GeekBand 极客班

互联网人才 + 油站!

GeekBand 极客班

www.geekband.com

Cocoa设计模式

孔祥波

什么是设计模式?

在软件工程中，**设计模式**（design pattern）是对软件设计中普遍存在（**反复出现**）的各种问题，所提出的解决方案。这个术语是由埃里希·伽玛（Erich Gamma）等人在1990年代从建筑设计领域引入到计算机科学的。

设计模式并**不直接**用来完成代码的编写，而是描述在各种不同情况下，要怎么解决问题的一种方案。面向对象设计模式通常以类或对象来描述其中的**关系和相互作用**，但不涉及用来完成应用程序的特定类或对象。设计模式能使不稳定依赖于相对稳定、具体依赖于相对抽象，避免会引起麻烦的紧耦合，**以增强软件设计面对并适应变化的能力**。

为什么学习设计模式?

- 软件开发过程中，容易忽视技术本身的**整体架构和基本原理**
- 控制开发成本的同时，通过缩短软件开发周期**促进程序员效率**
- Cocoa/Cocoa Touch 优雅的设计，大量使用设计模式
- 为那些**重复出现的编程问题**提供高质量的使用解决方案

设计模式的基本要素

- 模式名称
- 对模式的动机或者能解决的问题的简短描述
- 对模式的示例详细描述
- 使用此模式的成果

设计的指导原则

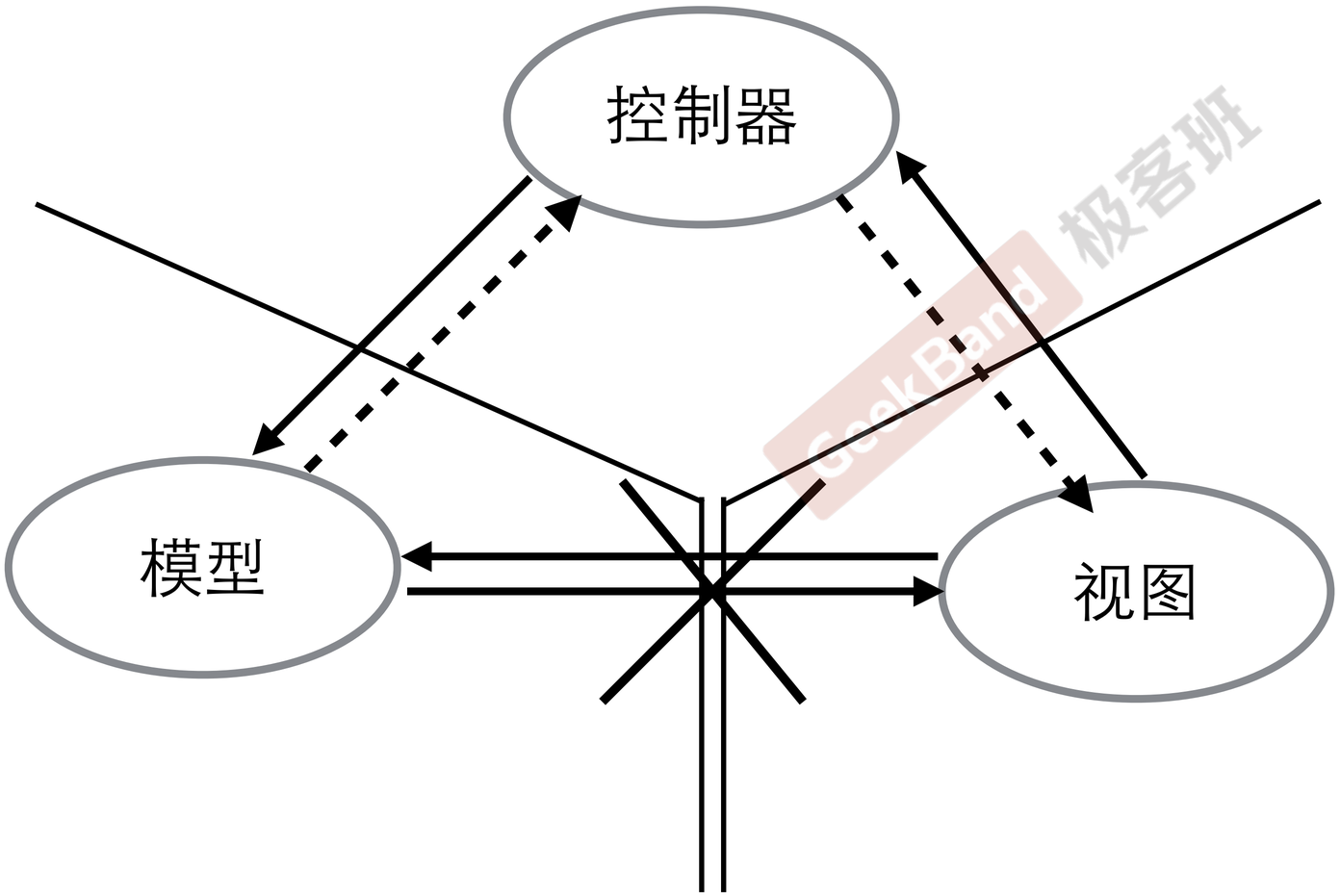
- 最小耦合
- 设计变更
- 强调接口而不是实现
- 找到最佳的粒度
- 使用复合更优于继承(父类和子类强耦合)

课程内容

- 深入理解**MVC**
- 两部创建
- 模版方法
- 单利模式
- 委托模式
- 观察者
- 通知模式
- **KVC/KVO**
- 归档与解档
- 层次结构
- 响应链
- **Prototype**

Model(模型)/View(视图)
Controller(控制器)

Model(模型),View(视图),Controller(控制器)



模型

- 数据是变化的(创建/修改/删除/查询)，管理应用程序数据和状态
- 和表现层(UI)无关
- 经常需要持久化保存(内存memory 外存Flash/SSD)
- 同一模型可以重用，甚至是不需要修改

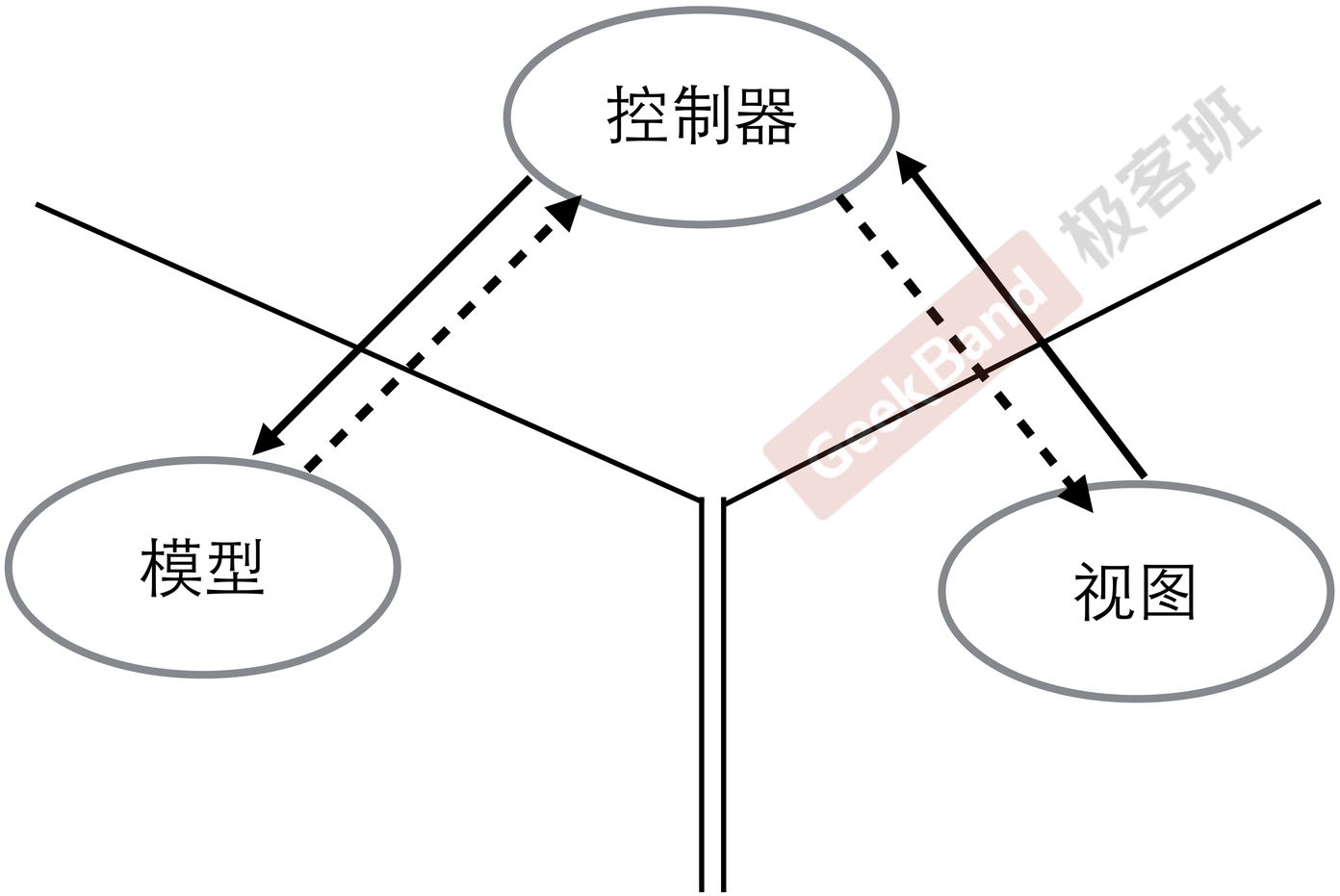
视图

- 使用恰当的方式将模型展示给用户(模型的简介/详请)
- 允许用户操作数据(创建/修改/删除/查询)
- 不存储数据(除保存临时数据cache)
- 简单可以重复使用，可配置显示不同类型数据(ImageView,Label)

控制器

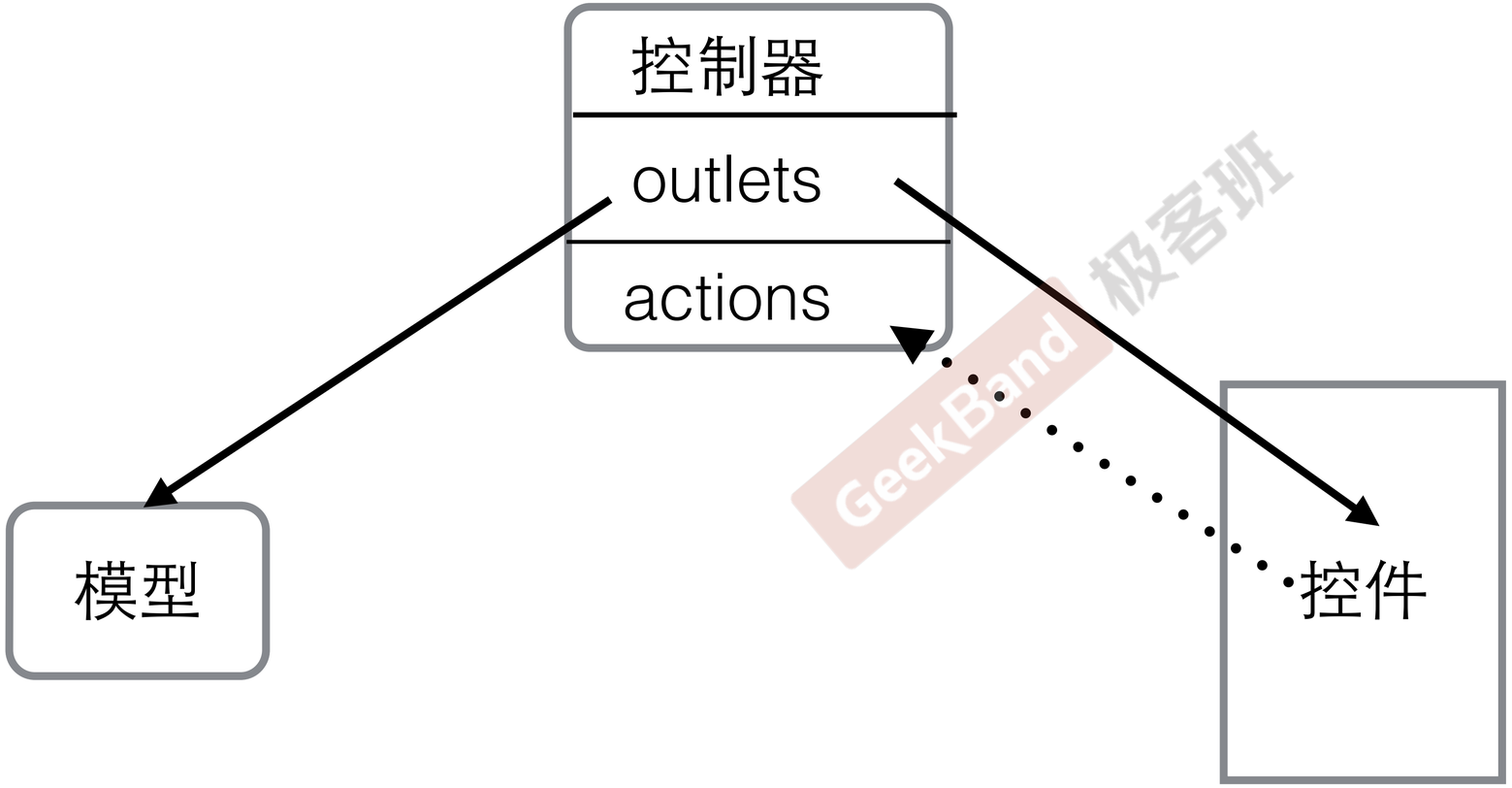
- 模型和视图的桥梁(模型和视图不允许互操作)
- 模型改变更新视图
- 用户操作视图，更新模型数据
- 一般维护应用程序核心逻辑(创建/下载数据等)

Model(模型),View(视图),Controller(控制器)

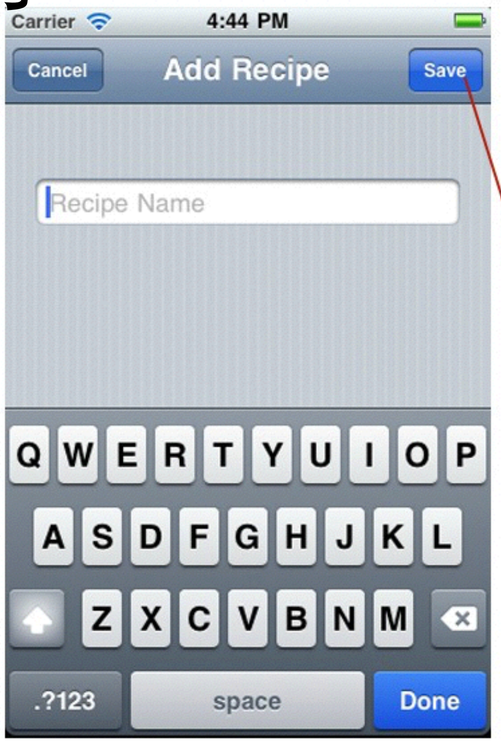


Target-Action 链接逻辑和控件

Target-Action 链接逻辑和控件



示例



Target

```
/*
     File: RecipeAddViewController.h
 Abstract: View controller to allow the user to add a
     new recipe
 |
  Version: 1.4

 */

@class Recipe;

@interface RecipeAddViewController : UIViewController
{
    Recipe *recipe;
    UITextField *nameTextField;
}

@property(nonatomic, retain) Recipe *recipe;
@property(nonatomic, retain) IBOutlet UITextField *
    nameTextField;

- (void)saveRecipe:(id)sender;
- (void)cancel;

@end
```

Action Method

```
UIBarButtonItem *saveBtn = [[UIBarButtonItem alloc] initWithTitle:@"Save"
                                                    style:UIBarButtonItemStyleDone
                                                    target:self
                                                    action:@selector(saveRecipe:)];
```

```
- (void)doSomething;
// OR
- (void)doSomething:(id)sender;
// OR
- (IBAction)doSomething:(id)sender;
// OR
- (IBAction)doSomething:(UIButton *) sender;
```

Control Event

```
[aUIButton addTarget:self
                action:@selector(doSomething:)
     forControlEvents:UIControlEventTouchUpInside];
```

```
typedef NS_OPTIONS(NSUInteger, UIControlEvents) {
    UIControlEventTouchDown                                         = 1 <<  0,      // on all touch downs
    UIControlEventTouchDownRepeat                                   = 1 <<  1,      // on multiple touchdowns (tap count > 1)
    UIControlEventTouchDragInside                                   = 1 <<  2,
    UIControlEventTouchDragOutside                                  = 1 <<  3,
    UIControlEventTouchDragEnter                                    = 1 <<  4,
    UIControlEventTouchDragExit                                     = 1 <<  5,
    UIControlEventTouchUpInside                                     = 1 <<  6,
    UIControlEventTouchUpOutside                                    = 1 <<  7,
    UIControlEventTouchCancel                                       = 1 <<  8,

    UIControlEventValueChanged                                      = 1 << 12,     // sliders, etc.
    UIControlEventPrimaryActionTriggered NS_ENUM_AVAILABLE_IOS(9_0) = 1 << 13,     // semantic action: for buttons, etc.

    UIControlEventEditingDidBegin                                   = 1 << 16,     // UITextField
    UIControlEventEditingChanged                                    = 1 << 17,
    UIControlEventEditingDidEnd                                     = 1 << 18,
    UIControlEventEditingDidEndOnExit                               = 1 << 19,     // 'return key' ending editing

    UIControlEventAllTouchEvents                                    = 0x00000FFF,  // for touch events
    UIControlEventAllEditingEvents                                  = 0x000F0000,  // for UITextField
    UIControlEventApplicationReserved                               = 0x0F000000,  // range available for application use
    UIControlEventSystemReserved                                    = 0xF0000000,  // range reserved for internal framework use
    UIControlEventAllEvents                                         = 0xFFFFFFFF
};
```

3 different flavors of action method selector types

```
- (void)actionMethod;
- (void)actionMethod:(id)sender;
- (void)actionMethod:(id)sender withEvent:(UIEvent *)event;
- (void)removeTarget:(id)target action:(SEL)action
        forControlEvents:(UIControlEvents)controlEvents;
```

selector是什么?

- the name used to select a method to execute for an object(unique identifier a method when source code is compiled)
- A selector by itself doesn't do anything
- SEL aSelector = @selector(methodName);
- SEL aSlector = NSSelctorFormString(@"methodName");
- [AObject performSelector:aSelector]
- BOOL yOrn=[AObject respondsToSelector:aSelector];

Interface Builder ,Xibs(Nibs)，Storyboard

UI Files 设计阶段

- 帮助你设计 “V”(视图) in MVC
- layout 用户界面元素
- 添加控制器对象
- 链接控制器和UI

Nib 加载

- 编译Xcode 使用ibtool 编译xib文件
- 二进制发布
- 运行时，对象被解档，values/settings restore
- 所有outlets和action 必须正确被链接

Nib 加载

```
@interface NSBundle(UINibLoadingAdditions)
- (NSArray *)loadNibNamed:(NSString *)name
    owner:(id)owner options:(NSDictionary *)
    options;
@end
```

Nib 加载

```
NS_CLASS_AVAILABLE_IOS(4_0) @interface UINib : NSObject {
  @private
    id storage;
}

// If the bundle parameter is nil, the main bundle is used.
// Releases resources in response to memory pressure (e.g. memory warning),
    reloading from the bundle when necessary.
+ (UINib *)nibWithNibName:(NSString *)name bundle:(NSBundle *)bundleOrNil;

// If the bundle parameter is nil, the main bundle is used.
+ (UINib *)nibWithData:(NSData *)data bundle:(NSBundle *)bundleOrNil;

// Returns an array containing the top-level objects from the NIB.
// The owner and options parameters may both be nil.
// If the owner parameter is nil, connections to File's Owner are not
    permitted.
// Options are identical to the options specified with -[NSBundle
    loadNibNamed:owner:options:]
- (NSArray *)instantiateWithOwner:(id)ownerOrNil options:(NSDictionary *)
    optionsOrNil;
```

自动创建对象需要自定义状态怎么办？

-awakeFromNib

NSObject(UINibLoadingAdditions) category定义

```
@interface NSObject(UINibLoadingAdditions)
- (void)awakeFromNib;
- (void)prepareForInterfaceBuilder
    NS_AVAILABLE_IOS(8_0);
@end
```

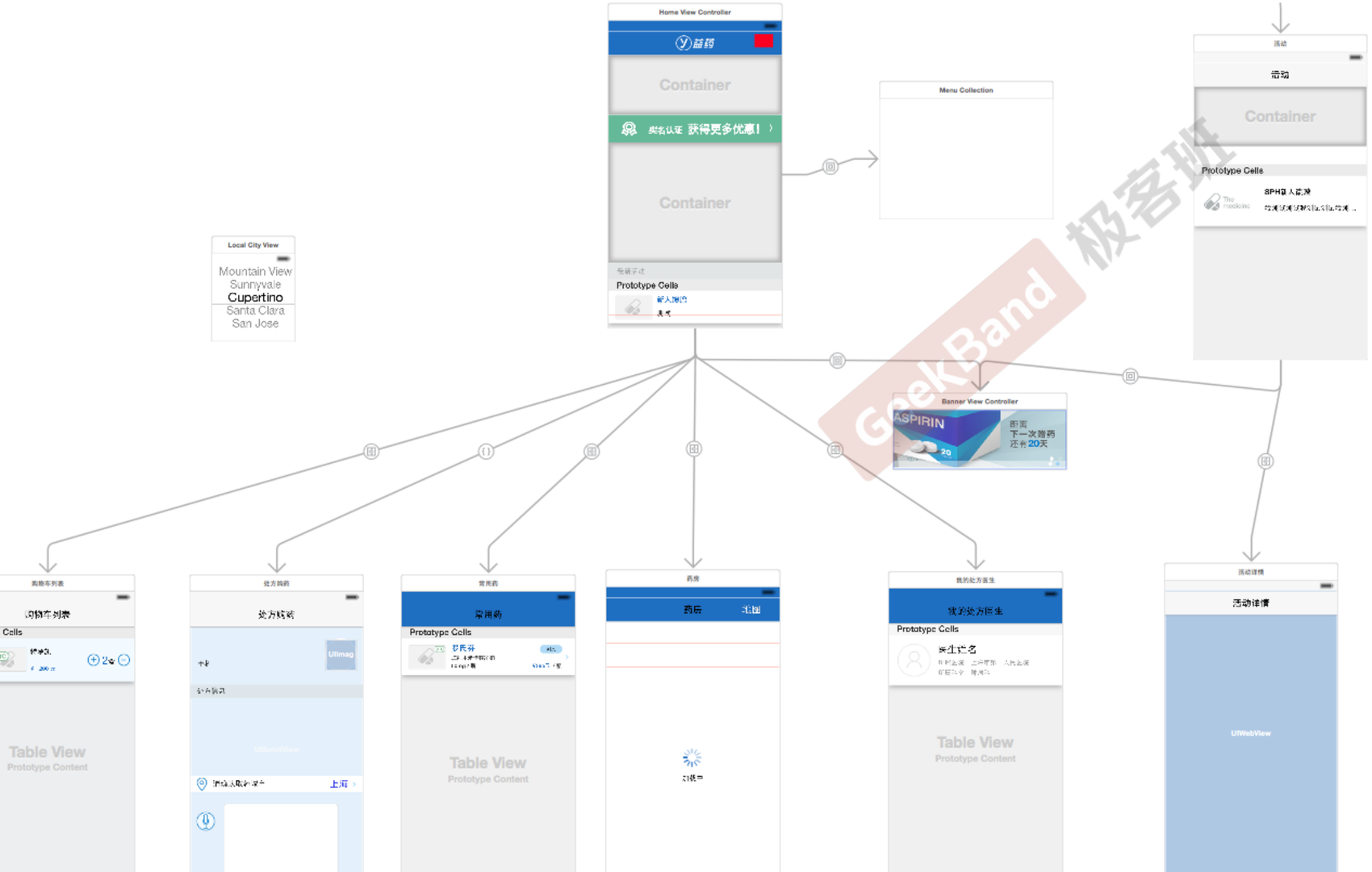
-awakeFromNib

- 加载nib后，可以实现自定义逻辑
- 缺省empty (NSObject)
- 对于controller 对象，经常用来恢复数据和状态

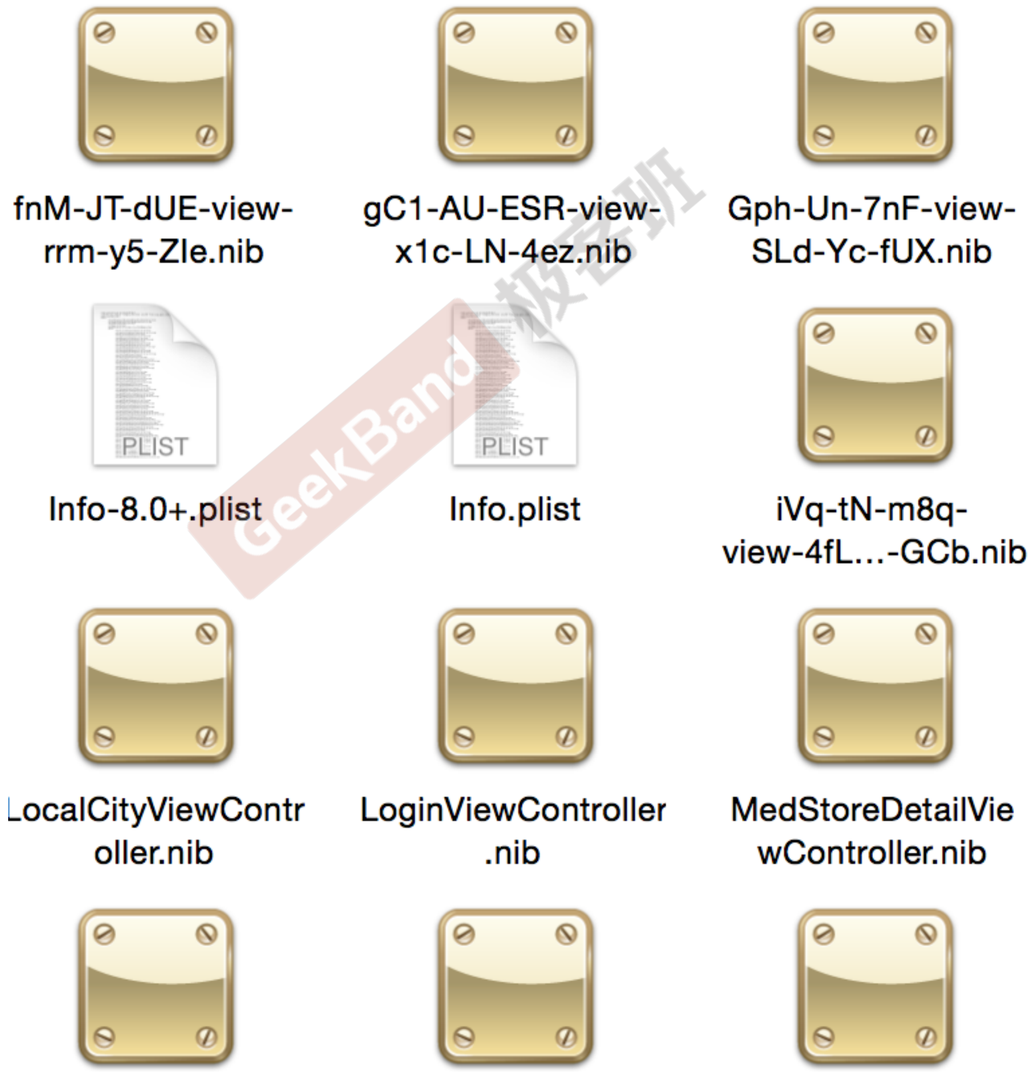
Storyboard

Scene

storyboard



Storybaord编译->Storybaordc文件包



Storyboard 故事板

- 控制view controller 之间关系和跳转
- Nib文件集合，Nib 对应scene
- scene 之间跳转segue（push / Modal）
- 通过实现-(void)prepareForeSegue(UIStoryBoardSegue*)segue sender(id)sender (页面跳转和数据传递)

Storyboard

每个UIViewcontroller 包含storyboard对象

```
@property(nonatomic, readonly, retain) UIStoryboard *storyboard
    NS_AVAILABLE_IOS(5_0);
```

通过storyboard文件创建UIViewcontroller 对象

```
NS_CLASS_AVAILABLE_IOS(5_0) @interface UIStoryboard : NSObject {
}

+ (UIStoryboard *)storyboardWithName:(NSString *)name bundle:(NSBundle *)
    storyboardBundleOrNil;

- (id)instantiateInitialViewController;
- (id)instantiateViewControllerWithIdentifier:(NSString *)identifier;
```

Storyboard 缺点

- 同时编辑，很容易产生冲突(SCM管理)
- 避免同时修改，可以拆分多个storyboard 文件
- scene 较多时，Xcode加载/编辑比较慢
- Storyboard References (**iOS9 WWDC15 Session 215**)

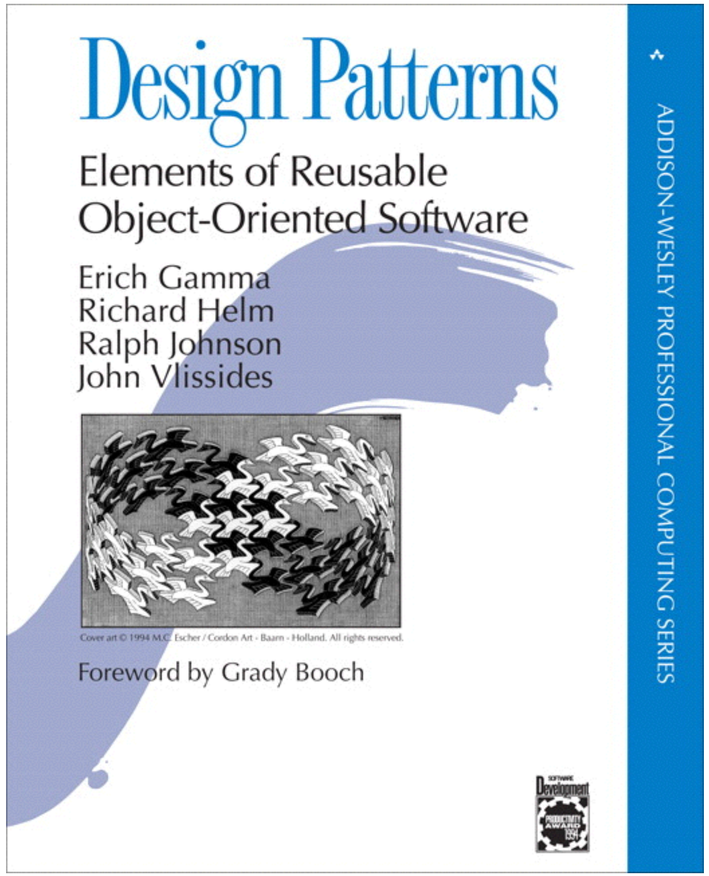
原型设计工具

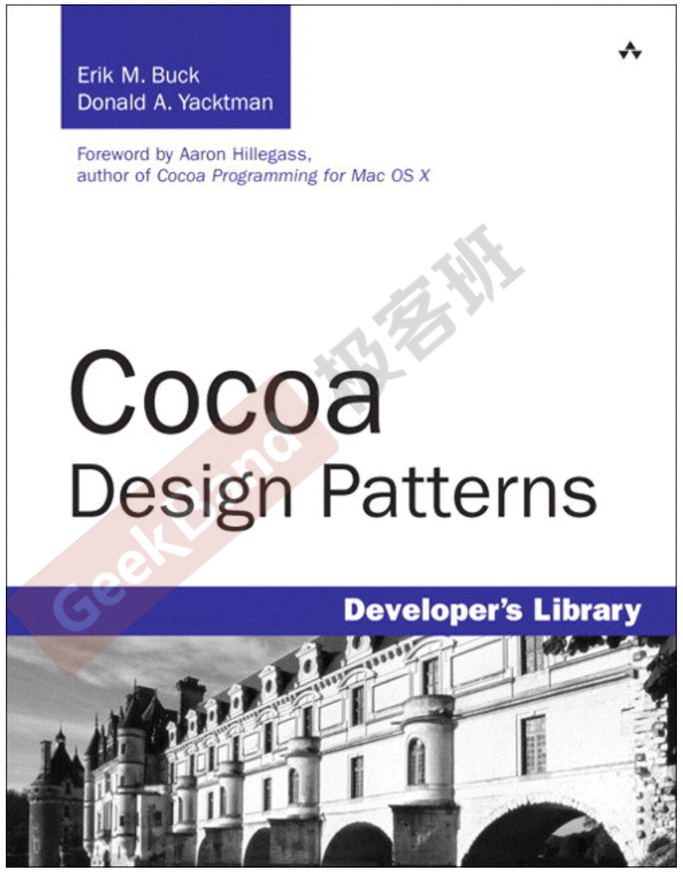
sketch

Omnigraffle

Keynote

书籍


Design Patterns
Elements of Reusable Object-Oriented Software
Erich Gamma
Richard Helm
Ralph Johnson
John Vlissides
Cover art © 1994 M.C. Escher / Cordon Art - Baarn - Holland. All rights reserved.
Foreword by Grady Booch
ADDISON-WESLEY PROFESSIONAL COMPUTING SERIES


Erik M. Buck
Donald A. Yacktman
Foreword by Aaron Hillegass, author of Cocoa Programming for Mac OS X
Cocoa
Design Patterns
Developer's Library
GeekBand 极客班

小结

- 设计模式
- MVC 模式
- target-action
- selector
- Nib/storyboard
- 原型设计

结束