

# Android Pebbleプラグイン Buildマニュアル Device Connect 1.0対応版

Version 1.0

1. Android PebbleのBuildに必要なパッケージ	3
2. Pebble SDKのインストール手順	3
3. Android プロジェクトのImport手順	3
3.1 android-support-v7-appcompat のImport	4
3.2 dConnectSDKAndroidのImport	6
3.3 dConnectDevicePluginSDKのImport	8
3.4 PebbleKit(PEBBLE_KITプロジェクト)のImport	11
3.5 Android PebbleプラグインのImport	13
3.6 PebbleAppのImport	16
4. PebbleAppのビルドと、Android プロジェクトへの登録	18
4.1 Androidの設定	18
4.2 PebbleAppのビルド	19
4.3 dConnectDevicePebbleプロジェクトへの登録	20
5. 通信プログラム作成時の注意点	21
5.1 Pebble Cプログラム作成上の注意	21
5.2 通信プログラム作成上の注意(Android側)	21
5.3 通信プログラム作成上の注意(Pebble側)	22
付録A: 更新履歴	24

# 1. Android PebbleのBuildに必要なパッケージ

Android PebbleプラグインのBuildに必要なパッケージは以下の通りである。

android-support-v7-appcompat	Android コンパティビリティー
dConnectDevicePluginSDK	デバイスプラグイン用のSDK。dConnectSDKAndroidをライブラリとして参照。
dConnectSDKAndroid	Androidに関連する部分のSDK
Pebble SDK	Pebble社が提供するSDK
dConnectDevicePebble	Android Pebbleプラグイン
PebbleApp	Pebble側アプリケーション

## 2. Pebble SDKのインストール手順

ターミナルより

```
curl -sSL https://developer.getpebble.com/install.sh | sh && source ~/.bash_profile
```

を入力する。shell は bash を使うこと。

以上で、Android側で使用される PebbleKit、Pebble側の開発環境がそろう。

## 3. Android プロジェクトのImport手順

AndroidプロジェクトのImportは下記の手順でおこなう。

3.1 android-support-v7-appcompat のImport

3.2 dConnectSDKAndroid のImport

3.3 dConnectDevicePluginSDK のImport

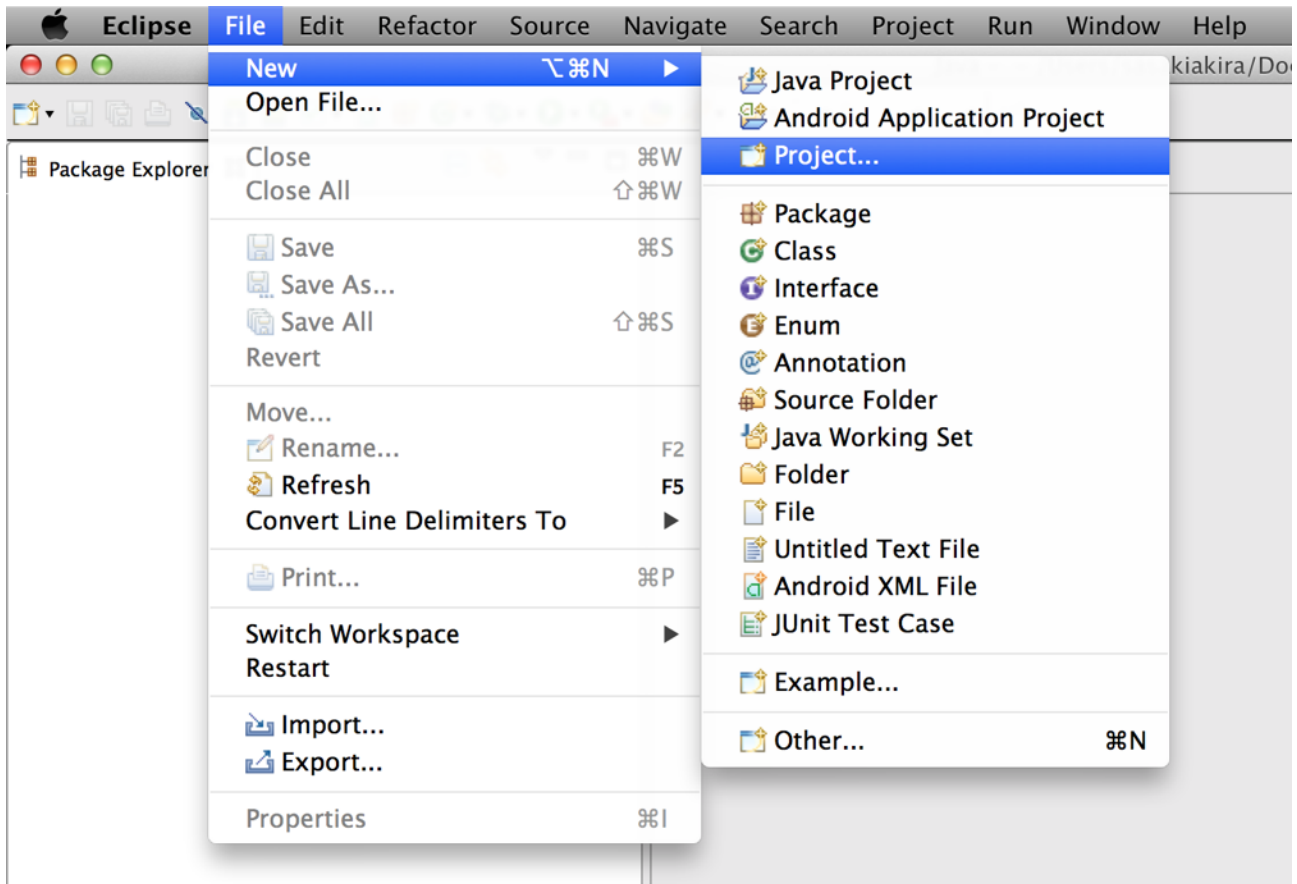
3.4 PebbleKit のImport

3.5 Android Pebbleプラグイン のImport

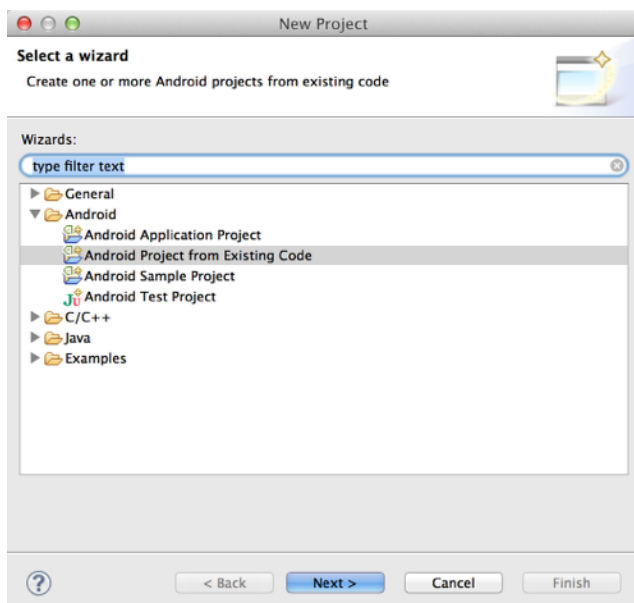
3.6 PebbleApp のImport(Pebble側アプリケーション)

## 3.1 android-support-v7-appcompat のImport

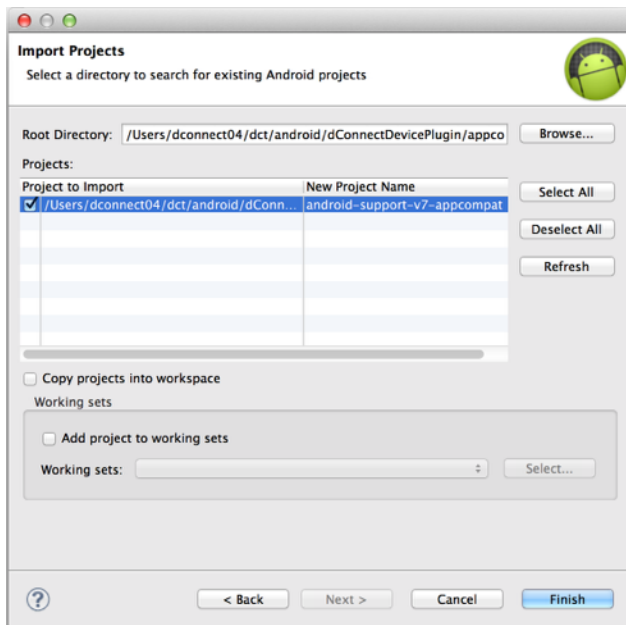
Eclipseのメニューから、[File]-[New]-[Project...]を選択する。



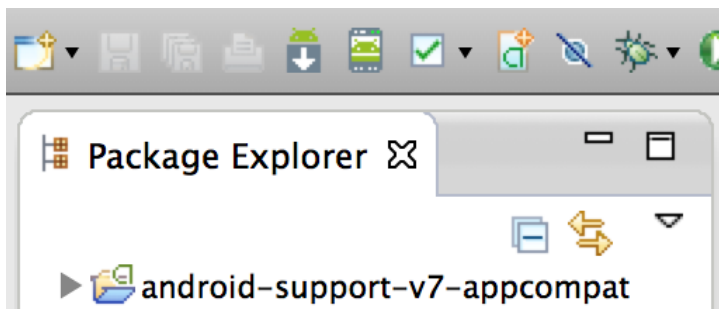
Android Project from Existing Codeを選択する。



android-support-v7-appcompat を選択し、Finishを押す。

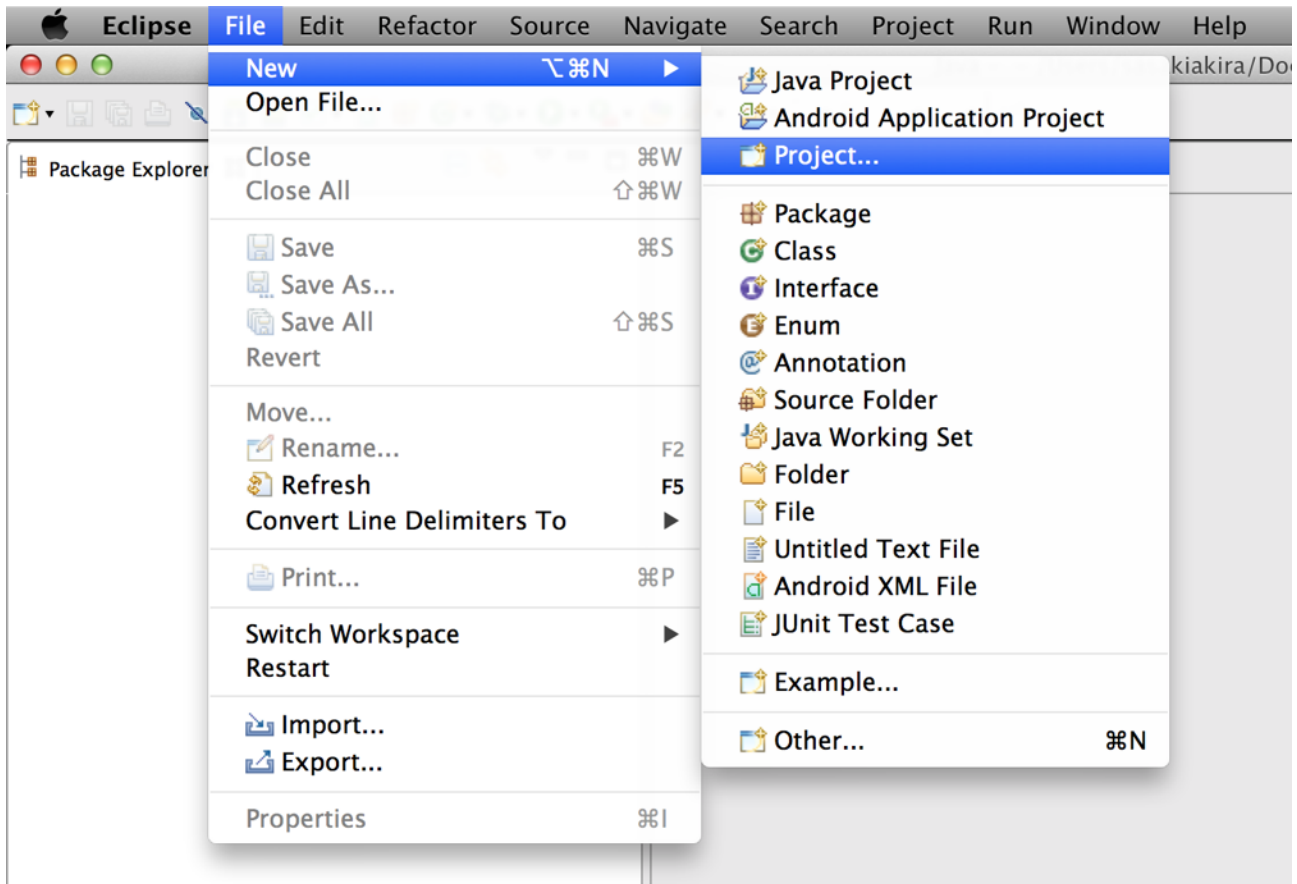


EclipseのPackage Explorerに、android-support-v7-appcompat が追加される。

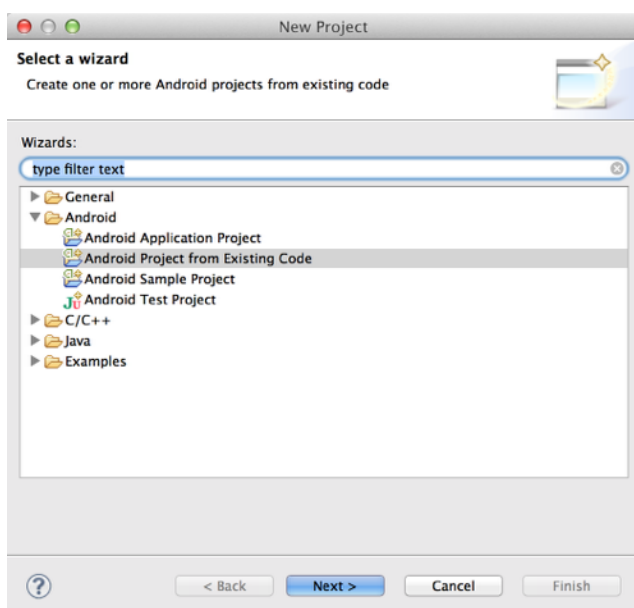


## 3.2 dConnectSDKAndroidのImport

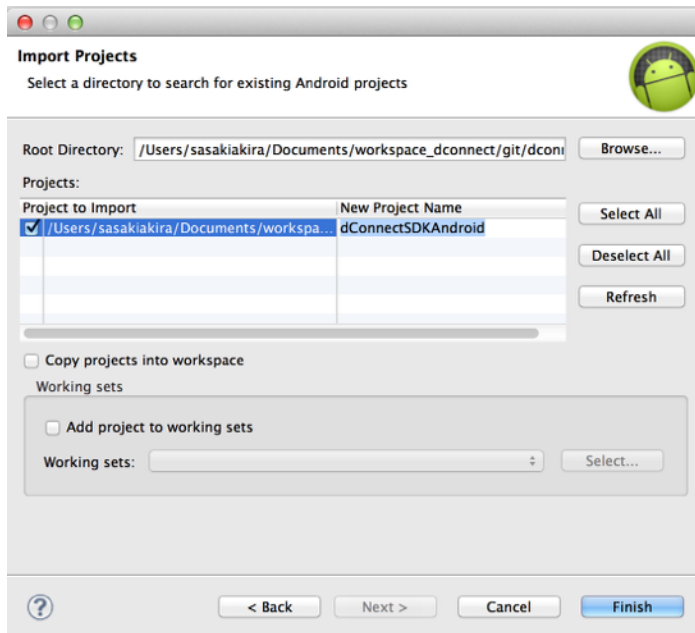
Eclipseのメニューから、[File]-[New]-[Project..]を選択する。



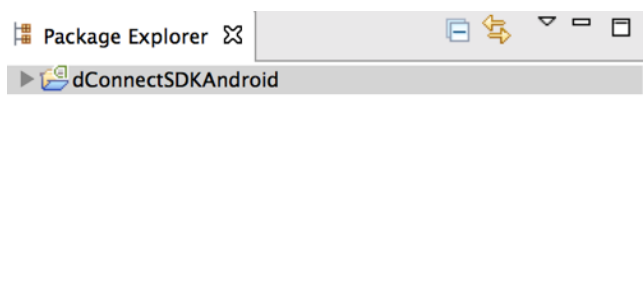
Android Project from Existing Codeを選択する。



dConnectSDKAndroidを選択し、Finishを押す。



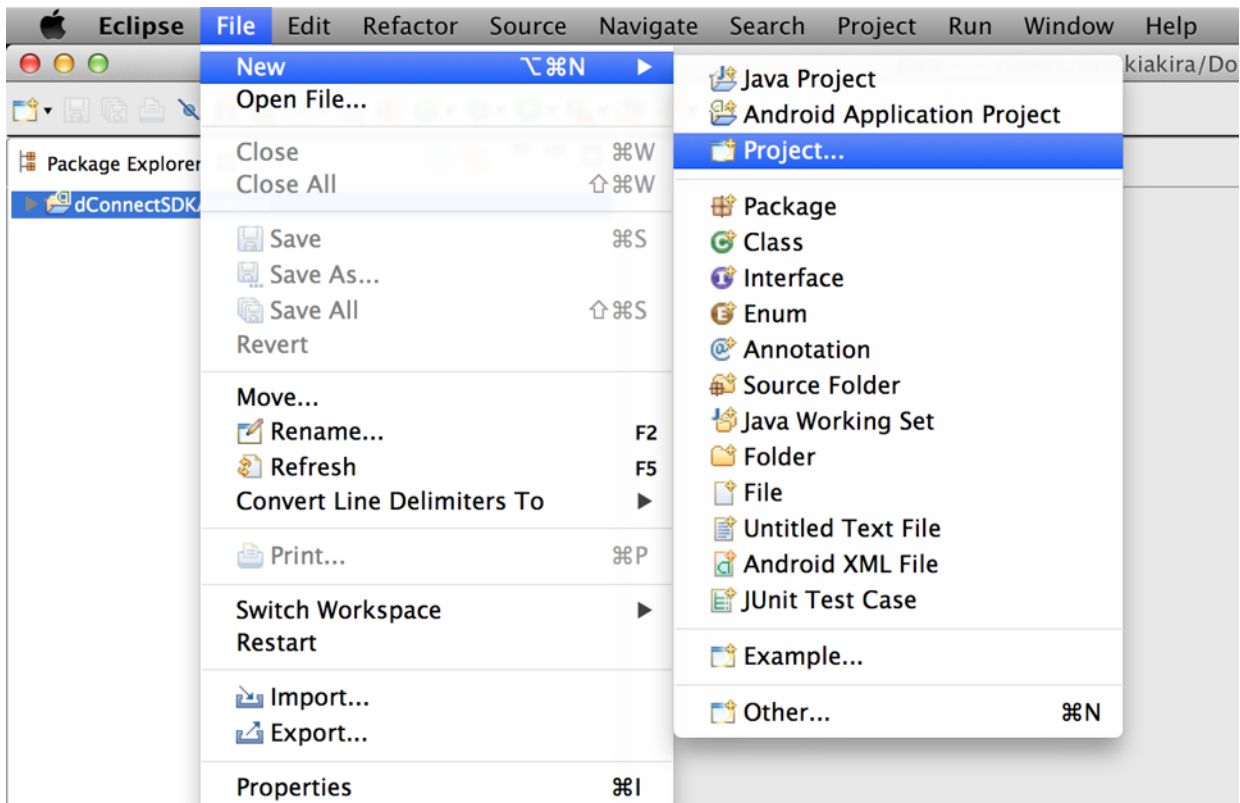
EclipseのPackage Explorerに、dConnectSDKAndroidが追加される。



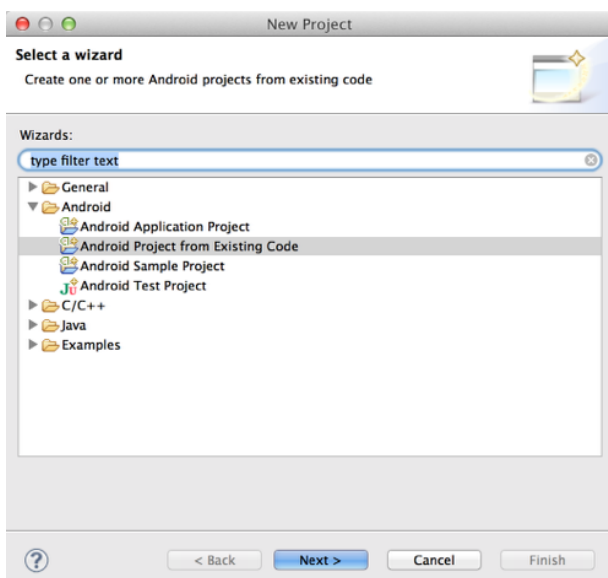
### 3.3 dConnectDevicePluginSDKのImport

次に、dConnectDevicePluginSDKをImportする。dConnectDevicePluginSDKは、dConnectSDKAndroidをライブラリとして参照する。

Eclipseのメニューから、[File]-[New]-[Project..]を選択する。

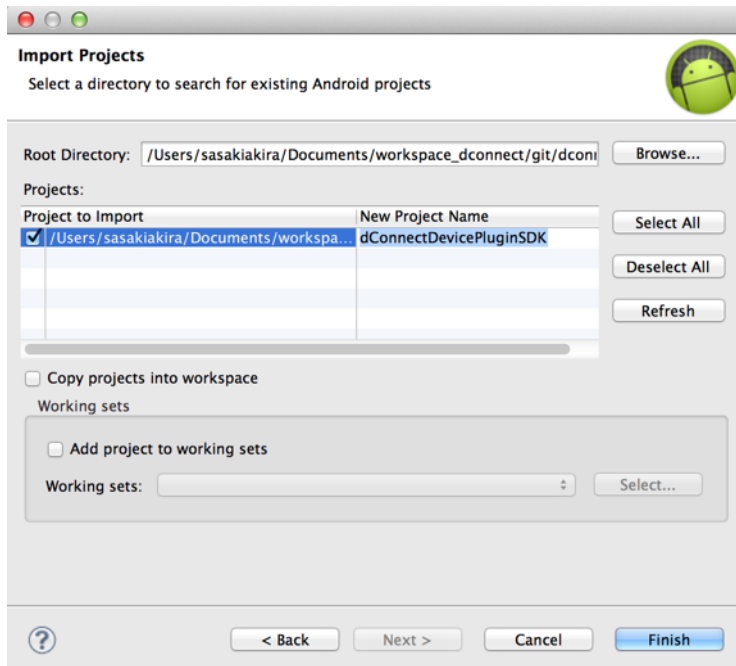


Android Project from Existing Codeを選択する。

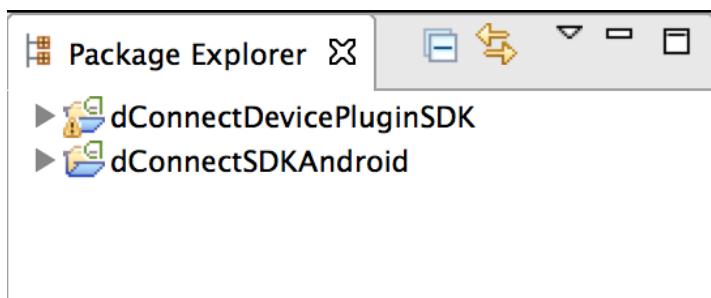




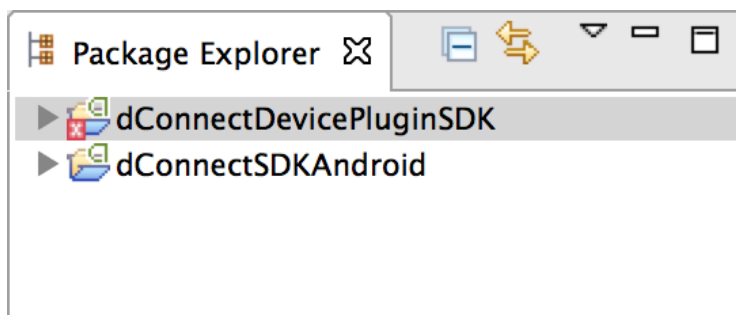
dConnectDevicePluginSDKを選択し、Finishを押す。



dConnectDevicePluginSDKがImportされます。

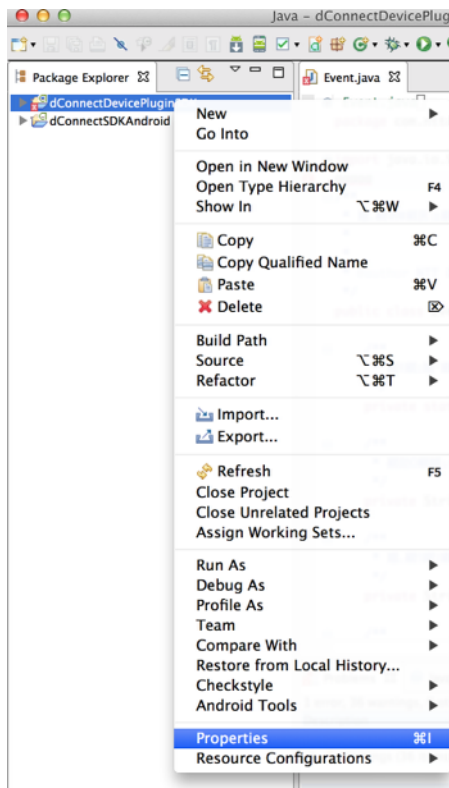


もし、エラーが消えない場合は、dConnectSDKAndroidの参照先が正しいか確認する。

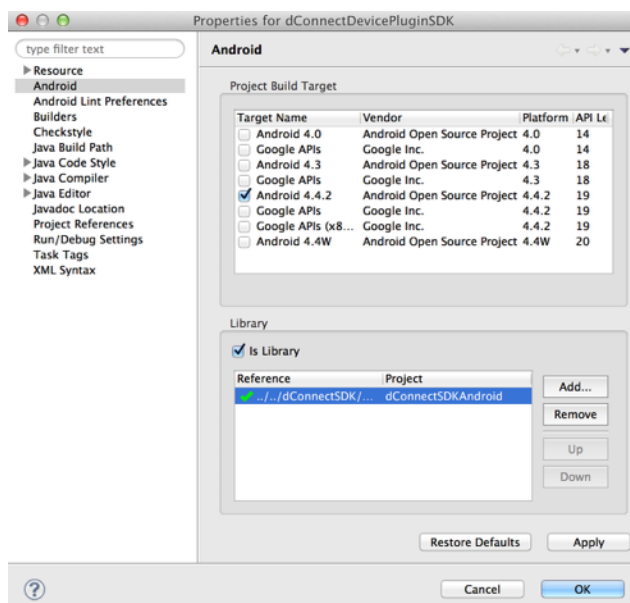


## エラーが消えない場合の対処方法

dConnectDevicePluginSDKの上で、右クリックを押し、ショートカットメニューを表示し、Propertiesを選択する。



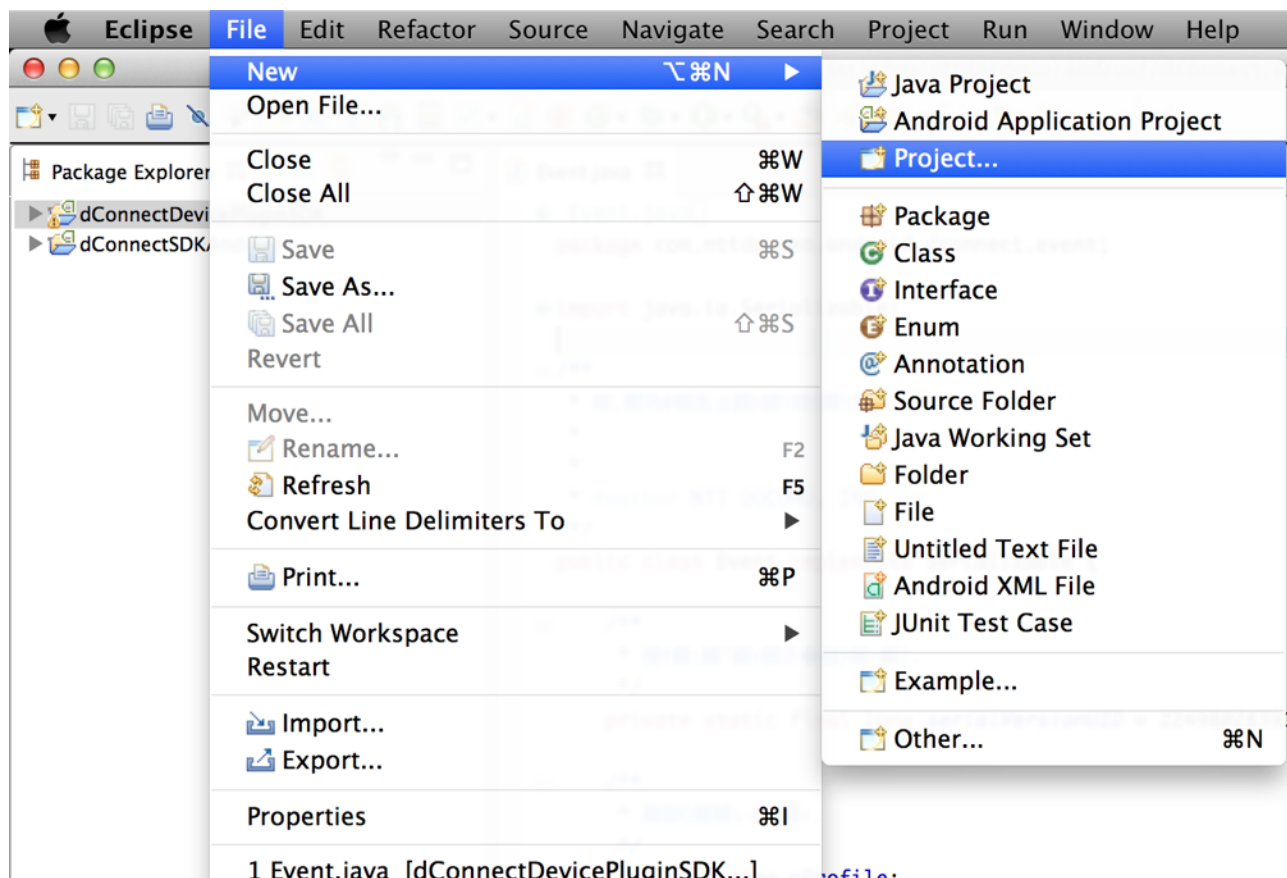
Androidの項目の、Is Libraryのチェックマークと、参照先のdConnectSDKAndroid指定フォルダが正しいか確認する。



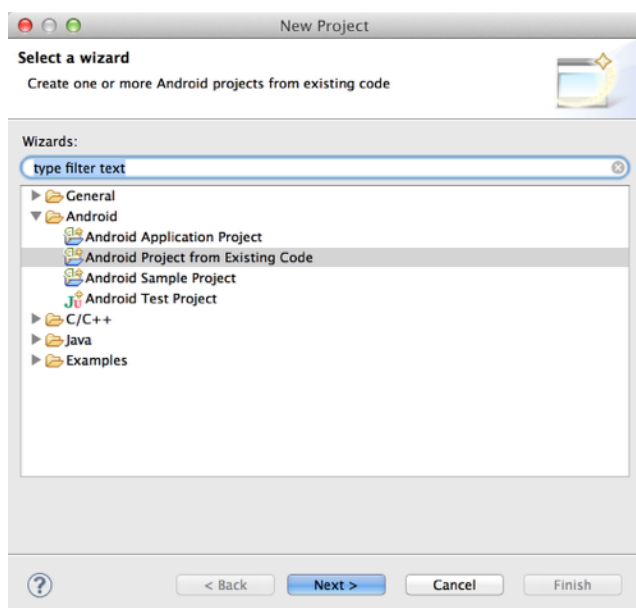
## 3.4 PebbleKit(PEBBLE\_KITプロジェクト)のImport

PEBBLE\_KIT をImportする。

Eclipseのメニューから、[File]-[New]-[Project..]を選択する。

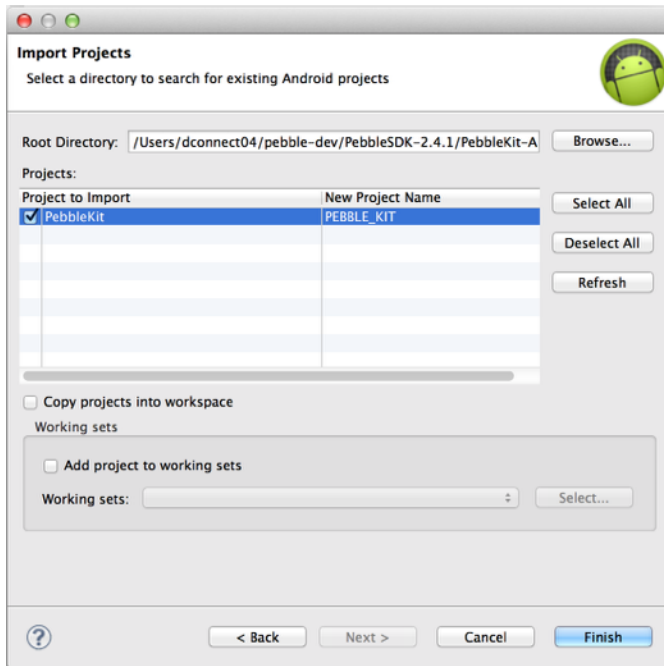


Android Project from Existing Codeを選択する。

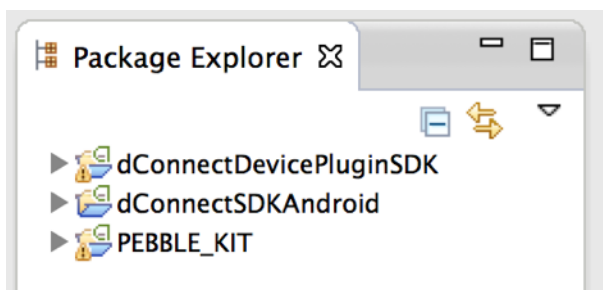


PEBBLE\_KITを選択し、Finishを押す。

Root Directory は標準で ~/pebble-dev/PebbleSDK-2.4.1/PebbleKit-Android/PebbleKit を選択する。



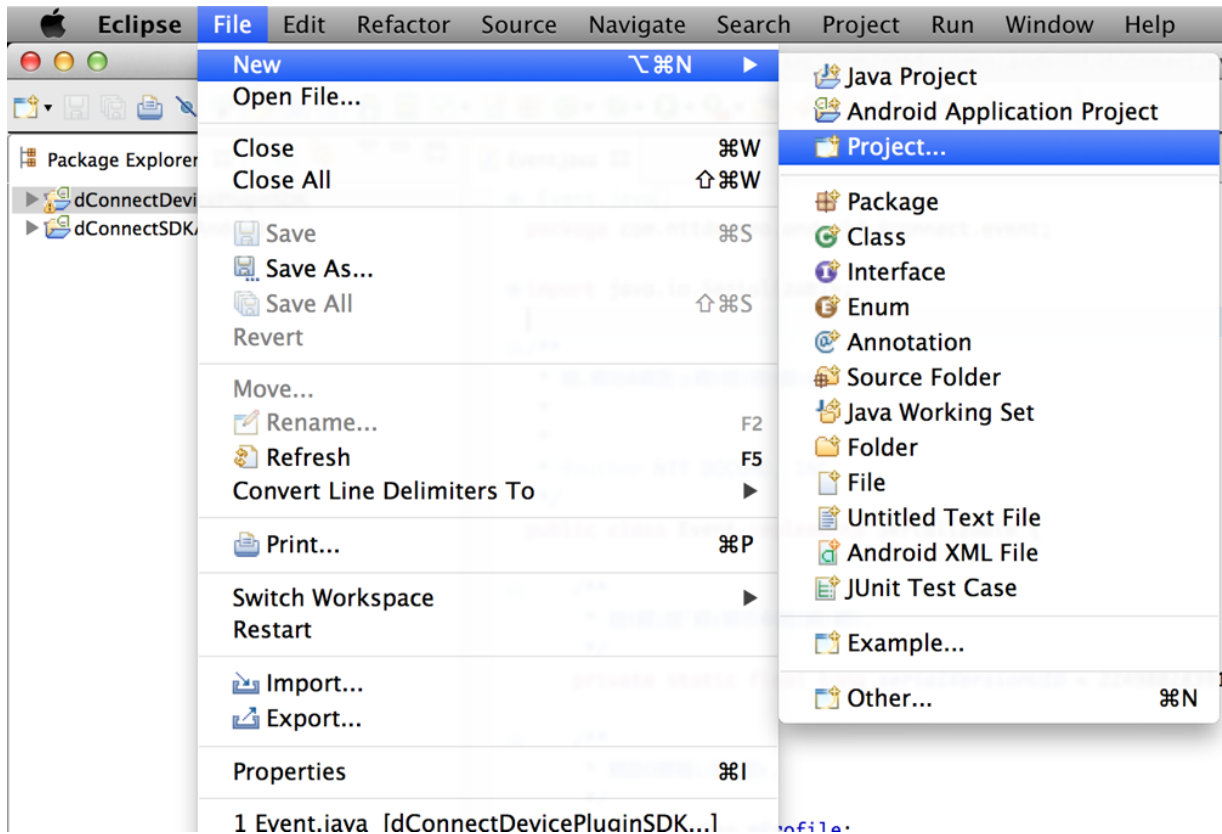
PEBBLE\_KITがImportされる。



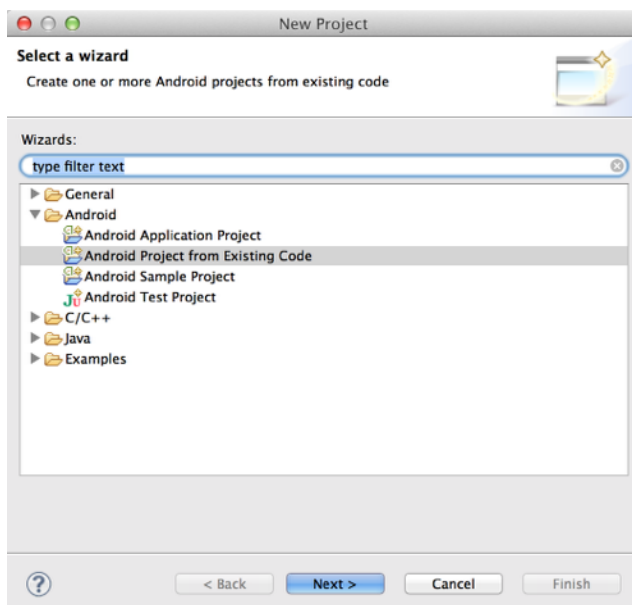
## 3.5 Android PebbleプラグインのImport

Android PebbleプラグインをImportする。

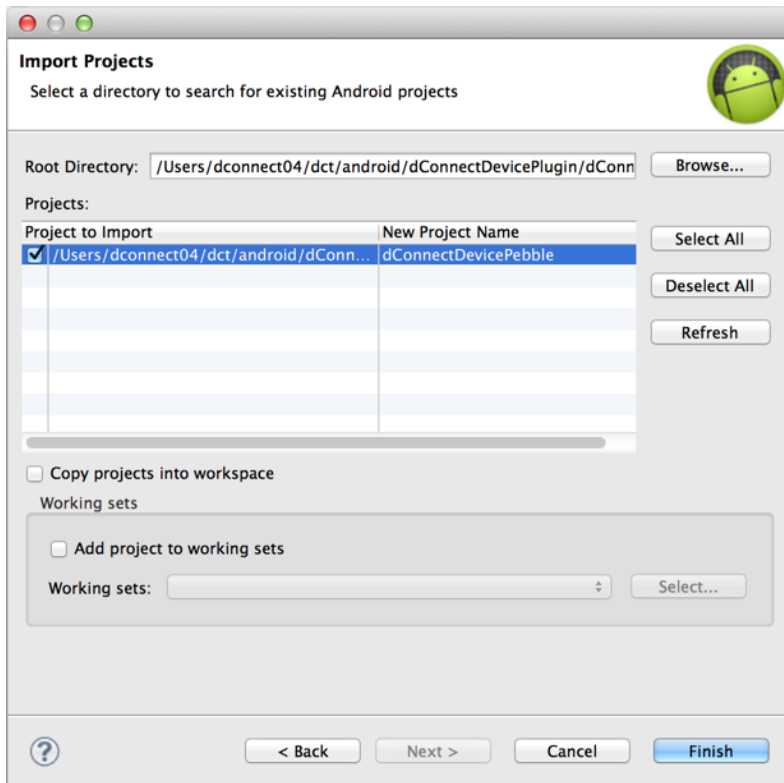
Eclipseのメニューから、[File]-[New]-[Project...]を選択する。



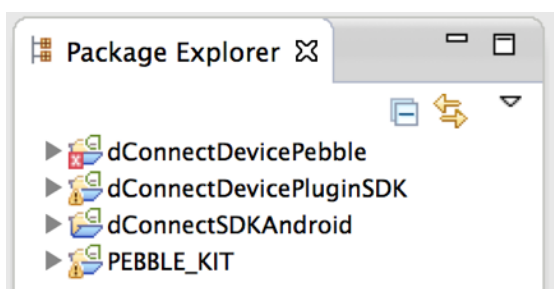
Android Project from Existing Codeを選択する。



dConnectDevicePebbleを選択し、Finishを押す。



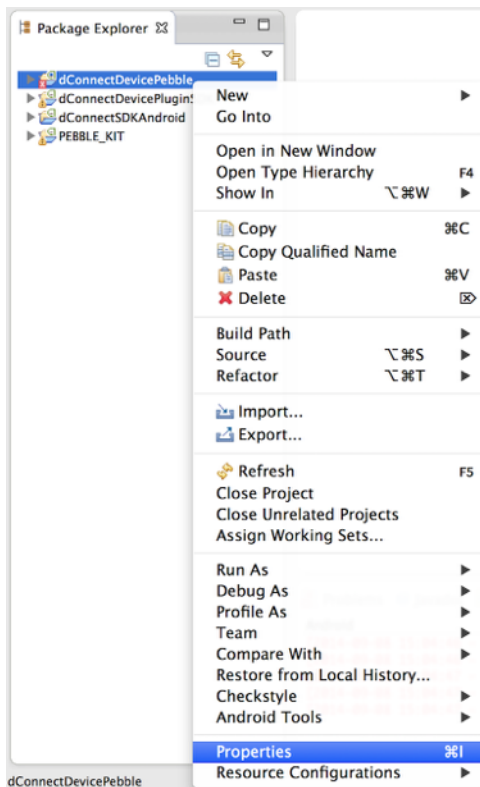
dConnectDevicePebbleがImportされる。



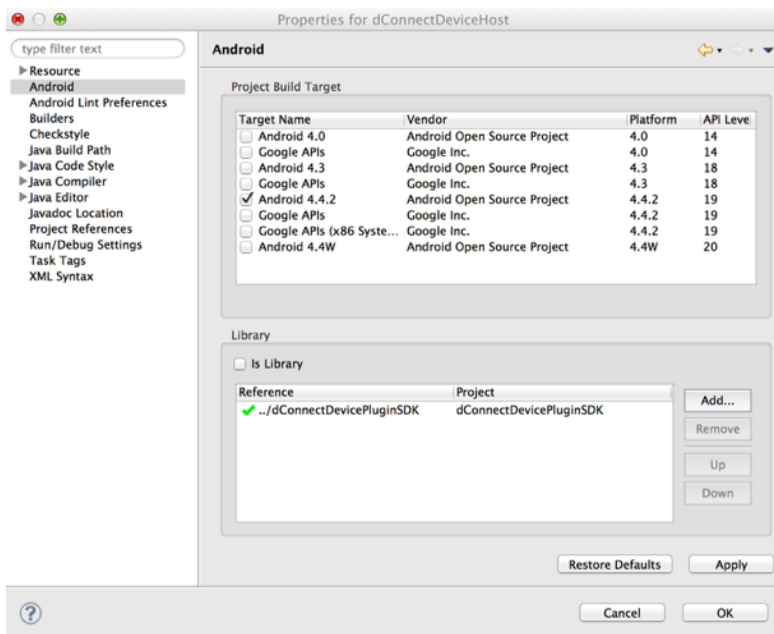
もし、エラーが消えない場合は、dConnectSDKAndroidの参照先が正しいか確認する。

## エラーが消えない場合の対処方法

dConnectDevicePebbleの上で、右クリックを押し、ショートカットメニューを表示し、Propertiesを選択する。

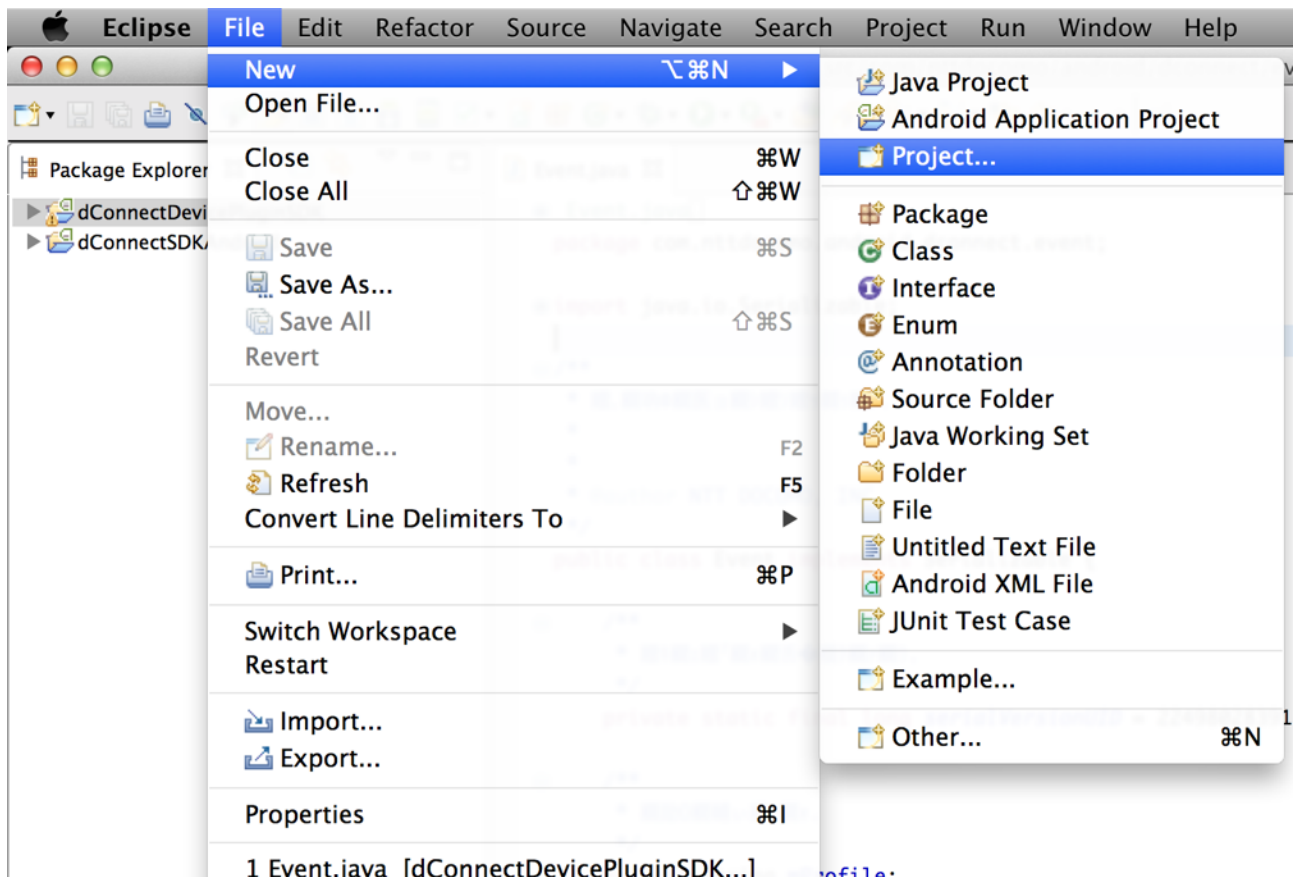


Androidの項目の、Is Libraryのチェックマークと、参照先のdConnectPluginSDK指定フォルダが正しいか確認する。

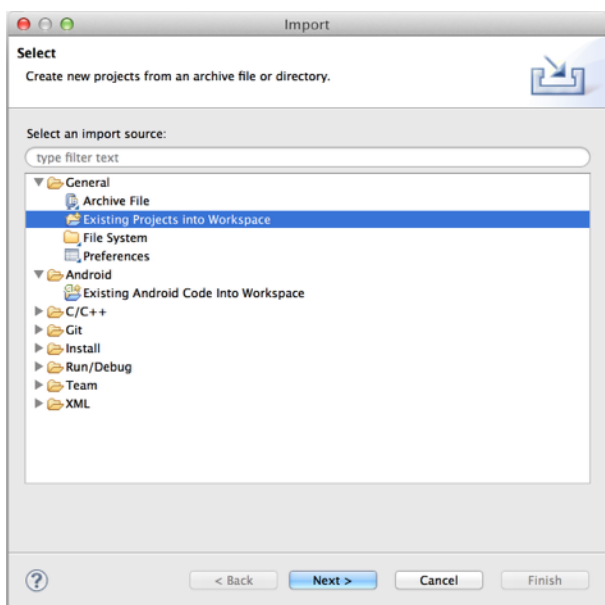


## 3.6 PebbleAppのImport

最後に、Pebble 側のアプリケーションのプロジェクトをインポートする。  
Eclipseのメニューから、[File]-[New]-[Project..]を選択する。

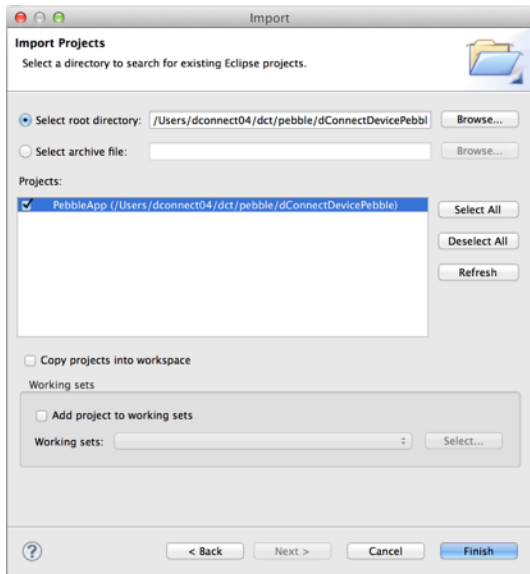


Existing Projects into Workspaceを選択する。

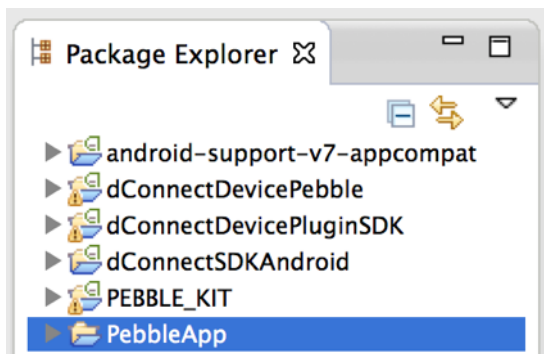




PebbleAppを選択し、Finishを押す。



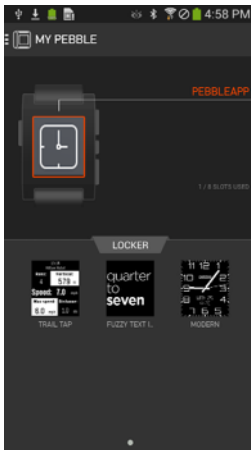
PebbleAppがImportされる。



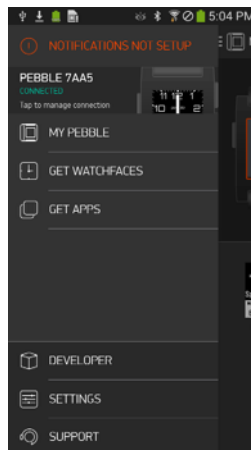
## 4. PebbleAppのビルドと、Android プロジェクトへの登録

### 4.1 Androidの設定

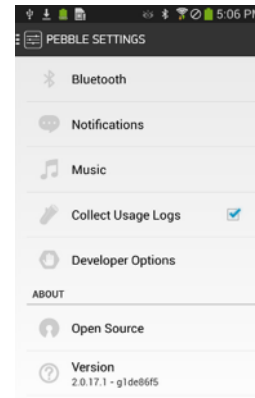
Google Play からダウンロードした Pebble アプリにて、開発用の設定を行う必要がある。  
Pebble 本体にプログラムをインストールする前に、Pebble アプリで必要な設定を行うこと。  
設定時に表示される IPアドレスは、Pebble 本体へのプログラムアップロード・デバッグに使用する。



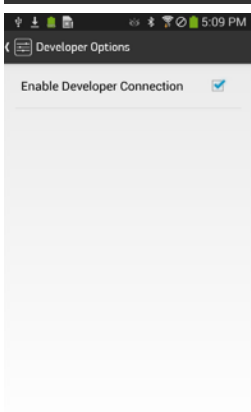
MY PEBBLE  
をタップ



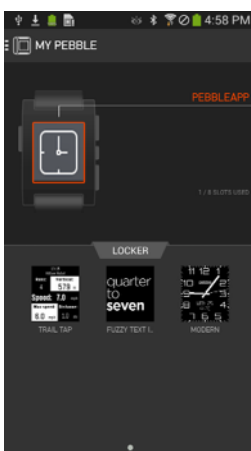
SETTINGS  
をタップを



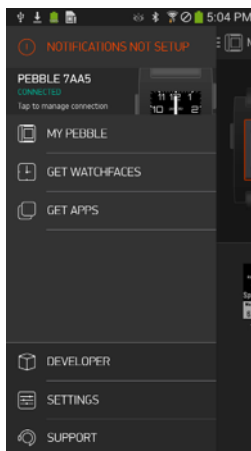
Developer...  
タップ



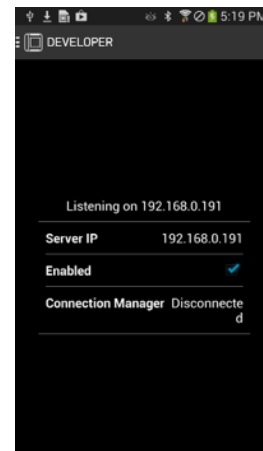
Enable Developer Connectionにチェック



MY PEBBLE  
をタップ



DEVELOPER  
をタップ



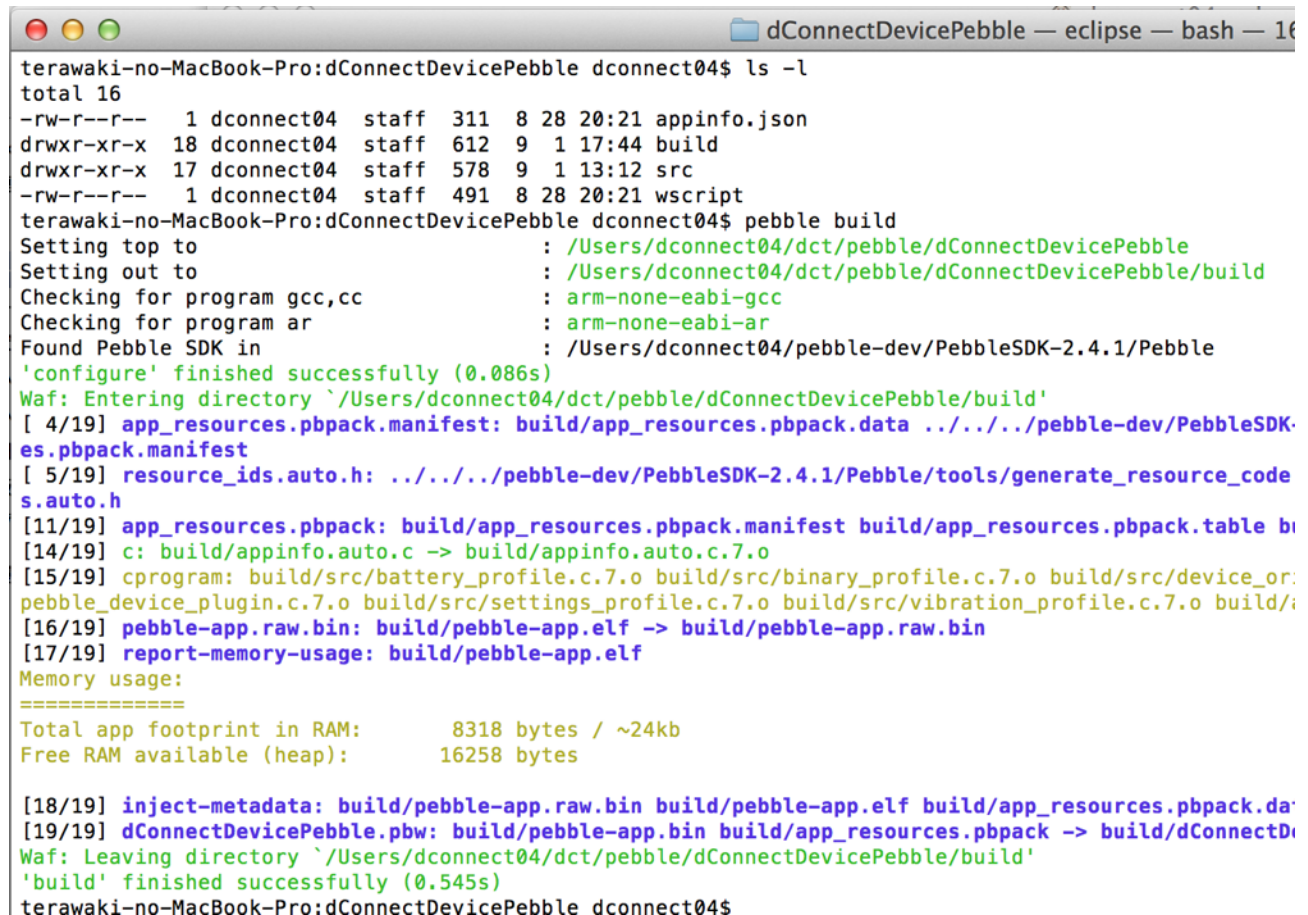
Enabled にチェック

## 4.2 PebbleAppのビルド

PebbleAppは、ターミナル上でビルドする。

src ディレクトリが存在するディレクトリにて、以下を実行する。実行ファイルは build/dConnectDevicePebble.pbw となる。

```
pebble build
```



```
terawaki-no-MacBook-Pro:dConnectDevicePebble dconnect04$ ls -l
total 16
-rw-r--r--  1 dconnect04  staff  311  8 28 20:21 appinfo.json
drwxr-xr-x  18 dconnect04  staff  612  9  1 17:44 build
drwxr-xr-x  17 dconnect04  staff  578  9  1 13:12 src
-rw-r--r--  1 dconnect04  staff  491  8 28 20:21 wscript
terawaki-no-MacBook-Pro:dConnectDevicePebble dconnect04$ pebble build
Setting top to                : /Users/dconnect04/dct/pebble/dConnectDevicePebble
Setting out to                : /Users/dconnect04/dct/pebble/dConnectDevicePebble/build
Checking for program gcc,cc   : arm-none-eabi-gcc
Checking for program ar       : arm-none-eabi-ar
Found Pebble SDK in          : /Users/dconnect04/pebble-dev/PebbleSDK-2.4.1/Pebble
'configure' finished successfully (0.086s)
Waf: Entering directory `/Users/dconnect04/dct/pebble/dConnectDevicePebble/build'
[ 4/19] app_resources.pbpack.manifest: build/app_resources.pbpack.data ../../../../pebble-dev/PebbleSDK-2.4.1/app_resources.pbpack.manifest
[ 5/19] resource_ids.auto.h: ../../../../pebble-dev/PebbleSDK-2.4.1/Pebble/tools/generate_resource_code_s.auto.h
[11/19] app_resources.pbpack: build/app_resources.pbpack.manifest build/app_resources.pbpack.table build/app_resources.pbpack.data
[14/19] c: build/appinfo.auto.c -> build/appinfo.auto.c.7.o
[15/19] cprogram: build/src/battery_profile.c.7.o build/src/binary_profile.c.7.o build/src/device_or_pebble_device_plugin.c.7.o build/src/settings_profile.c.7.o build/src/vibration_profile.c.7.o build/src/wscript
[16/19] pebble-app.raw.bin: build/pebble-app.elf -> build/pebble-app.raw.bin
[17/19] report-memory-usage: build/pebble-app.elf
Memory usage:
=====
Total app footprint in RAM:      8318 bytes / ~24kb
Free RAM available (heap):      16258 bytes

[18/19] inject-metadata: build/pebble-app.raw.bin build/pebble-app.elf build/app_resources.pbpack.data
[19/19] dConnectDevicePebble.pbw: build/pebble-app.bin build/app_resources.pbpack -> build/dConnectDevicePebble.pbw
Waf: Leaving directory `/Users/dconnect04/dct/pebble/dConnectDevicePebble/build'
'build' finished successfully (0.545s)
terawaki-no-MacBook-Pro:dConnectDevicePebble dconnect04$
```

全てを build しなおしたい場合には、以下を実行したのち、再度ビルドする。

```
pebble clean
```

デバッグ時、Build したプログラムをPebbleにダウンロードするには、以下のコマンドを実行しする。「4.1 Androidの設定」で取得した IP アドレスを指定する。

```
pebble install --phone 192.168.0.191
```

インストールと、その直後にデバッグログ表示を行う場合には、以下のコマンドを実行する。

```
pebble install --phone 192.168.0.191 --logs
```

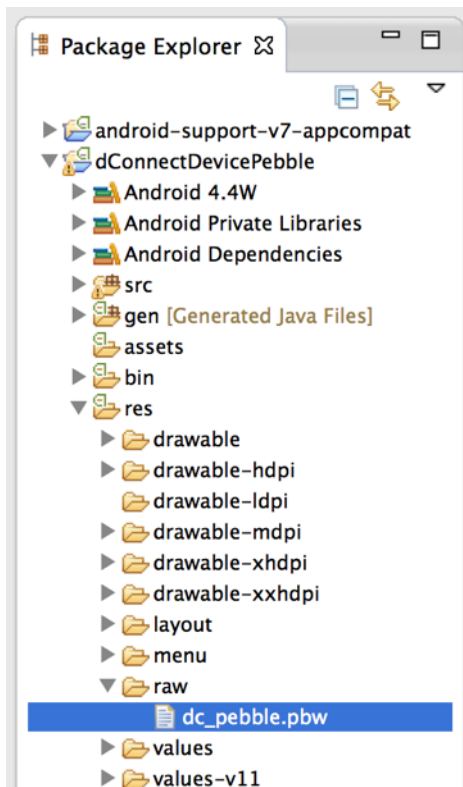
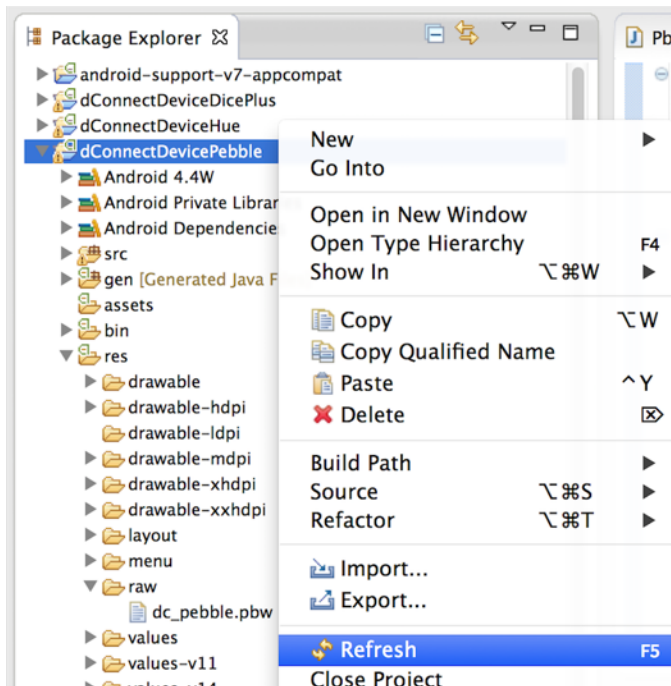
## 4.3 dConnectDevicePebbleプロジェクトへの登録

dConnectDevicePebble.pbw をコピーする。

ターミナルから、cp コマンドにてコピーする。以下は cp コマンドの例である。

```
cp PebbleApp/build/dConnectDevicePebble.pbw dConnectDevicePlugin/dConnectDevicePebble/res/raw/dc_pebble.pbw
```

コピー後には、プロジェクトをリフレッシュして、再ビルドする。



## 5. 通信プログラム作成時の注意点

### 5.1 Pebble Cプログラム作成上の注意

#### (1)使用できない標準関数

toa() strtok\_r() 等は用意されていない。  
strtok() 等はコンパイルは通るが、実行時エラーが発生する。  
strtok()等、内部で static 変数を使っている標準関数は使えない可能性が高い。  
itoa()のかわりに、snprintf(buf, sizeof(buf), "%d", data); を使う。

#### (2)Pebble のプログラムに ctype.h で定義されている isdigit() isalpha() 等

これらの関数のどれか1つでも使うと、約300byte プログラムサイズが増える。  
現プログラムでは、IsDigit()等の自作関数を使用している。

#### (3)コードサイズ

text/data/stack の領域の合計は、24Kbyte以下である。

#### (4)構造体を定義の、\_\_packed\_\_ 指定

構造体のサイズが最小になるという効果がある。

```
typedef struct __attribute__((__packed__)) {  
    .  
    .  
} TheStruct ;
```

#### (5)bool は 1byte なので、積極的に使用する。

#### (6)可変引数マクロ

デバッグ用に以下のようなものを使うと良い。

```
#define DEBUG_MODE //リリース時には、コメントにして build  
#ifdef DEBUG_MODE  
    #define DEBUG_MSG(args...) APP_LOG(APP_LOG_LEVEL_DEBUG, args)  
#else  
    #define DEBUG_MSG(args...)  
#endif
```

### 5.2 通信プログラム作成上の注意(Android側)

PebbleKit.sendDataToPebble(PebbleDictionary data) が送信を完了する前に、PebbleDictionary のインスタンスが消えてしまうと、壊れたデータが送られることになる。その時、Pebble 側は壊れたデータを受信した瞬間にハングアップする。

対策としては、PebbleKit.sendDataToPebble() の直後で以下のいずれかの処理を入れる。

- ・送信後に、ACK/NACK/TIMEOUT を待つ
- ・2秒程度のSleepを入れる(非推奨)
- ・PebbleDictionary data を永続化させる(クラス変数化)

以下は、Pebble側がハングアップするコードである。

```
public void sendWeatherDataToWatch(final Intent response, String dates) {
    PebbleDictionary data = new PebbleDictionary();
    data.addInt8(0, (byte) 4);
    data.addString(1, "");
    final Timer timer = new Timer();
    TimerTask task = new TimerTask() {
        @Override
        public void run() {
            PebbleKit.sendDataToPebble(getContext(), _UUID, data);
            //ここで、ACK/NACK/TIMEOUT を待つ。
            //または、推奨できないが、2秒程度のSleepを入れる。
        }
    }
}

//送信完了前にこのメソッドからリターンした場合、Pebbleが受信した瞬間にハングアップ
```

Pebble->Android の通信を 0.3秒毎に行っている最中、Android->Pebble の通信は失敗する確率が非常に高い。このような場合には、Android側からの通信は遠慮がちになるようであるが、Android->Pebble の送信回数が一定回数を超えると遠慮できずに、送信を行うようである。Android側で送信を、バッファリングするロジックがPebbleSDK 2.4.1 に入った可能性がある。

### 5.3 通信プログラム作成上の注意(Pebble側)

(1)送信側の問題(android,pebble 共に持っている問題)

受信側は、受信終了後 ACK を送信側に返しても、送信側でその ACK を受け取れないことがある。

この場合には送信側は送信エラーと判定するが、受信側は受信が正常に終了したと判断することに注意する。

(2)送受信可能なバイト数は以下の数値に左右される。今回作成したプログラムは、

以下の数値以上を設定することが必要である。

```
const int inbound_size = 128;
const int outbound_size = 128;
app_message_open(inbound_size, outbound_size);
```

現行のPebbleOSでは、PebbleDictionary 自体の最大サイズは、124byteとなっている。

PebbleDictionary に、2つの bytes を登録した場合の最大サイズは、それぞれは 64byte・44byteである。

文字列として送信可能なのは40文字程度である。

以下の設定でも、この制限は変わらない。

```
int inbound_size = app_message_inbox_size_maximum();
int outbound_size= app_message_outbox_size_maximum();
app_message_open(inbound_size, outbound_size);
```

### (3)連続送信

ハンドラー内では、2回以上の送信は不可能である。

タイマーハンドラー等を使用して、送信を時間的に分割すること。時間は3秒程度を取ること。後述するBluetooth の速度が高い場合には 1秒程度でも良い。

### (4) Bluetooth の通信速度

消費電力の関係で、デフォルトの状態では Bluetooth の動作速度は遅くなっている。この状態だと、Pebble が Android からの ACK を受信するのに2秒以上かかったり、ACK を取りこぼしたりする。

動作速度を上げる為には、app\_message\_outbox\_begin() の直前に以下の2行を追加する。

(常にこの2行が必要であることを注意すること。下の1行だけでは動作しない。)

```
app_comm_set_sniff_interval(SNIFF_INTERVAL_REDUCED);
app_comm_set_sniff_interval(SNIFF_INTERVAL_NORMAL);
```

逆に動作速度を下げて消費電力を提言する為には、app\_message\_outbox\_begin()の直前に以下の2行を追加する。

```
app_comm_set_sniff_interval(SNIFF_INTERVAL_NORMAL);
app_comm_set_sniff_interval(SNIFF_INTERVAL_REDUCED);
```

## 付録A: 更新履歴

変更日時	変更内容
2014/09/04	初版作成。
2014/09/10	画像・改ページの配置修正。
2014/09/10	内容の追加(初版を元に作ったので、福井さんの修正との整合性を取る必要がある。
2014/09/14	レビュー