EBPurchase

Simple In-App Purchase for iOS

EBPurchase adds simple In-App Purchase functionality to your iOS app. It wraps all of the necessary code for interacting with the StoreKit framework into a convenient little class, and provides you with easy-to-use methods. EBPurchase is non-ARC and has been tested to work with iOS 4.3 and higher.

IMPORTANT NOTE: EBPurchase was NOT created to be an "everything but the kitchen sink" solution. This class provides very simple In-App Purchase and Restore functionality, and was designed for developers with basic In-App Purchase needs and beginners interested in learning how to use the StoreKit framework. If you require additional features like checking the status of auto-renewable subscriptions or processing multiple product purchases, then please feel free to extend/modify this open source code for your own specific needs.

If this code saves you some development time and effort, and you feel like reciprocating the favor, downloading my iOS apps or buying one of my books is always much appreciated.

Due to my busy work schedule, I'm unable to provide support for this open source offering, but the instructions below and the included example project should provide all the guidance you need.

How To Use

1. Setup your App Listing and In-App Purchase item in your iTunes Connect account. For details, please read Apple's In-App Purchase Programming Guide. You'll need to configure your Xcode project's info plist file with your App Listing's Bundle Identifier.

2. Open the EBPurchase-Demo Xcode project and take a look at the related code there as you follow along with these instructions.

3. Add the StoreKit.framework to your Xcode iOS project.

4. Add the EBPurchase.h and EBPurchase.m files to your Xcode iOS project.

5. Import EBPurchase.h into the view controller header file where you wish to use it, assign that view controller as a delegate of EBPurchaseDelegate, and then add an EBPurchase property.

6. In that view controller's implementation file, create a new instance of that EBPurchase property in the viewDidLoad event, and assign its delegate to self.

7. In that view controller's implementation file, you'll want to implement the delegate methods of EBPurchaseDelegate, such as requestedProduct, successfulPurchase, failedPurchase, incompleteRestore, and failedRestore. When you call an EBPurchase method, the responses from the StoreKit framework are returned to your app in those delegate methods.

8. To check if your In-App Purchase item is available for purchase and that the price reflects the user's regional currency, your buy button should be dynamically modified at runtime to be enabled (or disabled) and display a dynamic price. This can be done by calling EBPurchase's requestProduct method in your viewWillAppear event. You pass your In-App Purchase Product ID as an NSString to that method. The StoreKit's response is returned to EBPurchase's delegate method requestedProduct, where you can then modify your buy button with the StoreKit information.

9. Tapping the buy button should call EBPurchase's purchaseProduct method. Since you previously called the requestProduct method, if your In-App Purchase item is available, then you'll have access to it as an SKProduct object, which you'll pass as a parameter to the purchaseProduct method. If the purchase failed or the user cancelled the action, the StoreKit's response is returned to EBPurchase's delegate method failedPurchase. If the user's purchase was successful, the StoreKit's response is returned to EBPurchase's delegate method successfulPurchase, where you should then unlock the purchased content, update the app's stored settings to reflect the purchase status, and thank the user!

10. Need to restore a previous purchase of a non-consumable or subscription In-App Purchase item in a re-installed app or on the user's other iOS device? Tapping the restore button should call EBPurchase's restoreProduct method. If the restore failed or the user cancelled the action, the StoreKit's response is returned to EBPurchase's delegate method failedRestore. If the restore queue did not include any previous transactions, either the user has not yet made a purchase or the user's prior purchase is unavailable, so then StoreKit's response is returned to EBPurchase's delegate method incompleteRestore, where you can notify the user to make a purchase within the app. If the user previously purchased the item, they will not be re-charged again, but it should restore their purchase. If the user's restore was successful, the StoreKit's response is returned to EBPurchase's delegate method successfulPurchase, where you should then unlock the purchased content, update the app's stored settings to reflect the purchase status, and notify the user.

11. In-App Purchase must be tested on an iOS device. I've included NSLog lines throughout the demo project code, so that when you run the app, you'll be able to easily trace which methods are firing in Xcode's console. I've also included an extensive amount of comments in the project, explaining the purpose of each piece of code. Enjoy!

License

Copyright (c) 2011 Electric Butterfly, Inc. - http://www.ebutterfly.com/

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to redistribute it and use it in source and binary forms, with or without modification, subject to the following conditions:

- This Software may be used for any purpose, including commercial applications.

- This Software in source code form may be redistributed freely and must retain the above copyright notice, this list of conditions and the following disclaimer. Altered source versions must be plainly marked as such, and must not be misrepresented as being the original Software.

- Neither the name of the author nor the name of the author's company may be used to endorse or promote products derived from this Software without specific prior written permission.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.