**IOSANEDevelopFramework 开发人员手册**

# IOSANEDevelopFramework是什么

IOSANEDevelopFramework 是一个基于XCode的开发框架，能够帮助ANE开发人员方便的在XCode中进行ANE的开发。它集成了ANE打包，IPA打包和IPA调试的功能，开发人员修改代码之后，只需要几步的点击操作就能完成打包和真机调试，对提高开发效率有很大的帮助。

# 如何开始

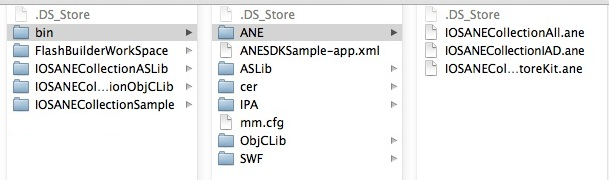
**你需要安装**

1. Flash Builder 4.6 或更高版本
2. XCode 4.3 或更高版本
3. AIR SDK 3.3 或更高版本
4. JRE
5. 推荐使用Mac OS 10.7 或更高版本的操作系统

# 工程结构介绍

**目录结构介绍**

|  |  |
| --- | --- |
| bin | 工程输出文件夹 |
| -ANE | 包含打包生成的ANE文件 |
| -ASLib | 包含由Flash库工程生成的swc文件，用于打包ANE |
| -IPA | 包含用于测试ANE的IPA示例文件 |
| -SWF | 包含Flash示例工程生成的swf 文件和对应的xml应用程序描述文件，用于打包IPA |
| IOSANECollectionASLib | Flash库工程文件夹，包含所有的Flash库工程及源码 |
| IOSANECollectionObjCLib | XCode库工程文件夹，包含XCode库工程及源码 |
| IOSANECollectionSample | Flash示例工程文件夹，包含所有用于测试ANE的Flash示例工程及源码 |



从目录结构中可以看到，整个工程结构分为三部分，Flash库工程，XCode库工程和Flash示例工程。ANE的开发只需要使用Flash库工程和XCode库工程，ANE的测试则需要引入Flash示例工程。

IOS本机扩展大集合项目包含很多的本机功能，这些功能最终以两种ANE的形式发布给用户，一种是所有的本机功能打包成一个ANE，另一种是每个功能打包成一个单独的ANE。ANE使用者可以根据自身的需要选择其中一种使用。包含所有功能的ANE文件名是IOSANECollectionAll.ane。包含单独功能的ANE文件名是IOSANECollection [LIB\_TYPE].ane

每个ANE都对应一个Flash库工程和XCode工程中的一个ObjC库构建目标。

**Flash 库工程介绍**

**全部Flash库工程都位于**IOSANECollectionASLib文件夹中，每个库都是一个Flash Builder工程，并以一个文件夹的形式存在，包含独立的功能，文件夹以库名命名。这些库都可以通过Flash Builder的导入Flash Builder项目功能导入。

其中ASLibCore工程是所有其他工程的依赖项目，包含一些基类和公共方法的实现。

**XCode 工程介绍**

和Flash Builder各独立的库工程不同，**XCode工程只有一个，位于**IOSANECollectionObjCLib文件夹中**。工程中包含很多构建目标，每个打包生成的ANE都是通过独立的构建目标实现的。**

**每个负责打包ANE的构建目标都有一个文件夹对应，文件夹里包含打包ANE需要的本机扩展描述文件extension.xml 和 链接库选项文件platformoptions.xml。**

**所有的源代码代码和打包需要的脚本都在**IOSANECollectionObjCLib/IOSANECollectionObjCLib 中。

**Flash 示例工程介绍**

**全部Flash示例工程都位于**IOSANECollectionSample文件夹中，每个示例都是一个Flash Builder工程，并以一个文件夹的形式存在，包含一个或多个ANE的使用示例，文件夹以示例名命名。这些示例都可以通过Flash Builder的“导入Flash Builder项目”功能导入。

这些示例生成的swf文件位于bin/SWF文件夹中。因为这些swf是依赖ane的，所以不能独立运行，需要被打包成IPA才能运行。XCode工程中有相应的构建目标实现了IPA的打包工作，打包生成的IPA位于bin/IPA中。

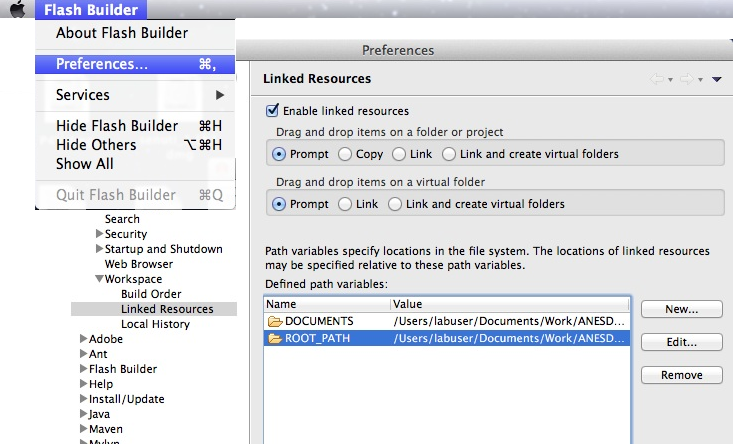
# 搭建开发环境

**Build Flash Lib Project 构建Flash库工程**

1. **打开Flash Builder Open Flash Builder。**
2. **在Flash Builder 当前的工作空间中添加一个Linked Resource**

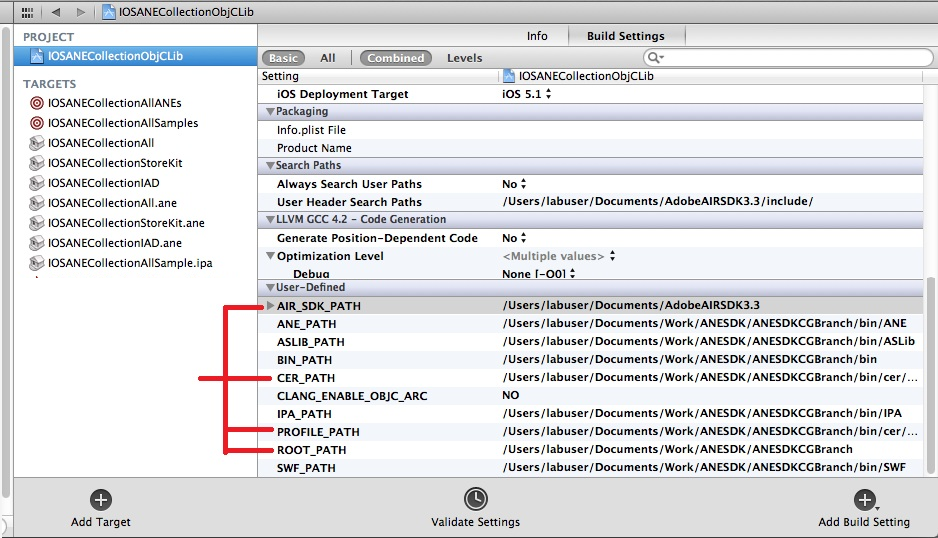
**“ROOT\_PATH” 指向你的IOSANECollection工程根目录。**

1. **导入IOSANECollectionASLib文件夹下所有Flash工程（如果你不想看全部的代码，至少要导入ASLibCore）。**
2. **编译工程。**
3. **检查bin/ASLib目录下是否成功生成编译工程对应的swc。**
4. **尝试编译IOSANECollectionASLib文件夹下其他的工程。**



**构建XCode工程**

1. **在IOSANECollectionObjCLib文件夹里找到IOSANECollectionObjCLib.xcodeproj 工程文件并打开。**
2. **打开工程的构建设置，修改下面的配置项**
   1. **ROOT\_PATH 指向你的IOSANECollection工程根目录。**
   2. **AIR\_SDK\_PATH 指向你的AIRSDK 目录。**
   3. **CER\_PATH 指向你的证书文件。**
   4. **PROFILE\_PATH 指向你的设备描述文件。**



**修改这些**

1. **在XCode的Scheme中选择IOSANECollectionAll.ane。**
2. **点击Run。**
3. **检查bin/ANE目录下是否成功生成IOSANECollectionAll.ane。**
4. **尝试构建其他的ANE目标。**

**如果上面的步骤都完成，说明您的开发环境已经配置完毕，可以开始开发ANE了。**

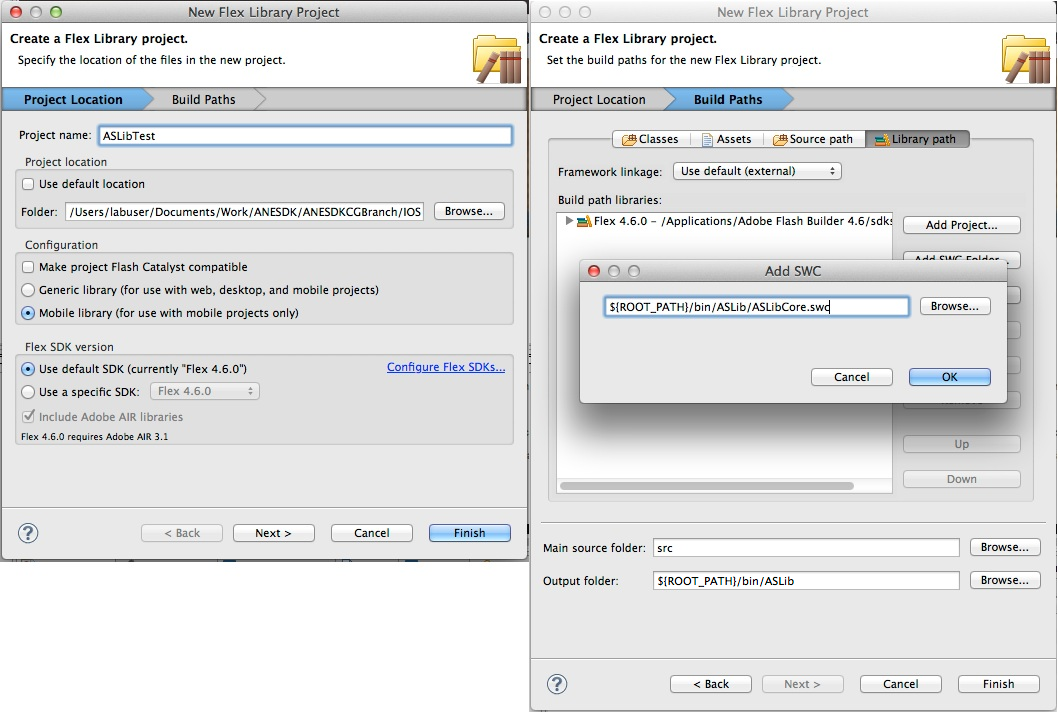
# 创建你的ANE项目

非常感谢您对IOS本机扩展大集合项目的关注，并愿意为这个项目做出自己贡献，并坚持阅读本文档。下面本文档将开始帮助您创建您的ANE项目了。

**创建Flash库项目**

首先打开Flash Builder，创建一个Flex Library Project。

1. 这里建议工程名字命名为ASLib+功能名，当然使用其他的名字也可以。
2. 注意选择工程路径要在**IOSANECollectionASLib文件夹下。**
3. **点击下一步。**
4. **选择依赖库标签，添加swc，路径：${ROOT\_PATH}/bin/ASLib/ASLibCore.swc。**
5. **修改输出文件夹：${ROOT\_PATH}/bin/ASLib。**
6. **点击完成。**



然后就是在Flash Builder中编码，定义ANE的AS接口。具体可以参考ASLibStoreKit的实现。

1. 创建包 com.adobe.anesdk.components.xxx. 为了防止命名冲突，所有代码都要写在这个包里。
2. 建议把ANE的AS接口分为四个模块 Context、Core、Helper、Test，其中Context是必须的其余三个可选。
3. 如果AS接口包含Core和Test模块，则建议在com.adobe.anesdk.components.xxx 包中分别创建core 包和test包 并把相应的代码放入。
4. Context 模块是对flash.External.ExtensionContext类的封装，负责ExtensionContext实例的创建、方法的调用和事件的监听。这些工作已经由ANESDKExtensionContextBase完成了一部分，所以您可以继承ANESDKExtensionContextBase类，子类只需要负责提供Extension ID和对本机功能调用的封装即可。子类的命名建议是xxxExtensionContext。

Core 和Helper 模块的开发建议

1. 如果本机扩展需要实现的功能比较简单，例如只有逻辑计算，且不需要大量依赖iOS SDK 中的类，功能本身也不需要暴露任何ActionScript的类供用户使用，则您完全可以不写Core模块。
2. Helper是为了方便用户使用，简化功能调用流程设计的，您可以根据具体的情况选择取舍，不过鉴于对使用者的友好性，还是建议保留。
3. 对于功能复杂且需要大量依赖iOS SDK的本机扩展，考虑到对外暴露功能的完整性和灵活性，建议您可以在Core模块中建立一套对iOS SDK 的映射类。映射类有和iOS SDK中对应类一样的类名、方法名和尽量一致的方法参数列表，且每个方法实现一样的功能。具体可以参考ASLibStoreKit的实现。当然凡事不是绝对的，您也可以根据不同的情况、条件限制等因素做功能的封装与暴露的平衡。

Test模块的开发建议

1. 因为本机扩展本身分为ActionScript端和本机端，测试起来也比较麻烦，出现问题调试起来就更麻烦，所以为了方便测试，方便问题的隔离，这里建议您为每个本机扩展的本机实现都写一个ActionScript的功能替身。这个替身就是放在Test包中的，替身命名建议为xxxExtensionContextForLocalTest。
2. 创建IxxxExtensionContext接口，令xxxExtensionContext类和xxxExtensionContextForLocalTest类 实现这个接口。在 xxxExtensionContext 类的 get instance() 方法中判断当前是否开启本地测试，如果是返回xxxExtensionContextForLocalTest 的实例 否则返回xxxExtensionContext的实例。详细的实现流程可以参考ASLibStoreKit。

其他建议

1. 暴露给使用者的接口代码尽量写详细的注释，注释的格式符合asdoc的规范。
2. ActionScript命名规范符合AS3 Coding Standards <http://techwiki.openstructs.org/index.php/AS3_Coding_Standards>

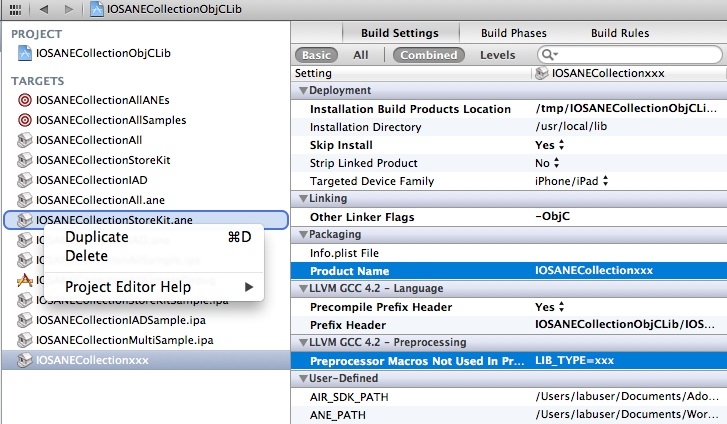
如果你的本机扩展ActionScript 端的编码已经告一段落，下面就可以开始XCode上的编码了，开始前记得检查：

1. ActionScript端是否已经成功在目录bin**/ASLib中**生成swc。
2. 设定的Extension ID 是否有代表性，和现有的ID有没有重名。

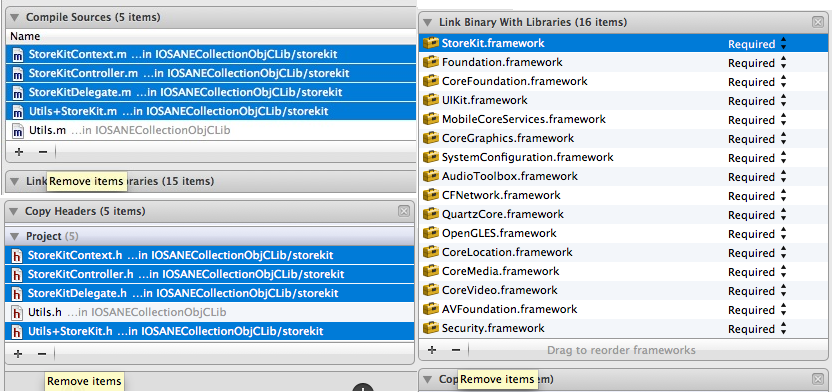
**XCode项目中添加构建目标**

首先打开XCode的工程，创建一个iOS库目标。比较简单的方法是复制一个现有的库目标，例如IOSANECollectionStoreKit。因为这样可以使一些构建参数的配置简单化。下面是复制现有库目标的流程：

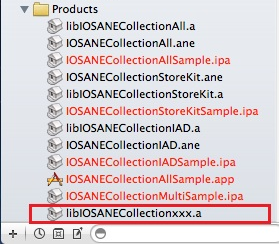
1. 右键点击一个现有的库目标(例如IOSANECollectionStoreKit)，选择Duplicate。
2. 将目标名和Product Name设置为您希望构建的ANE的名字，建议名字符合IOSANECollection+功能名的格式。注意功能名要有代表性，不要和现有的ANE功能名重名。建议这里的功能名和ActionScript端代码中需要传递的extensionType参数一致。
3. 修改Preprocessing 配置令LIB\_TYPE=功能名，这里建议功能名和上面的功能名一致 。



1. 因为目标是复制过来的，所以还要选择Build Phases 选项卡，删除一些不必要的配置
   1. 打开 Compile Sources 面板，删除除Utils.m以外的源文件 （如果您的代码中不需要调用Utils.m中提供的方法，就删除所有源文件）。
   2. 打开 Link Binary With Libraries 面板，删除StoreKit.framework （因为StoreKit.framework是StoreKit功能特有的需要动态链接的库）。
   3. 打开 Copy Headers 面板，删除除Utils.h以外的头文件（如果您的代码中不需要调用Utils.m中提供的方法，就删除所有头文件）。



1. 打开Manage Schemes… 面板，将对应库目标加进来。
2. 编译目标，检查目标输出目录Products是否有libIOSANECollectionxxx.a。如果有，说明上述步骤执行成功，您已经成功创建了一个空的ANE项目的ObjC库目标。



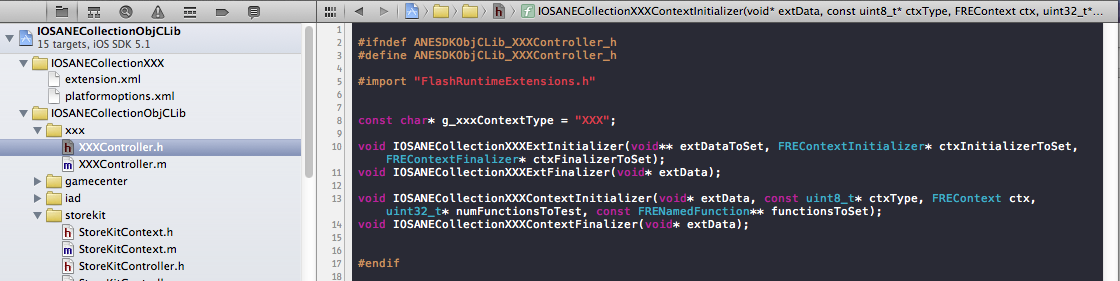
然后就是在XCode中编码，逐一实现功能。具体可以参考StoreKit的实现

1. 在工作区的IOSANECollectionObjCLib文件夹下创建一个Group，建议命名为ANE的功能名。您全部需要添加的代码都要放到这个Group里。
2. 相应的在文件系统的IOSANECollectionObjCLib/IOSANECollectionObjCLib文件夹下也创建一个相同名字的文件夹，用于存放您代码的文件实体。
3. 注意本机扩展初始化函数和上下文初始化函数的定义和实现，函数名不要和现有的初始化函数重名。建议函数名设定为：

IOSANECollection[功能名]ExtInitializer

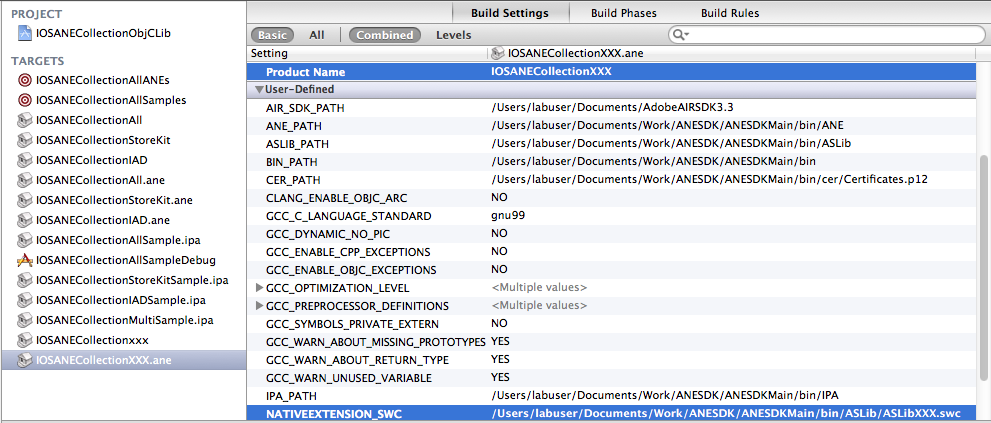
IOSANECollection[功能名]ContextInitializer

1. 编写代码实现功能。Utils.m里定义了很多实用的静态方法，如果想使用可以通过宏UTILS\_CLASSNAME来调用，这个宏在编译时会被替换为Utils类的类名。



编码工作告一段落时可以尝试打包ANE。

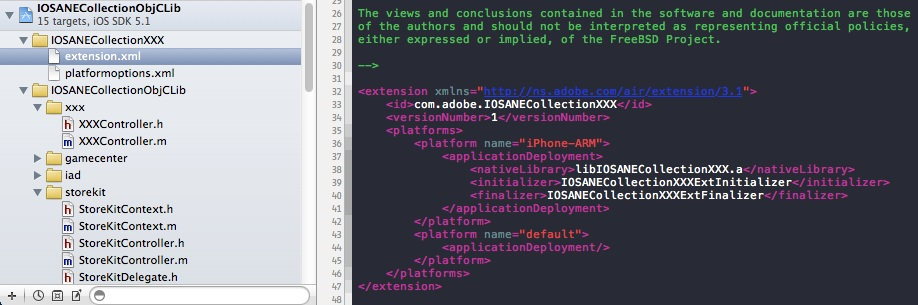
1. 点击工程图标进入工程配置面板，复制一个现有的用于生成ANE的目标（名字以.ane结尾的）
   1. 建议将目标名改名为IOSANECollection[功能名].ane。
   2. 将Product Name改为IOSANECollection[功能名] （必须和库目标的Product Name一致）。
   3. 将NATIVEEXTENSION\_SWC设置为指向ANE对应的swc 文件路径。



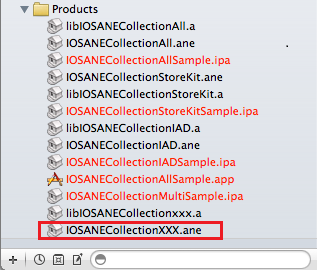
1. 在工作区中的根节点创建一个Group，命名为IOSANECollection+功能名（必须和Product Name一致）。

相应的在文件系统的IOSANECollectionObjCLib文件夹下也创建一个相同名字的文件夹，用于存放打包ANE的配置文件。

* 1. 在Group中创建extension.xml 和platformoptions.xml。（也可以直接从其他的用于打包ANE的Group中拷贝过来）注意这里的文件名不能随意起，必须叫extension.xml 和platformoptions.xml。
  2. 注意extension.xml 和platformoptions.xml文件实体在文件系统中的位置要在您刚刚创建的文件夹中。
  3. 修改extension.xml 和platformoptions.xml的内容以适应您的ANE。



1. 打开Manage Schemes… 面板，将对应库目标加进来。
2. 编译目标，检查目标输出目录Products中是否有IOSANECollectionXXX.ane。如果有，说明上述步骤执行成功，您已经成功创建了一个ANE并且可以在目录bin/ANE里找到它。



# Create Your ANE Testing Project创建你的ANE测试项目

**Create Flash Mobile Project for ANE Testing创建用于ANE测试的Flash手机项目**

**Add IPA Target in XCode Project for ANE Testing 在XCode项目中添加用于ANE测试的IPA目标**

**Debug your ANE Code 调试你的ANE代码**