极客学院
jikexueyuan.com

Swift图书展示项目开发实战（一）

Swift图书展示项目开发实战(一)

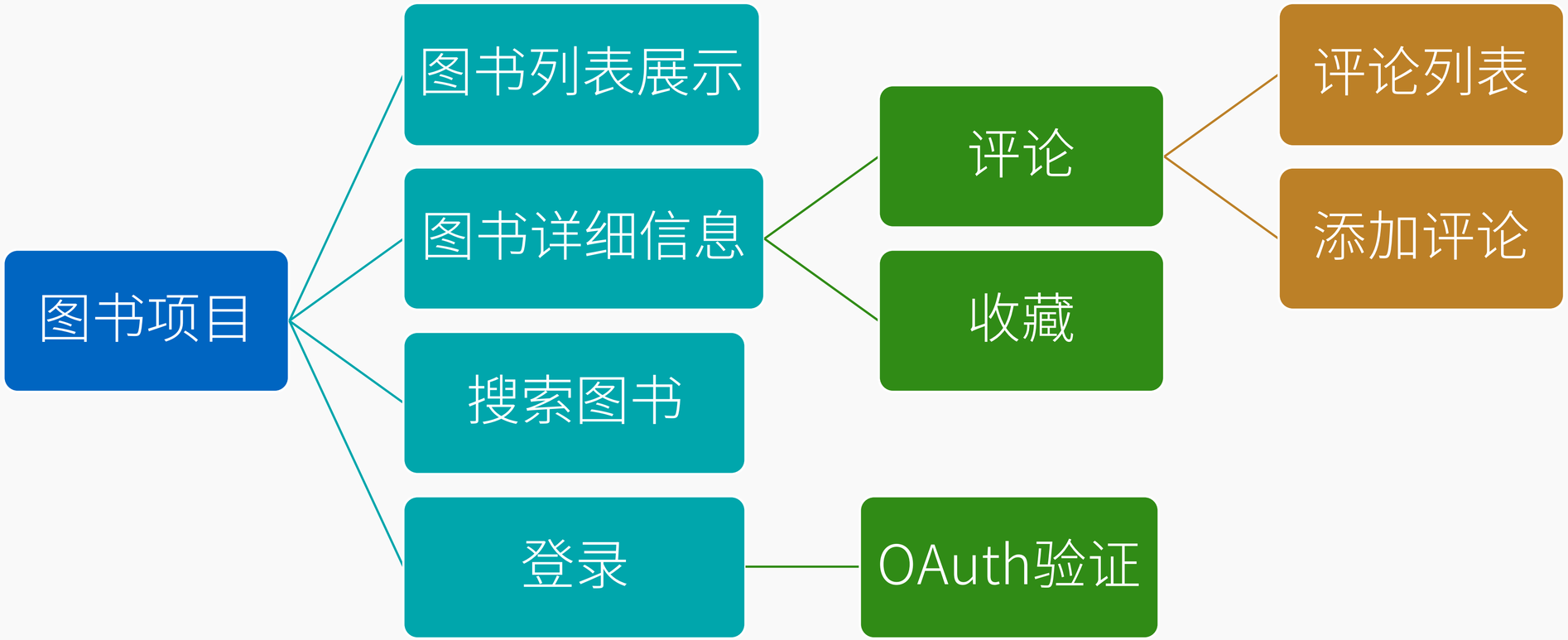
- 项目介绍
- Swift基础
- Swift常用技巧
- 使用CocoaPods管理framework

项目介绍

项目介绍

- 项目功能结构
- 重点学习目标

项目介绍 — **项目功能结构**



项目介绍 — **重点学习目标**

- Swift基本知识
- 如何运用Swift开发app
- 使用UITableView实现复杂的界面
- AutoLayout使用技巧
- 高度自适应

Swift基础

Swift基础

- Swift与Foundation
- 值类型和引用类型
- 命名空间
- OC与Swift互相调用
- 异常处理

Swift基础 — **Swift与Foundation**

无缝兼容 Foundation

- String - NSString
- Int, Float, Double, Bool 以及其他与数字有关的类型 - NSNumber
- Array - NSArray
- Dictionary - NSDictionary
- Set – NSSet
- AnyObject – NSObject

Swift原生类型是对Foudation更好的补充

Swift基础 — **值类型和引用类型**

- struct 和 enum 定义的类型是值类型，使用 class 定义的为引用类型。
- 基本数据类型全是struct
- 处理大量数据并且频繁增减元素时, NSMutableArray、NSMutableDictionary和NSMutableSet,其他情况 Array 、Dictionary和Set好一点

Swift基础 — **命名空间**

- 一个module 代表一个命名空间
- 主target就是一个module,创建framework和target就是新的module
- 不同module中就算类名相同,也可以通过module名进行区分, 不再需要给类名加上各种奇怪的前缀了

Swift基础 — **OC与Swift互相调用**

- OC调用Swift

```
# import "(moduleName)-Swift.h"
```

- Swift调用OC:

  动态库:

```
import moduleName
```

  静态库:

```
moduleName-Bridging-Header.h
```

  中引入OC头文件

Swift基础 — 使用C指针

```
const Type *    UnsafePointer<Type>
Type *          UnsafeMutablePointer<Type>
```

- int，bool 和 char 的对应 类型分别是 CInt，CBool 和 CChar
- 内存需要自己管理

Swift基础 — **泛型**

- 泛型代码可以让你写出根据自我需求定义、适用于任何类型的，灵活且可重用的函数和类型。它可以让你避免重复的代码，用一种清晰和抽象的方式来表达代码的意图。
- Swift 的数组和字典类型都是泛型集。

Swift基础 — **异常处理**

- ErrorType
- 同步 API 使用异常机制: do try catch throw throws
- 异步 API 使用泛型枚举

```
enum Result<T> {
    case Success(T)
    case Failure(NSError)
}
```

Swift常用技巧

Swift常用技巧

- guard
- 属性观察
- Extensions
- 协议扩展
- map flatMap filter
- 单例

Swift常用技巧 — **guard**

- 可以把guard近似的看做是Assert，但是你可以优雅的退出而非崩溃
- guard中解包得得到值可以用于后面的代码

Swift常用技巧 — **属性观察**

- willSet didSet
- 初始化对象并不会触发

Swift常用技巧 — 属性观察

```
@IBOutlet weak var myLabel: UILabel! {
      didSet {
         myLabel.textColor = UIColor.purpleColor()
      }
}
```

- 同样可以设置 `NSLayoutConstraint` 的 `constant`

Swift常用技巧 — **Extensions**

扩展：就是向一个已有的类、结构体、枚举类型或者协议类型添加新功能。这包括在没有权限获取原始源代码的情况下扩展类型的能力（即逆向建模）

Swift常用技巧 — **协议扩展**

在Swift 2.0中，可以对协议进行属性或者方法的扩展，和扩展类与结构体类似。这让我们开启了面向协议编程的篇章。

Swift常用技巧 — **map flatMap filter**

- map：得到一个由闭包里面的返回值组成的新序列
- flatMap：与map类似的功能,但是会过滤掉返回值里面的nil值
- Filter：得到一个由闭包返回值为true的值组成的新序列

Swift常用技巧 — **单例**

单例：表示一个类在系统中只有一个实例对象。通过全局的一个入口点对这个实例对象进行访问。

使用CocoaPods管理framework

使用CocoaPods管理framework

- framework介绍
- 管理framework

使用CocoaPods管理framework — framework介绍

- iOS 8后苹果开放了framework,也就是动态库的功能
- 和静态库在编译时和app代码链接并打进同一个二进制包中不同，动态库可以在运行时手动加载，这样就可以做很多事情

    1.应用插件化

    2.软件版本实时模块升级

    3.共享可执行文件(仅可用于App Extension)

使用CocoaPods管理framework — **管理framework**

- use_frameworks!
- CocoaPods生成一个动态框架，其中包含了所有pods而非一个静态库
- import moduleName

Swift图书展示项目开发实战(一)

本套课程中我们学习了Swift和CocoaPods。你应当掌握了以下知识：

- 熟悉项目的大体功能
- 熟悉Swift的一些基本知识和开发技巧
- 熟练运用CocoaPods管理项目的framework

如果想继续提高，你可以继续在极客学院学习后续的实战部分课程。

极客学院

jikexueyuan.com

中国最大的IT职业在线教育平台