iOS学习记录

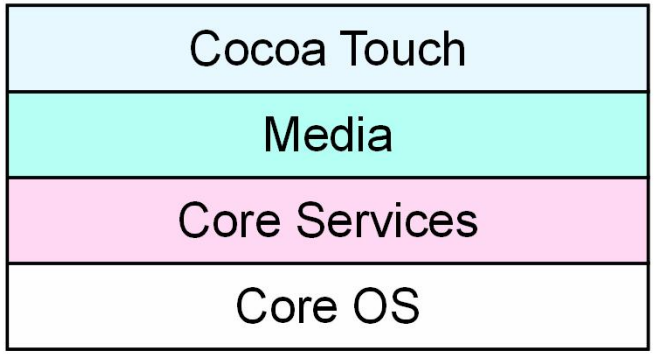
丁碧云

2016-12-4

Contents

- iOS系统架构
- 程序的实现

iOS系统架构

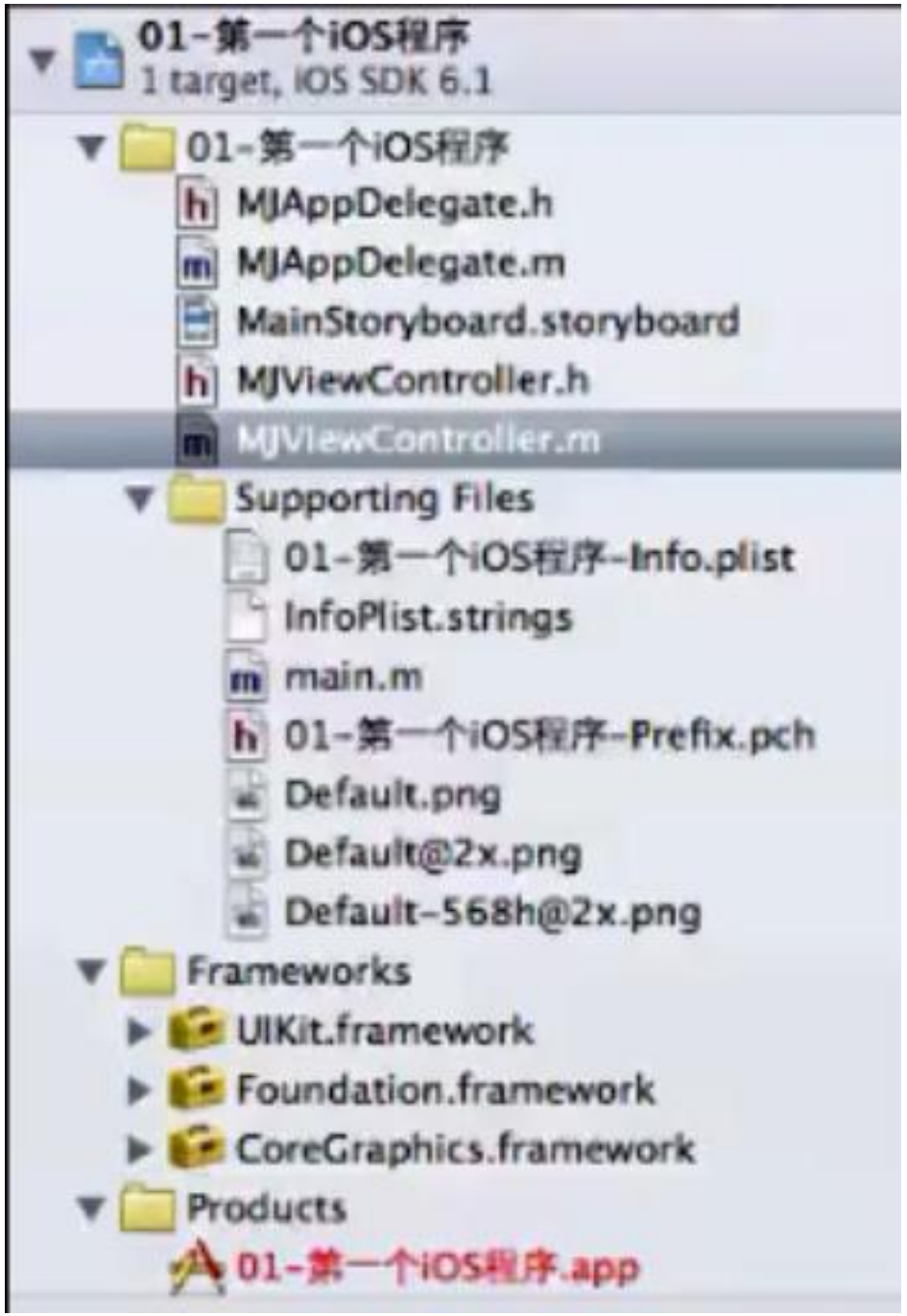


iOS的系统框架分为四个层次：

- **核心操作系统层（Core OS）**：包括内存管理、文件系统、电源管理以及一些其他的操作系统任务。可以直接和硬件设备进行加互。
- **核心服务层（Core Services）**：可以通过它来访问iOS的一些服务，比如网络连接、文件访问、数据库和用户定位等。
- **媒体层（Media）**：通过它可以在应用程序中使用各种媒体文件，进行音频和视频的录制，图形的绘制以及制作基础的动画效果。
- **可触摸层（Cocoa Touch）**：这一层为应用程序开发提供了各种游泳的框架，并且大部分与用户界面有关，本质上来说，它负责用户在iOS设备上的触摸交互操作。

一个基本的ios程序的文件包括

这些文件在建立一个Single View Application（单个界面）后是自动生成的一个模板工程
第一个例子中是做一个简单的计算器界面；



程序的实现

- OC中类分为2个文件
- .h，类的声明文件，用于声明变量、函数(方法)
- .m，类的实现文件，用于实现.h中的函数(方法)
- 类的声明使用关键字@interface、@end
- 类的实现使用关键字@implementation、@end

实现流程

MainStoryboard.storyboard文件

从iOS5开始，storyboard文件用于描述软件界面，作为程序入口，选定view->object，从下面的框中选择需要的控件，选定控件后用来设置其属性。

绑定控件与方法（属性），以实现对控件的操作

通过点击Editor的中间件，将控件拉线连接到控制的方法上，实现绑定。

UIViewController.h文件

在这个文件只能声明方法（监听按钮或是其触发事件），声明控件属性

UIViewController.m文件

在这个文件中，写出方法的具体操作

MJViewController.h申明方法和属性

```
//
//  MJViewController.h
//  01-第一个iOS程序
//
//  Created by apple on 13-11-21.
//  Copyright (c) 2013年 itcast. All rights reserved.
//

#import <UIKit/UIKit.h>

@interface MJViewController : UIViewController

// 声明一个方法来监听按钮点击  IBAction === void
- (IBAction)btnClick;

// 声明2个属性用来保存2个文本输入框
@property (nonatomic, weak) IBOutlet UITextField *num1;
@property (nonatomic, weak) IBOutlet UITextField *num2;

@property (nonatomic, weak) IBOutlet UILabel *result;

@end
```

文本输入的控件名

显示文本的控件名

表明已经实现了与控件的绑定

MJViewController.m方法的具体操作

```
#import "MJViewController.h"

@interface MJViewController ()

@end

@implementation MJViewController

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

#pragma mark 监听按钮点击
- (void)btnClick
{
    // 1.获得文本输入框的文字
    NSString *text1 = self.num1.text;
    NSString *text2 = self.num2.text;
//    _num1;

    // 2.将字符串转为整数
    int i1 = [text1 intValue];
    int i2 = [text2 intValue];


    // 3.将结果显示到右边的标签中
    self.result.text = [NSString stringWithFormat:@"%d", i1 + i2];

}
@end
```

转为字符串格式

.h中申明类，.m中实现类方法

```objc
//  Student.h
//  OC的类
//
//  Created by mj on 13-3-28.
//  Copyright (c) 2013年 itcast. All rights reserved.
//  只是用来声明Student这个类有哪些成员变量和方法

#import <Foundation/Foundation.h>

// @interface代表声明一个类
// : 代表继承
@interface Student : NSObject { // 成员变量要定义在下面的大括号中{}
    int age;
}

// 在这里声明的所有方法都是公共

// age的get方法
// - 代表动态方法  + 代表静态方法
- (int)getAge;

// age的set方法
- (void)setAge:(int)age;

@end
```

一个冒号代表一个参数，
后面跟参数类型和参数名

```objc
//  Student.m
//  OC的类
//
//  Created by mj on 13-3-28.
//  Copyright (c) 2013年 itcast. All rights reserved.
//

#import "Student.h"

@implementation Student

- (int)getAge {
    return age;
}

- (void)setAge:(int)newAge {
    age = newAge;
}

@end
```

类方法的实现

在main.m中的调用

```
// 方法名是setAge:andNo:
- (void)setAge:(int)newAge andNo:(int)no;
```

其中空格不属于方法名的一部分

```
// [person setAge:10];
person.age = 10;
```

点语法的运用，会自动调用类对属性值的操作方法。

```
// 这是错误的写法，会导致死循环，无限调用set方法
// self.age = newAge;// [self setAge:newAge];
```

```
15      @autoreleasepool {
16          // 创建一个Student对象:
17
18          // 1.调用一个静态方法alloc来分配内存
19          // 暂时把id当做是任何对象
20  //          Student *stu = [Student alloc];
21  //
22  //          // 2.调用一个动态方法init进行初始化
23  //          stu = [stu init];
24
25          Student *stu = [[Student alloc] init];
26
27          [stu setAge:100];
28
29          int age = [stu getAge];
30
31          NSLog(@"age is %i", age);
32
            // 释放对象
            [stu release];
        }
```

可代替上面两行diamante

注意参数前面要加:

注意:也是方法名的一部分

对象用完后释放对象

```
No Selection
All Output
2013-03-28 15:58:15.360 OC的类[3630:303] 调用了setAge方法
2013-03-28 15:58:15.362 OC的类[3630:303] 调用了getAge方法
2013-03-28 15:58:15.363 OC的类[3630:303] age is 100
```

类

- 方法的声明和实现，都必须以 + 或者 - 开头
- + 表示类方法(静态方法)
- - 表示对象方法(动态方法)
- 在.h中声明的所有方法都是public类型(通过Category可以实现private)

- 变量的作用域

@public 全局都可以访问
@protected 只能在类内部和子类中访问
@private 只能在类内部访问

- 变量必须定义在类的 { } 中

程序的启动过程

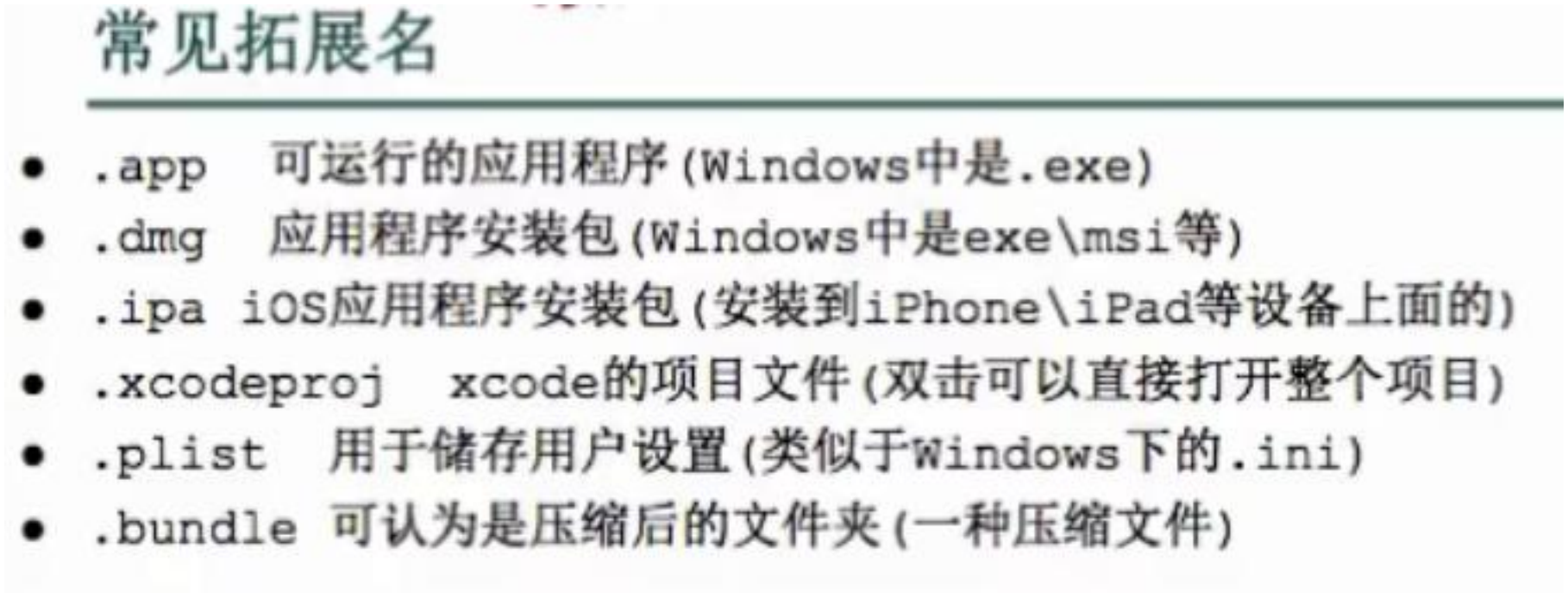
* 程序启动过程
1> 加载最主要的storyboard文件
2> 创建白色箭头所指的控制器对象（MJViewController）
3> 创建控制器内部的view，显示到用户眼前

- 点击对应的选项，就会切换到下一个新的界面，每一个新的界面都是一个新的 UIView，它们的尺寸接近屏幕大小
- 一般情况下，每一个“满屏”的 UIView 都交给对应的 UIViewController 去管理，像上图中的三个 UIView，都有自己的 UIViewController
- UIViewController 内部有个 UIView 属性，就是它负责管理的 UIView 对象

```
UIView          *_view;
@property(nonatomic,retain) UIView *view;
```

- UIViewController 的作用是：负责创建\销毁自己的 UIView、显示\隐藏 UIView、处理 UIView 和用户之间的交互（事件处理）。UIViewController 就是 UIView 的大管家。
- 因此，“设置”应用中的界面显示过程应该是：先创建一个 UIViewController，再由 UIViewController 创建自己的 UIView，最后把 UIView 显示到用户眼前。并且由 UIViewController 来处理 UIView 的事件，比如点击事件

Mac OS X 系统中的常见扩展名



谢谢大家！