实验 3

创建并推送内部/外部消息

实验结构

- 添加内部和外部或推送通知

实验目标

本实验结束后，你将能够：

- 在 Xcode 中创建本地通知
- 创建外部或推送通知

导论

iOS 应用中实现的本地通知能通知没在前台运行的应用，某一事件发生了，或是有新信息可用。例如，邮件应用在后台运行时你接收到了一封新邮件，本地通知就会创建，让你知道收件箱中有新邮件。

实验：添加内部和外部或推送通知

本地通知被 iOS 用于在相同设备上发起和传递。你可以按照日期和时间安排好本地通知。这些通知可以基于事件即时创建。你还可以添加声音、编号和标题给这些通知。

背景

你希望在应用中添加一个**内部通知**，在 iPhone 通知界面通知一条信息。通知界面也就是 iPhone 顶部手指往下滑动时出现的界面。

通知在应用开发中可以有很多有用的用途。例如，在应用 **Bands** 中，我们使用通知来跟踪和保存最喜爱的乐队名称，应用可以通过通知界面告诉用户最少播放的歌曲。此后，用户可以听、保存或是删除这首歌。

外部通知能够发送链接，用于下载软件、歌曲、电影、游戏。你可以在应用中实现这种链接，例如在玩游戏时，你可以邀请朋友并同时选择对手。

本实验中，你需要执行下面这些任务：

1. 添加本地通知
2. 添加推送或外部通知

实验准备

要执行这些任务，你需要有：

- 带有 Xcode 的 Mac
- 苹果开发者帐户

实验推荐解决方案

任务 1 解决方案：添加本地通知

要学习本地通知的用法，你可以创建一定时间间隔后发送通知的新应用。

1. 打开 Xcode 6.1.1 并选择 **Create a new Xcode project** 选项，如图 1 所示：

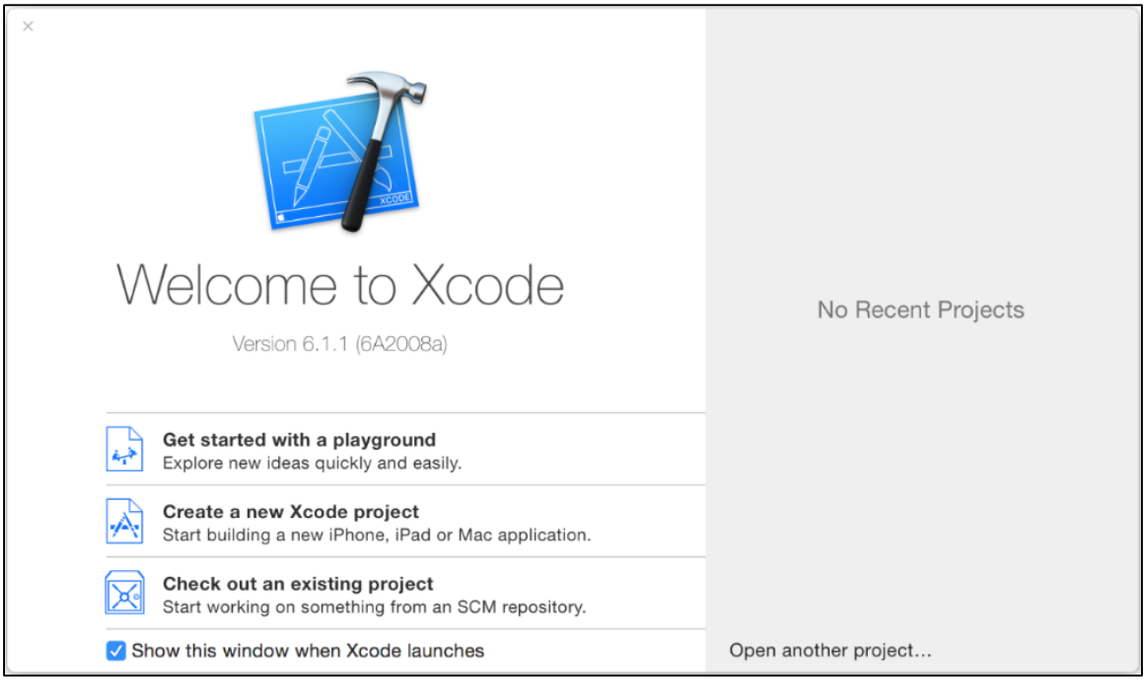

图 1: 选择创建新 Xcode 项目

2. 将项目命名为 **Notification**。可以选择 Objective-C 或 Swift 作为编程语言。本项目中，选择 **Objective-C**，如图 2 所示。点 **Next** 继续。

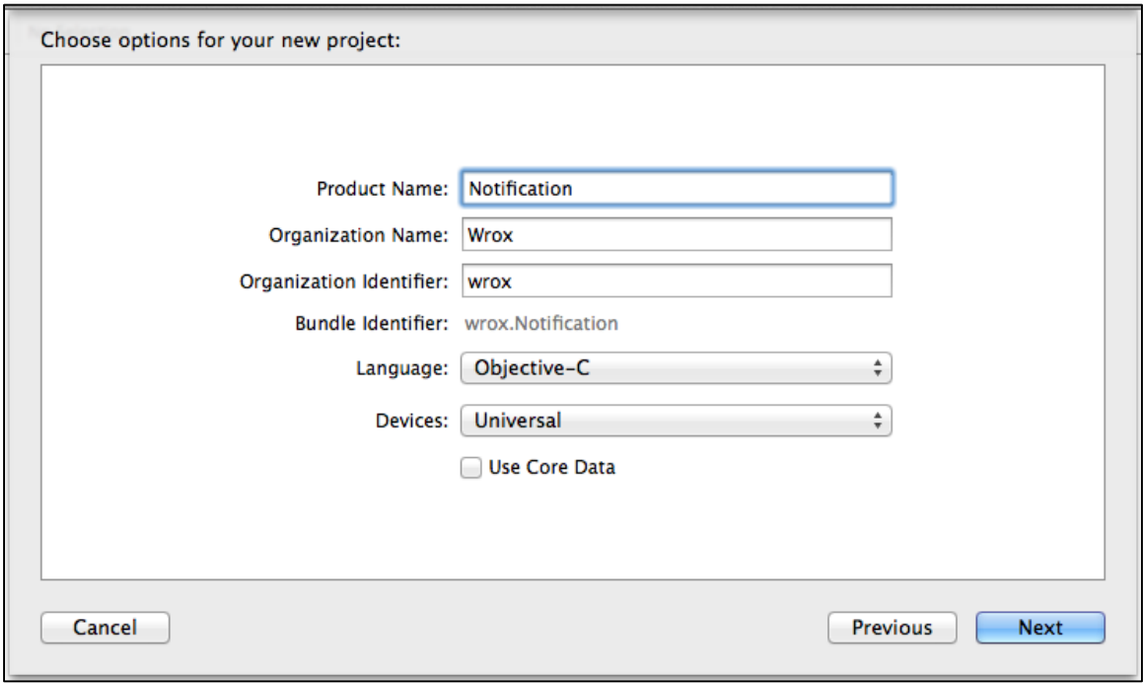

图 2: 为 Notification 项目确定各种选项

3. **Notification** 项目的 Objective-C 文件会显示出来，如图 3 所示：

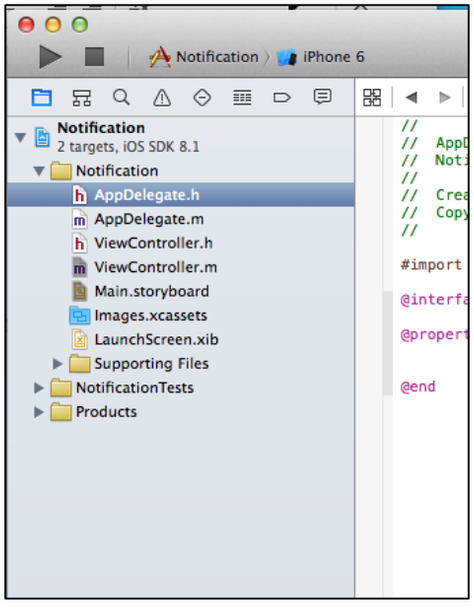

图 3: Notification 的 Objective-C 项目文件

4. 打开 `Main.storyboard` 文件。点实用工具面板的库选择器栏，为 Notification 应用设计 UI，如图 4 所示：

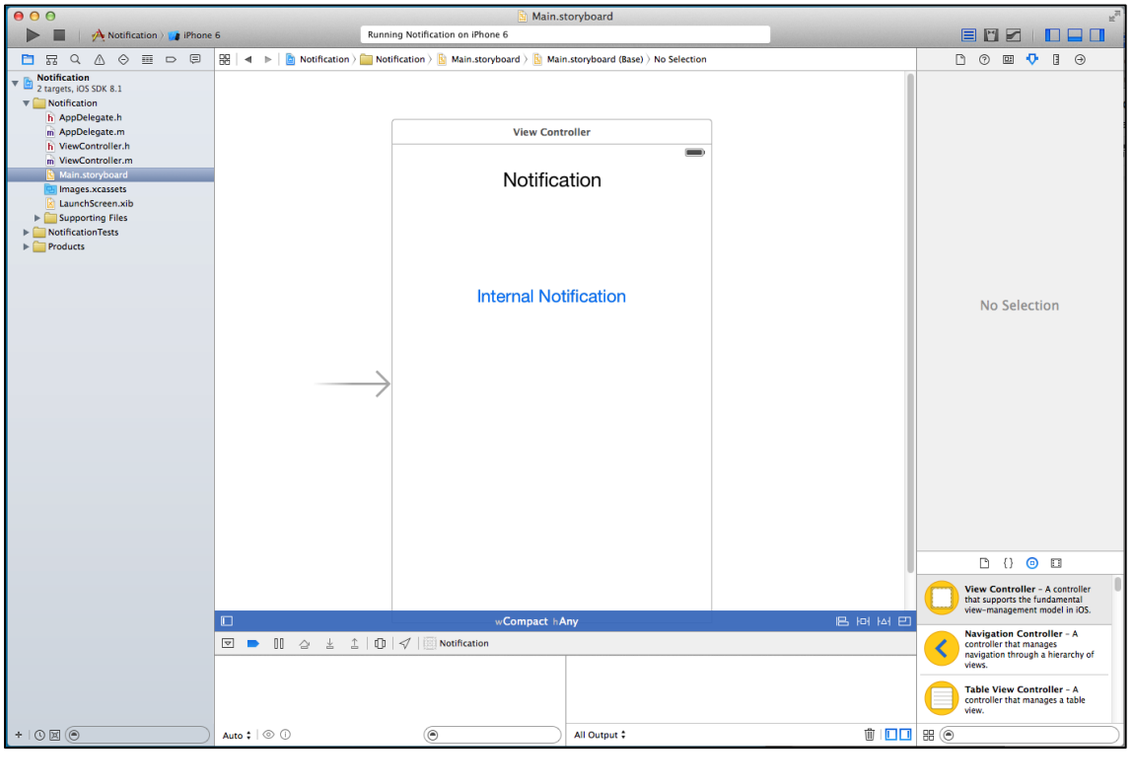

图 4: 在 `Main.storyboard` 文件中为 Notification 应用设计 UI

5. 点辅助编辑器来分屏。然后右键点击按住 **Internal Notification** 标签并拖放到 `ViewController.m`。在打开的弹窗中填入图 5 中所示的内容：

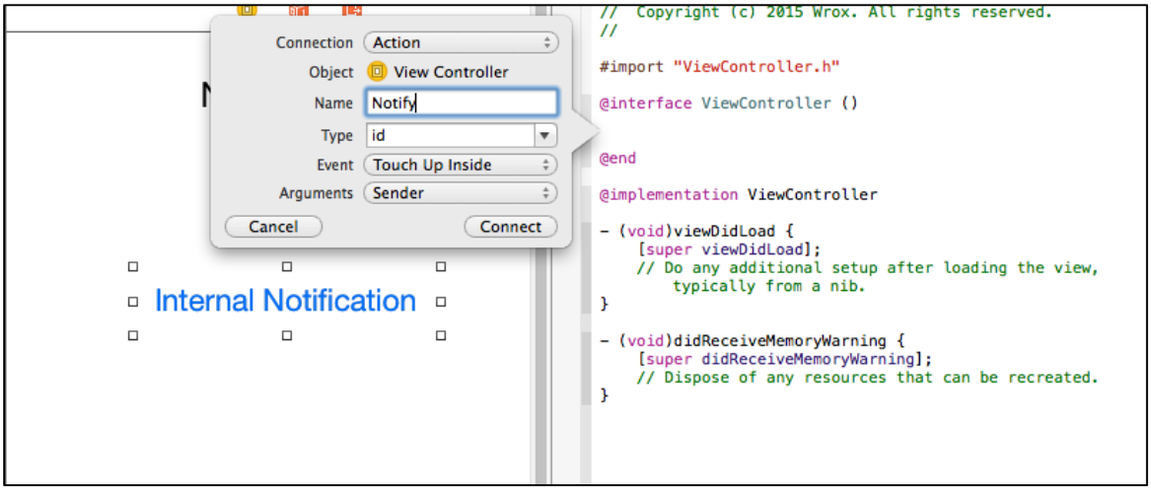

图 5: 在 ViewController.m 中为内部通知添加动作按钮

6. 在 ViewController.m 中，添加本地通知的逻辑，如下所示。在 Notification.alertbody 中添加通知正文。这里，你将它作为 Local Notification 添加。

```
(IBAction)Notify:(id)sender{
UILocalNotification *Notification = [[UILocalNotificationalloc] init];
Notification.fireDate = [NSDate dateWithTimeIntervalSinceNow:10];
Notification.alertBody = @“Local Notification”;
Notification.timeZone = [NSTimeZonedefaultTimeZone];
[[UIApplicationsharedApplication] scheduleLocalNotification:Notification];
}
@end
```

7. 在项目导航器中选择 AppDelegate.m 文件，并为本地通知添加如下代码。

```
- (BOOL)application:(UIApplication *)application
didFinishLaunchingWithOptions:(NSDictionary *)launchOptions {
       // Override point for customization after application launch.

   UILocalNotification *Notification =
[launchOptionsobjectForKey:UIApplicationLaunchOptionsLocalNotificationKey];

   if (Notification) {
         [applicationcancelAllLocalNotifications];
       }
   return YES;
   }
```

8. 上述代码在 iOS 7 中编译后能运行，但在 iOS 8 中不会编译。在 iOS 8 中，应用需要请求用户允许才能显示本地通知。要请求用户许可，需要在 AppDelegate.m 文件中添加如下代码。

```
   [applicationregisterUserNotificationSettings:[UIUserNotificationSettings
settingsForTypes:UIUserNotificationTypeAlert|UIUserNotificationTypeBadge|UIUserNotificationTyp
eSound categories:nil]];
```

9. 通过选择 **Project -> Run** 命令来运行模拟器。你会得到如图 6 所示的输出结果：

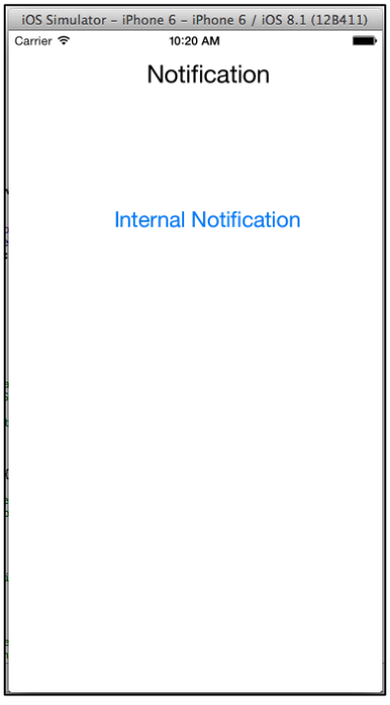

图 6: 运行应用

10. 点应用中的 **Internal Notification** 按钮后（参见图 6），你的应用会请求发送通知的许可，如图 7 所示。允许通知需要点击 **OK** 按钮。

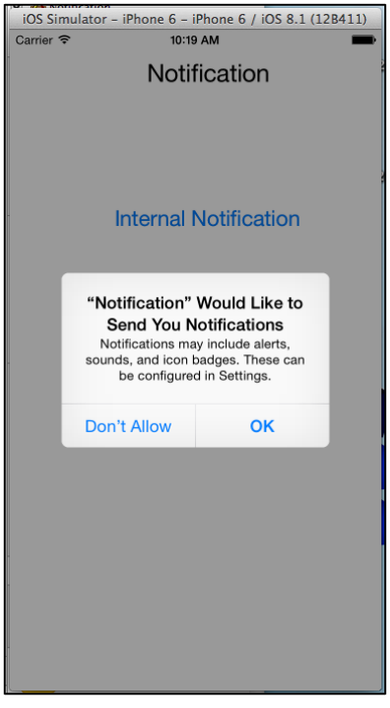

图 7: 应用请求发送通知的许可

11. 添加了请求用户许可的代码后，它会在 **Settings** 中生成 **Notifications** 选项，如图 8 所示。在 **Notifications** 选项中，你可以进行各种设置，包括声音、应用图标，如图 9 所示：

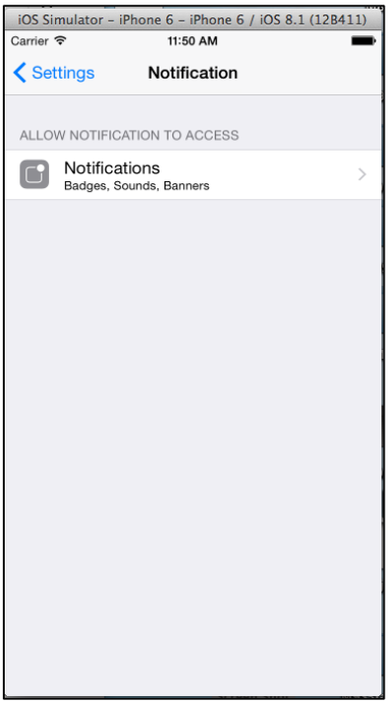

图 8: 设置中的通知选项

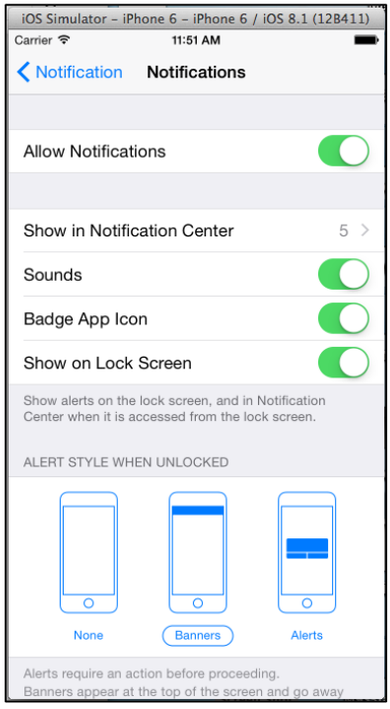

图 9: 设置通知

12. 改变 **Notification** 设置后，使用<shift>+Command+H 选项到主屏等待通知。你会得到如图 10 和 11 所示的通知：

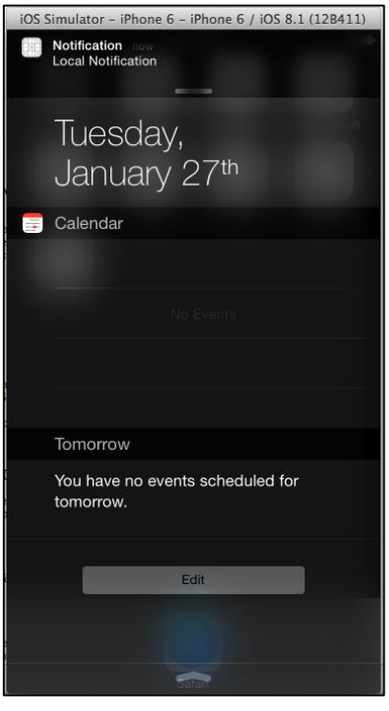

图 10: 作为横幅显示的通知

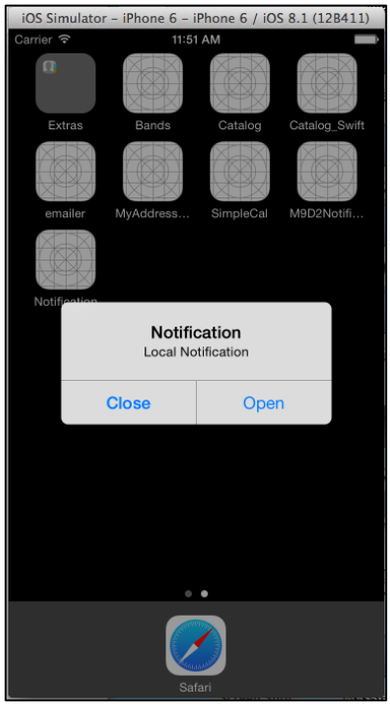

图 11: 作为警告显示的通知

源码

```
AppDelegate.m
#import "AppDelegate.h"

@interface AppDelegate ()

@end

@implementation AppDelegate

- (BOOL)application:(UIApplication *)application
didFinishLaunchingWithOptions:(NSDictionary *)launchOptions {
    // Override point for customization after application launch.

    [applicationregisterUserNotificationSettings:[UIUserNotificationSettings
settingsForTypes:UIUserNotificationTypeAlert|UIUserNotificationTypeBadge|UIUserNotificationTyp
eSound categories:nil]];
UILocalNotification *Notification =
[launchOptionsobjectForKey:UIApplicationLaunchOptionsLocalNotificationKey];

if (Notification) {
        [applicationcancelAllLocalNotifications];
    }
return YES;

}
```

任务 2 解决方案：添加推送通知

下面来学习执行推送通知的步骤：

1. 在 Mac 的 **Applications** 文件夹下打开 **Utilities** 文件夹，在 **Utilities** 文件夹中用搜索栏搜索 **Keychain Access**，如图 12 所示：

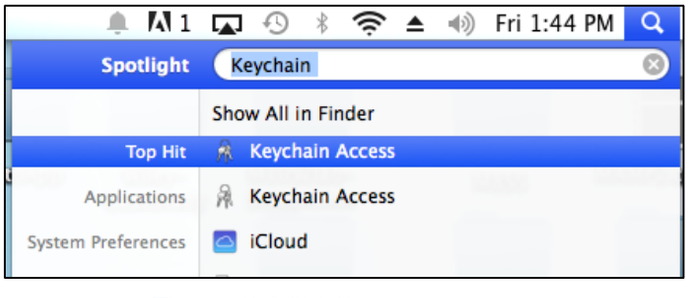

图 12: 在搜索栏中搜索 Keychain Access

2. 要请求证书，选择 **Keychain Access -> Certificate Assistant -> Request a Certificate From a Certificate Authority** 选项，如图 13 所示：

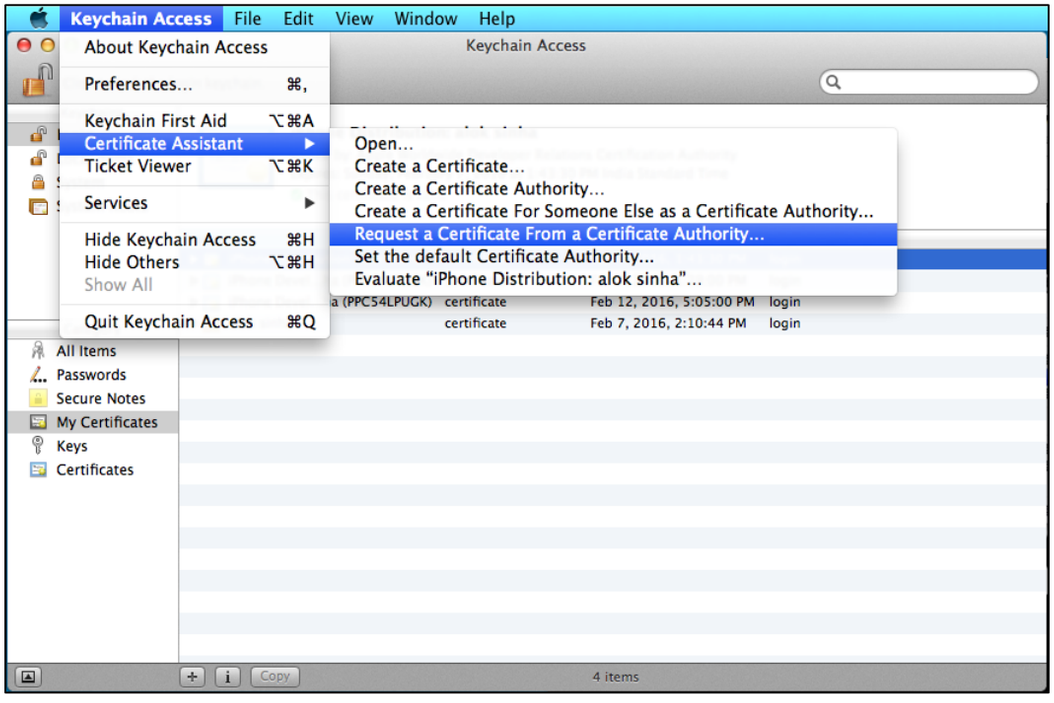

图 13: 从 Certificate Authority 请求证书

3. **Certificate Assistant** 向导会打开，如图 14 所示：

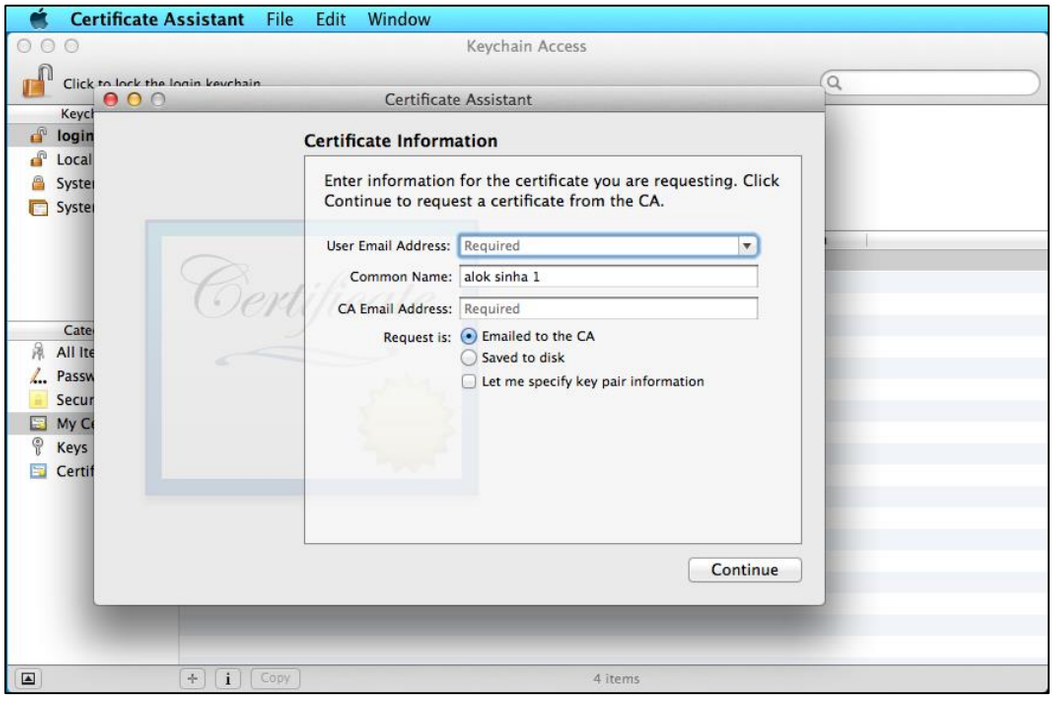

图 14: Certificate Assistant 向导

4. 提供用于从 certificate authority 创建证书的凭证，如图 15 所示。在 **User Email Address** 字段中输入你的电子邮箱地址，在 **Common Name** 字段中输入一个名称用于你的私钥。选择 **Saved to disk** 选项，点 **Continue**。

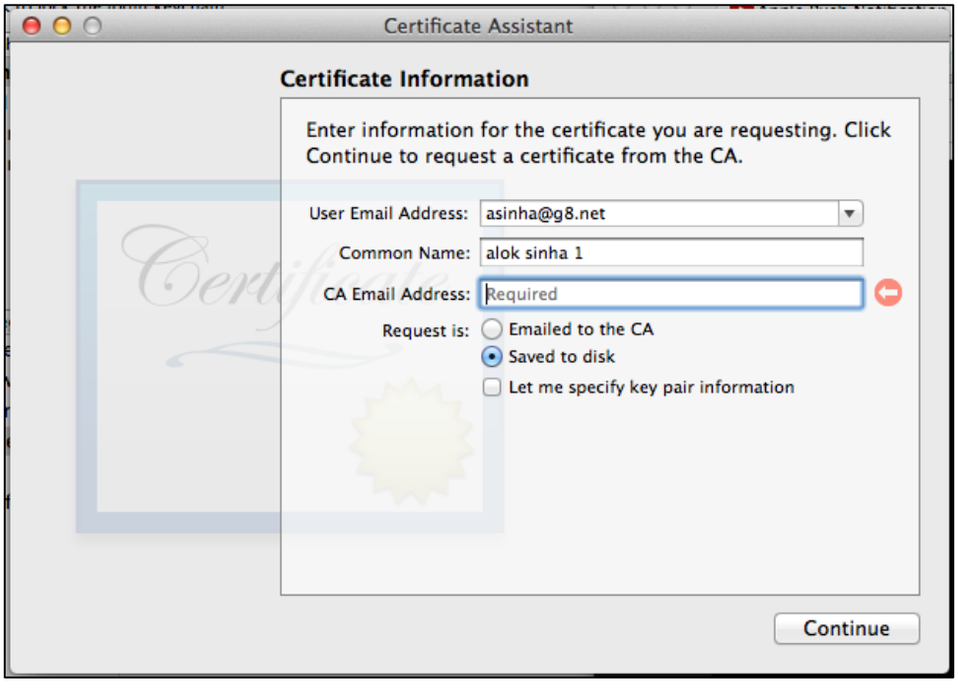

图 15: 为证书插入凭证

6. 在打开的对话框中（参见图 16），输入保存 CSR 文件的名称和位置，然后点 **Save**。

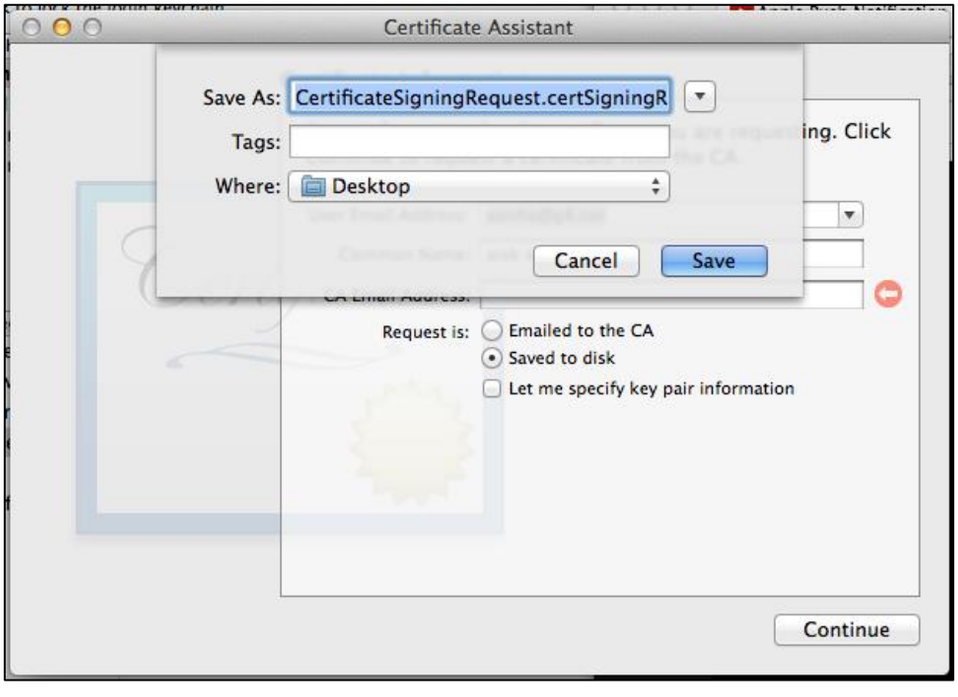

图 16: 将 CSR 保存到桌面

7. 证书请求创建并保存到了桌面，如图 17 所示。点 **Done**。

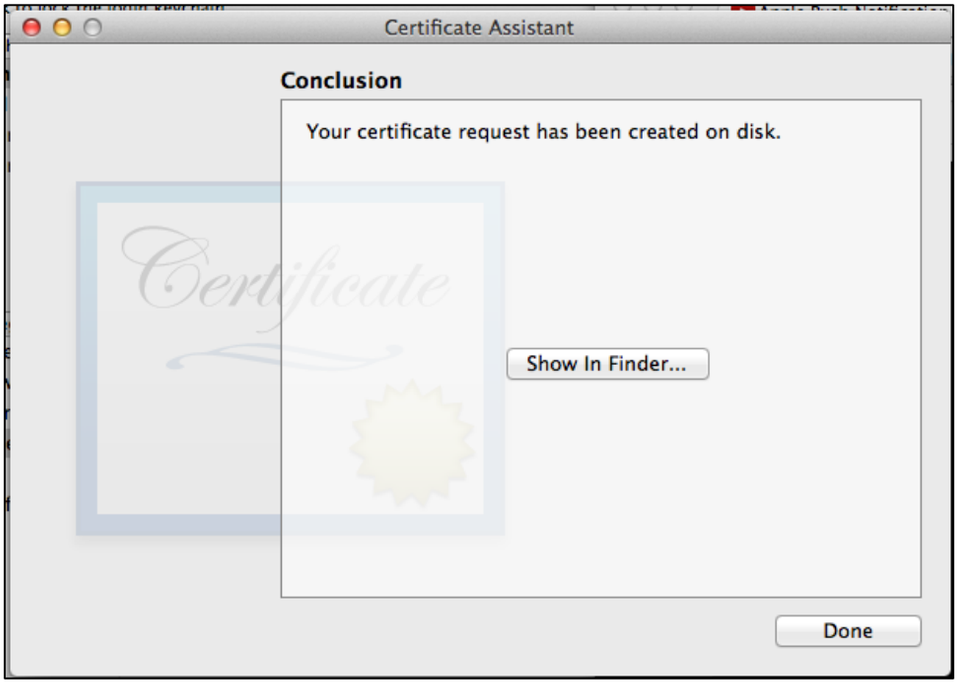

图 17: 证书请求已创建

8. 到链接 *https://developer.apple.com/devcenter/ios/index.action* 并用苹果 ID 登陆。图 18 是打开的 iOSDevCenter 页面。

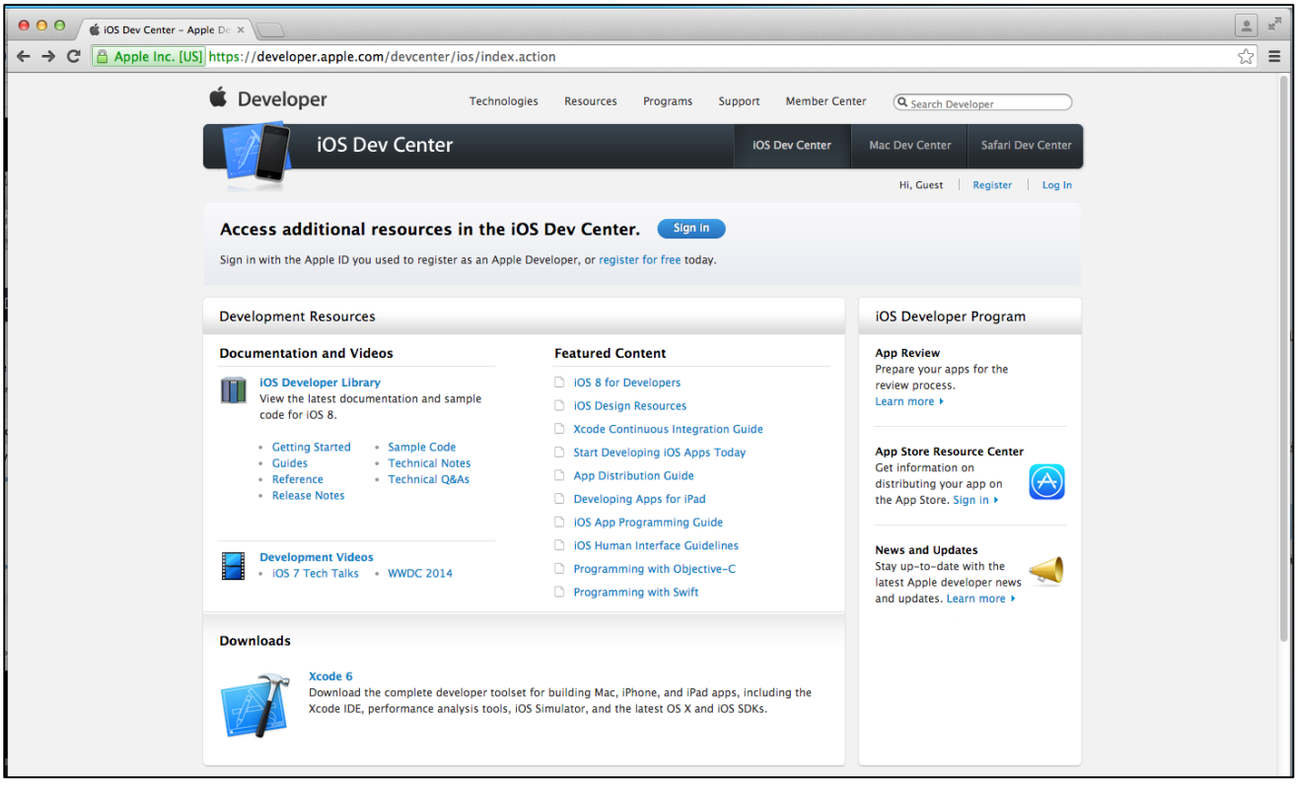

图 18: 登陆到 iOSDevCenter

9. 登陆到 DevCenter 之后，点 **Certificates, Identifiers & Profiles** 选项，如图 19 所示：

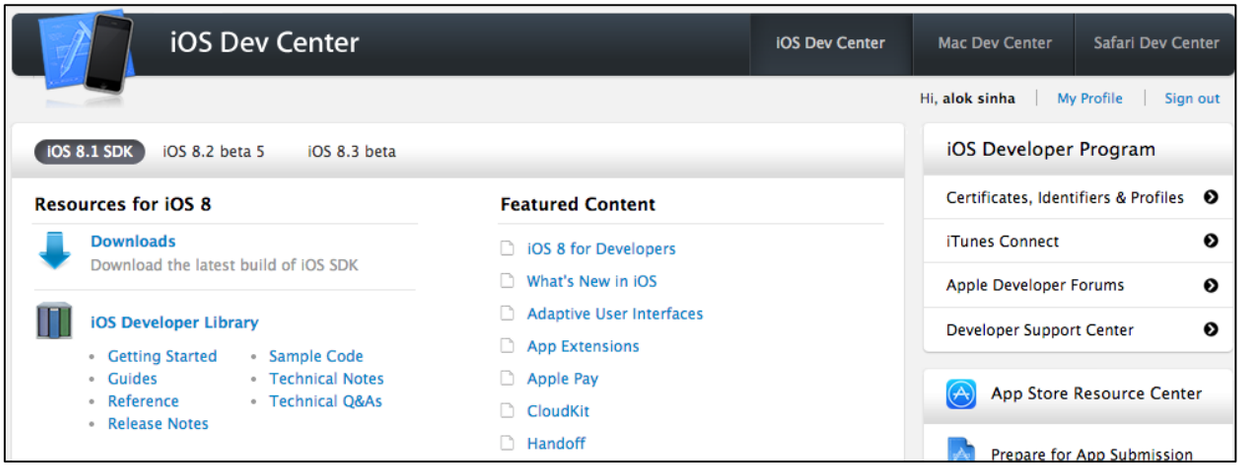

图 19: 点 Certificates, Identifiers & Profiles 选项

10. 在打开的页面上（参见图 20），选择 **Provisioning Profiles**。

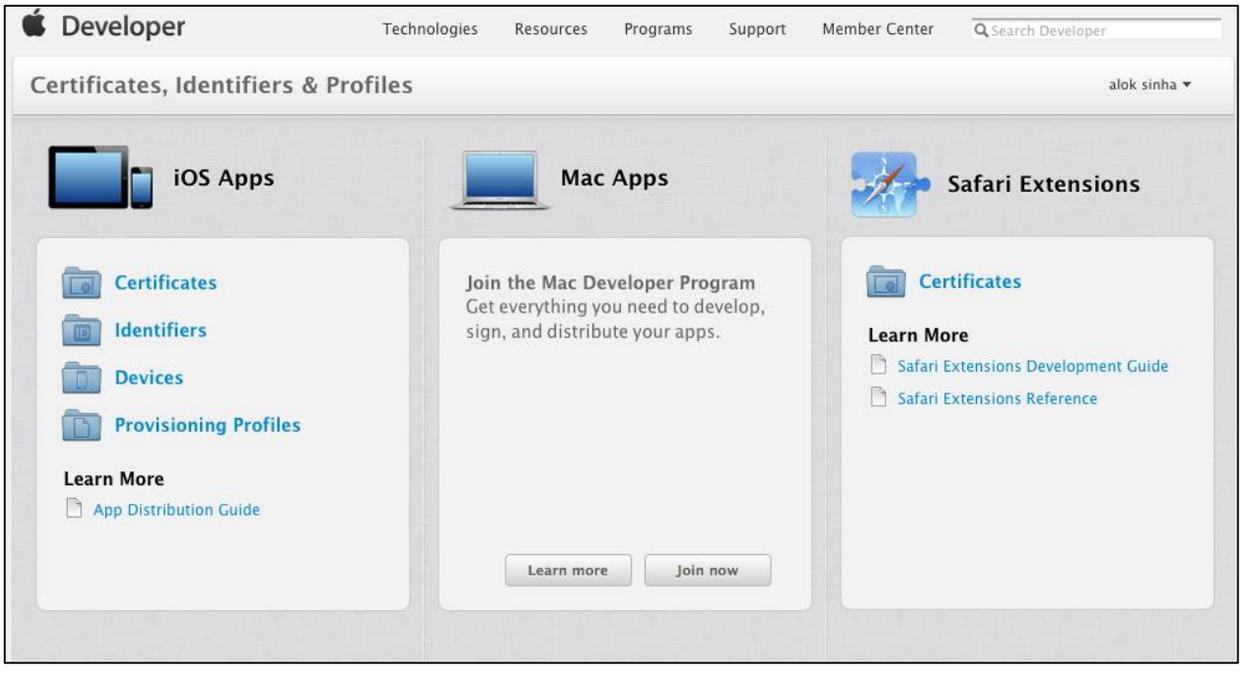

图 20: 选择 Provisioning Profiles 选项

11. 要浏览所有 iOS provisioning profiles，选择 **Provisioning Profiles** 下的 **All** 选项，如图 21 所示：

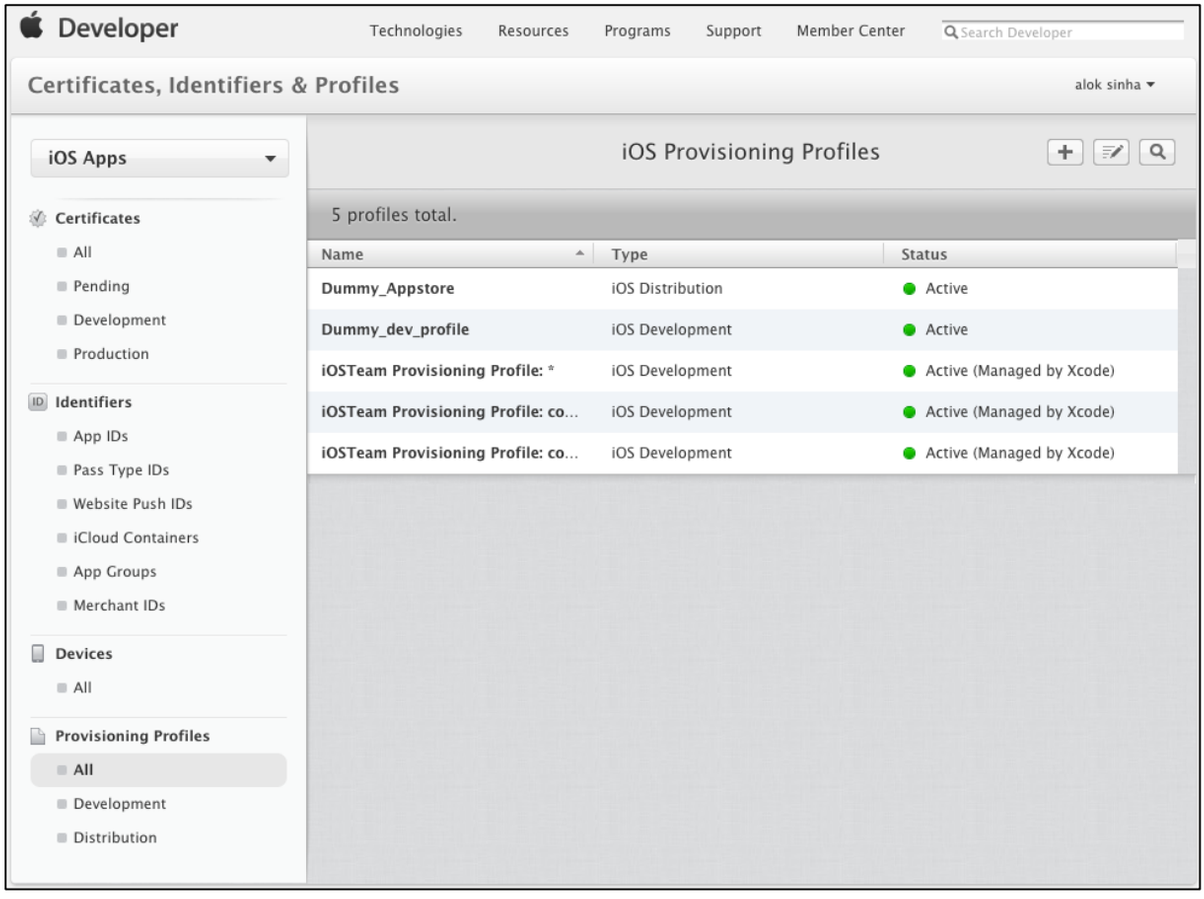

图 21: 浏览所有 Provisioning Profiles

12. 选择 **Identifiers** 部分的 **App IDs** 选项，如图 22 所示。这会显示创建的总 App ID。

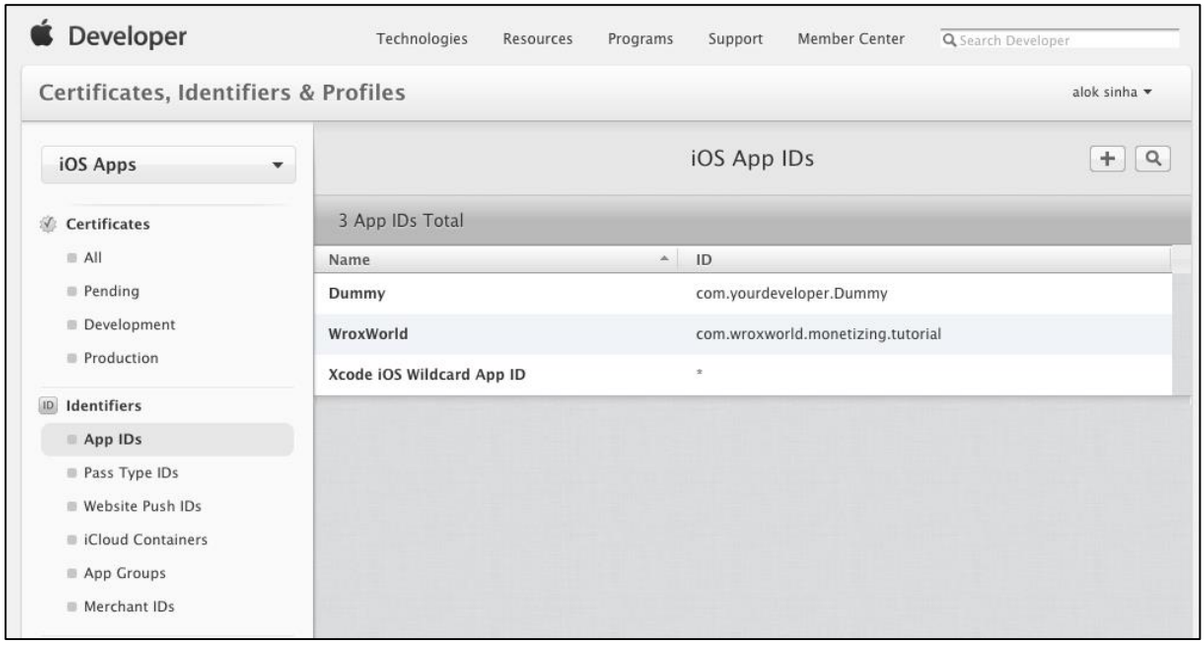

图 22: 选择 Identifiers 下的 App IDs

13. 选择要用的 App ID，例如 **WroxWorld**。这会显示所选 App ID 的应用服务，如图 23 所示：

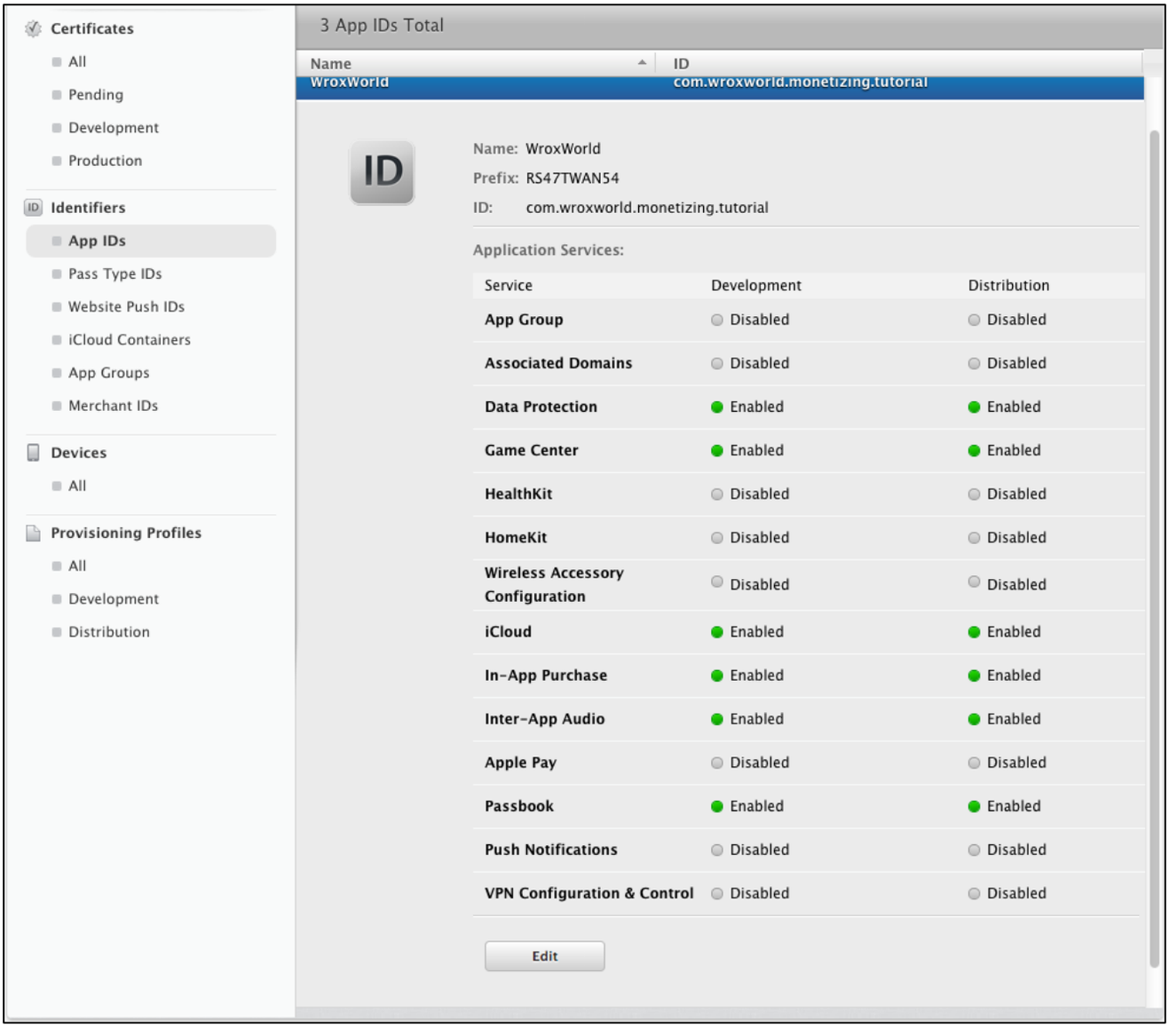

图 23: 显示所选 App ID 的应用服务

14. 默认情况下，推送通知服务是禁用的。要启用它，需要点 **Edit** 按钮（如图 23 所示），这会打开图 24 所示的页面，你要勾选 **Push Notifications** 复选框。

图 24: 点 Edit 并勾选 Push Notifications 复选框

15. 点 **App IDs** 选项并验证属性。**Push Notifications** 现在可以配置了，如图 25 所示。点 **Edit** 来进行配置。

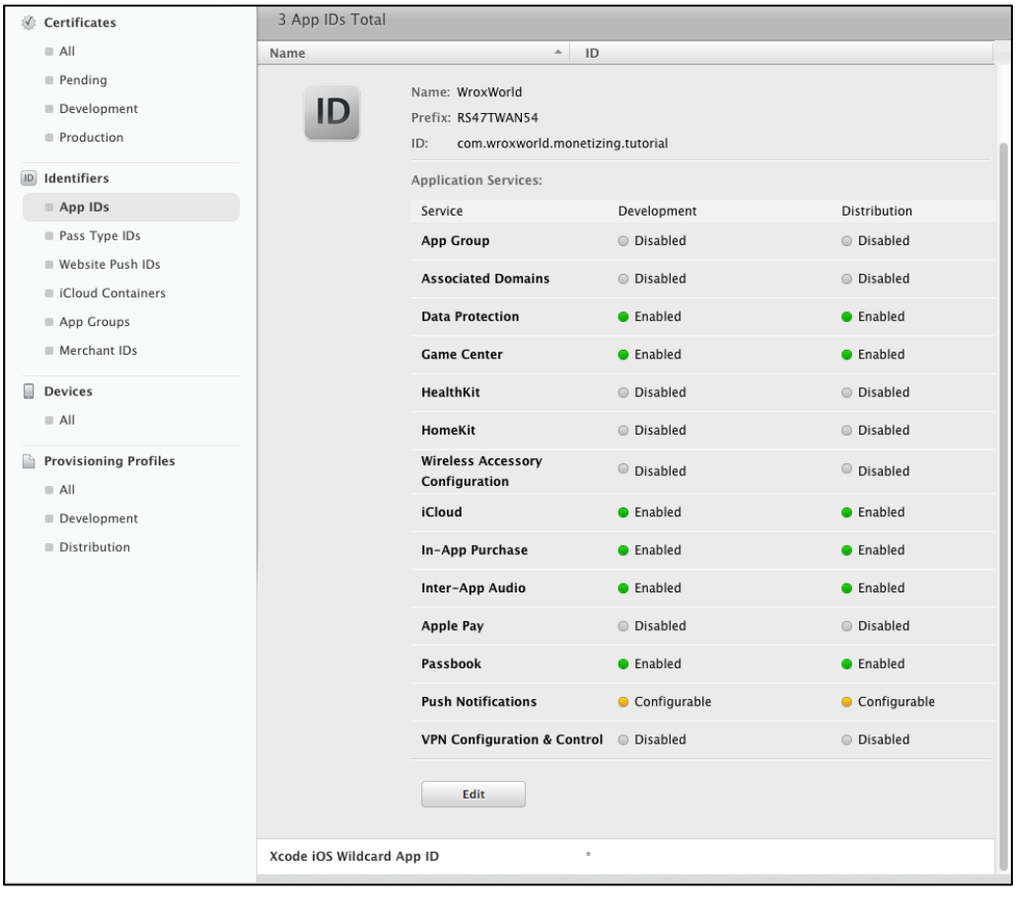

图 25: 检查 App IDs 属性

16. 在 **Development SSL Certificate** 下点 **Create Certificate** 按钮，如图 26 所示：

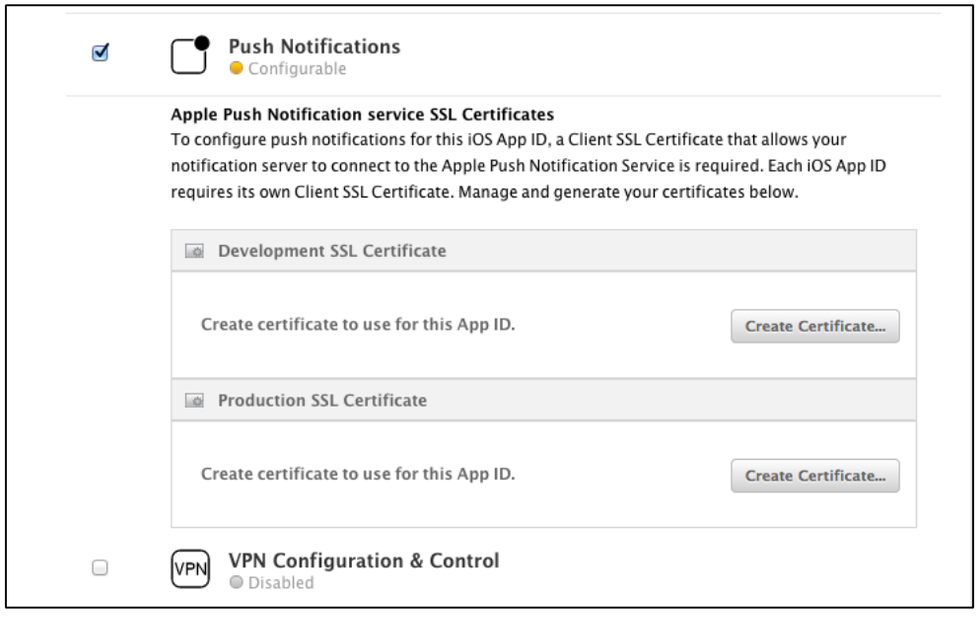

图 26: 创建 Development SSL Certificate

17. 在开启的 **Add iOS Certificate** 向导中（参见图 27），阅读如下说明然后点 **Continue**。

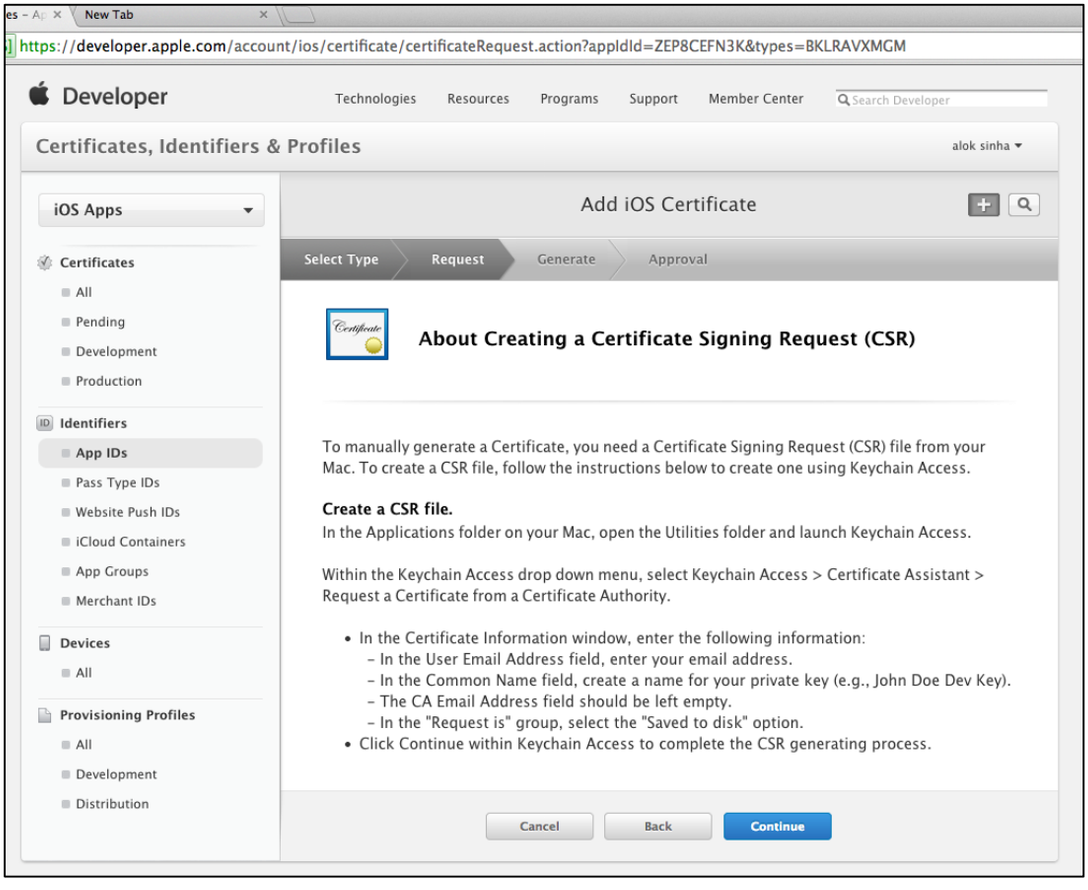

图 27: 开始添加 iOS 证书

18. 在下一个界面（参见图 28），点 **Choose File**，添加之前生成的 CSR 文件。

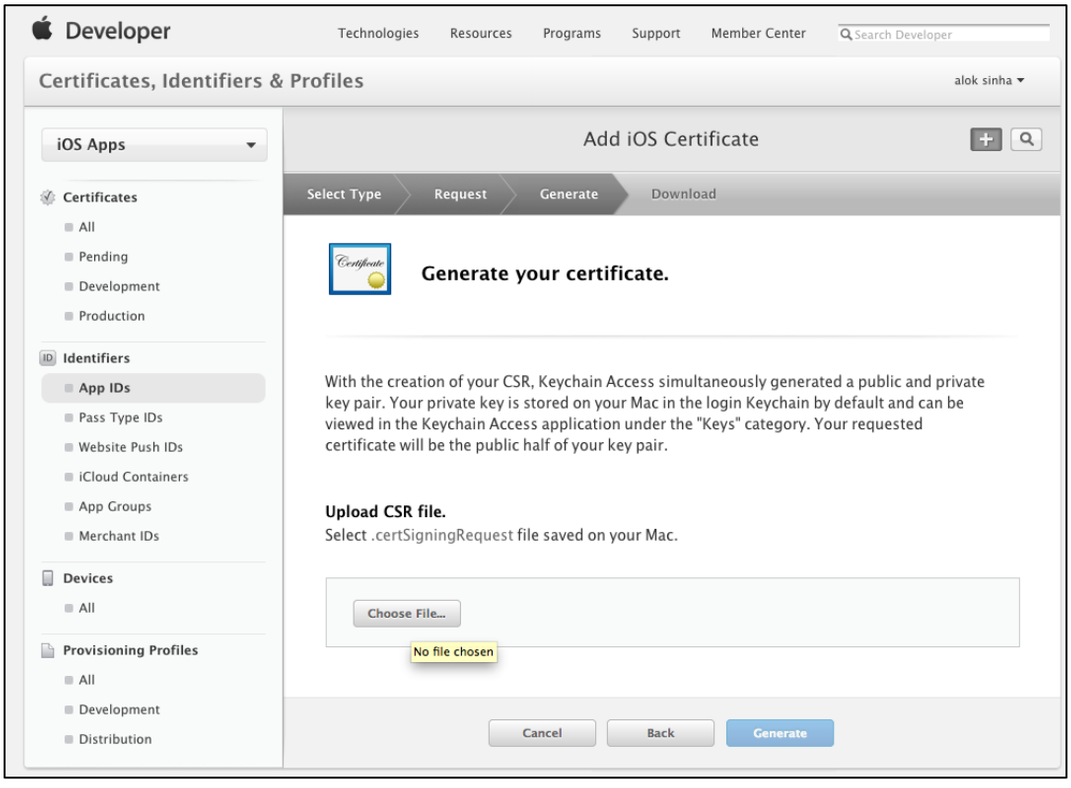

图 28: 点 Choose File

19. 从桌面选择之前生成的 CSR 文件，并点 **Open**，如图 29 所示：

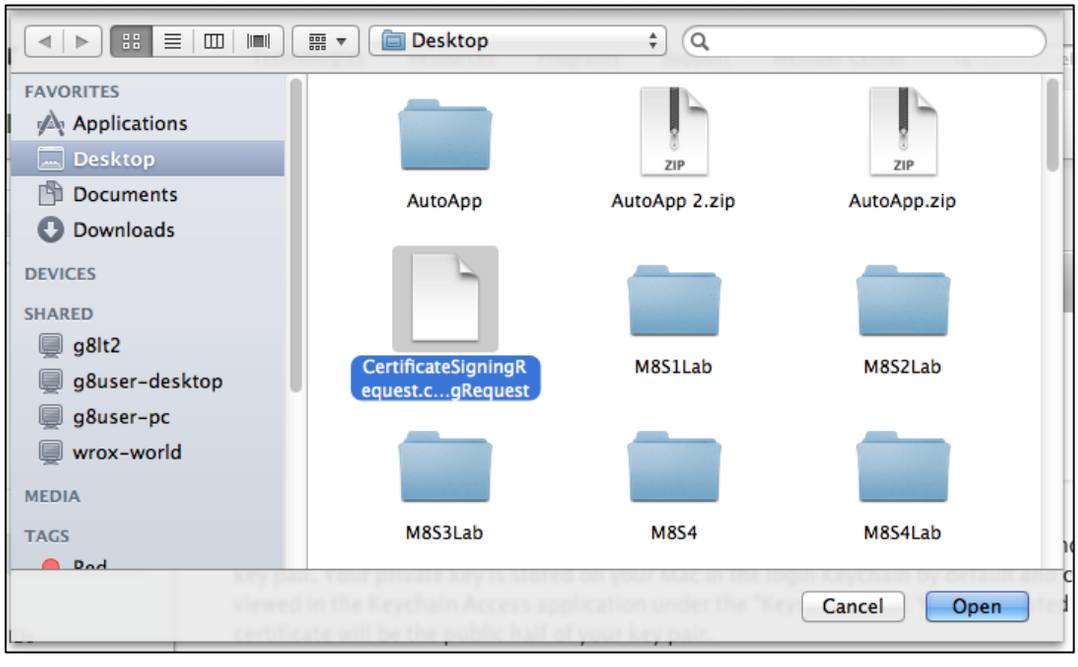

图 29: 选择之前生成的 CSR 文件

20. 添加 CSR 文件后，继续到下一个界面。你的证书这就准备好了，如图 30 所示。点 **Download**。

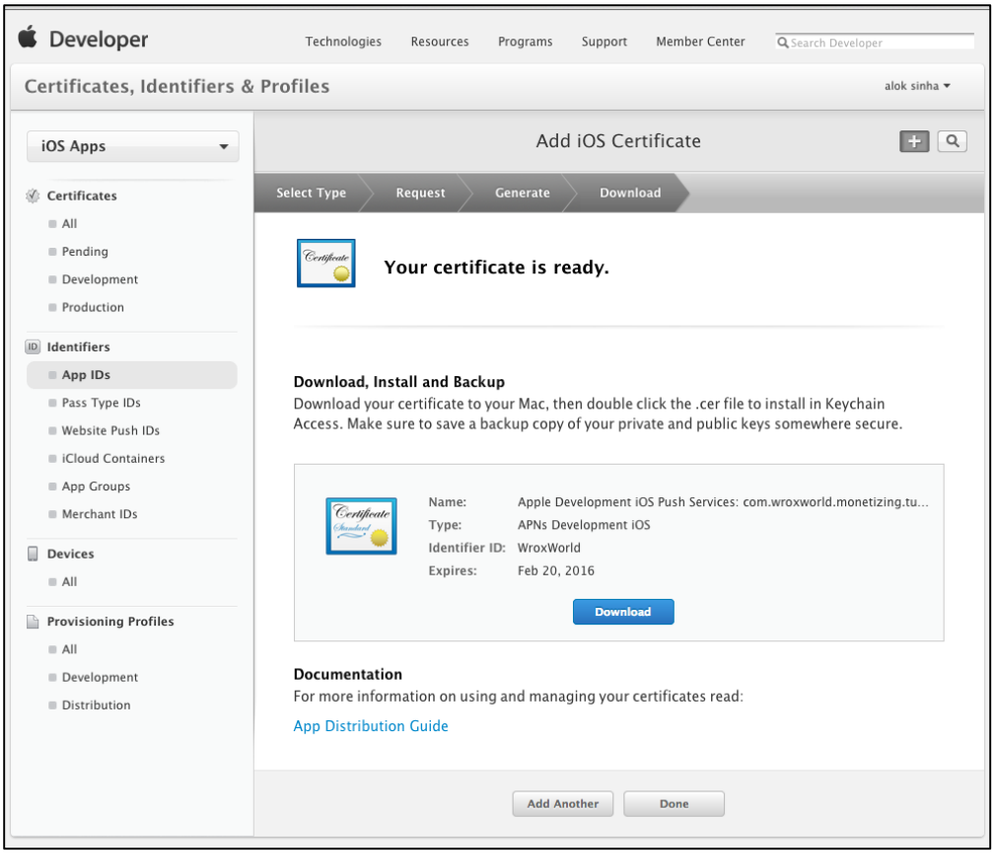

图 30: 下载证书

21. 打开 **Keychain Access** 文件夹并选择证书，如图 31 所示：

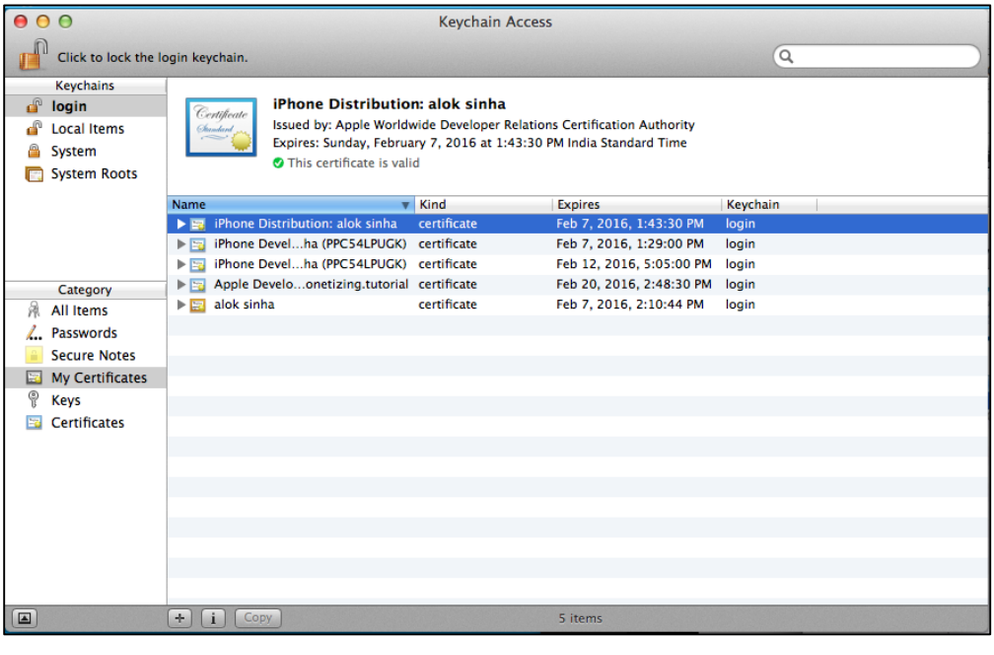

图 31: 在 Keychain Access 中选择证书

22. 右键点击证书并选择 **Export "Apple Development IOS Push Services:xxxx"**，从 **Keychain Access** 文件夹导出证书，如图 32 所示：

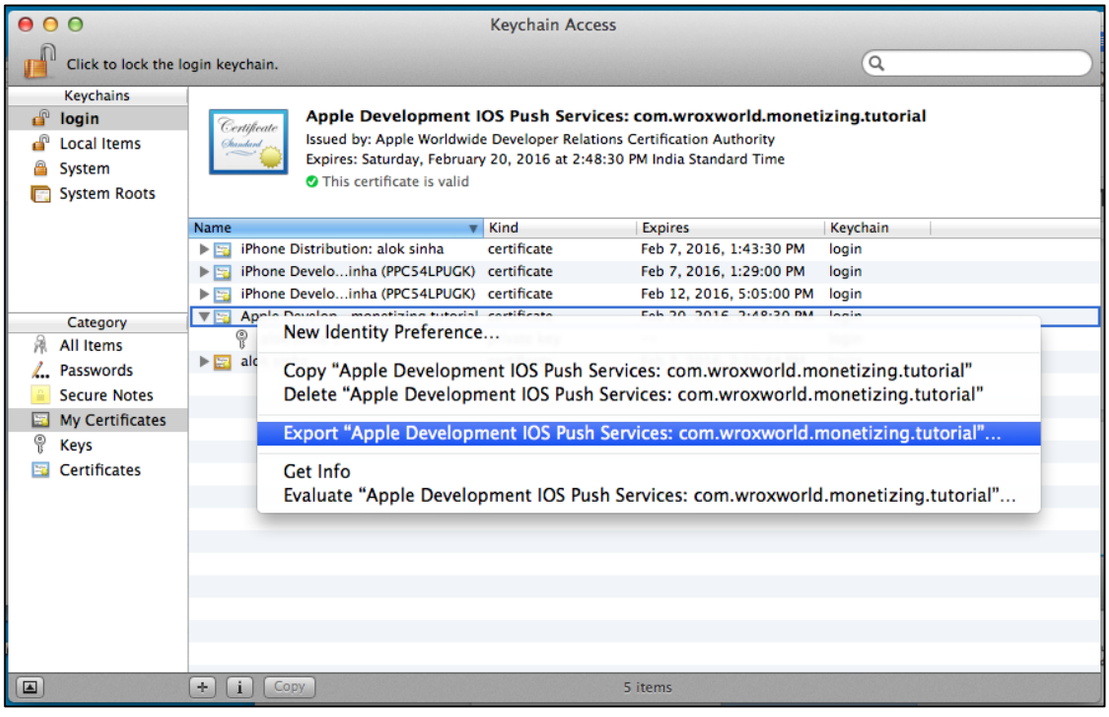

图 32: Export the Certificate from the Keychain Access

23. 在打开的对话框中，命名并保存导出的推送证书到你想要的位置，如图 33 所示。确保文件格式是 **Personal Information Exchange (.p12)**。

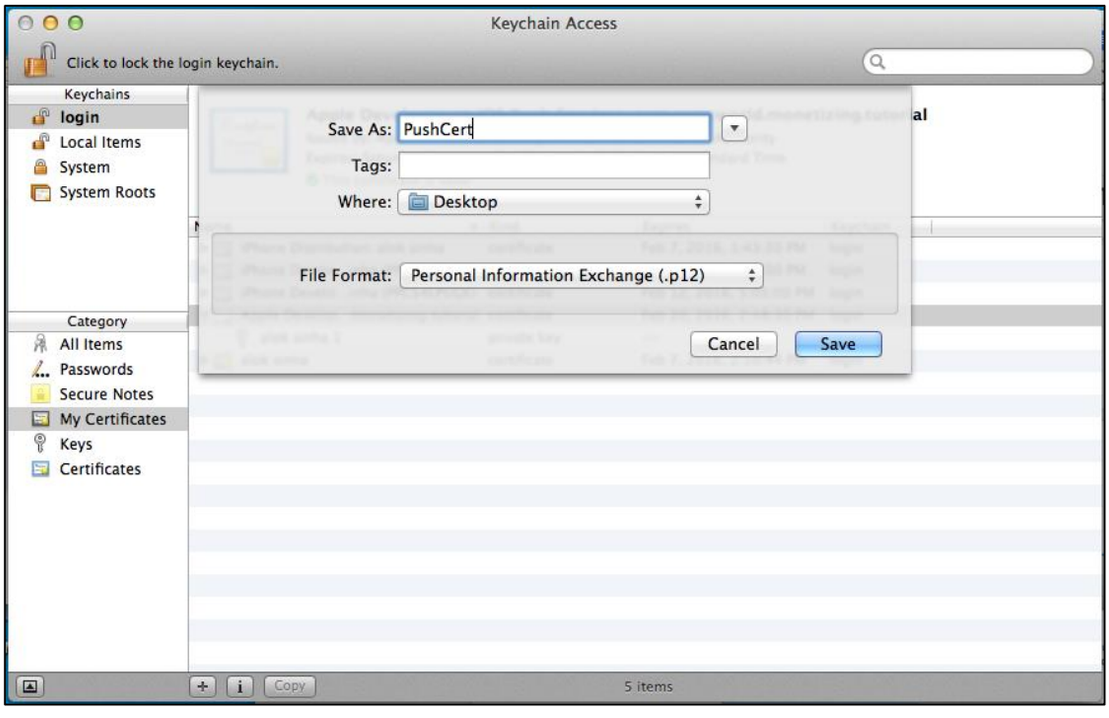

图 33: 保存导出的推送证书

24. 要从 **Keychain Access** 文件夹导出密钥，需要选择密钥并从右键菜单点 **Export "key name"**，如图 34 所示：

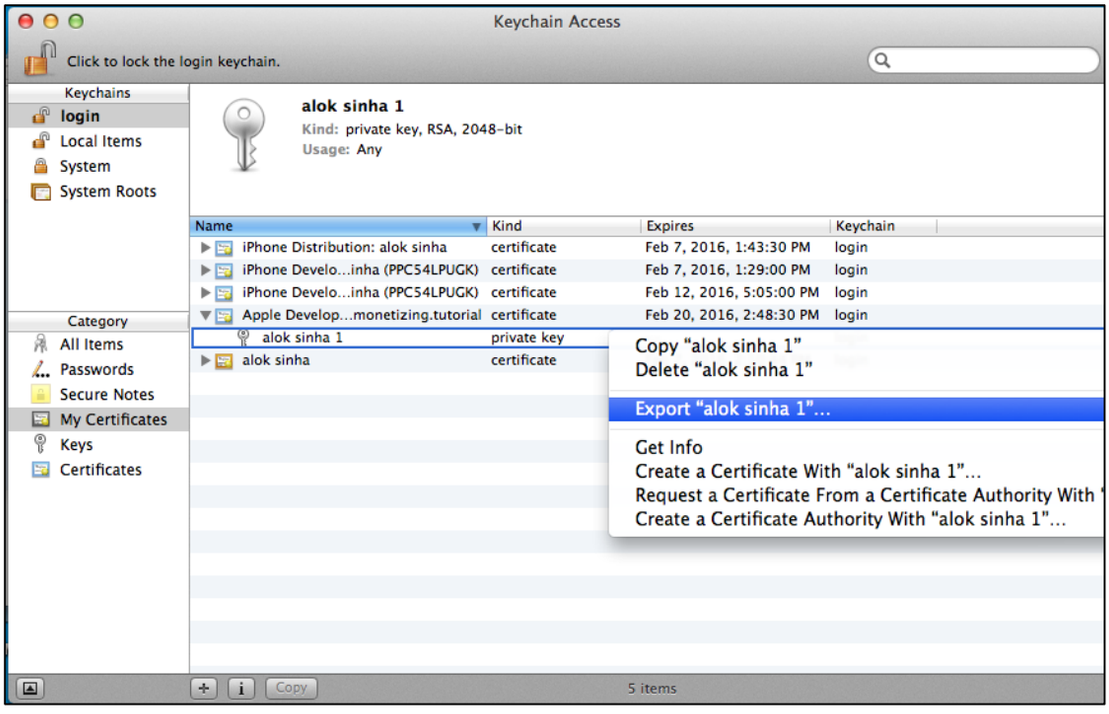

图 34: 导出密钥

25. 将导出的密钥保存为 **PushKey**，保存到需要的文件夹中，格式为.p12 文件格式，如图 35 所示。图 36 显示了为推送通知生成的所有证书。

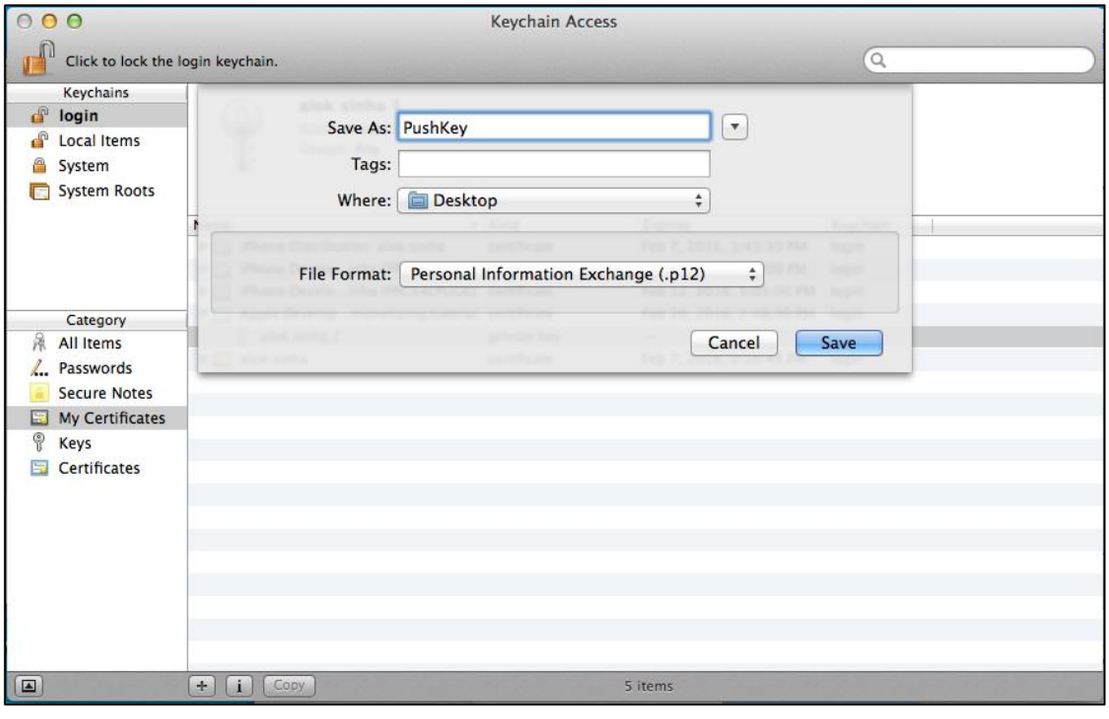

图 35: 保存导出的密钥

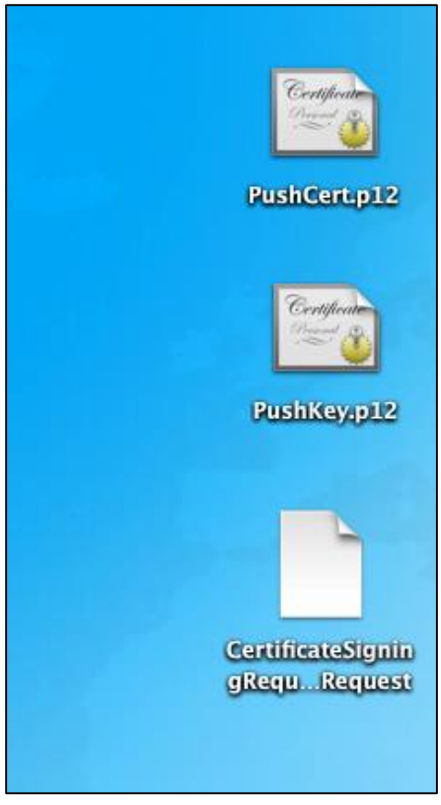

图 36: 为推送通知生成的证书

26. 打开 Terminal，编写如下代码来生成密钥对。这一密钥对将允许推送通知来接收任何苹果 iOS 8 兼容的物理设备。

```
   - (BOOL)application:(UIApplication *)application
didFinishLaunchingWithOptions:(NSDictionary *)launchOptions {
        // Override point for customization after application launch.
           [[UIApplicationsharedApplication]
registerForRemoteNotificationTypes:(UIRemoteNotificationTypeAlert) |
UIRemoteNotificationTypeBadge | UIRemoteNotificationTypeNone];

   return YES;
   }

   - (void)application:(UIApplication *)application
didRegisterForRemoteNotificationsWithDeviceToken:(NSData *)deviceToken
   {
   NSString *deviceTokenString = [NSStringstringWithFormat:@"%@", deviceTokenString];
   NSLog(deviceTokenString);

   }
```