
  func application(_ application: UIApplication,
didFinishLaunchingWithOptions launchOptions:
[UIApplicationLaunchOptionsKey: Any]?) -> Bool {

        window = UIWindow(frame: UIScreen.main.bounds)
        window?.rootViewController = RootViewController()

        window?.backgroundColor = #colorLiteral(red: 1, green:
1, blue: 1, alpha: 1) //Input Color Literal, Select Color

        window?.makeKeyAndVisible()

        return true
    }
```

- **Step 4： 在RootViewController中添加代码**

```
    override func viewDidLoad() {
        super.viewDidLoad()

        let screen = UIScreen.main.bounds
        let labelWidth : CGFloat = 100
        let labelHeight : CGFloat = 20
        let labelTop : CGFloat = 200

        let label = UILabel(frame: CGRect(x:(screen.width -
labelWidth) / 2, y: labelTop, width: labelWidth, height: labelHeight))

        label.text = "Hello World!"
        label.textColor = UIColor.blue
        label.textAlignment = .center

        view.addSubview(label)
    }
```

- **Step 5： 调整项目属性（也适用XIB情况）**
  - General



  - 如果项目崩溃，则关掉重启。