1. KVC

最基本的KVC有NSKeyValueCoding协议提供支持

- setValue:forKey

对于setValue:属性值forKey:@“name”

底层实现机制：看是否有名为setName的setter－>检查是否有名为\_name的成员变量－>检查是否有名为name的成员变量－>如果都没有执行setValue：forUndefinedKey方法

- valueForKey

对于valueForKey:@“name”

底层实现机制：看是否有名为name的getter－>检查是否有名为\_name的成员变量－>检查是否有名为name的成员变量－>如果都没有执行valueforUndefinedKey方法

- Key路径

用于操作对象的“复合属性”（属性中含有属性，使用.来说明内部属性路径即可）

setValue:forKeyPath

valueForKeyPath

- 处理不存在的key

当既不存在对应的setter和getter，也不存在对应的成员变量时，KVC将自动调用setValue：forUndefinedKey：和value：forUndefinedKey:方法。这两个方法只是淡出引发异常，没有进一步处理

- 处理nil值

付过为不接受nil值得属性设置值为nil，将会引发setNilValue：forKey：方法

1. KVO

iOS应用通常将程序组件分为数据模型组件和视图组件。使用KVO实现当数据模型组件中数据改变时视图组件实时显示的功能。

最基本的KVO由NSKeyValueObserving协议提供支持

- addObserver:forKeyPath:options:context:

- removeObserver:forKeyPath:

- removeObserver:forKeyPath:options:context:

1. iOS NSNotification
2. iOS项目

main函数中调用UIApplicationMain函数，隐式创建一个UIApplica对象，并设置其delegate属性为AppDelegate对象（函数的第四个参数）

* 1. 应用程序委托对象

是整个iOS应用的通信中心，还负责应用程序执行过程中的事件循环（也就是负责iOS应用的各种生命周期事件）

要求：继承自UIResPonder和实现UIApplicationDelegate协议

1. 观察者模式
2. MVC

观察者模式下，观察者和被观察者有可能是两个对等的对象，而MVC下被观察者往往只是单纯的数据体，而观察则是单纯的视图组件

1. iOS事件机制

获取控件的方式：IBOutlet或者控件的tag属性

* 1. IBAction
  2. 为UI控件的疼事件绑定事件监听器

能与用户交互的控件大都继承自UIControl基类，可调用其addTarget：action：forControlEvents:方法注册事件处理方法

* 1. 将UI控件的某些生命周期事件委托给对应的委托对象处理

如UITable

1. iOS生命周期
   1. Not running
   2. Inactive

在前台运行，但是不能接收事件处理

* 1. Active
  2. Bacground

在后台，并且还在执行代码

* 1. Suspend

在后台，但是没有执行代码

1. iOS的运行
   1. 响应中断

警告框式的中断（如电话来电）到来时，从Active转到Inactive

Banner中的通知将会被放在应用窗口的上边沿，应用仍然处在Active状态并且可以接收触碰事件；用户拉下Banner查看通知内容时，应用切换到Inactive

锁屏时，系统屏蔽所有触碰事件，应用转入Background，状态转换为UIApplicationStateInactive.