

ウェアラブル端末用 Android Wear アプリ開発の基礎知識

Android Wear アプリの開発環境「Android Studio」とは

Android Wear アプリの開発には、Eclipse ではなく Android Studio という新しい開発環境が必要になります。



図 1 Android Studio のアイコン。中で Android が働いている

Android Studio は 2013 年 5 月に開催された「Google I/O 2013」で発表された IntelliJ IDEA ベースの Android 開発です。本連載第 43 回の「[Eclipse ADT に代わる IDE となるか？Android Studio の基礎知識](#)」で取り上げていますので、インストール方法などはそちらを参照してください。

「Eclipse を 10 年以上使用してきて、いまさら別の IDE に変更するのは苦痛だ」という意見はよく分かります。筆者もその一人です。そうした開発者は世界中にたくさんいるようです。調べてみたところ、Android Wear の開発を [Eclipse + ADT](#) でできないわけではないようです。現に、Android Wear プレビュー版は Eclipse + ADT で開発するスタイルでした。

今、Android Studio を使用するメリットは何でしょうか？

Android Studio はライブラリの依存関係を自動的に解決してくれます。これは、Android Studio では「[Gradle](#)」という Groovy の DSL であるビルドシステムが標準で利用されているためです。Gradle は Apache Ivy ベースの依存関係管理機能に加え、Maven のセントラルリポジトリも利用できるため、ライブラリの依存関係解決能力が極めて高いです。

Eclipse ではビルドシステムの標準に Ant が利用されていますが、Ant はこうしたライブラリ依存関係解決機能は提供されていないため、**手動でライブラリの設定をプロジェクトに対して行わなければならない**。これは Eclipse を使っている以上普通のことですが、Android Studio で**自動的にライブラリがプ**

プロジェクトに追加され、コンパイルエラーなどが直ってしまうのを目の当たりにすると、「乗り換えても良いかなあ」と思われます。

Eclipse+ADT で Android Wear 開発を行おうとすると、Android Wear のサポートライブラリが必要になります。プレビュー版ではGoogleから JAR ファイルとして提供されていましたが、正式版のサポートライブラリの 在りかは公表されていません。つまり Eclipse+ADT で開発しようとして、どこからサポートライブラリを持ってくるか、Android SDK Manager のリポジトリにライブラリの場所を追加するなりしてダウンロードできるようにする必要があります。

調べれば海外の大手質問サイトなどでその答えが見つかると思いますが、Google公認の方法ではなく、「その時点ではその方法で解決できた」という方法であることに留意してください。

さて、Android Studio は発表されて 1 年以上が経過していますが、Eclipse+ADT を置き換えるほどまでは Android アプリ開発者に浸透してはいないよう です。Googleとしては、Android Wear を皮切りに、Android Studio を使ってもらいたい、という意図があるように思えます。

ということで今回はその意図に乗ってみようと思います。今回の記事は Android Studio を使用して解説します。解説内容をトレースしてみたい方は、「[Android Studio | Android Developers](#)」より Android Studio をダウンロードしてインストールしてください。

なお、Windows 版で Android Studio を動作させるためには 64bit 版の JDK が必要です。



図 2 32bit JDK+Android

Studio の組み合わせのエラーダイアログ

筆者はいろいろな都合で 32bit 版の JDK が引き続き必要であるため、今回は Mac OSX を利用しています。

Android Wear アプリの開発に使う実機は？

Android Wear アプリの開発は、エミュレーター上でも可能ですが、本格的なアプリを開発する場合は Android Wear 端末(以下「実機」)が必要になってきます。

2014 年 8 月の原稿執筆現在、Google Play Store 上で購入可能な実機は、以下の 2 つです。



Samsung Gear Live

- ディスプレー：1.63 インチ
- サイズ：37.9 x 56.4 x 8.9 mm
- 重さ：59g
- 電池：300mAh
- CPU：1.2GHz
- Bluetooth：BLE 4.0
- メモリ：4GB ストレージ、512MB RAM
- センサー：加速度計／コンパス／ジャイロ스코ープ／心拍計
- 価格：2 万 2000 円



LG G Watch

- ディスプレー：1.65 インチ
- サイズ：37.9 x 46.5 x 9.95 mm
- 重さ：63g
- 電池：400mAh
- CPU：1.2GHz
- Bluetooth：BLE 4.0
- メモリ：4GB ストレージ、512MB RAM
- センサー：加速度計／コンパス／ジャイロ스코ープ
- 価格：2 万 2900 円

それぞれの端末の詳細に関しては以下の記事を参照してください。

- [Android Wear 端末、LG と Samsung は本日予約開始、Motorola は今夏](#)
- [iFixit、LG G Watch と Samsung Gear Live を解剖](#)
- [徹底レビュー：「Samsung Gear Live」の毎日着けたいくなる使用感と痛過ぎる欠点](#)
- [腕時計だけど時計じゃない「Gear Live」を買って開けてみた](#)
- [「Samsung Gear Live」を買って開けてみた](#)
- [LG G Watch レビュー（後編）——通知、Google Now、OK Google、Android Wear アプリを試す](#)
- [LG G Watch レビュー（前編）——開封して初期設定をしました](#)

スペックや価格は、どちらもほとんど同じですが、筆者は「Samsung Gear Live」を選択しました。LG G Watch が搭載していない心拍計を Samsung Gear Live が搭載しているのが決め手です。電池は小容量なので、毎日充電が必要です。GPS がないのが残念ですが、もし搭載していたら価格がもっと上がってしまうでしょうからしょうがないですね。

実機でまず行うべきは「システムアップデート」です。筆者は端末到着後、数日間「どうやって操作すれば良いのかよく分からない不思議なインターフェースだなあ」と思いながら機能確認を行っていました。ところが、どうやら、まともに操作することも難しいような状態のファームウェアが搭載されていただけのようです。

「システムアップデート」を選択するのに苦労しました。OK ボタンをタップしても反応しないためです。何故アップデートできたのかは不明ですが、繰り返していると成功すると思います。システムアップデートは充電状態にして行うことを推奨します。

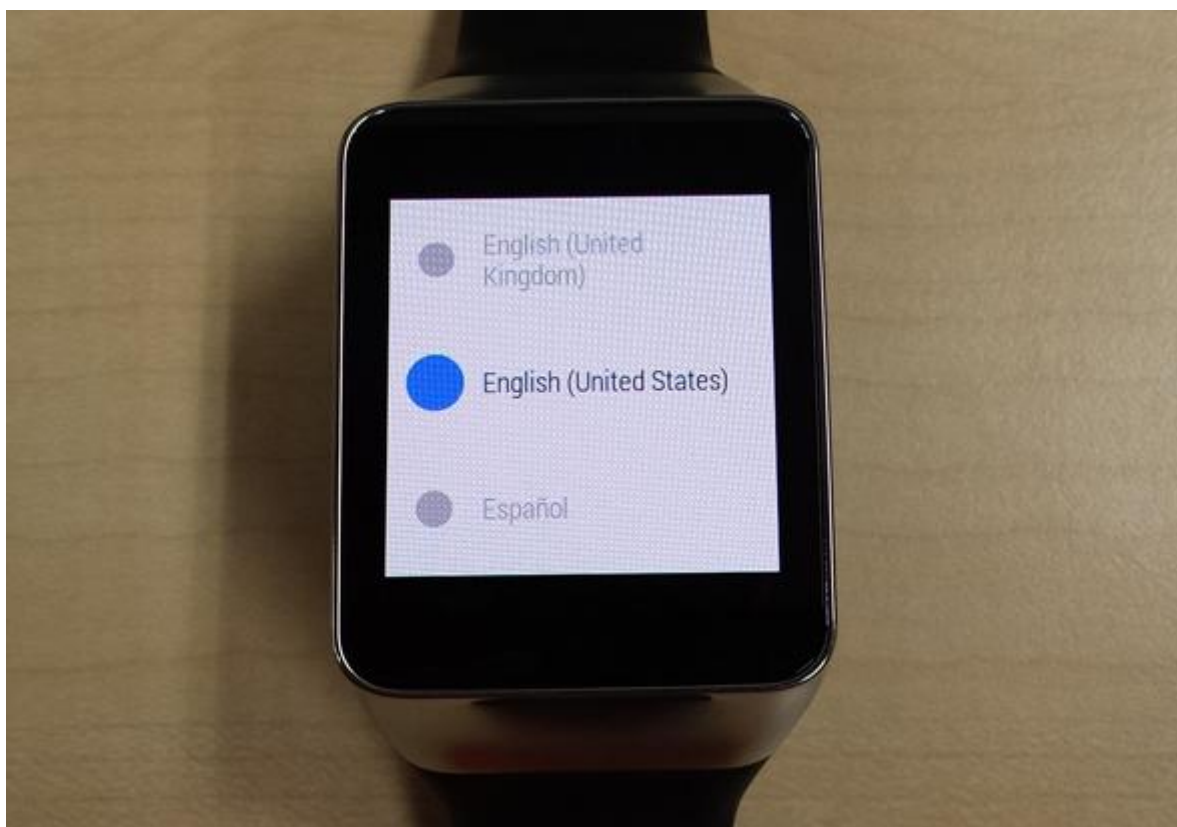
また、プリインストールされているファームウェアは出荷時期によって更新されると思いますが、念のため最新であるか確認しておくことを推奨します。

Android Wear アプリを開発するには、Android スマホとのペアリングが必要

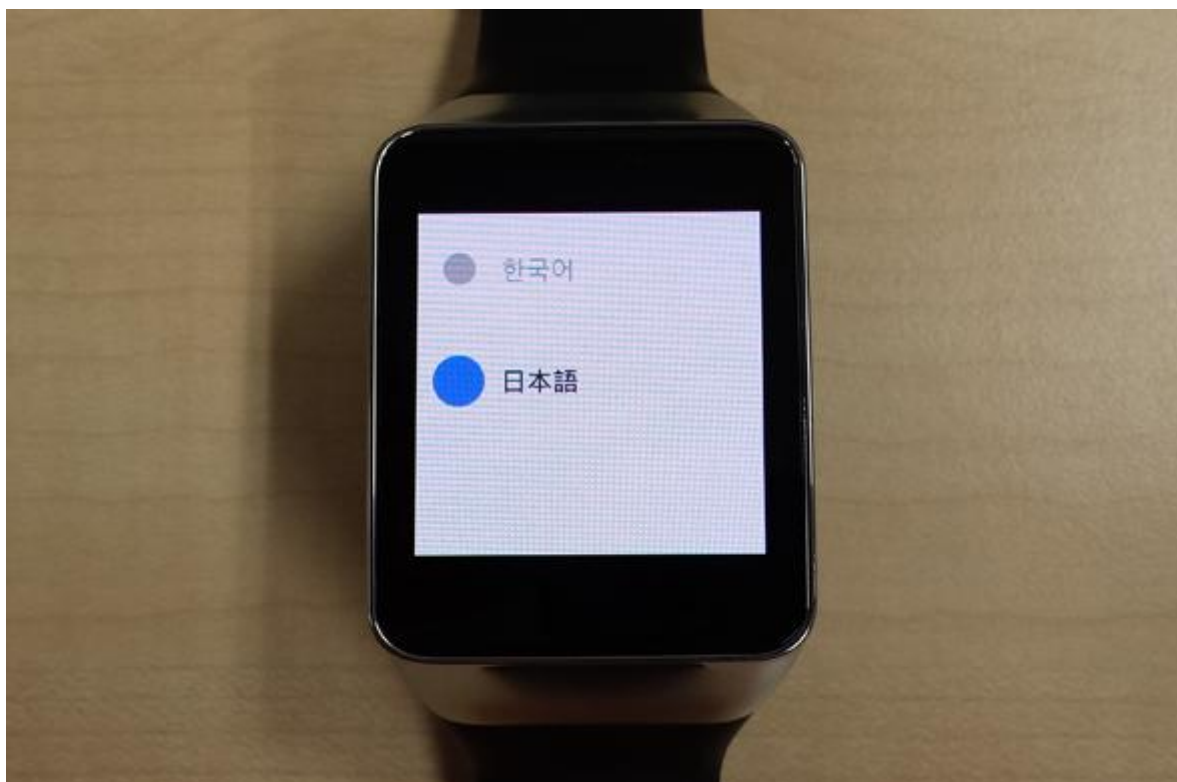
Android Wear アプリを開発する際には、ホストとなる Android 4.3 以上のスマートフォン（以降「ホスト」）に、「Android Wear」という端末操作用アプリをインストールして、ホストとペアリングしておく必要があります。実機とは [BLE \(Bluetooth Low Energy\)](#) でペアリングします。エミュレーターとペアリングすることもできます。

以下よりホストと実機のペアリング手順を説明します。

実機は起動直後、以下のように言語選択画面が表示されています。デフォルトでは英語です。



下にスクロールして、日本語を選択してタップします。ただし、この言語選択は別段重要ではありません。というのもペアリング途中でホストの表示言語に合わせて実機の言語も自動的に変更されるためです。



またホストの表示言語を、例えば英語に変更した場合、まもなく実機の表示言語も英語に自動変更されます。時間もホストと同期するようになっていて、実機側の設定が最小限で使えるようにしてあります。この辺りはうまく作られていると感心しました。

Android Wear アプリをホストにインストールするように指示があります。ホストに Android Wear アプリをインストールしてペアリングしなければ、この実機はこの画面のまま先に進めず使用できません。



Android Wear アプリは以下リンクまたは QR コードから Google Play Store に接続してインストールしてください。



<https://play.google.com/store/apps/details?id=com.google.android.wearable.app&hl=ja>

ホストに Android Wear アプリをインストールして起動すると、ペアリングを促すウィザード画面が表示されます。



Android Wear でどのようなことができるのか説明する画面が表示されます。

android
wear

時計をスマートウォッチにする

Android Wearと携帯端末は常に同期しています。



Google Now

端末でGoogle NowがONになっていると、最新の天気予報や通勤経路の交通状況、フライト情報など、検索しなくても必要なときに情報が表示されます。端末がロックされているときでも、同じ情報のほとんどが時計に送信されます。



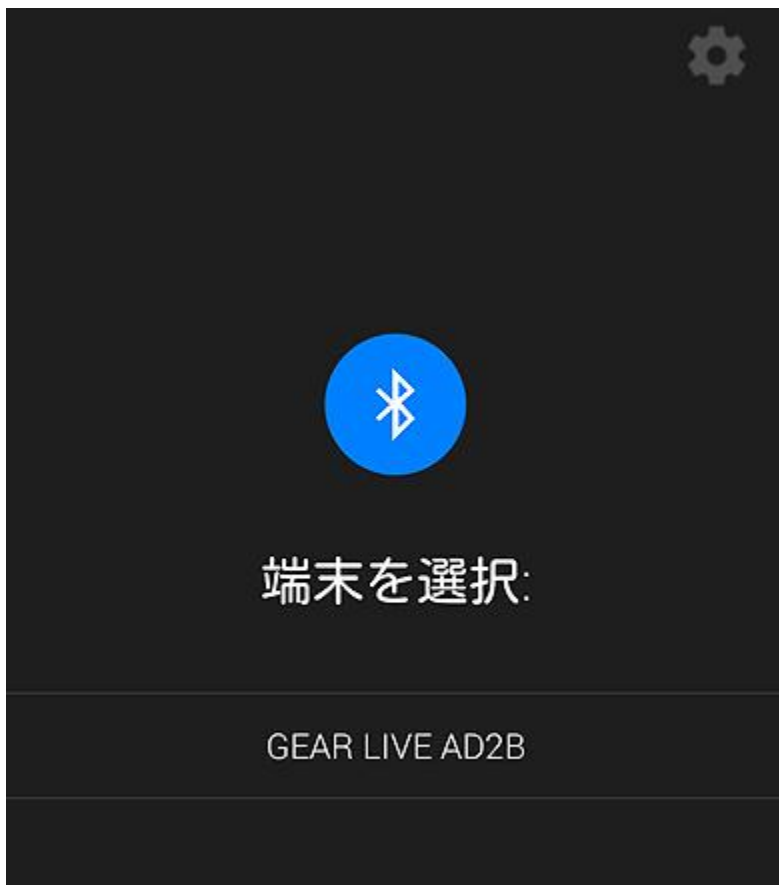
位置情報

Googleの位置情報サービスがONになっていると、お使いの端末の現在地が時計に送信され、端末がロックされているときでもナビを利用できます。

下にスクロールして利用規約とプライバシーポリシーに同意します。



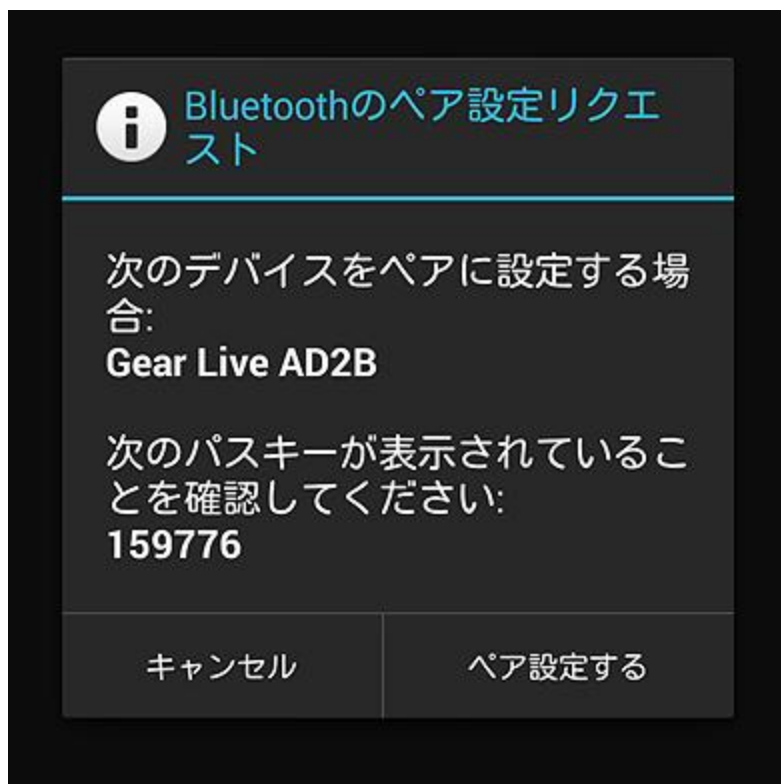
近くの実機を検知してリストに表示します。ここでは Gear Live AD2B が1つだけ検知されているので、それをタップします。



タップするとホストの名称とペア設定コードが表示されます。間違いがなければチェックボタンをタップします。



実機の名称と実機に表示されているパスキーに間違いがなければペア設定するをタップします。



ホストは、しばらくした後、以下ペアリング完了画面が表示されます。

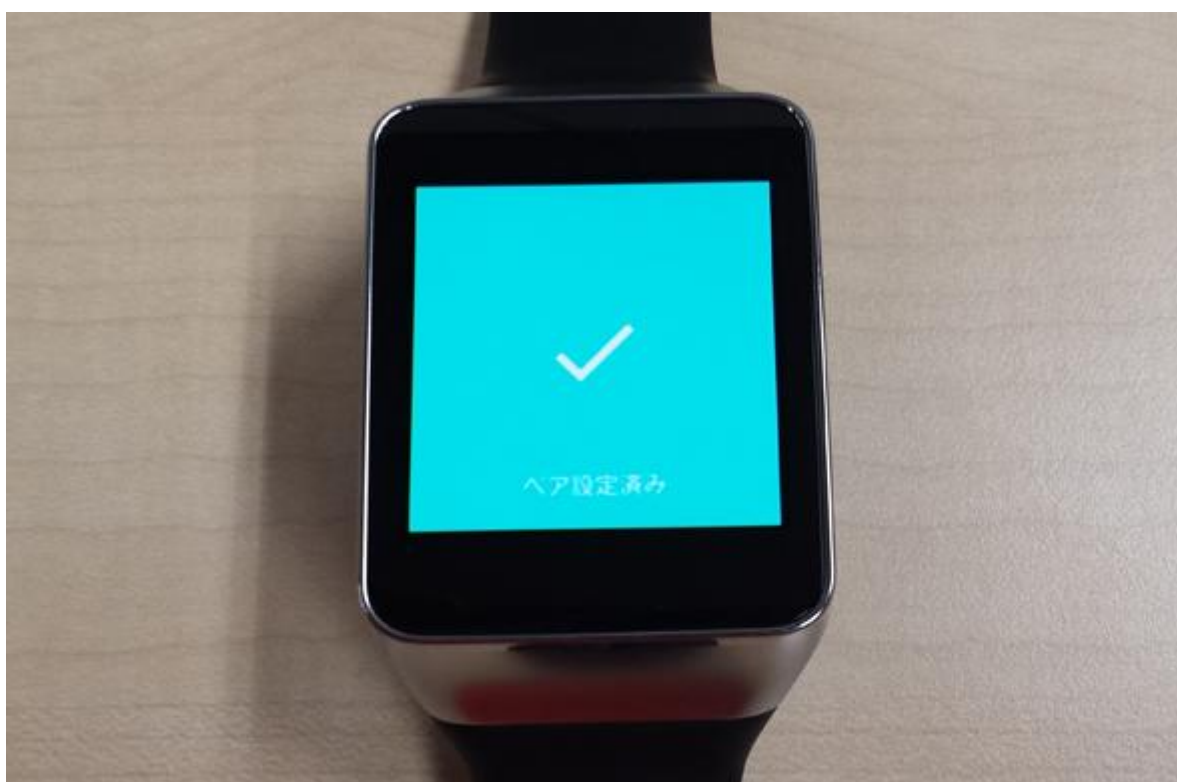


ホストはこの後、自動的にチュートリアルが開始されます。実機の基本的な使い方を理解する上でも見ておくといいでしょう。

実機は下記のようなペアリング処理がしばらく続きます。



完了するとペアリング完了画面が表示されます。



この画面も自動的に終了します。実機側でもチュートリアル用のカードがいくつか存在する状態で時計表示されるので、全てのチュートリアル用カードを読んで、その指示に従いつつ基本的な使い方を習得することをおすすめします。

実機の開発者向けオプションを設定

ペアリングが完了したら、次は設定です。実機の開発者向けオプションを以下の手順で有効にしておく必要があります。



図 4 設定画面

まず、実機の端末情報を選択します。上記スクリーンショットはすでに開発者向けオプションが有効になっています。

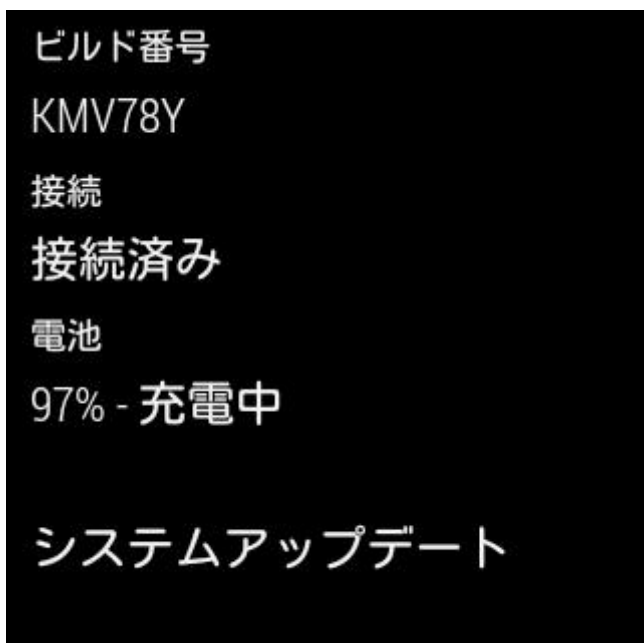


図 5 端末情報画面

ビルド番号を複数回タップします。タップを繰り返すと「デベロッパーになるまであと 3 ステップです。」というメッセージが表示され、さらにタップを繰り返すと「これでデベロッパーになりました！」とメッセージが表示されます。

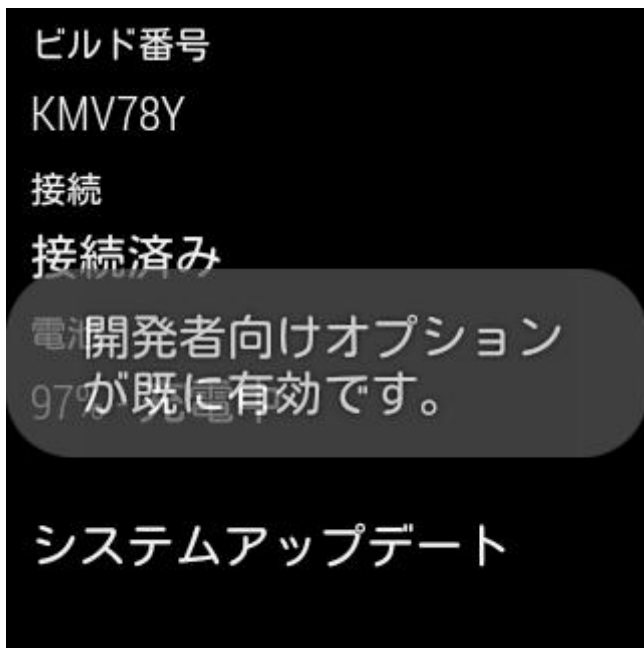


図6 開発者向けオプションメッセージ

すでに開発者向けオプションが有効な場合は、上記スクリーンショットのようにメッセージが表示されます。

開発者向けオプションでは、以下を変更します。



図7 開発者向けオプション画面

- 充電中に画面をロックしない：無効 → 有効

これは好みで行ってください。実機はバッテリーの消耗を抑える目的だと思うのですが、すぐにディスプレイがオフになります。そしてその時間を変更するオプションは用意されていません。

- ADB デバッグ：無効 → 有効

これを有効にしないと開発ができません。

- Bluetooth 経由でデバッグ：無効 → 有効

Bluetooth 経由のデバッグも好みで行います。実機を PC に USB で接続して開発する場合は無効のまま、ホストを PC に USB で接続して開発する場合は有効にします。この項目が有効になっていると、実機を PC に接続しても PC で認識されません。Bluetooth 経由のデバッグを有効にした場合、ホストの Android Wear アプリでも Bluetooth 経由のデバッグを有効にしておく必要があります。



図 8 Bluetooth 経由のデバッグを有効にし、ターゲットと接続済みになっていることを確認

この状態でホストを PC に接続すると、以下のような確認ダイアログが表示されます。



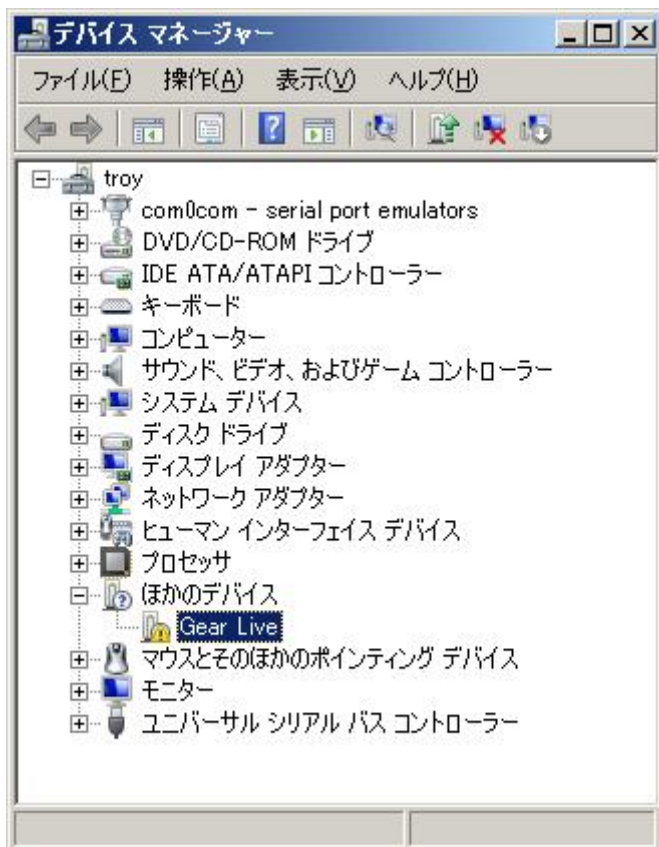
図9 デバッグ許可をホストで代行

OK をタップすることで、実機のデバッグが可能になります。実機の画面は狭いので、こうした込み入った表示や確認はホストで行うわけですね。

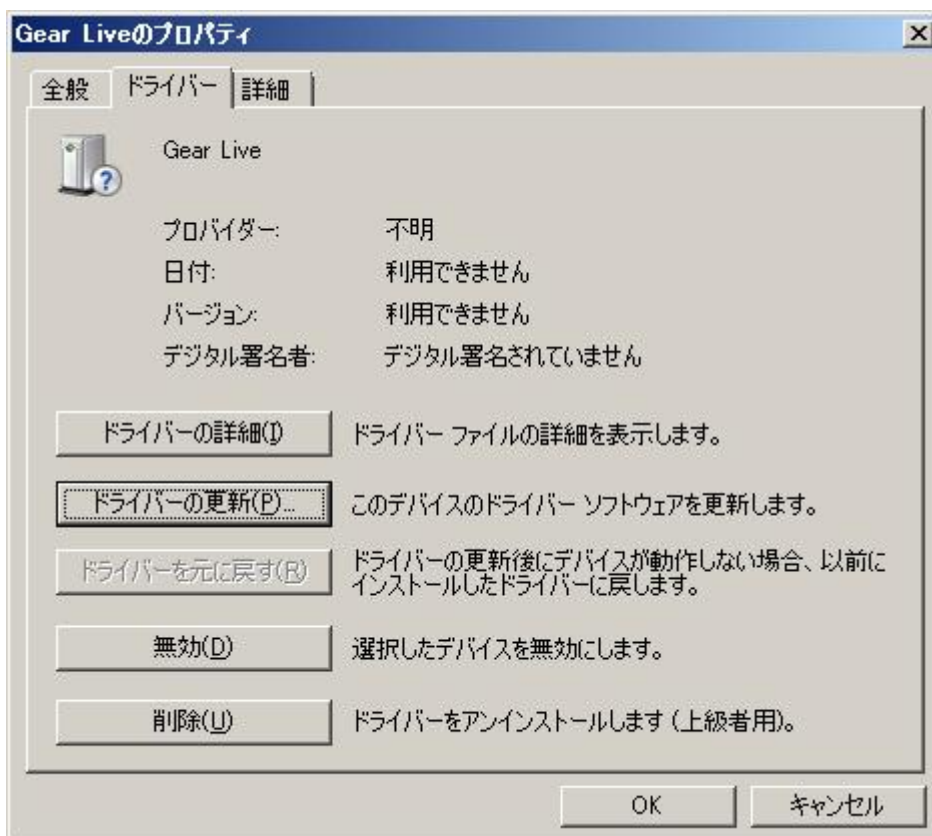
コラム「Windows 上でのデバイスドライバー設定方法」

2014 年 8 月の原稿執筆時現在、Windows 上で Android Wear アプリを開発するには、Android Wear 端末のデバイスドライバーを手動でセットアップしなければなりません。これは Android SDK で配布されるデバイスドライバーの定義ファイルに Android Wear 端末の VID/PID が含まれていないためです。

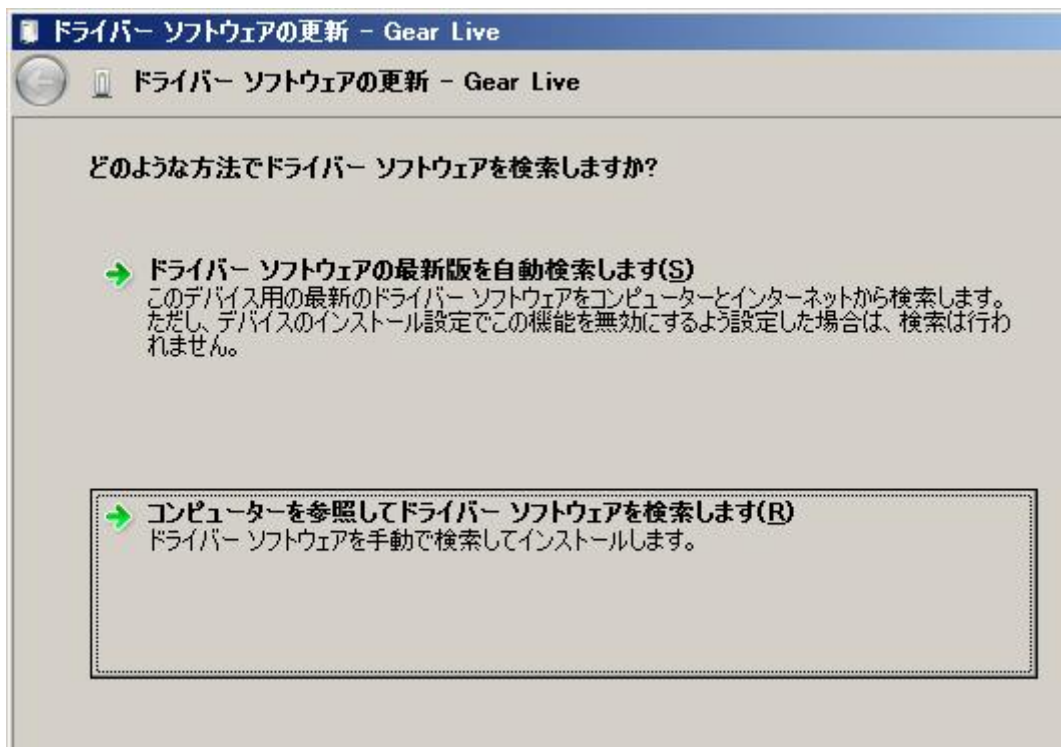
ただし、既存の Android 端末のデバイスドライバーが利用できるので、手動で Android Wear 端末にデバイスドライバーを設定して ADB が利用できるようにすることは可能です。ここでは、その方法を解説します。



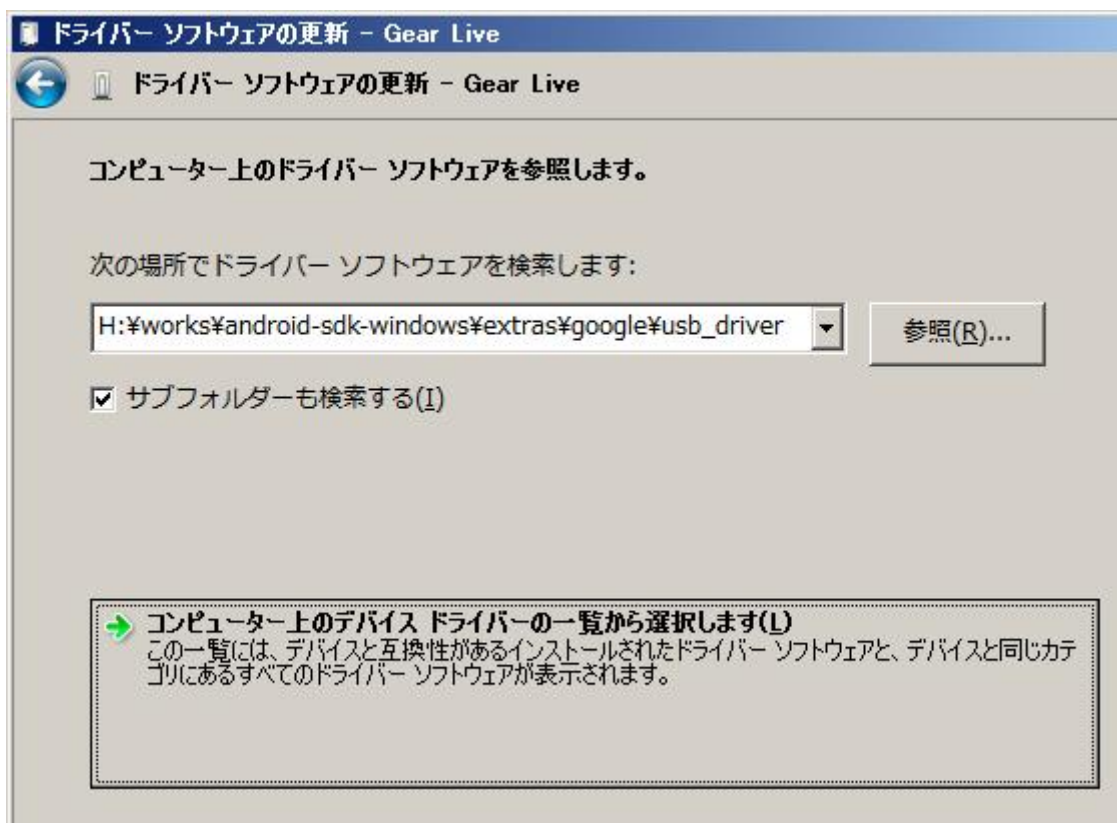
実機を Windows PC に接続すると、デバイスドライバーのインストールに失敗し、デバイスマネージャで上記のように警告マークが表示された状態になります。このデバイスのプロパティを開きます。



ドライバータブの「ドライバーの更新」をクリックします。



下側の選択肢をクリックします。



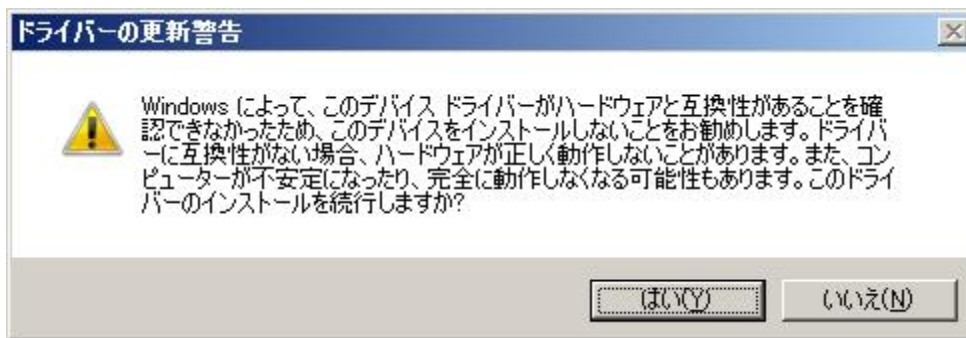
下側の選択肢をクリックします。



ADB Interface を選択し、「次へ」をクリックします。



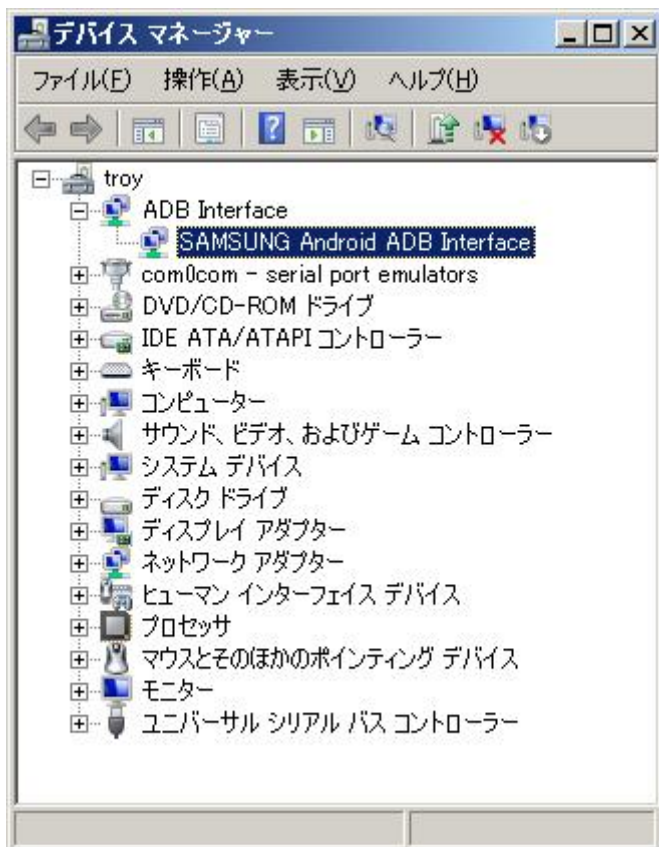
いくつか候補が出てきた場合は、日付の一番新しいものを選択し、「次へ」をクリックします。



ドライバーの更新警告ダイアログが出ますが、続行するために「はい」をクリックします。



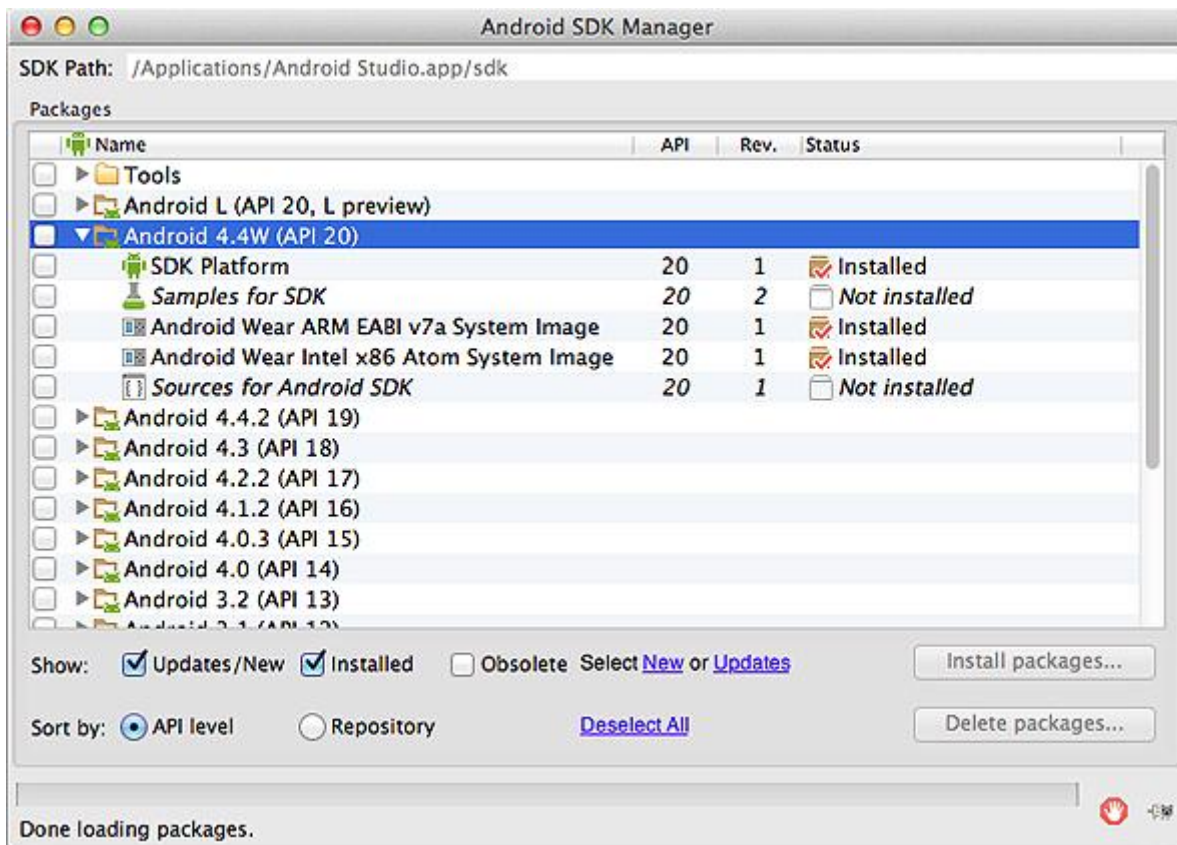
デバイスドライバーのインストールが完了したら「閉じる」をクリックします。



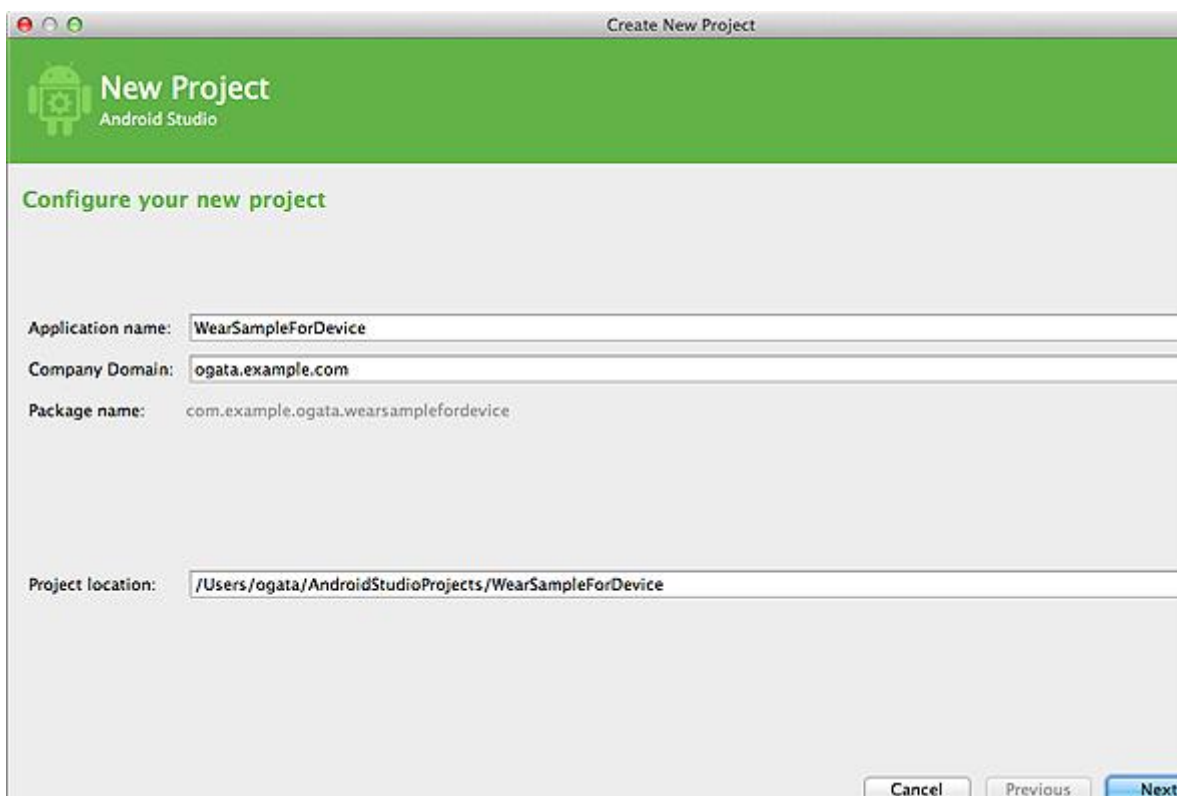
実機が ADB Interface で認識されていることを確認します。この状態になっていれば、adb コマンドで認識される状態になっています。

Android Wear アプリプロジェクトの作成

Android Studio で Android Wear アプリプロジェクトを作成する手順を解説します。作業は Mac OS Xで行っていますが Windows でも手順自体は変わらないと思います。



プロジェクト作成の前に、Android SDK の状態を最新にしておいてください。特に「Android 4.4W の SDK Platform」と System Image がインストールされているか、それらが最新であるかは必ず確認してください。



トップ画面から「New Project」を選択してアプリ名、ドメイン、保存場所を入力します。パッケージ名はアプリ名とドメインから自動的に生成されますが、変更することも可能です。

factors your app will run on

require separate SDKs

☐ Phone and Tablet
Minimum SDK: API 15: Android 4.0.3 (IceCreamSandwich)

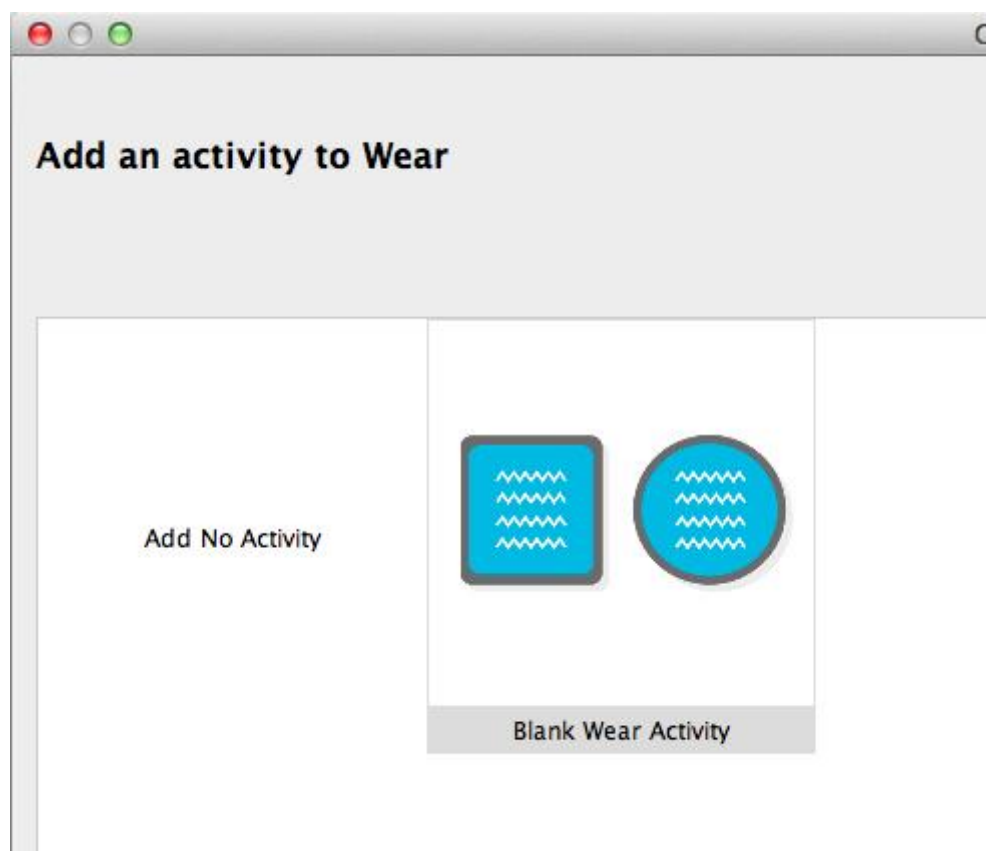
☐ TV (Not Installed)
Minimum SDK:

☒ Wear
Minimum SDK: API 20: Android 4.4 (KitKat Wear)

☐ Glass (Not Installed)
Minimum SDK:

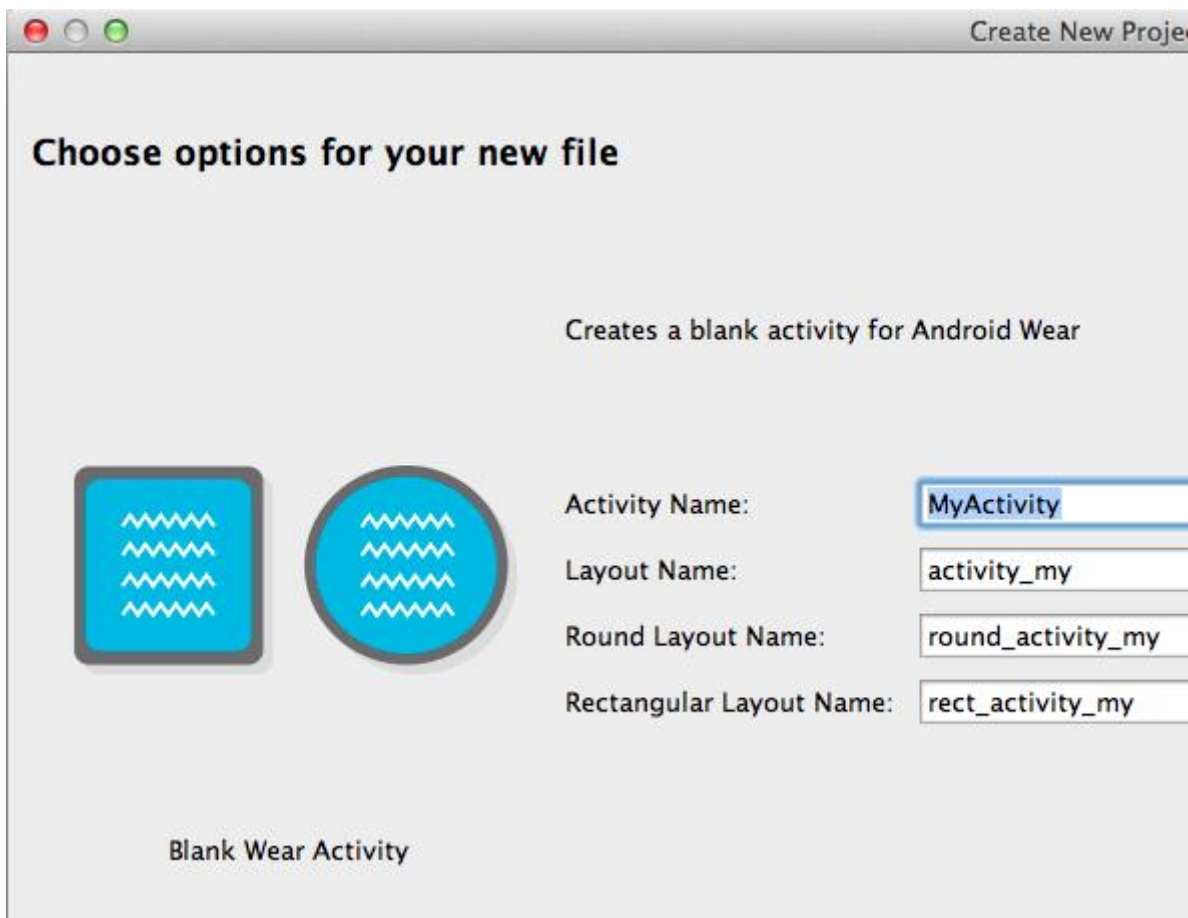
Lower API levels target more devices, but have fewer features available. By targeting API 15 and later, your app will run on approximately 84.3% of the devices that are active on the Google Play Store. [Help me choose.](#)

作成するアプリの種別を選択します。今回作成するのは Android Wear アプリなので、Wear にチェックを入れます。Android SDK でプラットフォームがダウンロードしてあれば、TV アプリや Glass アプリも選択肢として選べるようになります。

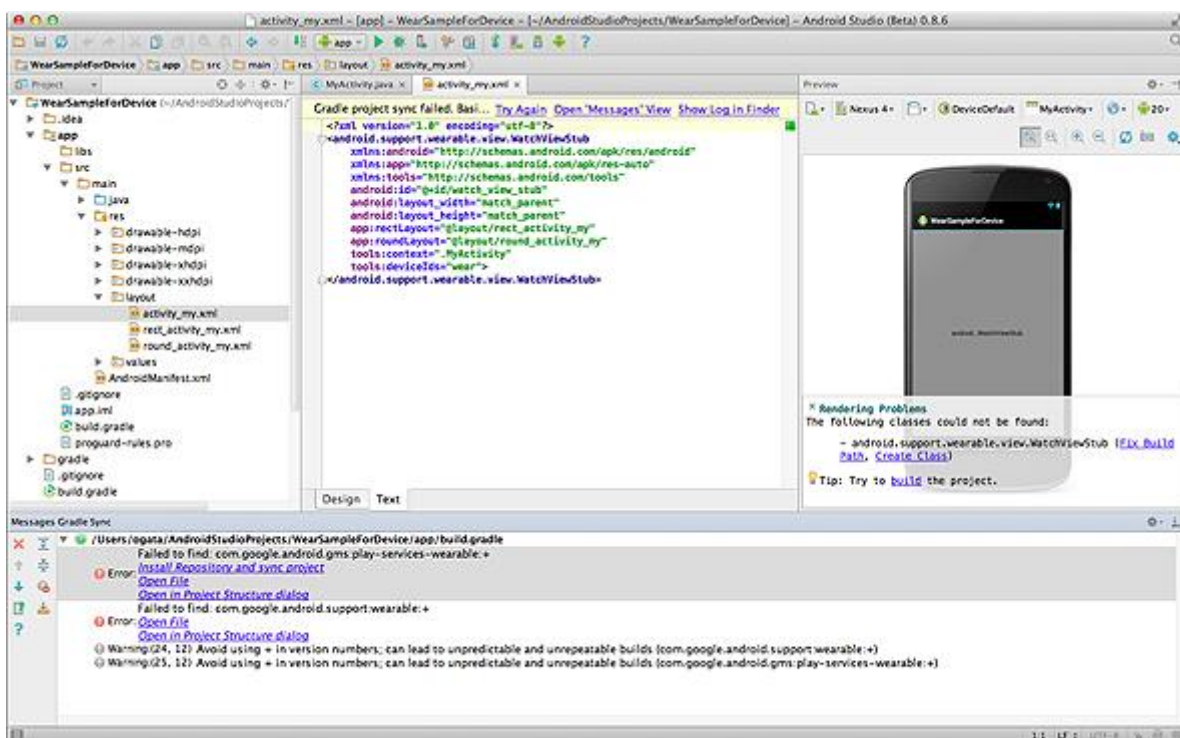


Activity のないプロジェクトか、「Hello」を表示する Activity があるプロジェクトにするかを選択します。Android Wear は四角い画面の他に、丸い画面を持つデバイスも想定されていて、「Blank Wear Activity」

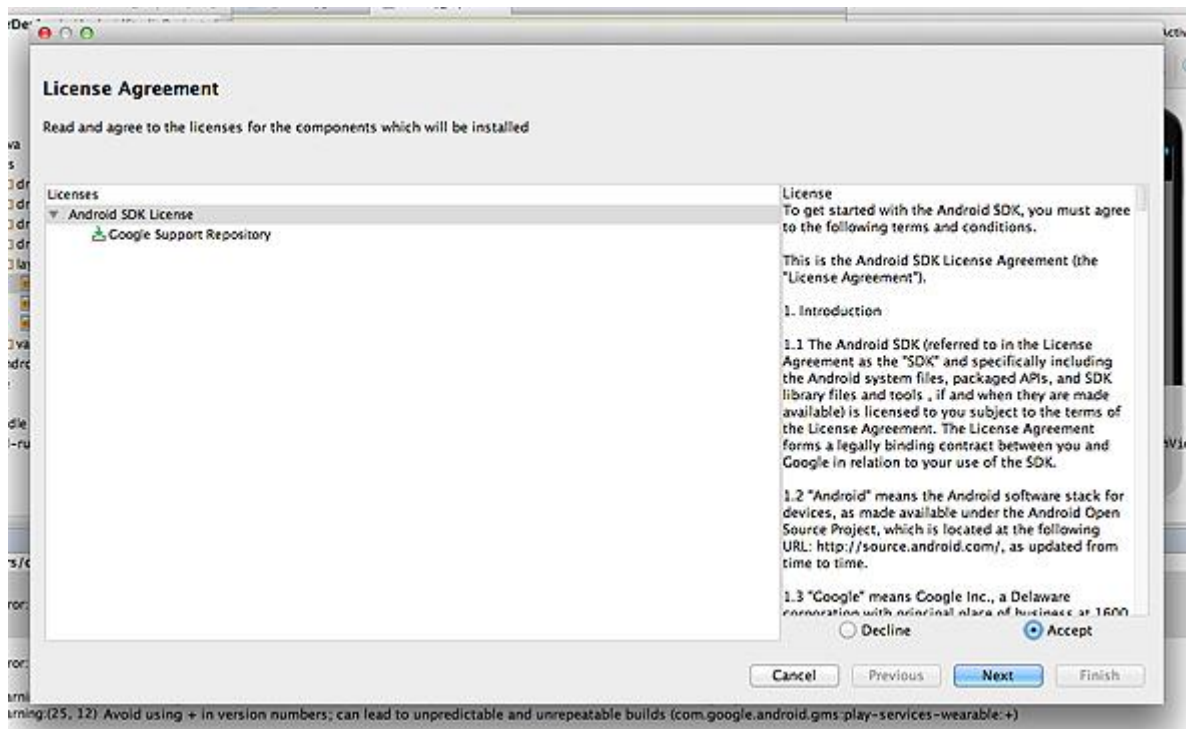
を選択することで、両方のリソースが自動生成されます。今回は自動生成されたものを利用することになります。



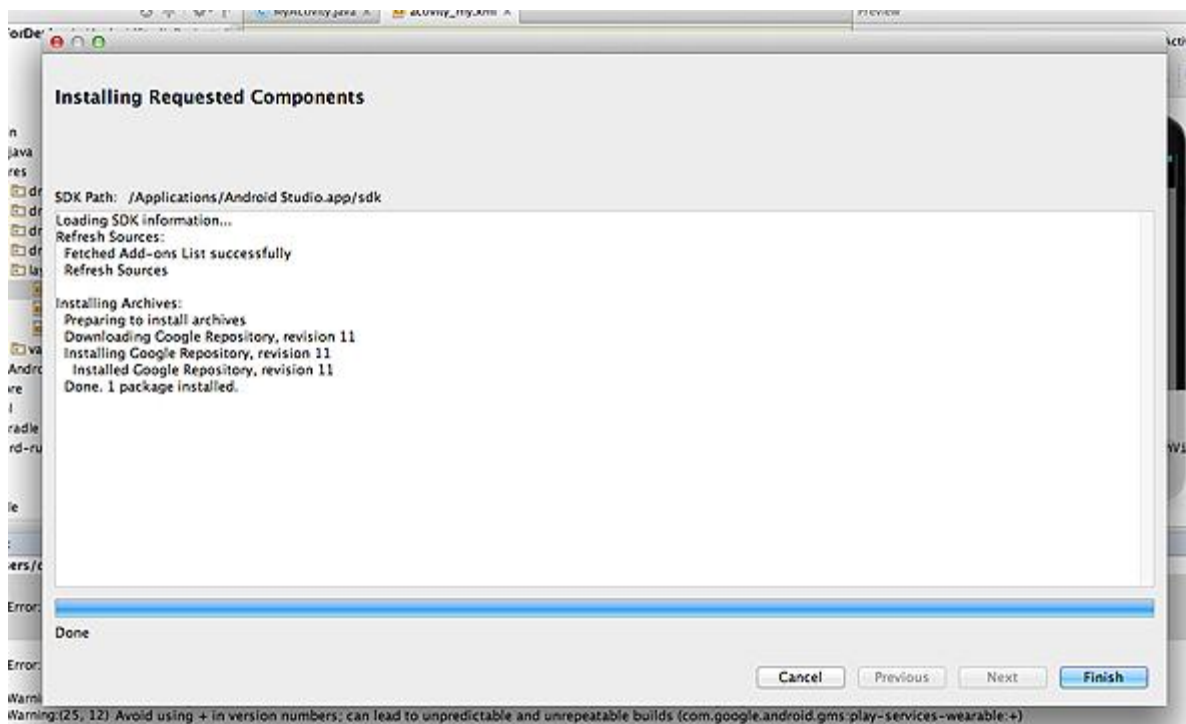
Activity の名前を入力します。ここではデフォルトの MyActivity のままにしていますが、Activity の名前を変えると、レイアウト名も自動的に変更されるようになっています。



プロジェクト生成直後に自動ビルドが行われます。画面下部のメッセージエリアにエラーが出ている状態ですが、ここで「Install Repository and sync project」リンクをクリックします。



「Google Support Repository」を追加するためのライセンスに同意します。

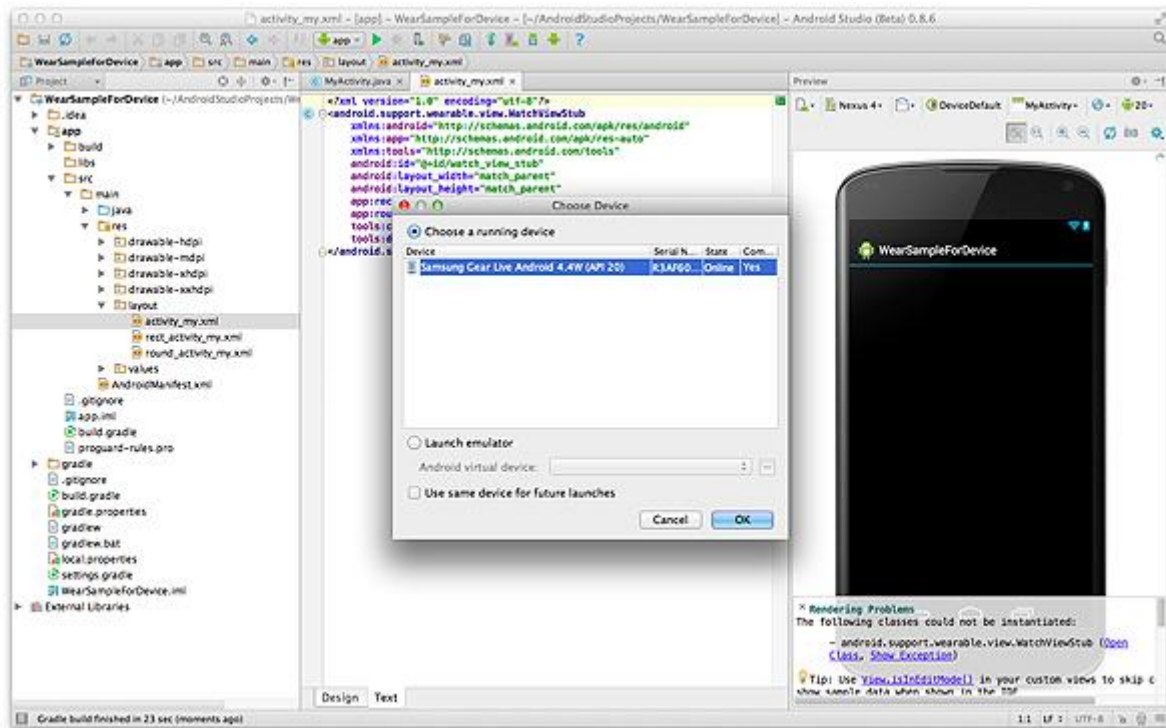


リポジトリが設定され、ライブラリがインストールされました。Finish をクリックすると、自動ビルドが開始され、プロジェクトのビルドに成功します。

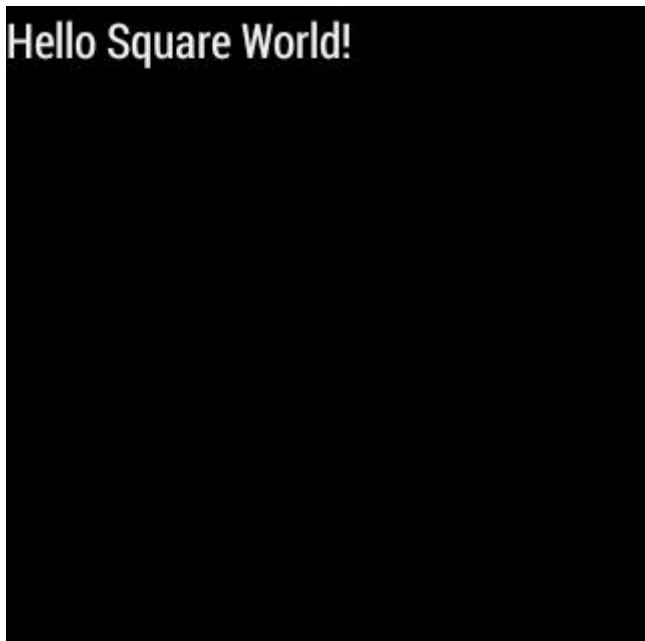
これが Gradle の能力なのでしょうか。外部のライブラリを必要とする以上、コンパイルエラーが発生してしまうのは致し方ありません。しかしそれをクリックだけで解決できてしまうという仕組みが Eclipse ユーザーとしては感動的でした。

Android Wear 実機デバッグ

デバッグボタンをクリックすることでデバッグを開始できます。



どのデバイスでデバッグするかを選択するダイアログが表示されます。実機が直接接続されているか、Bluetooth 経由でデバッグが有効になっていてホストが接続されていれば、実機が選択肢として表示されます。



アプリが起動してメッセージが表示されました。

このアプリはウィザードで作成した状態のままですが、以下よりダウンロードできます。

- [WearSampleForDevice.zip](#)

今回は、この Hello アプリをベースに、Android Wear にもっと踏み込んだアプリ開発を解説しますので、楽しみに。

Android Wear アプリ開発に関する 3 つの注意点

最後に、筆者が気付いた Android Wear アプリにおける注意点を 3 つ紹介します。

【1】 バッテリー容量の少なさ

Samsung Gear Live のバッテリーは毎日充電が必要である旨が公式サイトにも記載されています。卓上に置き時計として放置するなら 3 日ぐらいいは持ちますが、腕に装着して それなりに活用するとなると、やはり毎日充電が必要になってしまうでしょう。毎日充電するのは良いとして、Samsung Gear Live は気の利いたクレードルがないので、充電が面倒くさいです。

さて、Samsung Gear Live のバッテリー容量は 300mAh ですが、いったいどれぐらいの容量なのでしょう。今回ホストに使用した Samsung Galaxy Nexus は 2000mAh なので、スマートフォンの約 6 分の 1 ぐらいの容量だということですね。ちなみに、単三乾電池は約 2500mAh です。

【2】 ペ어링解除ができない

実機側は一度ペ어링した設定は解除できません。解除する必要に迫られた場合、端末をリセットして工場出荷時状態に戻して、最初からペ어링をやり直す必要があります。

実機は 4GB のストレージがあるため、それなりのサイズのデータを蓄積することが可能です。ただし、実機がペ어링するホストを変更する唯一の 手段がファクトリリセットであり、ファクトリリセット後はストレージ上のファイルが全て削除されてしまうことから、実機上で動作するアプリはデータをホスト側に定期的に送っておき、いつか行われるファクトリリセットに常に備えておく必要があります。

【3】 アプリのアンインストールができない

実機にインストールしたアプリをアンインストールするインターフェースが提供されていません。アプリのアンインストールは「adb uninstall」コマンドでしか行えないのです。

開発者はそれでよいとしても、一般ユーザーにしてみればインストールしたアプリが削除できない、という事態に陥ってしまいます。どうしても削除したいアプリがある場合、ファクトリリセットされてしまうかもしれません。今後のバージョンアップでアンインストール用のインターフェースが提供されることを期待したいです。

新 IDE : Android Studio を使ってみよう！

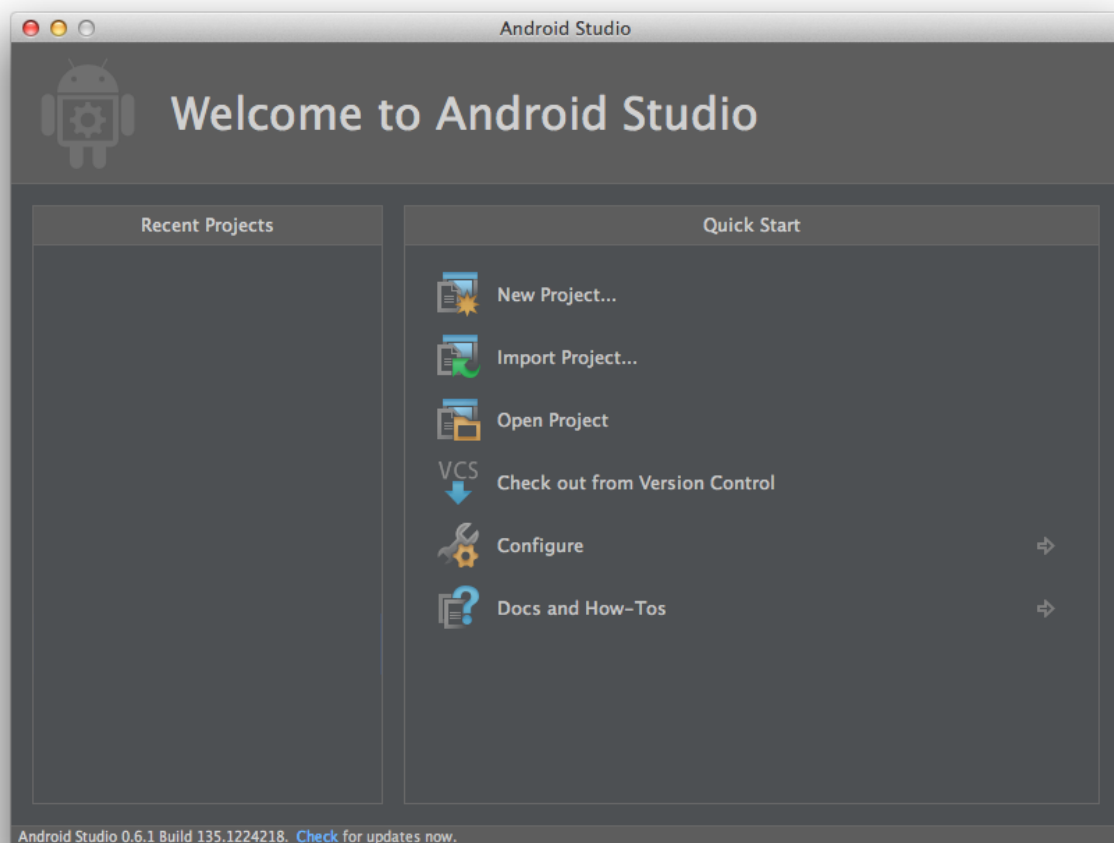
Android Studio を導入するにあたり、知っていると便利な情報をまとめてみました。そろそろ本格導入したいと考えている方はこの機会にチャレンジしてみてもいいのではないでしょうか。

インストール

まずは [公式ページ](#) から配布パッケージをダウンロードし、インストールします。

Windows なら適当なフォルダへ展開し、exe ファイルを起動してください。
Mac なら dmg ファイルを開き、Application フォルダへコピーします。
その後、コピーした app ファイルを起動しましょう。

スプラッシュ後に以下のような画面が表示されればインストールは完了です。



Android Studio の設定

インストールしたら早速プロジェクトを作成して開発を始めてもいいのですが、まずはいくつかの設定をしておきましょう。

起動した直後には、先ほどの Welcome 画面が出ていると思います。その右側ペインにある Configure をクリックします。その先の画面の Preferences を開くことで色々と設定を行うことができます。

もし、すでに新しいプロジェクトを作成してしまっていた場合でも、メニューバーにある Android Studio から Preferences を選択すれば同じ画面を開くことができます。

ここから先は Preferences 画面を開いているという前提で進めていきます。

ベース色の設定

ベース色の設定を行いましょう。これは開発のモチベーションをあげる大事な要素になります！

インストールしたばかりの Android Studio のテーマは、白背景がデフォルトで設定されています。これを変更するには IDE Settings → Appearance を選択し、Theme 横のボックスもしくは下矢印をクリックしましょう。すると、白ベースの Default と 黒ベースの Darcula を選択することができますので、好きな方を選びましょう。

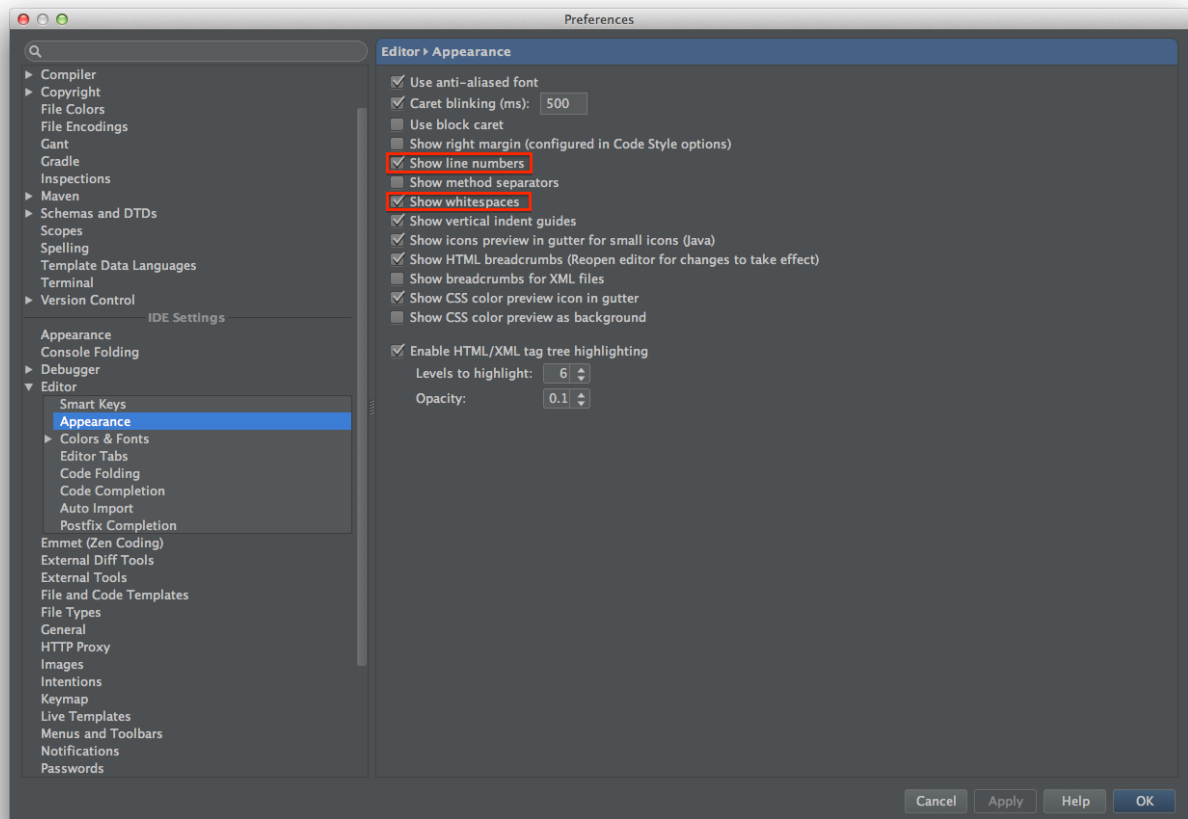
ちなみに、オススメは黒ベースの Darcula です。背景が黒いので落ち着いていて目にやさしく、何よりクールでカッコイイです！特に色のこだわりがないのであれば、黒に変更しちゃいましょう！

行番号とスペースの表示

Editor → Appearance を開きます。

Show line numbers にチェックを入れると、行番号が表示されるようになります。

Show whitespaces にチェックを入れると、空白文字を可視化することができます。

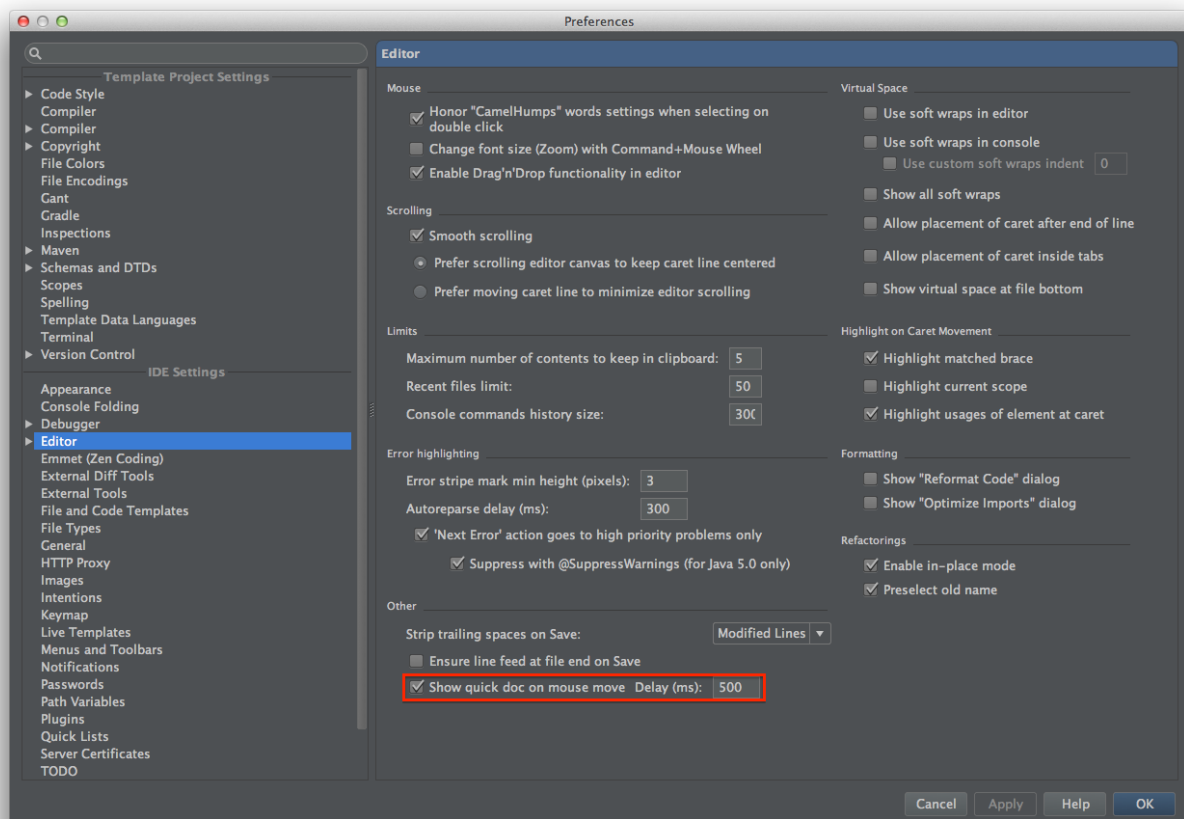


最近の IDE では、GUI 上にエラー情報が表示されるため、行番号を表示するメリットは大きく無いですが、それでも行番号を表示しておいて良いこともあると思います。また、全角スペースが入っていることでコンパイルエラーが起こる場合もあります。両方とも出しておいて損は無いでしょう。

マウスオーバーでドキュメントを表示する

Editor → Show quick doc on mouse over

にチェックを入れます。右側に、マウスオーバーしてから表示されるまでの遅延時間を設定することができるので、快適だと思える時間を入れましょう。



自動インポート

Android Studio では編集する度に自動的に保存が行われています。そこで、保存時にインポートを自動で行ってくれるよう設定することで、インポートに関して気にすることが(ほとんど)なくなります。

Editor > Auto Import > Optimize import on the fly

Editor > Auto Import > Add unambiguous on the fly

これらにチェックを入れましょう。

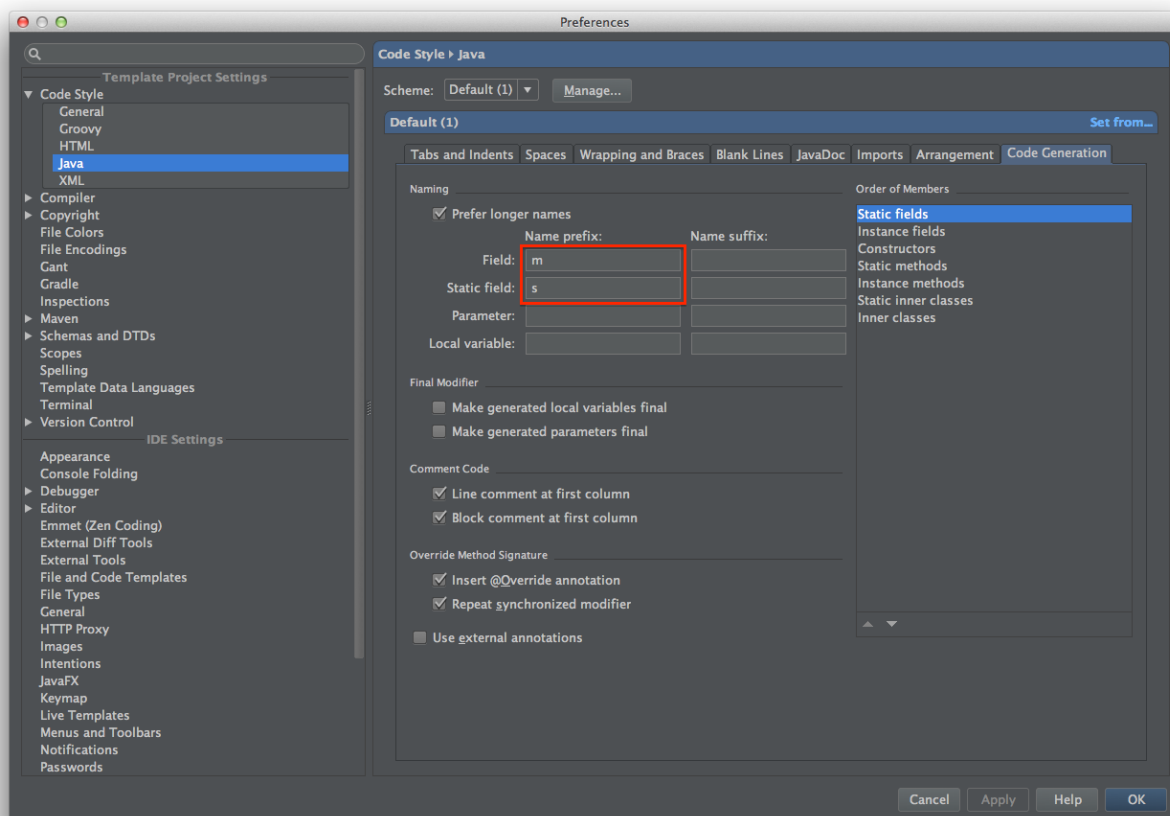
prefix を設定

この設定をしておくことで、コーディング規約に則る場合、恩恵を受けることのできる機能が数多く存在しています。

例えば、getter や setter を生成する際に、先頭に付いている m や s などのプレフィックスを無視して生成することが可能になります。

設定は Template Project Settings にある、Code Style -> Java を選択し、右側ペインに表示されているタブの中から Code Generation を選択します。

Naming と書かれている部分の Field と Static field の Name prefix カラムに、それぞれ m と s を入力することで完了です。



keyMap の変更

KeyMap を設定しておくことで、開発の効率は格段に上がるでしょう。
IDE Settings の方にある Keymap を選択します。

最上部にある Keymaps という部分から、Keymap のセットを選択することができるので、Eclipse や、Emacs などのキーバインドの中から使い慣れたものを選ぶと良いでしょう。

しかし、このキーバインドはあくまで Eclipse 風や Emacs 風になり、完全に模倣することは出来な
いようです。なので足りない部分は自分なりのカスタマイズをしていくことになると思います。

まずは Keymap のセットを選択した後に、そのすぐ右にある Copy ボタンをクリックします。するとそ
のセットをコピーすることができるので、コピーしたセットに好きな名前を設定し、カスタマイズを行いま
しょう。

編集した機能は、文字が青色になるため、ひと目でどこを編集したのかがわかるようになっていま
す。

Tool Button の表示

この設定は Preferences にはありません。ですが、これもかなり重要な設定になりますので初期の段階でやることをお勧めします。

メニューバーにある View → Tool Button を選択し、チェックを付けることで表示させることができます。

すると、様々な Tool Button が画面の周りを囲むように出てくると思います。この中には、Android 開発には欠かせない LogCat や、TODO を管理することのできるビューなどを表示するためのボタンがあります。

Android Studio を使う

ここからは、Android Studio を使っていく上で、知っておくと得ることを紹介したいと思います。

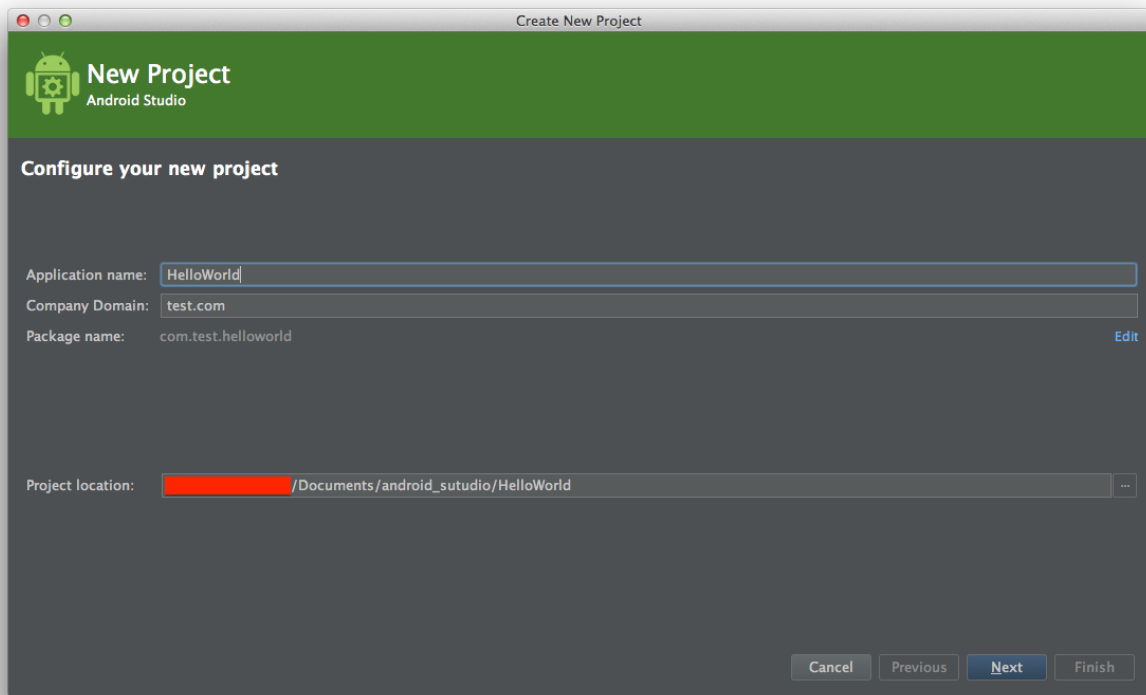
プロジェクトの作成

はじめに、Android Studio で新しくプロジェクトを作成する手順について説明します。この手順にそって行えば、問題なく新しいプロジェクトを作成することが出来ます。

まずは、Welcome 画面の右ペインから New Project... を選択しましょう。すると、Application name と Company Domain 等が設定できる画面に遷移します。

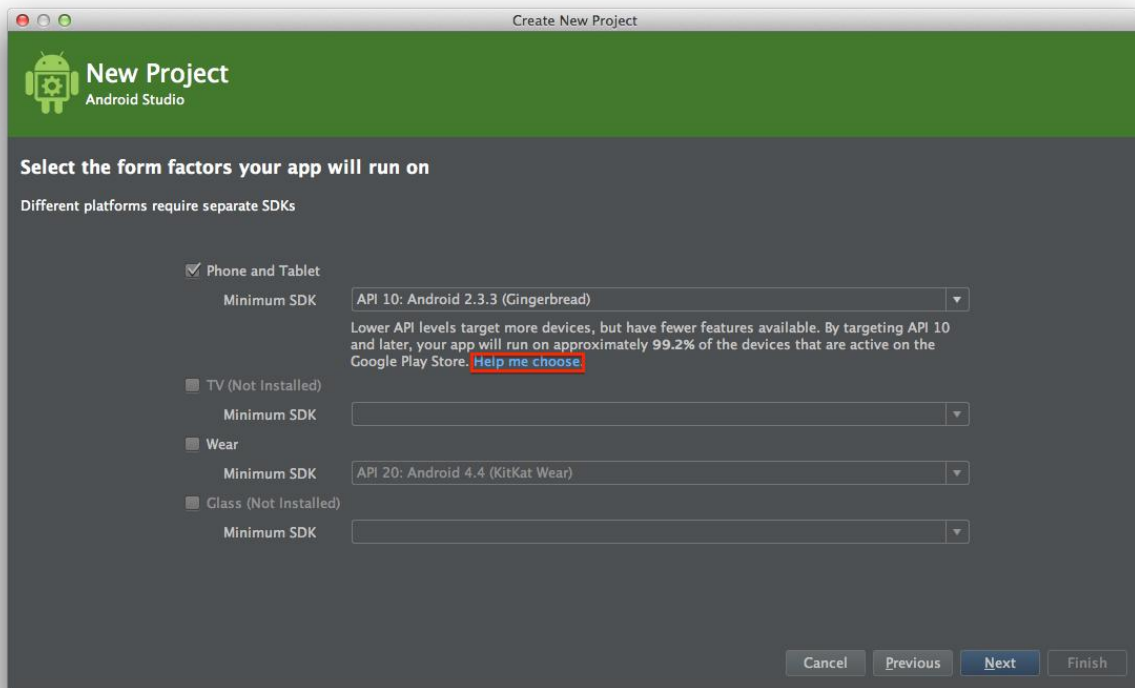
ここで、Application name には好きな名前を、Company Domain には使用するドメインを書きましょう。ここに記述するドメインは、使用するドメインの逆順にするのが Android の公式な Package Name の付け方なのですが、Android Studio では Company Domain にそのままの順番で書くと、逆順にした Package Name を自動で作成してくれます。また、右端に Edit と書かれた部分があり、そこをクリックすると直接編集することも出来ます。

後は Project location を、自分の作りたいディレクトリへ設定して next を押します。

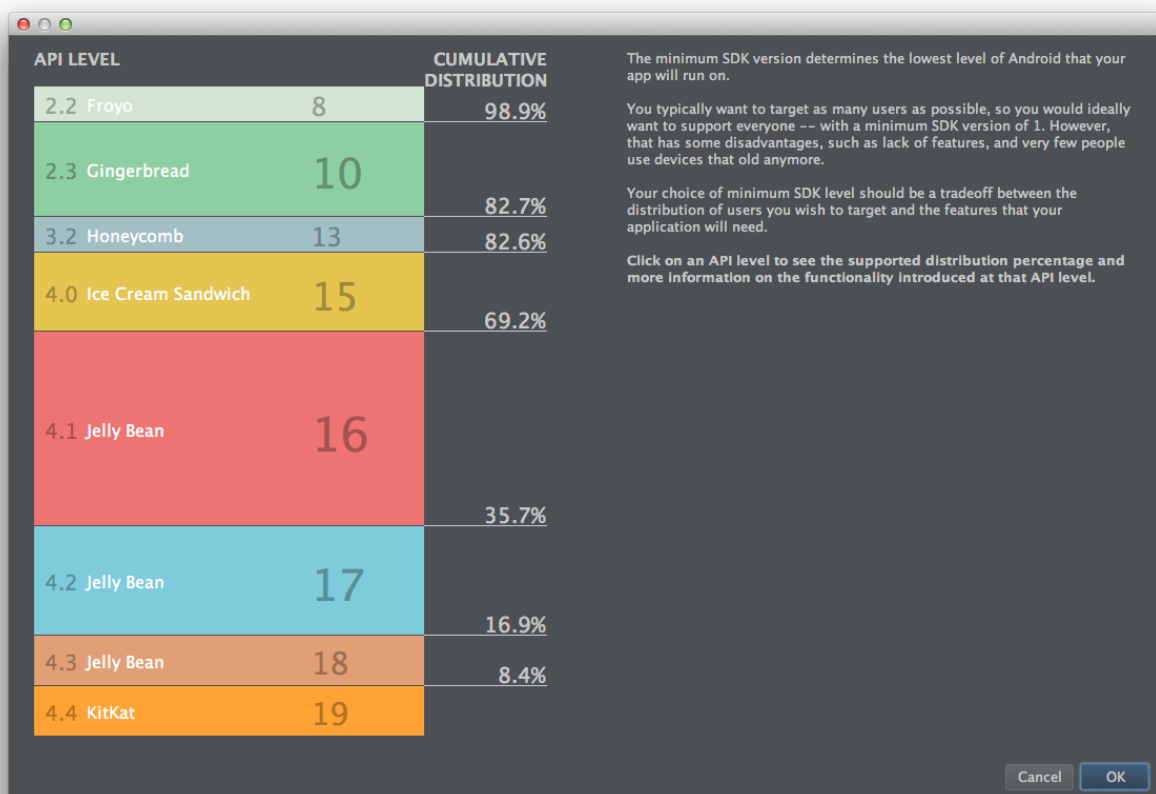


次の画面は Minimum SDK を決める画面です。ここで、開発するアプリをインストールすることのできる OS のバージョンを選択することが出来ます。基本的にそれだけなので、以前までのバージョンの Android Studio では別な画面と同一の場所で設定を行なっていたのですが、なんとこの画面で現在の Android OS のシェアを確認することができるようになったのです！

※ 追記:最新版(0.8.2)では、Android TV、Android Wear、Google Glass の SDK をアプリ内に含むかどうかをこの画面で設定できるようになりました。



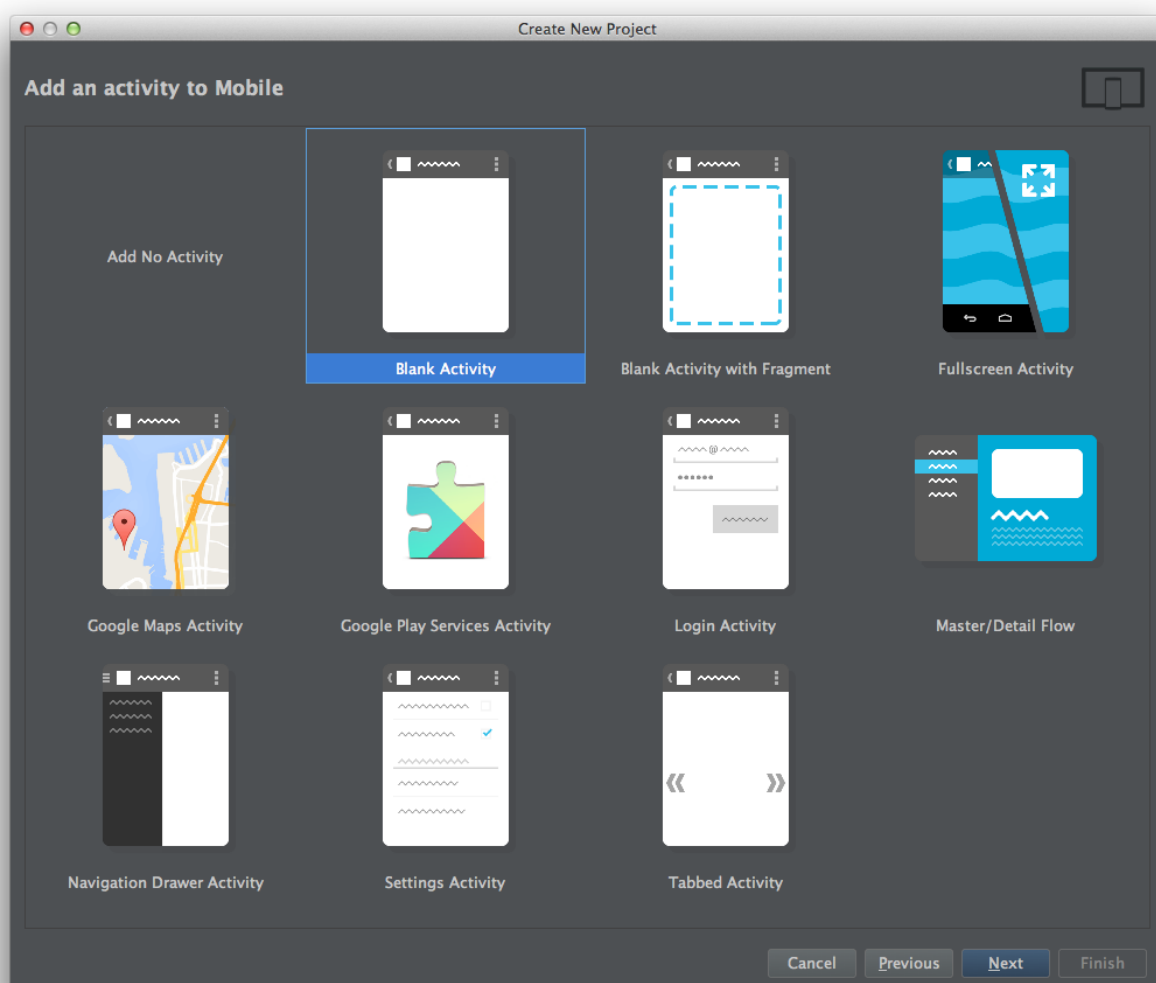
上の画像が、Minimum SDK を選択する画面なのですが、この画面の赤い四角で囲んでいる場所をクリックすることで、シェアを確認する画面に移行することが出来ます。



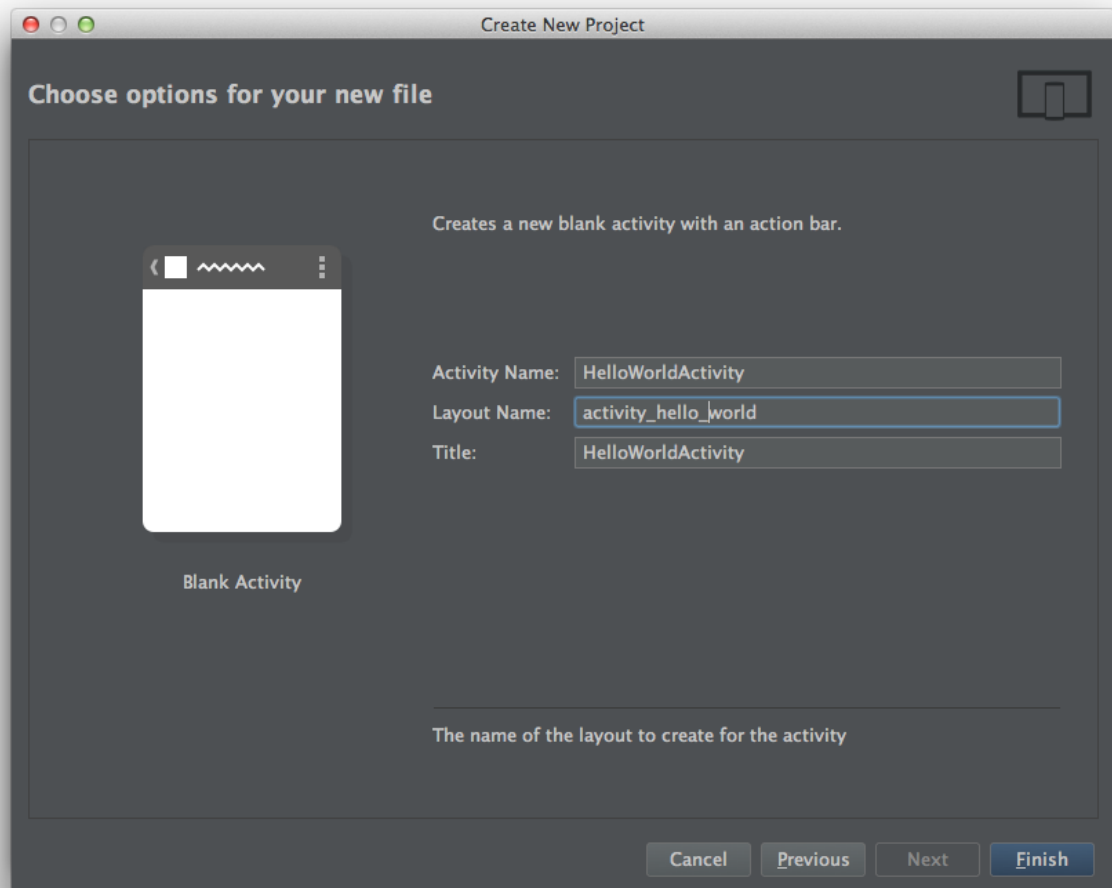
このシェアの画面を見て、自分のアプリをどのバージョンまでサポートするか選択することが出来ます。適した Minimum SDK を設定したのなら、next を押してください。

次の画面では、Activity のタイプを選ぶことが出来ます。とてもグラフィカルになり、どのような物が選べるのかがとてもわかり易くなりました。色々とあり、目移りしてしまいそうですが最初はシンプルに Blank Activity を選択し、next をクリックしましょう。

使ってみて慣れるということも大事なので、別なものを使用してみるのもいいかもしれませんが、最初のうちは混乱するだけなので、シンプルなものから作成していきましょう。もうすでに慣れているという方は、自由に選んでも良いでしょう。



最後に Activity の名前を決めて終了です。Activity の名前は自由に決めてしまっても構いません。適切な名前を ActivityName に入れ、最後に Finish ボタンをクリックすれば、新しいプロジェクトの作成は完了です。



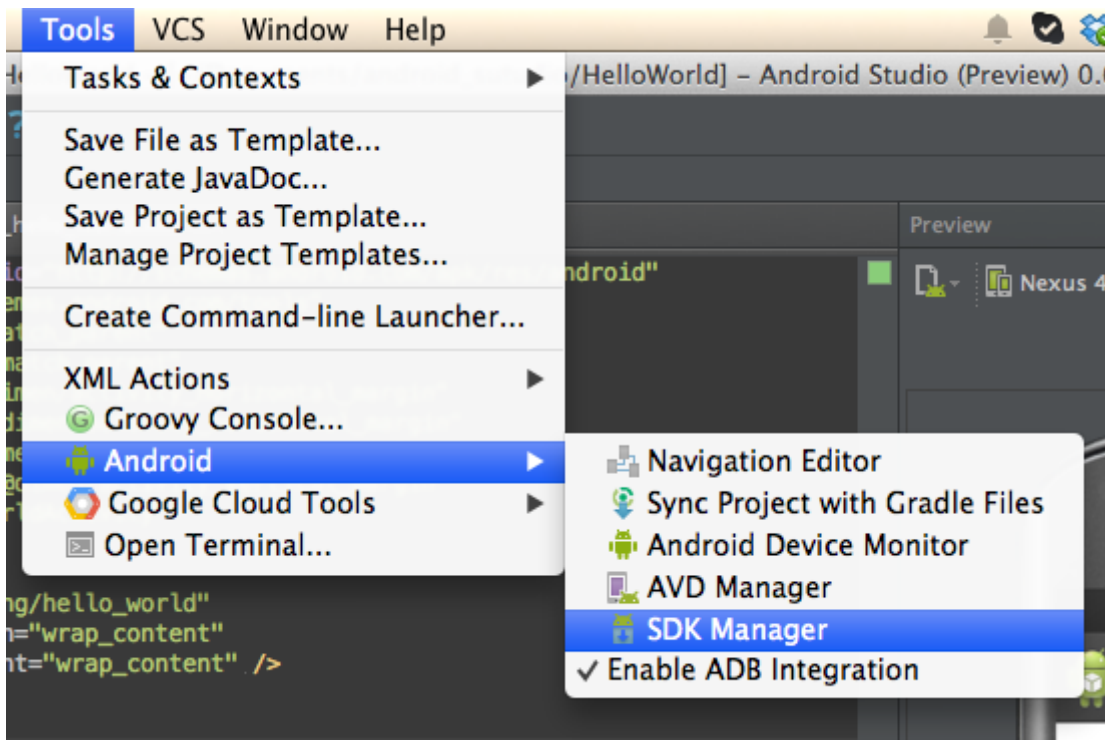
これで新しいプロジェクトは完成です！

後は、実機やエミュレータでビルドしたアプリを動かしてみましよう。画面上部に Hello World という文字が表示されるはずです。

Build に失敗する場合

Error: The SDK Build Tools revision (19.0.3) is too low for project ':app'. Minimum required is 19.1.0

新しく Android Studio をインストールした場合には起こらないと思いますが、Android Studio をバージョンアップして使用する場合には上記のようなエラーが発生し、ビルドに失敗することがあります。そんな時は、まず SDK マネージャを開きましょう。SDK マネージャは、Android Studio 画面上部にあるドroids君のアイコンの隣や、メニューバーの Tools -> Android -> SDK Manager から立ち上げることが出来ます。



SDK マネージャを開いたら、Android SDK Build-tools の 19.1 と書かれたバージョンをインストールしましょう。これでビルドに必要な環境は整いました。しかし、このままではもう一度ビルドしても同じエラーが出てしまうことがあります。その時は build.gradle ファイルを編集する必要があります。

プロジェクトツリーから Project Name -> app -> build.gradle を開きましょう。

このファイルの中の「buildToolsVersion」が、'19.0.3' のままになっているはずなので、'19.1.0' と書き換えましょう。

これで問題なくビルドが通るようになるはずです。

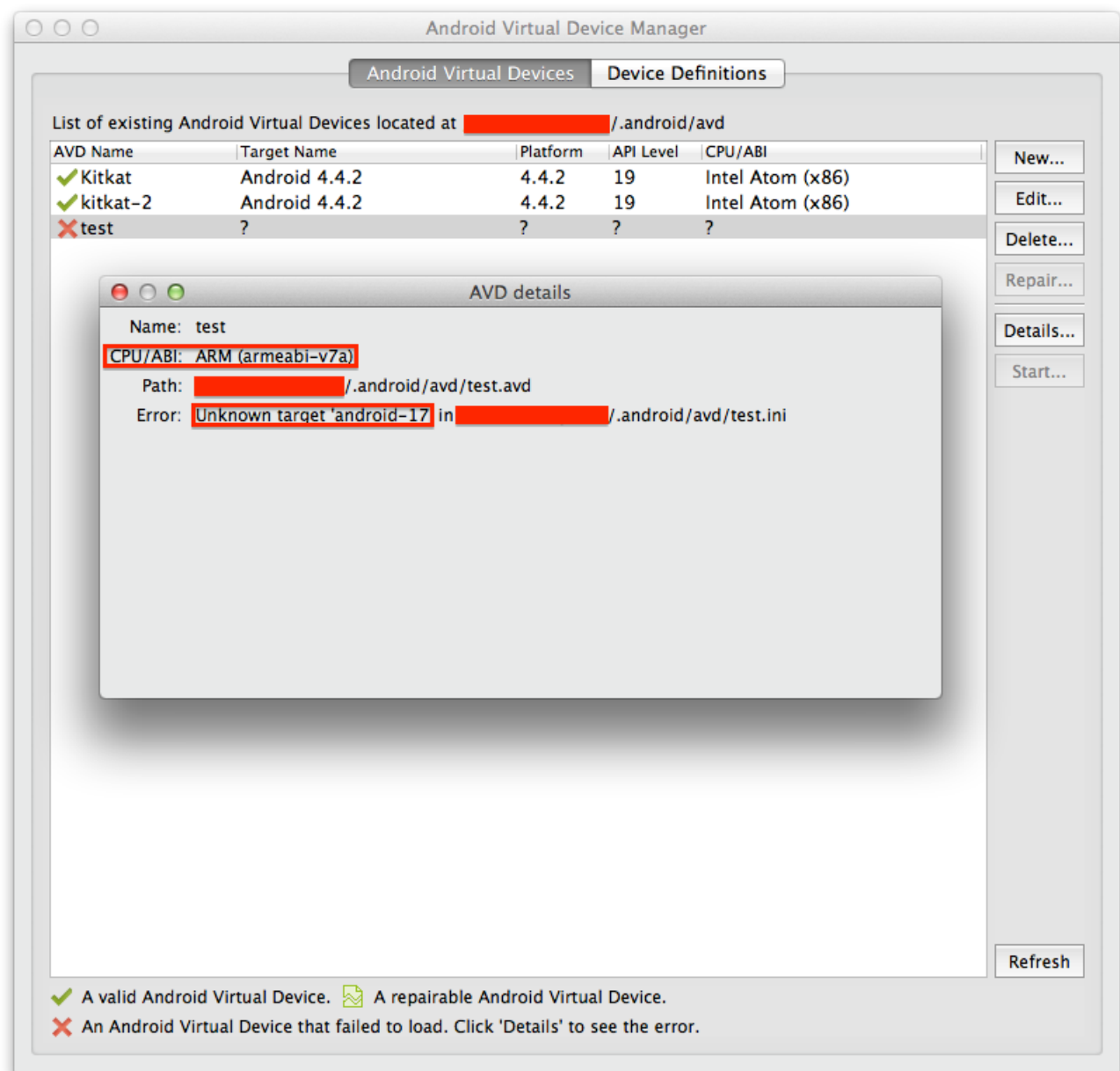
AVD Manager にある壊れたデバイス

AVD マネージャが参照している Path は、(ホームディレクトリ)/.android 以下になるため、以前 Eclipse 等で作成したエミュレータの設定が残っていることがあります。しかし、それらは壊れていて使用することができなくなっていると思います。

その理由としては、新しくインストールした Android Studio では、SDK にインストールされている System Image が足りていないためです。

この問題を解決するには、まず AVD マネージャを立ち上げ、×の付いているデバイスの Details を参照します。そうすると、使用されている System Image がわかるので、それを SDK マネージャの方で探し、インストールしましょう。

例えば以下の画像の場合、Error に書かれている内容と、CPU/ABI にかかっている内容から、API 17 の ARM System Image が必要であることが分かります。



足りなかった System Image のインストールが完了すると、バツだった部分が チェックマークになり、その AVD を使用することができるようになります。もし変化が見られない場合は、AVD マネージャを再起動してみてください。