

第十部分 其他主题



授课教师：杨文川



其他主题 (2学时)

本章目的在于使学生掌握iOS应用开发中涉及到的一些相对分散又比较重要的知识点，包括：在Xcode中调试程序、项目的本地化以及应用发布等。

- Xcode调试技术
- 国际化与本地化
- 应用发布



Xcode调试技术

- 在开发应用时，不可避免会犯错(语法错误和逻辑错误)。当发现错误后，如何定位错误并改正错误呢？

Xcode提供了强大的错误调试工具，帮助开发人员识别并定位错误，然后通过代码执行过程来检查控制流和数据结构的状态找到错误的原因，直到修复所有错误。

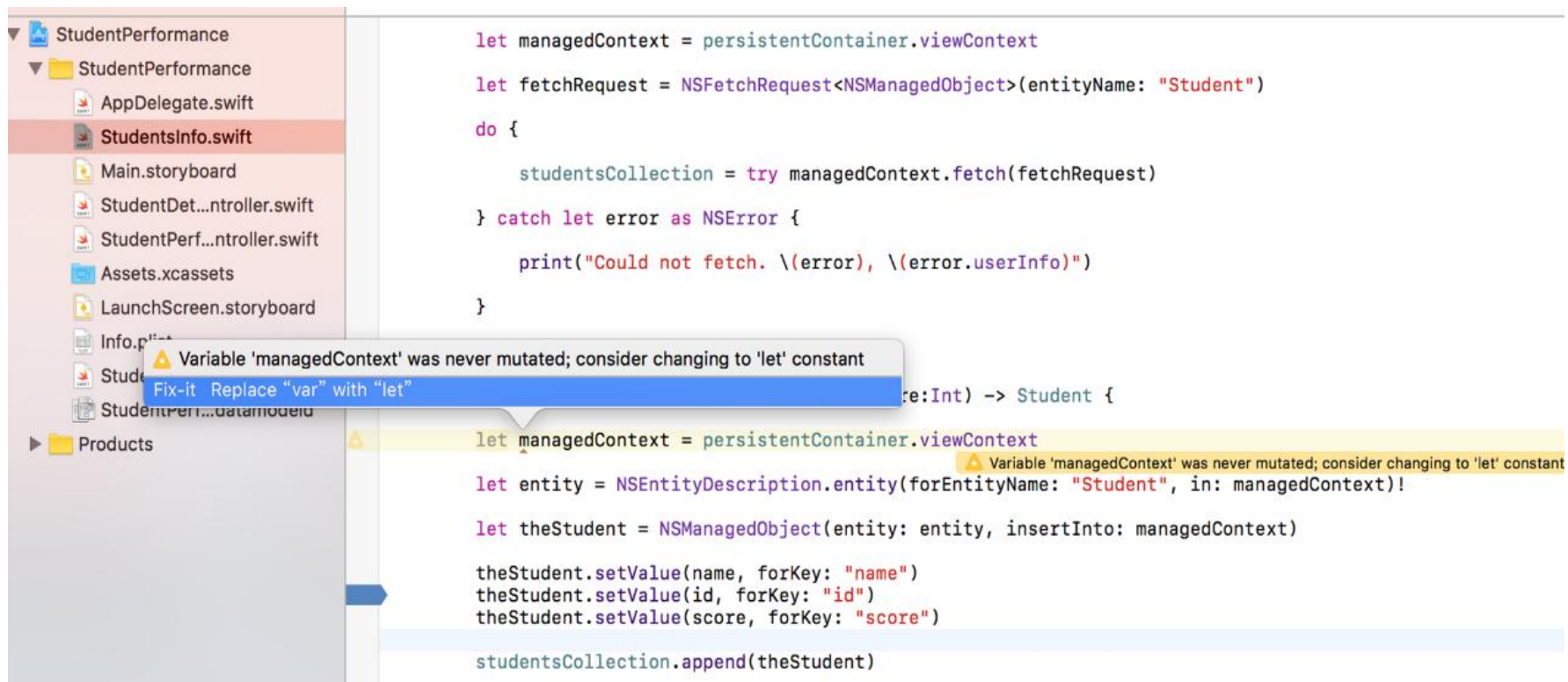
- 语法错误

对于语法错误或可能引起错误的情况，Xcode会即时的在问题语句行显示，并给出修复方案。对于开发者来说，只需要确认错误及修复方案，直接点击确认就一键完成了修复。

- 逻辑错误

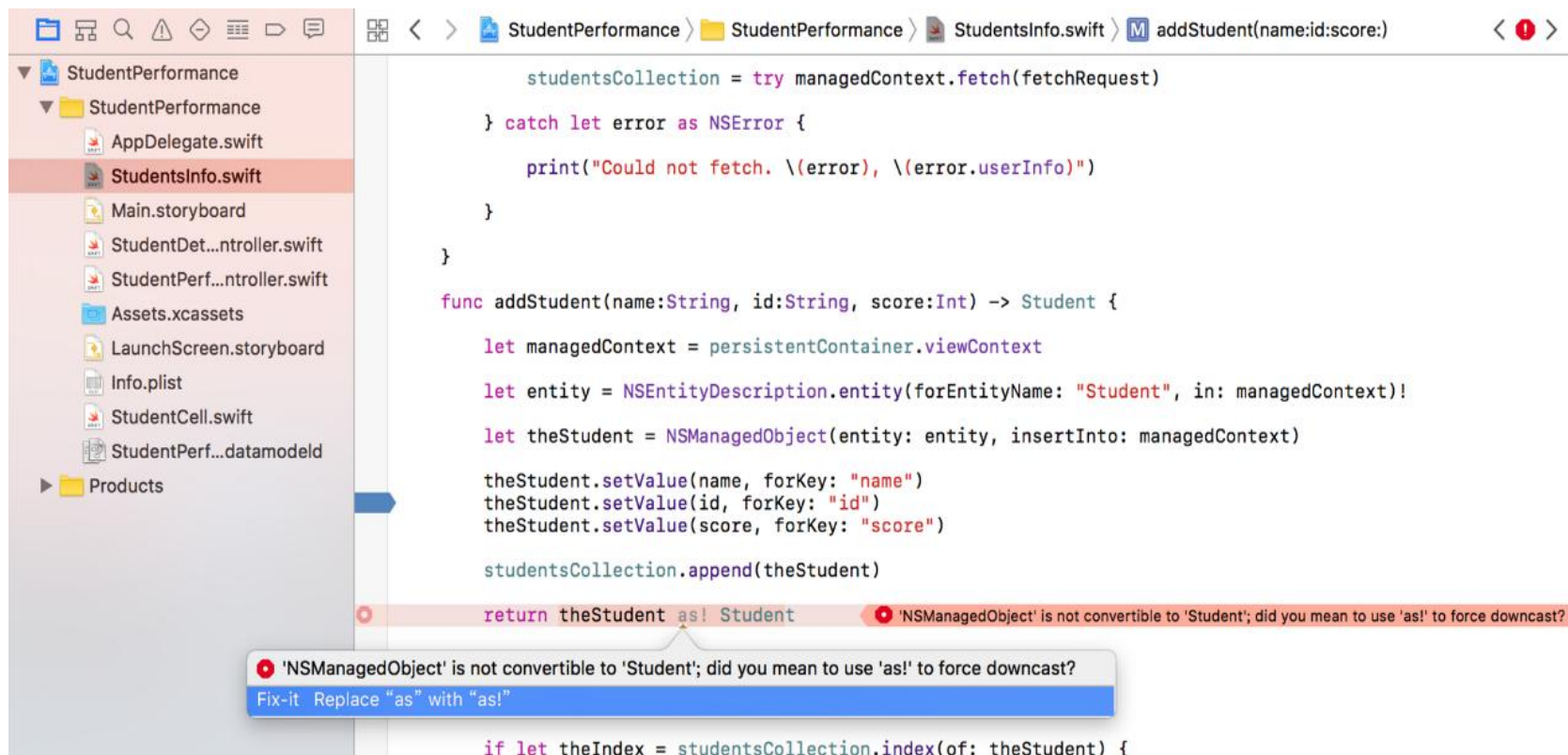
逻辑错误大部分需要在运行阶段才能发现并定位。修复逻辑错误相对来说比较复杂，往往需要获取程序运行阶段各种数据，根据这些数据进行分析后，才能定位错误并最终解决错误。

语法错误实例



代码参见iosPrj的Chapter07-StudentPerformance6.2
将managedContext前的let变为var看看啥现象

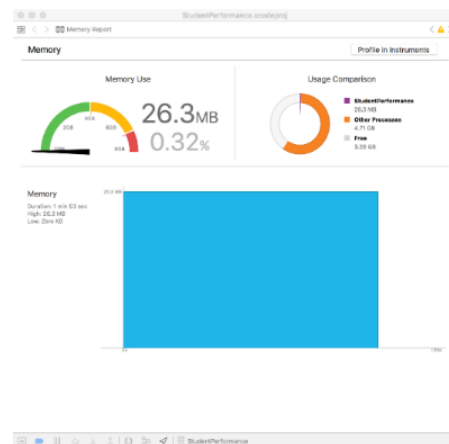
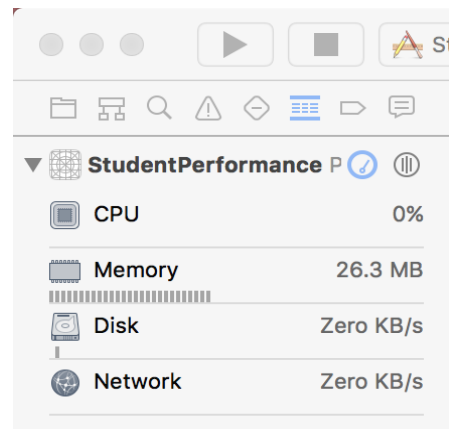
语法错误实例



将addStudent最后的as! 改为as，看看啥现象

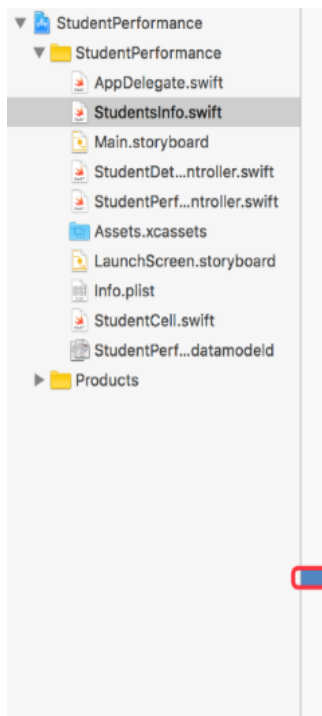
常用调试工具

- 调试测量器：当项目进入到调试状态后，点击最左侧工具栏的第六个标签，即可打开。该测量器动态显示了正在运行的应用的资源占用情况，包括：CPU、内存、硬盘以及网络。
- 调试测量器通过列表简要显示了CPU、内存、硬盘和网络的资源占用情况，点击其中任意一个条目，还可以进一步打开详情窗口。



设置断点

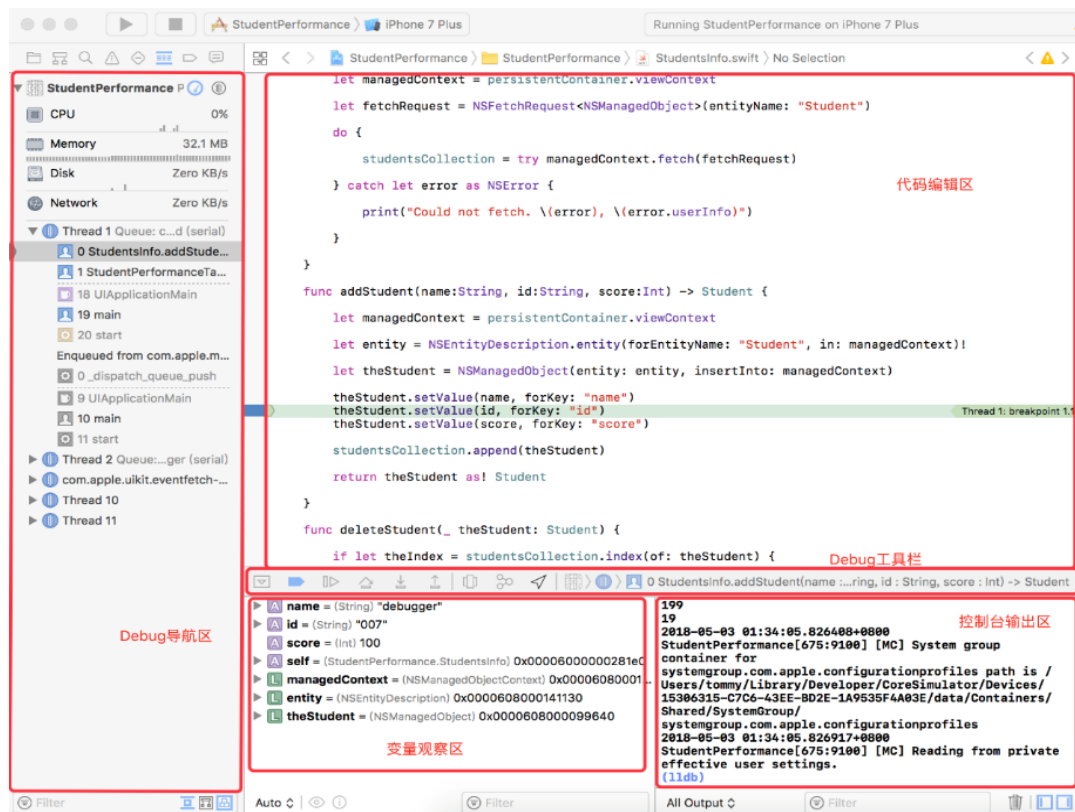
- 设置断点是一个广泛使用的程序调试方法，它可以通过检查程序运行过程中的实际状态来帮助开发者判断错误的根源。
- 断点会中断程序的正常执行，使开发者可以单步跟踪程序执行中的各个状态，包括：变量的值、控制流的执行过程等。在编辑界面中设置一个断点非常便利。



```
}  
  
init() {  
    let managedContext = persistentContainer.viewContext  
    let fetchRequest = NSFetchRequest<NSManagedObject>(entityName: "Student")  
    do {  
        studentsCollection = try managedContext.fetch(fetchRequest)  
    } catch let error as NSError {  
        print("Could not fetch. \(error), \(error.userInfo)")  
    }  
}  
  
func addStudent(name:String, id:String, score:Int) -> Student {  
    let managedContext = persistentContainer.viewContext  
    let entity = NSEntityDescription.entity(forEntityName: "Student", in: managedContext)!  
    let theStudent = NSManagedObject(entity: entity, insertInto: managedContext)  
    theStudent.setValue(name, forKey: "name")  
    theStudent.setValue(id, forKey: "id")  
    theStudent.setValue(score, forKey: "score")  
    studentsCollection.append(theStudent)  
    return theStudent as! Student  
}
```

断点调试

- 设置完断点后，直接点击运行程序，程序运行到断点处时会自动切换到调试界面。
- 调试界面分为四个主要的区域，包括：Debug导航区、变量观察区、控制台输出区、代码编辑区。
 - Debug导航区由两部分组成：调试测试器和过程查看器。过程查看器用来检查当前应用的执行控制流。
 - 变量观察区显示当前各个相关变量的值。控制台输出区用来动态显示系统信息。
 - 代码编辑区显示当前正在执行的源代码文件，并高亮显示正在执行的语句行。





调试工具栏

- 在调试过程中，最常用的就是调试工具栏。

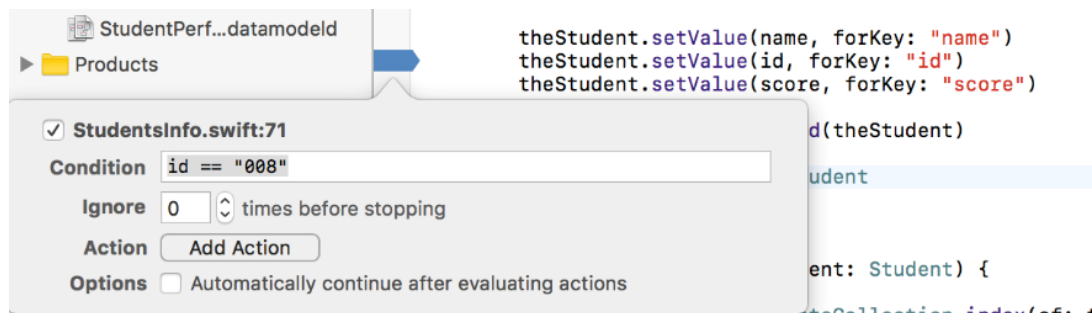
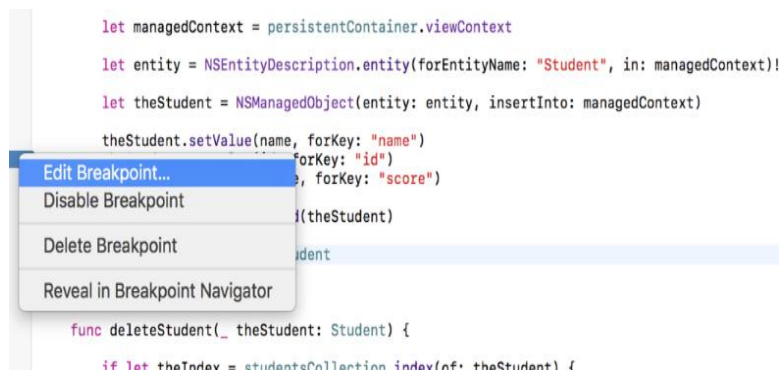
从左到右依次为隐藏/显示的切换按钮、断点激活按钮、继续执行/中断的状态切换按钮、以及三个单步跟踪按钮。

三个单步跟踪按钮分别为：step over (执行到下一行语句)、step into (进入当前语句的内部，通常为函数)、step out (跳出当前函数)。

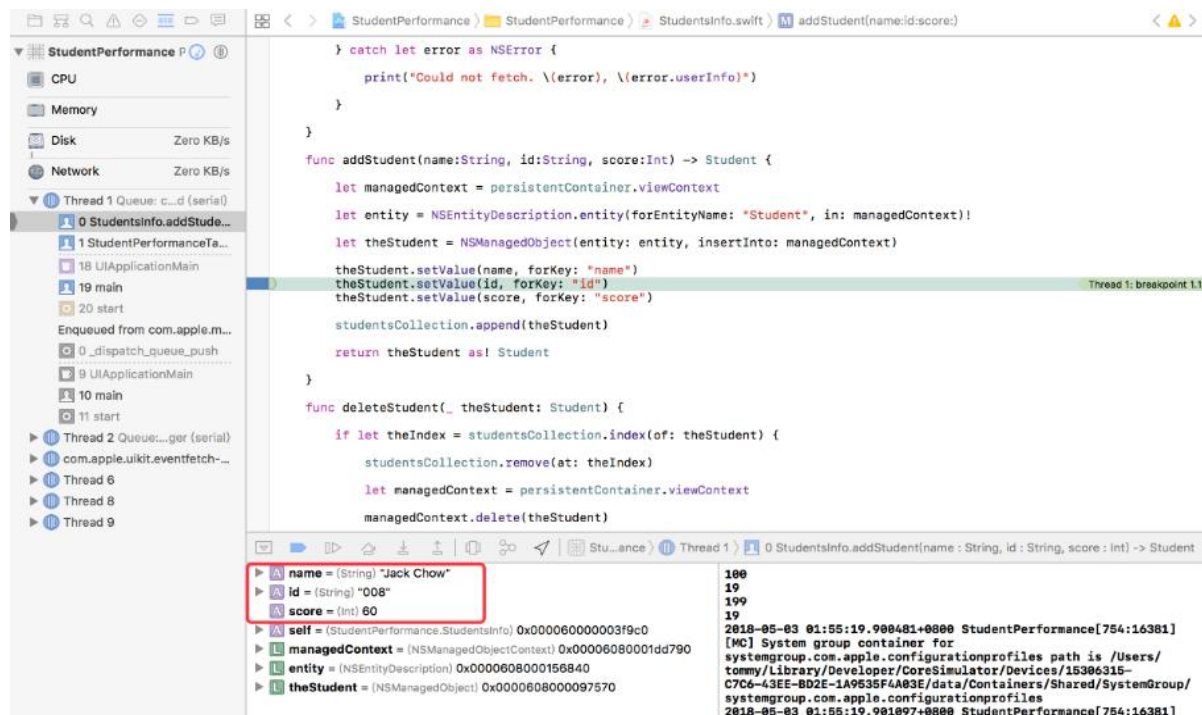


编辑断点

- 在设置断点的时候，还可以对断点进行编辑。在断点处点击鼠标右键会弹出一个菜单。通过该菜单，可以对断点进行编辑、失效、删除、显示等。
- 在断点编辑窗口中，可以为该断点设置触发条件、忽略执行的次数、触发的动作等。在这里，添加一个断点触发条件，即：`id == "008"`。应用执行过程中，当添加的学生的id等于008，则程序执行到该断点处，断点被激活，进入到调试模式；如果添加的学生的id不等于008，则该断点无效，程序不会中断执行。



运行断点





第十部分 其他主题

本章目的在于使学生掌握iOS应用开发中涉及到的一些相对分散又比较重要的知识点，包括：在Xcode中调试程序、项目的本地化以及应用发布等。

- Xcode调试技术
- 国际化与本地化
- 应用发布



国际化与本地化

- 本地化就是将应用中的一种语言翻译为多种语言的过程。而要实现应用的本地化，就必须先进行国际化。国际化就是使应用具备支持不同语言显示的能力的过程。
- 目前，苹果的App Store已经可以在150多个国家使用，国际化是任何一个应用想要打开全球市场所必经的一步。因此，应用的国际化与本地化是每一个开发者的必修内容。



实例：汇率转换工具2.0

- 要求

- 在实例“汇率转换工具”的基础上，实现在简体中文和英文两种系统语言下都能够正确显示相应的语言文字。

代码参见iosPrj的Chapter09-
LocalizationSample

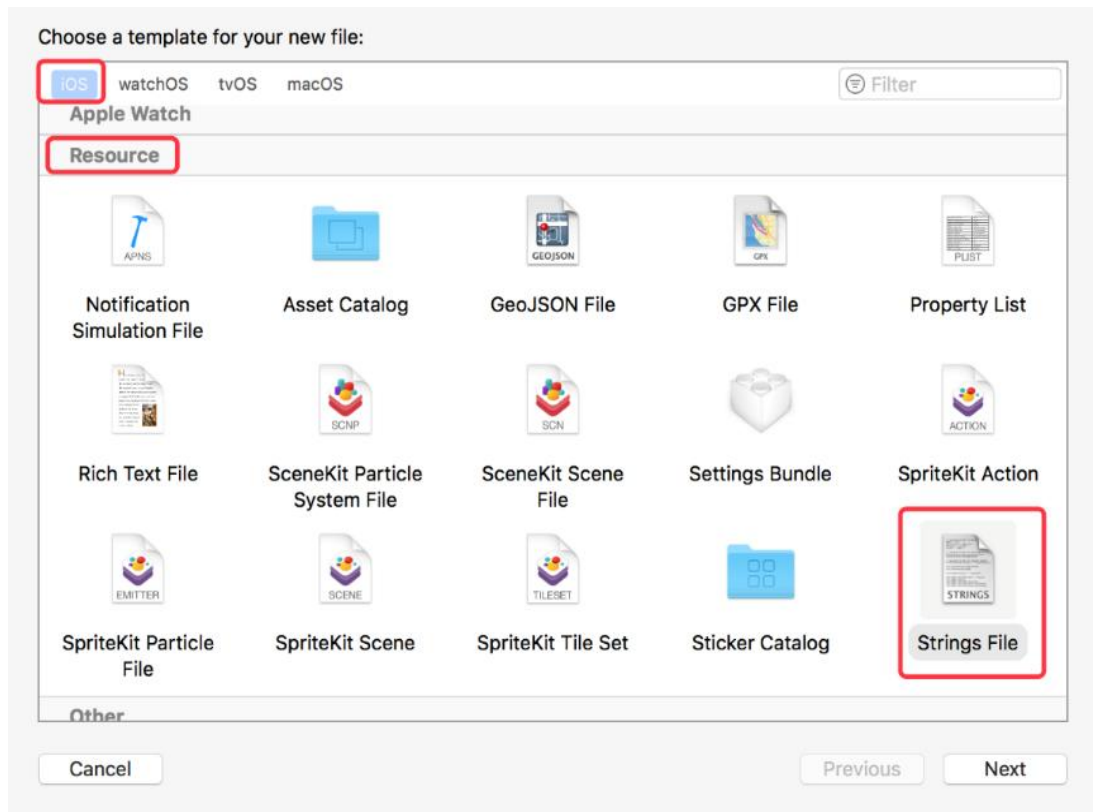
重置界面文字

- 原来的项目中是将控件显示的字符串直接写在属性文件中的，要实现这些字符串的本地化就必须将其放入到一个独立的文件，然后根据不同的系统语言来获取不同的字符串。
- 将这四个控件中预先写入的文字修改为控件的名字。而控件应该显示的文字，将在运行阶段写入。



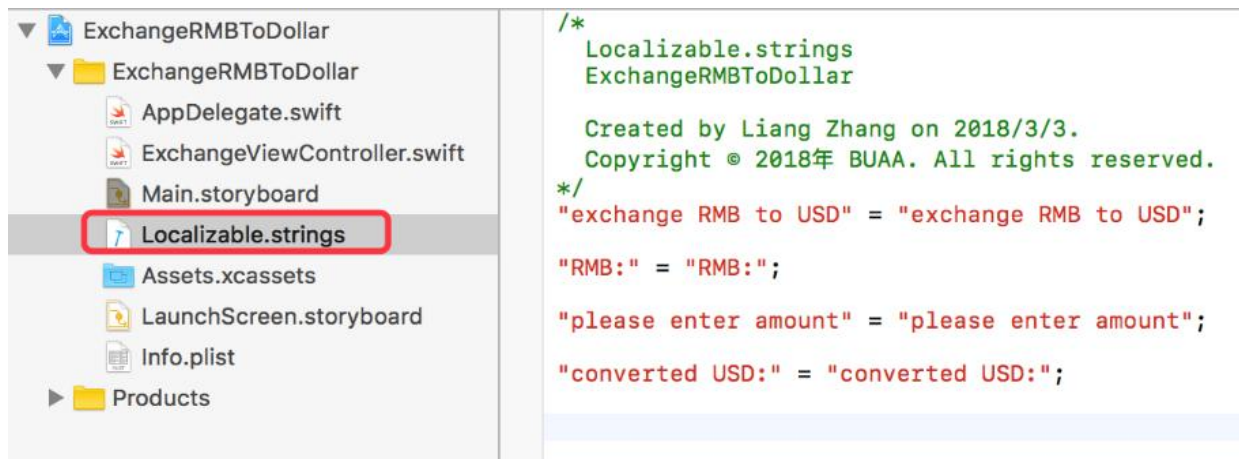
创建本地化文件

- 在iOS中，应用中不同语言所使用的本地化字符串都是保存在.strings本地化文件中的。
- 创建本地化文件：
通过菜单创建一个新文件，在文件类型对话框中选择iOS\Resource\Strings File，然后点击“Next”
文件名保存为Localizable。
Localizable.strings是iOS系统中缺省的本地化文件名。



编辑本地化文件

- Localizable.strings文件是按照“键-值”对的方式来保存数据的。
- 向文件中添加四个控件中要显示文字的字符串，等号左边为键，右边为值。





初始化显示文字

- 在视图控制器 ExchangeViewController 中重载函数 `viewDidLoad`，实现视图载入之前进行四个控件中文字内容的初始化。函数

`NSString(_key:tableName:bundle:value:comment:)`

用来根据 `key` 获取本地化文件中对应的值，

这里 `tableName`、`bundle`、`value` 都可以取缺省值，因此调用该函数时只需要提供 `key` 和 `comment`。

```
override func viewDidLoad() {  
    super.viewDidLoad()  
  
    titleLabel.text = NSLocalizedString("exchange RMB to USD", comment: "title label")  
    hintLabel.text = NSLocalizedString("RMB:", comment: "hint label")  
    infoLabel.text = NSLocalizedString("converted USD:", comment: "info label")  
    dollarTextField.placeholder = NSLocalizedString("please enter amount", comment:  
        "place holder of text field")  
}
```

运行效果

iPhone 7 – iOS 10.3 (14E8301)
Carrier 6:00 PM

exchange RMB to USD

RMB:

please enter amount

converted USD: ???

添加中文的本地化支持

The screenshot shows the Xcode interface with the 'ExchangeRMBToDollar' project selected. The left sidebar shows the project's file structure, including source files like AppDelegate.swift and ExchangeViewController.swift, as well as assets like Main.storyboard and Localizable.strings. The middle pane shows the 'PROJECT' and 'TARGETS' sections, with 'ExchangeRMBToDollar' selected under 'TARGETS'. The right pane shows the 'Build Settings' for the selected target. The 'Deployment Target' is set to 'iOS Deployment Target 10.3'. The 'Configurations' section shows 'Debug' and 'Release' configurations, both with 'No Configurations Set'. The 'Localizations' section shows 'English — Development Language' with '2 Files Localized'. A red box highlights the '+' button at the bottom of the 'Localizations' section, indicating where to click to add a new language.

PROJECT

TARGETS

ExchangeRMBToDollar

Build Settings

Deployment Target

iOS Deployment Target 10.3

Configurations

Name	Based on Configuration File
Debug	No Configurations Set
Release	No Configurations Set

+ —

Use Release for command-line builds

Localizations

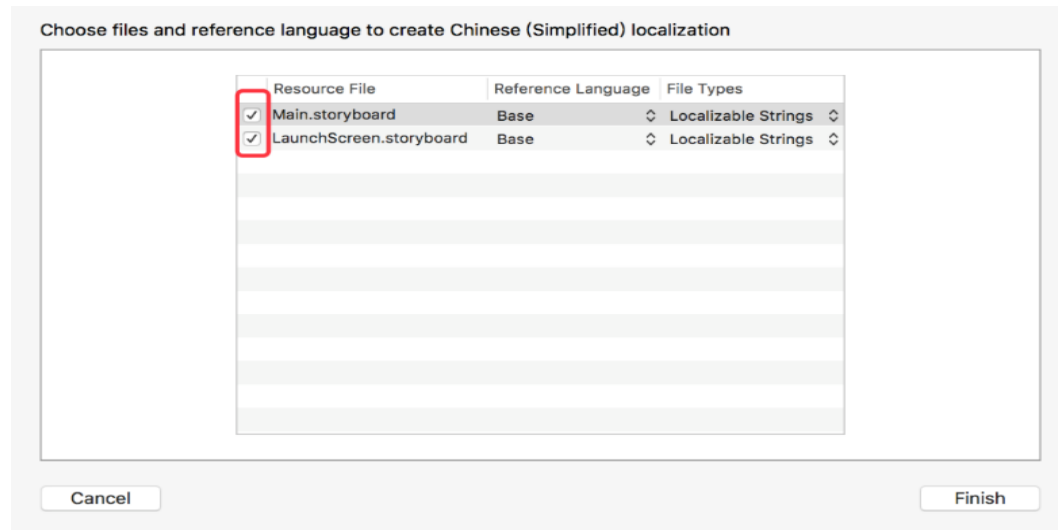
Language	Resources
English — Development Language	2 Files Localized

+ —

☒ Use Base Internationalization

选择需要本地化的文件

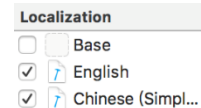
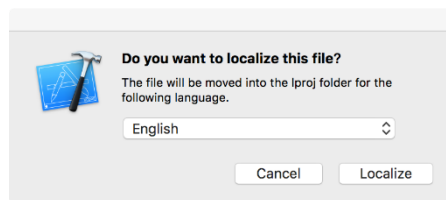
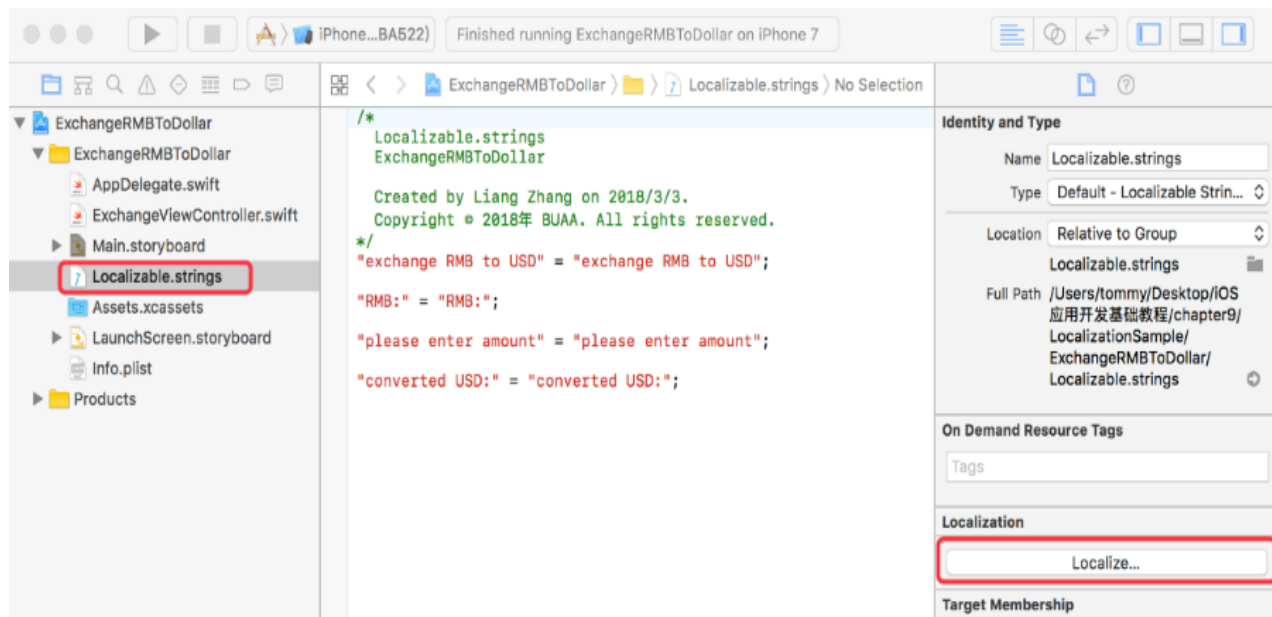
- 在弹出的对话框中选择需要本地化为简体中文的项目文件
- 打开项目工程文件目录，可以检查新创建的简体中文本地化文件“zh-Hans.lproj”
- “Base.lproj”为项目缺省的本地化文件夹，应用运行的时候如果找不到对应语言的本地化文件，则会使用该文件夹中的资源。



名称	修改日期	大小	种类
AppDelegate.swift	2018年3月3日 下午6:48	2 KB	Swift Source
Assets.xcassets	2018年3月3日 下午7:15	--	文件夹
Base.lproj	今天 下午5:59	--	文件夹
ExchangeViewController.swift	今天 下午5:58	2 KB	Swift Source
Info.plist	2018年3月3日 下午6:48	1 KB	Property List
Localizable.strings	今天 下午5:58	292 字节	Strings File
zh-Hans.lproj	今天 下午6:03	--	文件夹

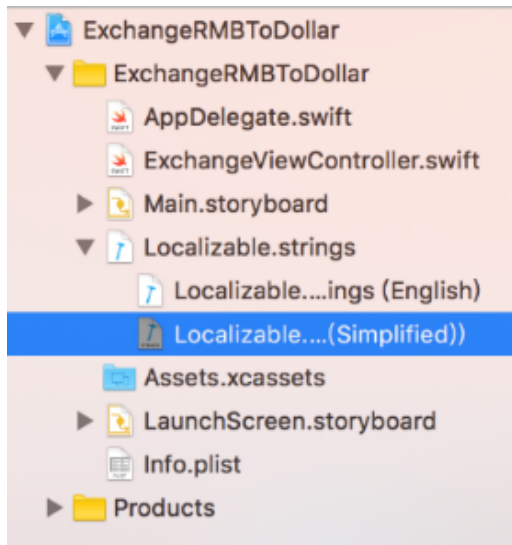
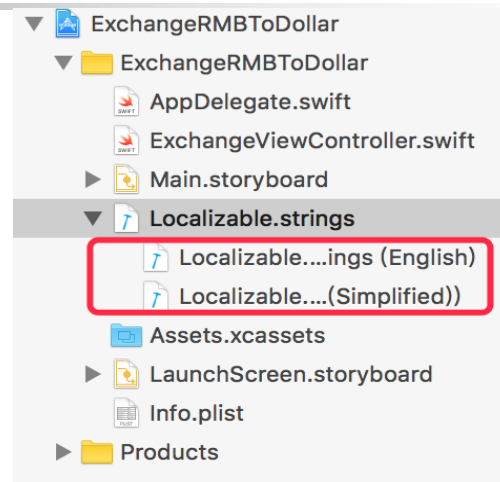
装载本地化文件

- 将已经创建的 Localizable.strings 文件放入到各个本地化文件夹中
- 系统会弹出对话框询问该文件(即：Localizable.strings)属于何种语言，这里我们选择英语
- 在本地化文件浏览窗口中会显示已经支持本地化的语言，在这里复选“English”和“Chinese”



编写中文本地化文件

- 展开项目文件浏览器中的文件 Localizable.strings，可以看到分别生成了英文和简体中文的本地化文件
- 选中 Localizable.strings(Simplified)，将每个键对应的值修改为相应的中文字符串



```
/*
 Localizable.strings
 ExchangeRMBToDollar

 Created by Liang Zhang on 2018/3/3.
 Copyright © 2018年 BUAA. All rights reserved.
 */
"exchange RMB to USD" = "人民币兑换为美元";

"RMB:" = "人民币:";

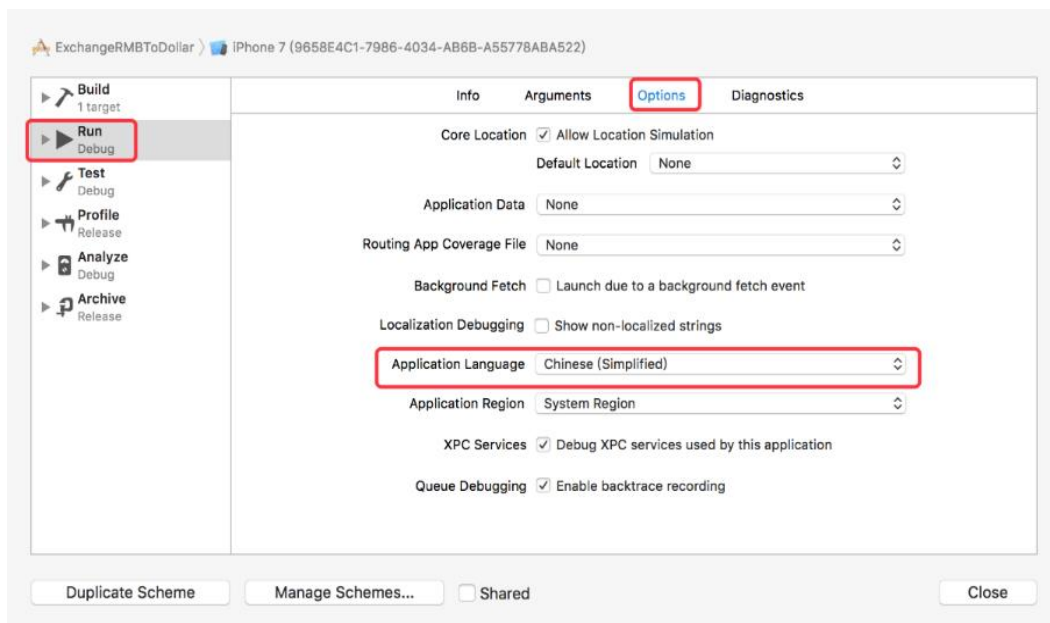
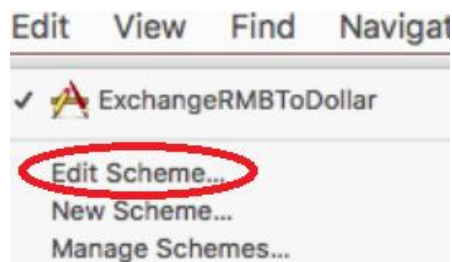
"please enter amount" = "请输入金额";

"converted USD:" = "可转换为美金:";
```

运行前的设置

■ 检验效果：

- 点击项目运行的脚本，选择“Edit Scheme”，然后打开脚本编辑窗口。
- 选择Run\Options\Application Language\Chinese(Simplified)，点击Close。
- 这样，设备运行时的系统语言就设置为了简体中文，而缺省的情况为英文。



运行效果

iPhone 7 - iOS 10.3 (14E8301)
Carrier 6:15 PM

人民币兑换为美元

人民币:

请输入金额

可转换为美金:

???



第十部分 其他主题

本章目的在于使学生掌握iOS应用开发中涉及到的一些相对分散又比较重要的知识点，包括：在Xcode中调试程序、项目的本地化以及应用发布等。

- Xcode调试技术
- 国际化与本地化
- 应用发布

代码参见iosPrj的Chapter09-SampleRelease

应用图标

代码参见iosPrj的Chapter09-SampleRelease

- 打开工程，在项目文件浏览窗口中选择文件 Assets.xcassets，在中间栏中选择AppIcon条目，即打开了应用图标设置窗口。

- Assets.xcassets是系统自动生成的项目图片资源文件夹，可以在工程文件夹下找到这个目录。系统中要用到图片文件都可以添加到该文件夹下。

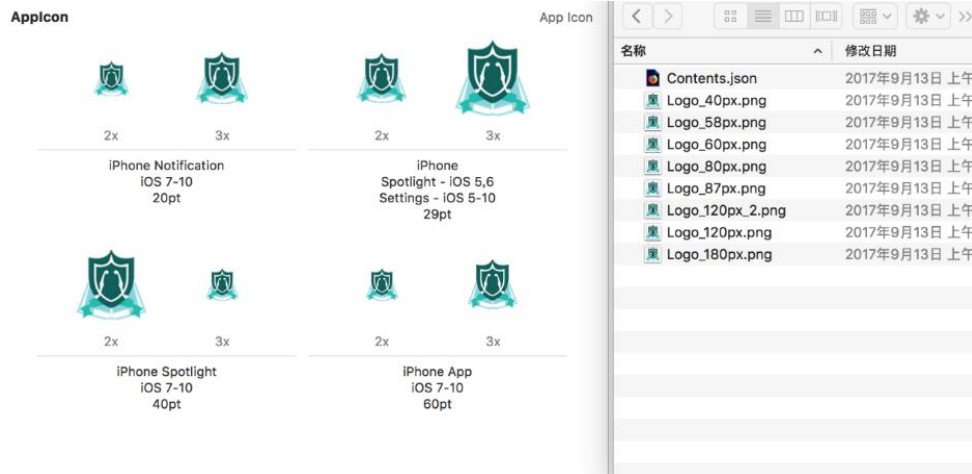
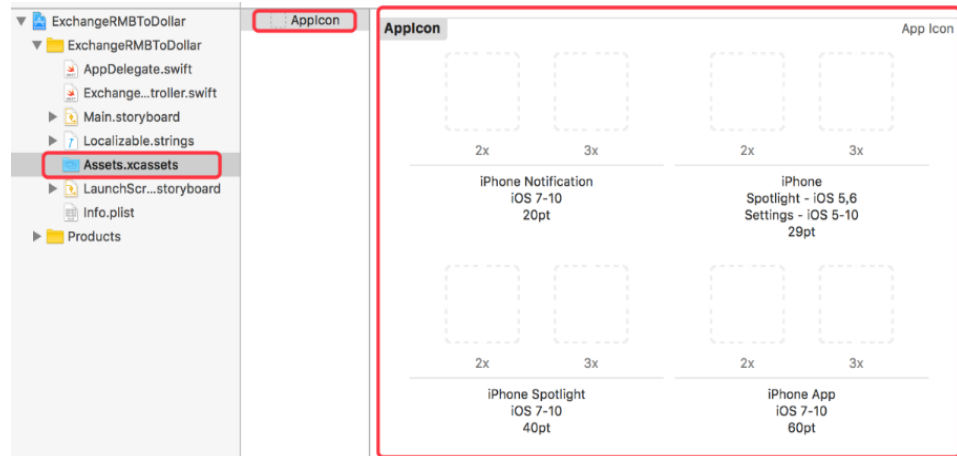
- Assets.xcassets中的子文件夹AppIcon是由系统自动创建的，用来存放应用图标。

- 系统根据运行设备自动显示了全部图标，包括

- Notification(通知栏中的应用图标)、
- Spotlight(搜索栏中的应用图标)、
- Settings(系统设置中的应用图标)、
- App(桌面上的应用图标)。

- pt表示单位“点”，2x表示2倍尺寸。

- 2x和3x对应图片尺寸分别为：
 $60\text{pt} \times 60\text{pt} @ 2x = 120\text{px} \times 120\text{px}$;
 $60\text{pt} \times 60\text{pt} @ 3x = 180\text{px} \times 180\text{px}$ 。



运行效果



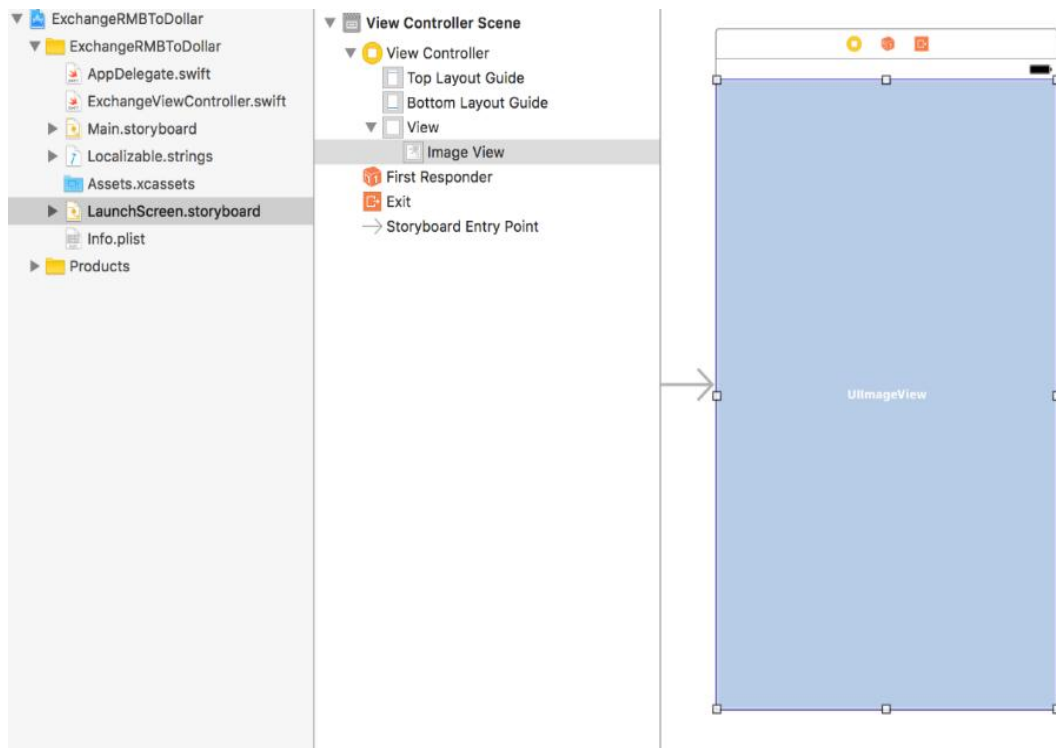
启动画面

- 应用启动时，启动画面会首先出现，然后它会被应用的主界面所替代。
- 启动画面是为了提升应用的响应速度，避免用户产生等待过久的感觉。
- 苹果官方要求所有的应用都必须有一个启动画面。
- 由于苹果设备屏幕尺寸的多样性，启动画面的尺寸要求也各不相同，右表列出了启动画面的图片规格。
- 建议使用故事板来实现启动画面的管理，除了项目主界面的故事板以外，可以创建一个启动画面专用的故事板。
- 在设计启动画面的时候要尽量避免使用任何文字，因为启动画面是静态的，无法进行本地化。

设备型号	竖屏尺寸	横屏尺寸
iPhoneX	1125px*2436px	2436px*1125px
iPhone8 Plus	1242px*2208px	2208px*1242px
iPhone8	750px*1334px	1334px*750px
iPhone7 Plus	1242px*2208px	2208px*1242px
iPhone7	750px*1334px	1334px*750px
iPhone6s Plus	1242px*2208px	2208px*1242px
iPhone6s	750px*1334px	1334px*750px
iPhoneSE	640px*1136px	1136px*640px

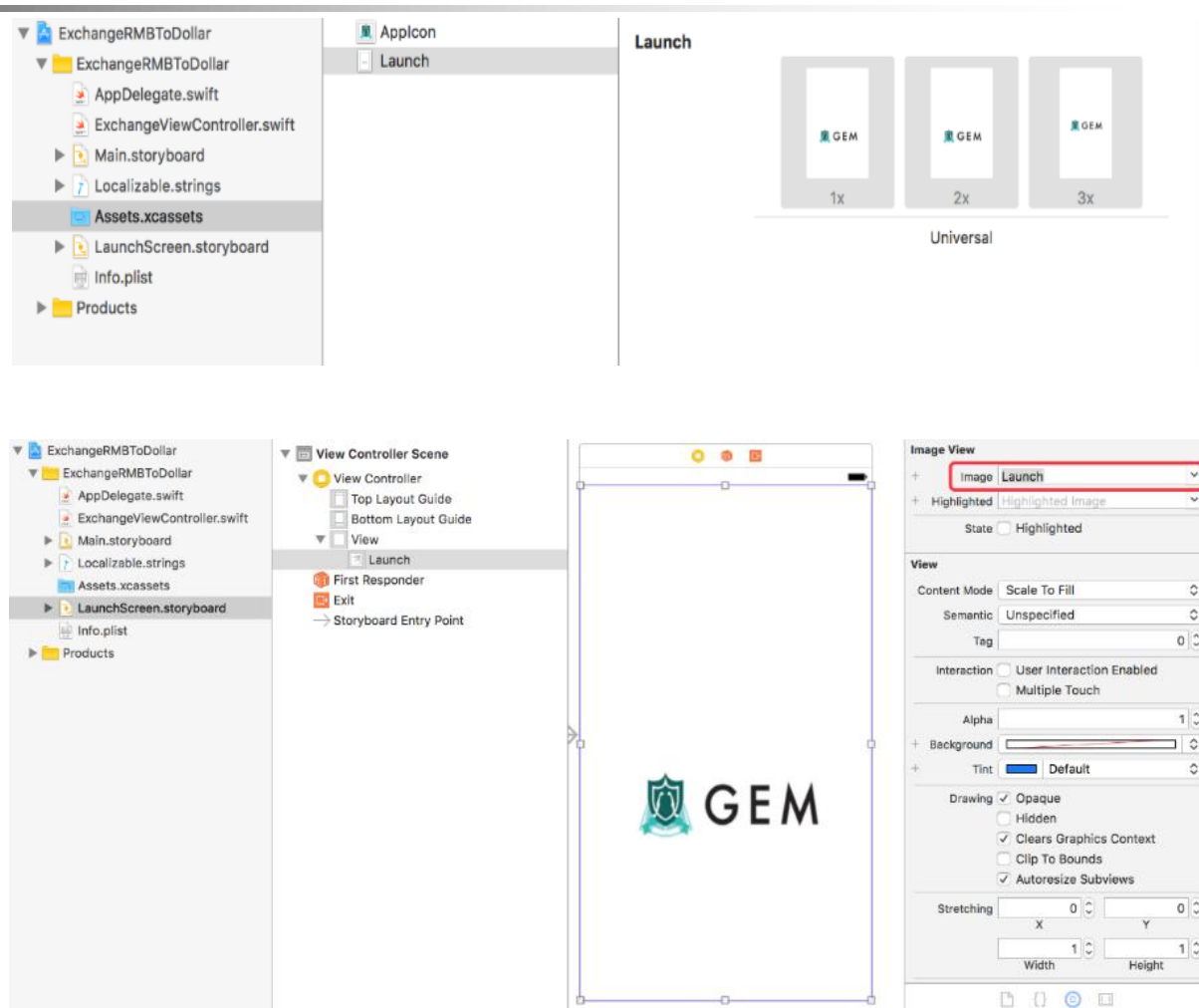
配置启动画面故事板

- 启动画面用到的图片素材是UI设计师根据项目要求提前准备好的。
- 在工程文件浏览窗口中发现，系统在创建的时候已经缺省为我们建立了一个空的启动画面故事板文件 LaunchScreen.storyboard。
- 选中文件 LaunchScreen.storyboard，向视图控制器中拖拽进一个 UIImageView，并调整好尺寸。



设置启动画面的图片

- 打开图片资源文件 Assets.xcassets，在中间的文件浏览框中创建一个新的资源文件“Launch”，将准备好的不同尺寸的启动图片拖入相应的空白框中。
- 将启动画面故事板中 UIImageView 的 Image 属性设置为 Launch。



运行效果



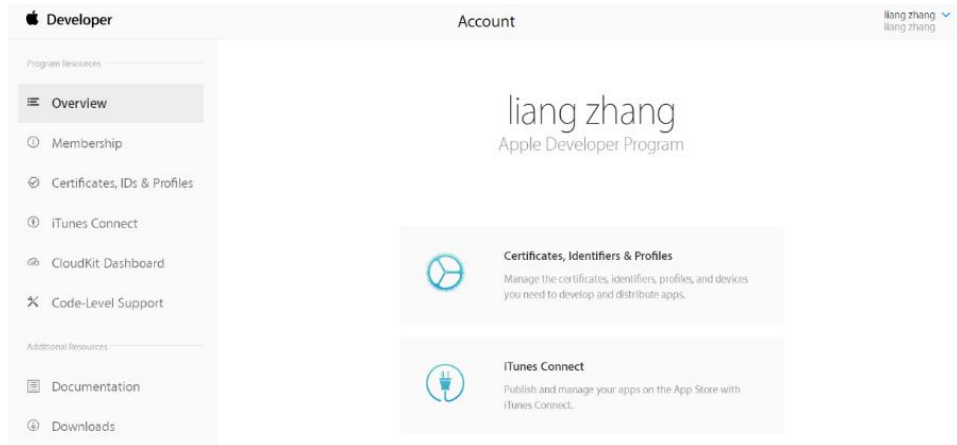
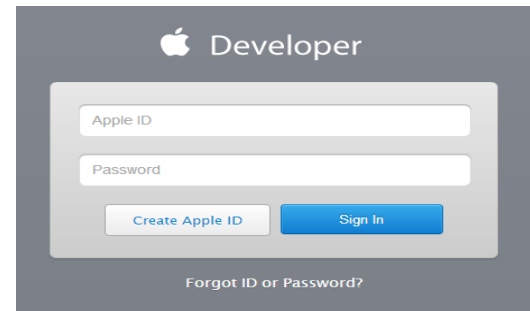


发布应用

- 申请开发者证书
- 创建描述文件
- 设置产品的标识和部署信息
- 应用提交

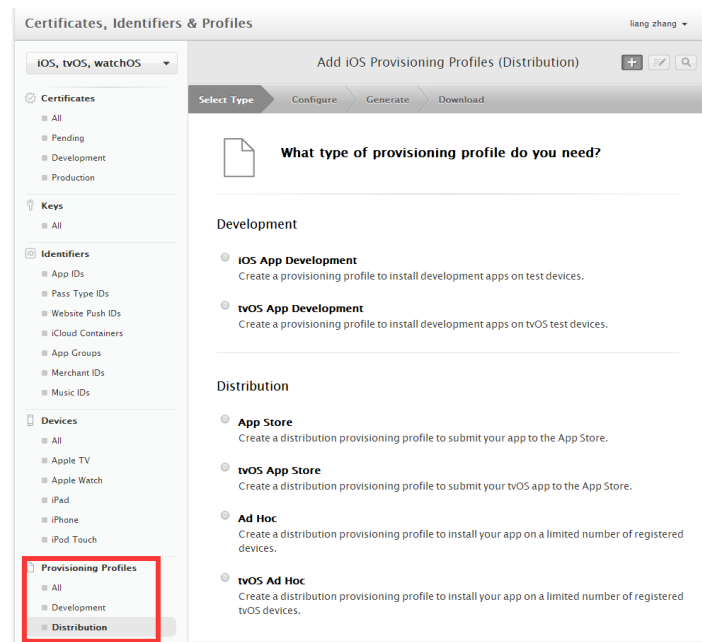
申请开发者证书

- 要将应用发布到App Store，就必须要有开发者证书。开发者证书有两种：个人开发者和企业开发者。申请开发者证书可在网站 <https://developer.apple.com> 上完成。打开该网站，点击“Account”即可进入登录/注册界面。
- 如果已经有了Apple ID可以直接登录，如果还没有的话，需要先注册Apple ID。我们使用已有的Apple ID登录进入。
- 在该界面上可以按照系统的提示，一步步完成开发者证书的申请，并交纳一年的license授权费。苹果审核通过后，会发邮件到注册邮箱，然后就可以使用了。



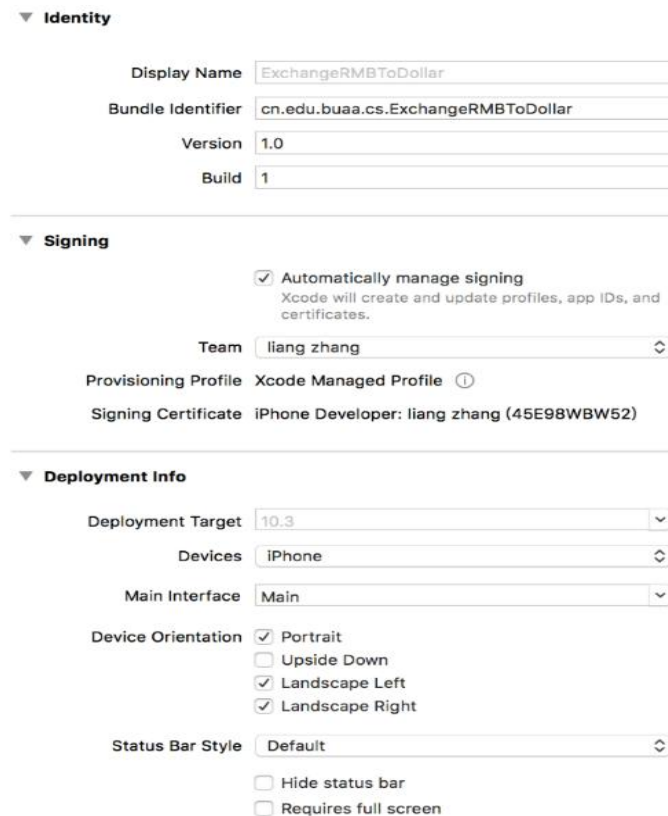
创建描述文件

- 描述文件(Provisioning Profile)是应用在设备上编译时使用的。有两种描述文件，分别为开发描述文件和发布描述文件。
- 在开发者管理中心可以找到描述文件标签。在此页面上，可以为开发和发布分别创建描述文件，具体操作可根据系统提示一步步完成。



设置产品的标识和部署信息

- 有了开发者证书和描述文件后，我们需要在项目的签名信息中指定相应开发者ID，并导入描述文件。
- 需要指定本次产品发布的标识，包括：App显示的名字、包标识符、版本号等。最后，要根据实际的部署目标设备来设置部署信息。完成设置以后，重新进行编译，然后就可以向App Store提交应用了。



The screenshot shows the Xcode project settings interface, divided into three sections: Identity, Signing, and Deployment Info.

Identity

- Display Name: ExchangeRMBToDollar
- Bundle Identifier: cn.edu.buaa.cs.ExchangeRMBToDollar
- Version: 1.0
- Build: 1

Signing

- ☒ Automatically manage signing
Xcode will create and update profiles, app IDs, and certificates.
- Team: liang zhang
- Provisioning Profile: Xcode Managed Profile ⓘ
- Signing Certificate: iPhone Developer: liang zhang (45E98WBW52)

Deployment Info

- Deployment Target: 10.3
- Devices: iPhone
- Main Interface: Main
- Device Orientation:
 - ☒ Portrait
 - ☐ Upside Down
 - ☒ Landscape Left
 - ☒ Landscape Right
- Status Bar Style: Default
 - ☐ Hide status bar
 - ☐ Requires full screen

应用提交

- 应用的提交需要使用Apple ID在网站 <https://itunesconnect.apple.com>中完成。
- 点击“我的App”，按照系统提示一步步完成操作即可完成提交。提交后，要等待苹果审核，如果审核不通过，会反馈意见，如果通过，就可以在App Store上线了。





谢 谢
