智捷iOS课堂

iOS中的商业模式-应用内购买

主要内容

- 概述
- 测试环境搭建
- 在程序中添加实现应用内购买
- 测试应用内购买

概述

应用内购买产品类型

- 消耗性，产品购买之后即被消费，再次购买该产品时还需要支付，只能应用于当前设备。
- 非消耗性，该类产品一旦购买可以一直使用，而且可以在与该用户账号关联的多个设备上使用。App Store会保留用户的购买记录。
- 订阅类，订阅类产品在订阅周期内如同非消费型购买一样，在订阅期过后如消费型购买一样。作为开发者需要确保用户订阅的内容在其iTunes同步的设备上都有效，可以在程序内部加入自己的订阅计划更新机制。苹果期望订阅类产品可以通过外部服务器交付。另外，订阅类产品可以在与该用户账号关联的多个设备上使用。订阅类产品又可以细分为：自动再生订阅类、自动再生订阅类和免费订阅类。

交付模式

- 内置产品类型，需要交付的产品已经在程序内部，通常用于一些功能的锁定，这些功能原本是在程序中，但是需要购买这些功能才能解锁，开发人员需要记录这些购买记录，并且能够备份和恢复这些信息。它的优点是能很快交付产品给客户，大多数的内置产品应用为非消耗性产品。这种模式是我们本书重点介绍的模式。
- 服务器产品类型，该种模式下需要开发商或运行商提供另外的服务器，将要交付的内容、服务和订阅的产品更新到服务器上。应用程序与服务器和App Store交互获取信息。这种方式非常灵活，但是投入比较大，适合于订阅、内容和服务类产品。

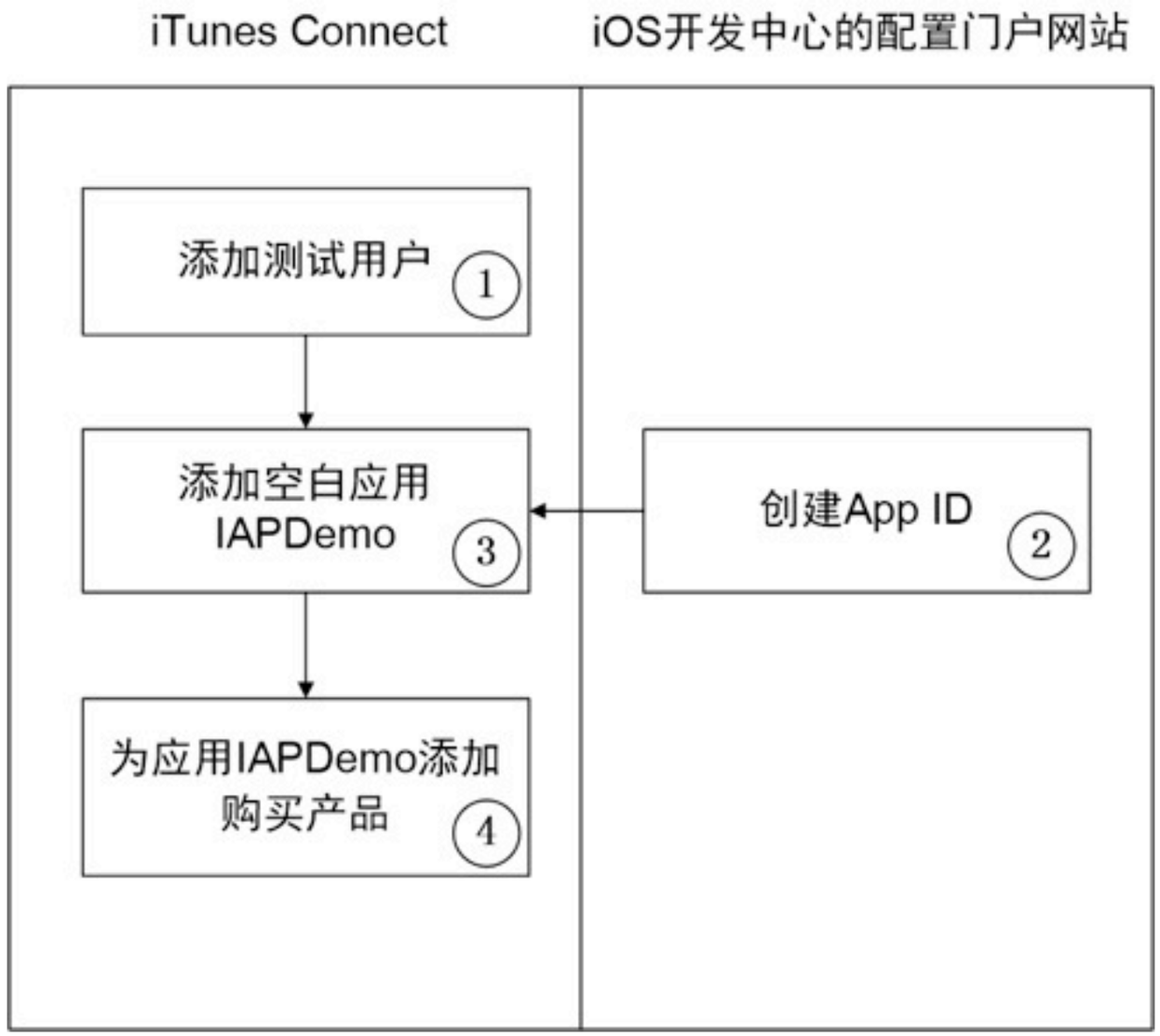
应用内购买案例

国际化与本地化是相反的过程


CMCC
3:30
恢复
应用内购买
Maps
¥6.00
Tool
¥6.00

测试环境搭建

应用内购买测试环境搭建流程


iTunes Connect
iOS开发中心的配置门户网站
添加测试用户
1
添加空白应用
IAPDemo
3
创建App ID
2
为应用IAPDemo添加
购买产品
4

- 添加测试用户
- 创建App ID
- 添加空白应用IAPDemo
- 为应用IAPDemo添加购买产品

在程序中添加实现应用内购买

创建工程、初始化处理


恢复
应用内购买
Navigation Controller
Table View
Prototype Content
View Controller – 应用内购买

ViewController.h

```
#import <UIKit/UIKit.h>
#import <StoreKit/StoreKit.h>

@interface ViewController : UITableViewController
<SKProductsRequestDelegate,SKPaymentTransactionObserver>

//点击刷新按钮
- (IBAction) request:(id)sender;
//点击恢复按钮
- (IBAction)restore:(id)sender;
```

为工程添加必要的框架StoreKit.framework

```
//刷新按钮属性
@property (weak, nonatomic) IBOutlet UIBarButtonItem *refreshButton;
//恢复按钮属性
@property (weak, nonatomic) IBOutlet UIBarButtonItem *restoreButton;
//产品列表
@property (nonatomic,strong) NSArray* skProducts;
//数字格式
@property (nonatomic,strong) NSNumberFormatter * priceFormatter;
//产品标识集合
@property (nonatomic,strong) NSSet * productIdentifiers;

@end
```

ViewController.m

```
- (void)viewDidLoad
{
    [super viewDidLoad];

    //设置数字格式
   self.priceFormatter = [[NSNumberFormatter alloc] init];
   [self.priceFormatter setFormatterBehavior:NSNumberFormatterBehavior10_4];
   [self.priceFormatter setNumberStyle:NSNumberFormatterCurrencyStyle];

    //从ProductIdentifiers.plist文件读取应用内产品标识
   NSString* path = [[NSBundle mainBundle]
                     pathForResource:@"ProductIdentifiers" ofType:@"plist"];
   NSArray* array = [[NSArray alloc] initWithContentsOfFile:path];
    //从NSArray转化为NSSet
   self.productIdentifiers = [[NSSet alloc] initWithArray:array];

    // 添加self作为交易观察者对象
   [[SKPaymentQueue defaultQueue] addTransactionObserver:self];

}
```

获得产品信息

request:方法

```
- (IBAction)request:(id)sender {
    //检查设备是否有家长控制，禁止应用内购买
    if ([SKPaymentQueue canMakePayments]) {
        //没有设置可以请求应用购买信息
        SKProductsRequest *request= [[SKProductsRequest alloc]
                  initWithProductIdentifiers:self.productIdentifiers];
        request.delegate = self;
        [request start];

        self.navigationItem.prompt = @"刷新中...";
        self.refreshButton.enabled = NO;
        self.restoreButton.enabled = NO;
    } else {
        //有设置情况下
        UIAlertView *alertView  = [[UIAlertView alloc]
         initWithTitle:@"访问限制" message:@"您不能应用内购买！" delegate:nil
                                cancelButtonTitle:@"Ok" otherButtonTitles: nil];
        [alertView show];
    }
}
```

家长控制设置


iPod
下午5:26
设置
通用
iTunes Wi-Fi 同步
Spotlight 搜索
自动锁定
3 分钟
密码锁定
关闭
访问限制
打开
日期与时间
键盘
多语言环境
iPod
下午1:51
通用
访问限制
电视节目
全部
应用程序
全部
应用程序内购买
要求密码
15 分钟
需要输入密码才能购买。
隐私:
定位服务
通讯录
日历
提醒事项

SKProductsRequestDelegate协议

```
- (void)productsRequest:(SKProductsRequest *)request didReceiveResponse:
(SKProductsResponse *)response
{
    NSLog(@"加载应用内购买产品...");

    self.navigationItem.prompt = nil;
    self.refreshButton.enabled = YES;
    self.restoreButton.enabled = YES;


    self.skProducts = response.products;
    for (SKProduct * skProduct in self.skProducts) {
        NSLog(@"找到产品: %@ %@ %0.2f",
              skProduct.productIdentifier,
              skProduct.localizedTitle,
              skProduct.price.floatValue);
    }

    [self.tableView reloadData];
}
```

UITableViewDataSource协议

```
- (UITableViewCell *)tableView:(UITableView *)tableView
cellForRowAtIndexPath:(NSIndexPath *)indexPath {

    ... ...

    int row = [indexPath row];
    SKProduct * product = self.skProducts[row];
     cell.textLabel.text = product.localizedTitle;
    [self.priceFormatter setLocale:product.priceLocale];
    cell.detailTextLabel.text
                = [self.priceFormatter stringFromNumber:product.price];

//从应用设置文件中读取 购买信息
    BOOL productPurchased = [[NSUserDefaults standardUserDefaults]
boolForKey:product.productIdentifier];
    if (productPurchased) {
        cell.accessoryType = UITableViewCellAccessoryCheckmark;
        cell.accessoryView = nil;
    } else {
       ... ...<设置按钮>
    }
    return cell;
}
```

设置按钮

```
UIImage *buttonUpImage = [UIImage imageNamed:@"button_up.png"];
UIImage *buttonDownImage = [UIImage imageNamed:@"button_down.png"];

UIButton *button = [UIButton buttonWithType:UIButtonTypeCustom];
button.frame = CGRectMake(0.0f, 0.0f, buttonUpImage.size.width,
                                      buttonUpImage.size.height);
[button setBackgroundImage:buttonUpImage forState:UIControlStateNormal];
[button setBackgroundImage:buttonDownImage
                  forState:UIControlStateHighlighted];

[button setTitle:@"购买" forState:UIControlStateNormal];
button.tag = indexPath.row;

[button addTarget:self action:@selector(buttonTapped:)
                  forControlEvents:UIControlEventTouchUpInside];

cell.accessoryView = button;
```

buttonTapped:方法

```
- (void)buttonTapped:(id)sender {

    UIButton *buyButton = (UIButton *)sender;
    //通过按钮tag获得被点击按钮的索引，使用索引从数组中取出产品SKProduct对象
    SKProduct *product = self.skProducts[buyButton.tag];
    //获得产品的付款对象
    SKPayment * payment = [SKPayment paymentWithProduct:product];
    //把付款对象添加到付款队列中
    [[SKPaymentQueue defaultQueue] addPayment:payment];

}
```

处理交易结果

SKPaymentTransactionObserver协议

```
- (void)paymentQueue:(SKPaymentQueue *)queue updatedTransactions:(NSArray
*)transactions
{
    for (SKPaymentTransaction * transaction in transactions) {
        switch (transaction.transactionState)
        {
            case SKPaymentTransactionStatePurchased: //交易完成
                [self completeTransaction:transaction];
                break;
            case SKPaymentTransactionStateFailed:    //交易失败
                [self failedTransaction:transaction];
                break;
            case SKPaymentTransactionStateRestored:  //交易恢复
                [self restoreTransaction:transaction];
            default:
                break;
        }
    }

}
```

响应处理交易的方法

```
//交易完成
- (void)completeTransaction:(SKPaymentTransaction *)transaction {
    NSLog(@"交易完成...");
    [self provideContentForProductIdentifier:
                    transaction.payment.productIdentifier];
    //把交易从付款队列中移除
    [[SKPaymentQueue defaultQueue] finishTransaction:transaction];
}

//交易恢复
- (void)restoreTransaction:(SKPaymentTransaction *)transaction {
    NSLog(@"交易恢复...");

    self.navigationItem.prompt = nil;
    self.refreshButton.enabled = YES;
    self.restoreButton.enabled = YES;

    [self provideContentForProductIdentifier:
           transaction.originalTransaction.payment.productIdentifier];
    [[SKPaymentQueue defaultQueue] finishTransaction:transaction];
}
```

响应处理交易的方法

```
//交易失败
- (void)failedTransaction:(SKPaymentTransaction *)transaction
{

    NSLog(@"交易失败...");
    if (transaction.error.code != SKErrorPaymentCancelled)
    {
        NSLog(@"交易失败: %@", transaction.error.localizedDescription);
    }

    [[SKPaymentQueue defaultQueue]
                        finishTransaction: transaction];
}

//购买成功
- (void)provideContentForProductIdentifier:(NSString *)productIdentifier
{
    [[NSUserDefaults standardUserDefaults]
                          setBool:YES forKey:productIdentifier];
    [[NSUserDefaults standardUserDefaults] synchronize];
    [self.tableView reloadData];
}
```

恢复交易

restore:方法

```
- (IBAction)restore:(id)sender {
    self.navigationItem.prompt = @"恢复中...";
    self.refreshButton.enabled = NO;
    self.restoreButton.enabled = NO;
    [[SKPaymentQueue defaultQueue] restoreCompletedTransactions];
}
```

测试应用内购买

修改本地资源文件ProductIdentifiers.plist

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/
DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<array>
   <string>com.51work6.IAPDemo.map</string>
   <string>com.51work6.IAPDemo.tool</string>
   <string>com.51work6.IAPDemo.elves</string>
   <string>com.51work6.IAPDemo.Equipment</string>
</array>
</plist>
```

测试

谢谢