

# 第4章

## 開発環境 セットアップ

著：柴田文彦

# 4-1

# Androidアプリって どうやって作るの？

著：柴田文彦

Androidアプリケーションは、開発環境とよばれるソフトで作成します。開発環境はパソコンにインストールした後に、いくつかの設定が必要です。慣れた人なら30分～1時間で開発環境を構築できるでしょう。



## 4-1-1 開発環境ってどんなもの？

### 開発環境は、コードを書いたりコンパイルしたりするソフト

Androidのアプリケーションのプログラムは、基本的にプログラミング言語Javaによって記述します。従って、Androidアプリの開発ツールとしては最小限Javaのコンパイラ（コンピューターが理解できるかたちに翻訳するソフト）が必要となります。しかし、それだけではAndroidアプリを作成することはできません。

Androidアプリとして必要な要素を付加したり、Androidデバイス上で動作可能な形式に変換し、デバイスに転送して動かしたりするためには、さまざまなツールが必要となります。こうしたAndroid固有の開発ツールは、それぞれの機能別に用意されています。それらは、Androidアプリ開発環境としてサポートされているWindows、OS X、Linux上で動作するコマンド（コンピューターへの命令）として提供されています。Javaのコンパイラに加えて、こうしたコマンドを組み合わせて使うことで、Androidアプリを作成することができます。

ただし、こうしたツールを一つづつ個別に起動しながらAndroidアプリを作成するのは非常に手間がかかります。通常は、こうした一連のツールの使用を自動化する方法を利用します。そのための方法にも何通りかありますが、もっとも有力なのは、「IDE（Integrated Development Environment=統合開発環境）」と呼ばれるアプリケーションを使うことです。一般的なIDEは、個別の開発ツール群と同様、パソコンの上で動作します。IDEは、個別のツールを適切な順番で組み合わせて利用しながら、大きな目的であるアプリケーションの作成を実行します。いずれにしても、あるソフトウェア（ここではAndroidアプリケーション）の開発に必要なツール類一式を総称して「開発環境」と呼びます。

開発環境にも色々な種類がありますが、この講座では古くからAndroidアプリの開発に利用されてきた「Eclipse（エクリプス）」と呼ばれるものを使います。Eclipse

Eclipseは、Android専用というわけではありません。一般的なJavaアプリケーションや、C++プログラムの開発などにも使える汎用性の高いものです。

をAndroidアプリの開発のために使うには、さまざまなツール類をプラグイン(コラム参照)としてEclipse本体に追加します。そうすることで、EclipseはAndroidアプリ開発用の開発環境としてあつらえられたツールのように機能します。

Eclipseは、Androidの開発者向けサイト(<http://developer.android.com>)から1つのパッケージとしてダウンロードすることができるようになっています。このパッケージを「ADT Bundle(ADTバンドル)」と呼びます。この圧縮されたADT Bundleのパッケージを展開してディスクに保存するだけで、Javaのコンパイラーを除くAndroidの開発環境を整備することができます。その具体的な方法については次の節で順を追って詳しく説明します。いずれにしても、JavaコンパイラーやEclipseといった汎用のソフトウェアを除けば、Android専用のツールやプラグインは、すべてAndroidの開発者サイトから無償でダウンロードすることができます(図1)。

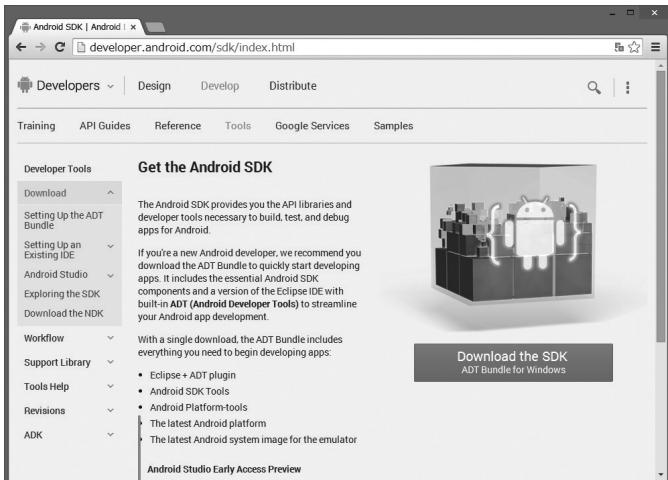


図1:Androidの開発者向けサイトの、開発ツールをダウンロードするためのページ。サイトのトップページから「Develop」、「Tools」のリンクを順にクリックするだけでたどり着くことができる

なお、現状(2014年5月現在)ではADT Bundleとして供給されているEclipseをベースとする開発環境が、Androidアプリ開発のための標準的なものです。しかしADT Bundleの供給元であるグーグルは、それとは別に新たな開発環境「Android Studio(Androidスタジオ)」を準備しています。すでに、先行試用版(Early Access Preview)として供給し、誰でも利用可能な状態になっています。近い将来、このAndroid Studioが、ADT Bundleに代わって、Androidの正式な開発環境となることが予想されます。しかし、現状ではまだ開発途上のものであり、頻繁にバージョンが更新される、やや不安定な環境です。このため、本講座ではADT BundleによるAndroidアプリの開発を説明していきます。

以前は、汎用のEclipseをインストールした後、Android開発用のプラグインを手動でインストールして、開発環境を整える必要がありました。今は、あらかじめAndroid用のプラグインをインストールした状態のEclipseを入手可能になりました。



### プラグイン(Plug-in)

プラグインは、略さずに言えば、「プラグインソフトウェア」のことと、「アドオンソフトウェア」と呼ばれることもあります。プラグインとは、本来「プラグを差し込む」という意味で、プラグを差し込むように簡単に(一定の規格に従って)追加できるソフトウェアのことです。

Androidアプリ開発用のプラグインとしては、アプリ構築に不可欠なツール以外に、ユーザーインターフェースの設計ツール、AndroidのSDK(Software Development Kit:開発キット)の管

理ツール、Androidデバイスがなくても、エミュレーター上でAndroidアプリを起動できるようにする仮想デバイスの管理ツールなどがあります。

Eclipseのプラグインの中には、Eclipseのメニュー表示を日本語に変換してくれるようなものもありますが、この講座では、Androidアプリを開発するための機能に直接関わらないプラグインは使用しない、という前提で話を進めます。



## 4-1-2 開発環境の中身はこんな感じになってる

### 開発ツールはJavaアプリケーション

Androidアプリ開発環境として、ADT Bundleとして提供されているEclipseと、付随するツール類を使う前提で、これまでのところを整理しておきましょう。Androidアプリ開発に必要となる環境を、もう一度確認します。

JDKは、オラクルが提供しているJava言語のコンパイラなどの開発ツール群のことです。

まず言うまでもないことですが、開発環境が動作するベースとなるパソコンとOSが必要です。この講座では、OSとしてはWindowsを使うことになっています。その上で、「JDK(Java Development Kit)」を使います。これについても詳しくは次節で説明します。開発環境であるEclipse自体も一種のJavaアプリケーションなので、動作するにはJava環境が必要です。これはJDKをインストールしてあれば十分です。さらにその上にEclipseと、Android開発ツール類をインストールすることになりますが、これはADT Bundleとして提供されているものを使うので、まとめて1つのソフトウェアとして扱うことができます。この環境全体の構成を簡単に図に示しておきましょう(図2)。

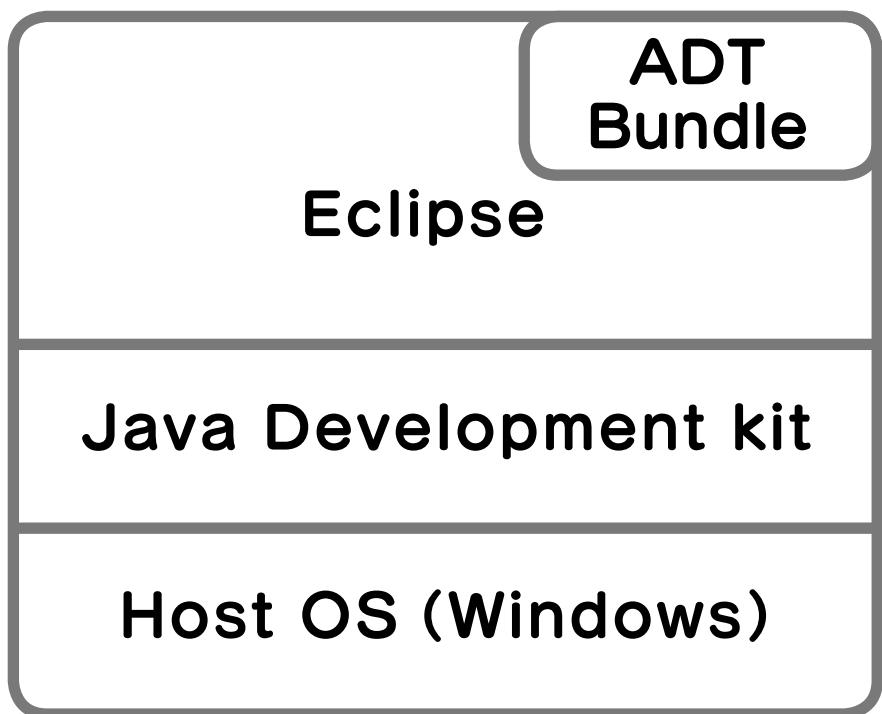


図2:Androidアプリ開発環境全体の大まかな構造。パソコンのOS(Windows)の上にJava環境があり、その上にADTを含んだEclipseが乗っている

次節からは、順にこの開発環境の構築方法について説明します。その前に、そのための手順を簡単に示しておきましょう。手順は大きく4つのステップに分けて考えることができます(図3)。

これらのうち、特にStep 1と3が重要で、Step 2と4は、それに従属するものと見ることができますが、いずれも必要なものです。特にStep 2は、Eclipseの動作には不可欠な操作となります。

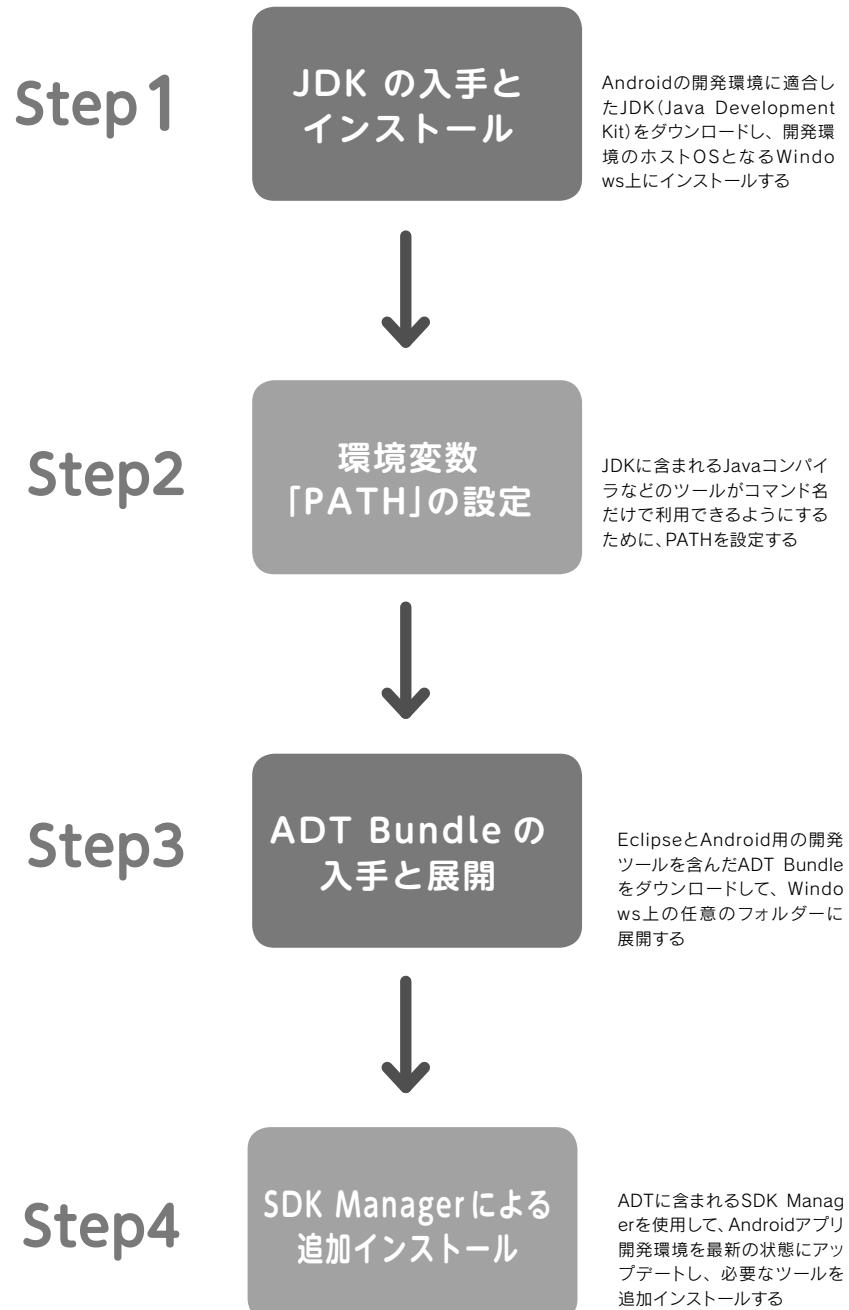
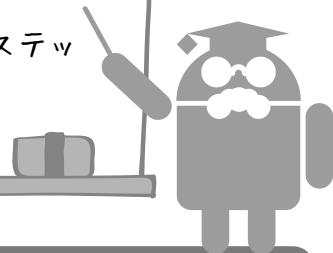


図3:Androidアプリ開発環境をセットアップするための手順。大きく4つのステップに分かれている

# 4-2 開発環境の導入方法は？

著：柴田文彦

開発環境は、Android開発者サイトのSDKをダウンロードするページから入手することができます。通常のアプリケーションにくらべてインストール作業が面倒ですが、最初のステップなのでしっかりとやっておきましょう。



## 4-2-1 Java開発キットのインストール

「JDK」とはJavaの開発キットという意味で、Javaを利用したソフトウェアを開発する際に必要となるツールです。Javaを利用したソフトウェアを実行するだけなら、「JRE(Java Runtime Environment:Java実行環境)」というツールを利用すればよいのですが、開発のためにJDKが不可欠となります。

JDKのバージョンは6ということですが、もう少し細かく言えば、Standard Editionのバージョン6(Java SE 6)を選べば良いことになります。

### JDKの入手とインストール

JDKのインストーラーは、オラクルのサイトから無償でダウンロードすることができます。ただし、JDKにはさまざまなタイプとバージョンの組み合わせがあり、いったいどれをダウンロードしてインストールすれば良いのか、迷ってしまうかもしれません。この場合には、新しければ新しいほど良いという原則は通用しないと考えるべきでしょう。

Android開発者サイトのSDKをダウンロードするページにある「SYSTEM REQUIREMENTS(システム要件)」には、「JDK 6」という指定があります(図1)。

The screenshot shows a web browser displaying the Android SDK setup page at developer.android.com/sdk/index.html. The URL bar shows the full URL. The page has a navigation menu on the left with links like 'Developer Tools', 'Download', 'Setting Up the ADT Bundle', etc. The main content area is titled 'SYSTEM REQUIREMENTS'. It lists 'Operating Systems' (Windows XP/Vista/7, Mac OS X 10.5.8 or later, Linux), 'Eclipse IDE' requirements (Eclipse 3.7.2 or greater, note about Helios support), and 'Other development environments' (JDK 6, Apache Ant 1.8 or later). A note at the bottom states that some Linux distributions may include JDK 1.4 or Gnu Compiler for Java, both of which are not supported for Android development.

図1:Android SDKページにある「SYSTEM REQUIREMENTS」の記述。Android開発環境に適合したJavaのバージョンは6だということがわかる

このシステム要件の指示には、必要なツール類へのリンクがあるので、「JDK 6」をクリックすると、オラクルのJavaダウンロードページを開くことができます。ただし、そこはJavaの最新版(2014年6月現在ではJava SE 8u5)をダウンロードすることを主眼としたものです。SE 6以前のバージョンをダウンロードするには、そのページのいちばん下にある「Previous Releases - Java Archive」の「DOWNLOAD」ボタンをクリックします。それによって開く「Oracle Java Archive」ページで、さらに「Java SE 6」のリンクをクリックすることで、目的のJava SE 6のダウンロードページを開くことができます(図2)

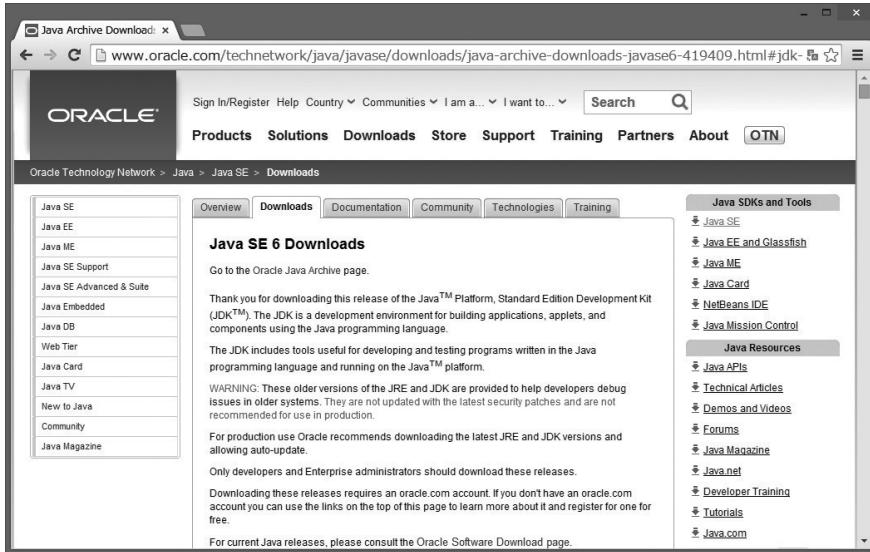


図2:Oracleの「Java SE 6 Download」のページ。ここから、バージョン6として最新の6u45のJDKをダウンロードする

ここでは、さらにSE 6として最終(最新)版の「Java SE Development Kit 6u45」のリンクをクリックして、ページを目的の場所にスクロールさせます。ここでは、まず「Accept License Agreement」のラジオボタンを選択してから、目的のプラットフォーム(OS)の種類を選びます。この講座で使用するWindowsは64ビット版なので、「Windows X64」の右にある「jdk-6u45-windows-x64.exe」のリンクをクリックして、ダウンロードします(図3)。

Java SE Development Kit 6u45		
You must accept the Oracle Binary Code License Agreement for Java SE to download this software.		
Thank you for accepting the Oracle Binary Code License Agreement for Java SE; you may now download this software.		
Product / File Description	File Size	Download
Linux x86	65.46 MB	<a href="#">jdk-6u45-linux-i586.rpm.bin</a>
Linux x86	68.47 MB	<a href="#">jdk-6u45-linux-i586.bin</a>
Linux x64	65.69 MB	<a href="#">jdk-6u45-linux-x64-rpm.bin</a>
Linux x64	68.75 MB	<a href="#">jdk-6u45-linux-x64.bin</a>
Solaris x86	68.38 MB	<a href="#">jdk-6u45-solaris-i586.sh</a>
Solaris x86 (SVR4 package)	120 MB	<a href="#">jdk-6u45-solaris-i586.tar.Z</a>
Solaris x64	8.5 MB	<a href="#">jdk-6u45-solaris-x64.sh</a>
Solaris x64 (SVR4 package)	12.23 MB	<a href="#">jdk-6u45-solaris-x64.tar.Z</a>
Solaris SPARC	73.41 MB	<a href="#">jdk-6u45-solaris-sparc.sh</a>
Solaris SPARC (SVR4 package)	124.74 MB	<a href="#">jdk-6u45-solaris-sparc.tar.Z</a>
Solaris SPARC 64-bit	12.19 MB	<a href="#">jdk-6u45-solaris-sparcv9.sh</a>
Solaris SPARC 64-bit (SVR4 package)	15.49 MB	<a href="#">jdk-6u45-solaris-sparcv9.tar.Z</a>
Windows x86	69.85 MB	<a href="#">jdk-6u45-windows-i586.exe</a>
Windows x64	59.96 MB	<a href="#">jdk-6u45-windows-x64.exe</a>
Linux Intel Itanium	53.89 MB	<a href="#">jdk-6u45-linux-i64-rpm.bin</a>
Linux Intel Itanium	56 MB	<a href="#">jdk-6u45-linux-i64-bin</a>
Windows Intel Itanium	51.72 MB	<a href="#">jdk-6u45-windows-i64.exe</a>

図3:同じJavaバージョンにも、それが動作するプラットフォームによって多くの種類がある。ここでは「Windows X64」用を選ぶ

この後、オラクルのユーザーとしてのログインを求められます。すでにアカウントを登録してある場合には、ユーザー名とパスワードを入力してログインし、ダウンロードを

ダウンロードページを直接開くためのURLは「<http://www.oracle.com/technetwork/java/javase/downloads/java-archive-downloads-javase6-419409.html#jdk-6u45>」となります。

アカウントが未登録なら、その場で作成してから、改めてログインします。Oracleのアカウントは無償で作成することができます。このテキストでは、そのための手順は割愛します。

実行します。

ダウンロードしたJDKは、そのままダブルクリックすれば起動して、インストールが始まります。インストールの途中には、オプションを設定する「カスタムセットアップ」の画面が表示されます(図4)。



図4:JDKインストールの際に設定可能なオプション。「開発ツール」だけをインストールすればいい

このまま、デフォルトの設定でインストールを続行しても構いませんが、Androidアプリの開発には不要な要素も多いので、必要なものだけを選んでインストールすると良いでしょう。選択可能な3つの要素のうち、Androidアプリ開発に必要なのは「開発ツール」だけです。開発ツール以外とを「×」(インストールしない設定)にして「次へ」ボタンをクリックすれば、実際のインストールが始まり、しばらく待てば完了します。

## 忘れがちな環境変数「PATH」の設定

JDKをインストールした後は、やや面倒な作業が必要です。JDKは一般的なアプリケーションとは異なり、細かなツールの集合であって、アイコンをダブルクリックすれば起動するという類いのものではありません。そこで、これらのツールがインストールされた場所をWindowsのシステムに教えてあげる必要があるのです。それが、ここで言う環境変数「PATH(パス)」の設定です。

JDKがインストールされた場所は、実は上の図4で示したインストール時のオプションを選択するダイアログに表示されていました。そのダイアログで変更することもできるのですが、デフォルトでは「C:\Program Files\Java\jdk 1.6.0\_45」という場所になっています。

まず、Windowsのエクスプローラーで、この場所を開いてみましょう。「PC」→「OS (C:)」→「Program Files」→「Java」→「jdk1.6.0\_45」と順にたどっていくと、その下に「bin」というフォルダーがあるので、それを開きます。この「bin」こそが、

JDK本体が格納されているフォルダーです。の中には、「java.exe」や「javax.exe」といったファイルが並んでいることが確認できます。

このフォルダーを開いたら、アドレスバーの上で右ボタン(またはControlキーを押しながら左ボタン)をクリックしてメニューを表示させ、「アドレスをテキストとしてコピー」を選びます(図5)。これにより、JDK本体のフォルダーの場所が、テキスト形式でクリップボードにコピーされました。

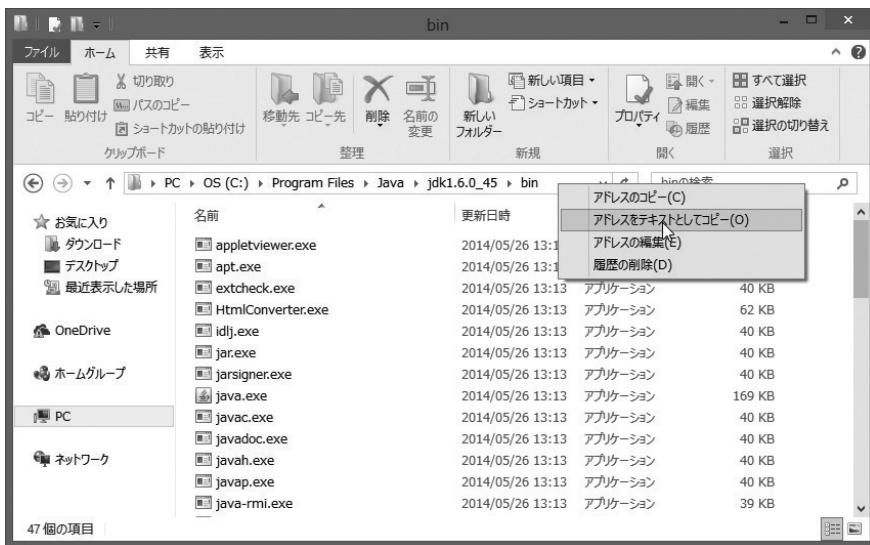


図5:[C:\Program Files\Java\jdk 1.6.0\_45\bin]を開き、アドレスバーで右ボタンクリックして、パスをテキストとしてコピーする

次に、この場所を表すテキストを、システムの環境変数に書き込みます。そのためには、まずデスクトップ左下にある「スタート」を右クリックしてメニューを開き、「システム」を選びます(図6)。

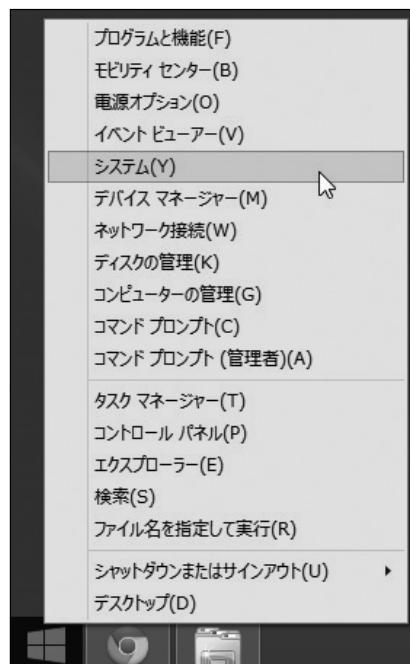


図6:デスクトップ左下の「スタート」ボタンを右クリックして開くメニューから「システム」を選択する

その結果、「システム」というウインドウが開きます。ここではその左側のメニューにある「システムの詳細設定」のリンクをクリックします(図7)。



図7:「システム」ウィンドウの左側にある「システムの詳細設定」をクリックする

すると、こんどは「システムのプロパティ」というダイアログが開きます。この中は5つのタブに分かれています。ここでは、その中から「詳細設定」タブをクリックして表示し、その中の「環境変数(N)...」ボタンをクリックします(図8)。

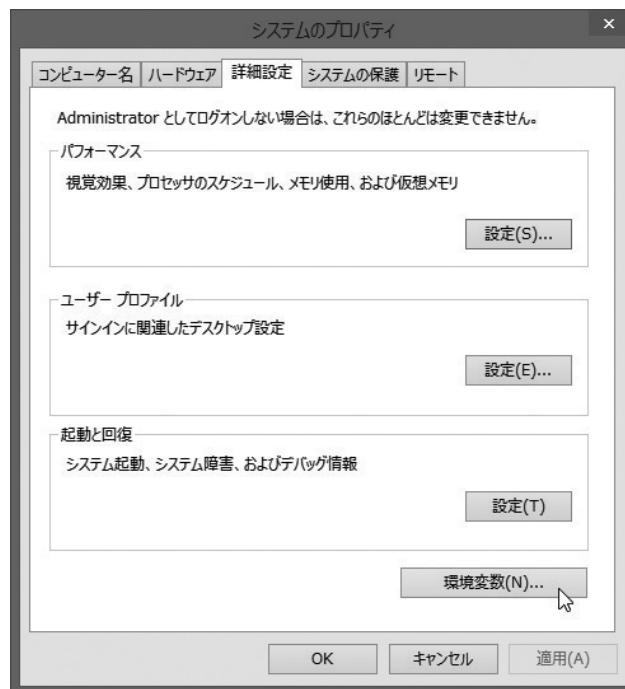


図8:「システムのプロパティ」ダイアログの「詳細設定」タブにある「環境変数(N)...」ボタンをクリックする

「システム」の開き方はWindows 8.1の場合で解説していますが、7以前でも同様に操作できます。

次に開くのは「環境変数」ダイアログです。このダイアログの中には、あらかじめ「TEMP」と「TMP」という2つのユーザー環境変数が登録されているのが分かるでしょう。ここに「PATH」を追加するために「新規(N)...」ボタンをクリックします(図9)。



図9:「環境変数」ダイアログの「新規(N)...」ボタンをクリックする

すると「新しいユーザー変数」というダイアログが表示されます。ここではその中の「変数名」欄に「PATH」とタイプ入力します。そして「変数値」欄には、JDK本体の場所を入力すれば良いわけです。先ほどの作業で、この場所はクリップボードに入っているので、それを貼り付けます。この枠の中で右クリックしてメニューを表示させ、「貼り付け」を選ぶだけです(図10)。

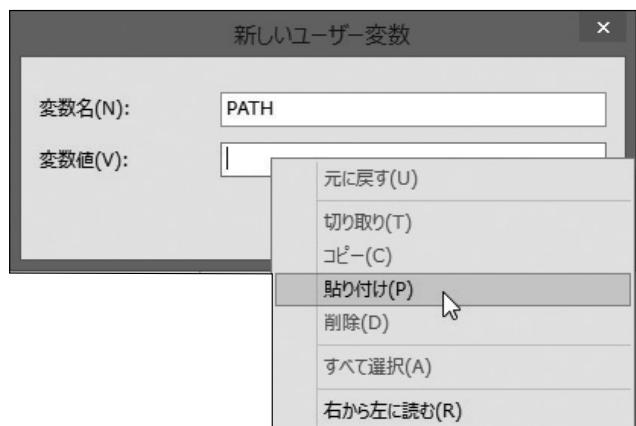


図10:「新しいユーザー変数」ダイアログでは、「変数名」に「PATH」と入力し「変数値」には、すでにコピーしてある場所のテキストを貼り付ける

これで、「変数名」欄にJDKのコマンドが格納されているフォルダーの場所が入力されるはずです(図11)。その後「OK」をクリックして、順にダイアログを閉じていきましょう。

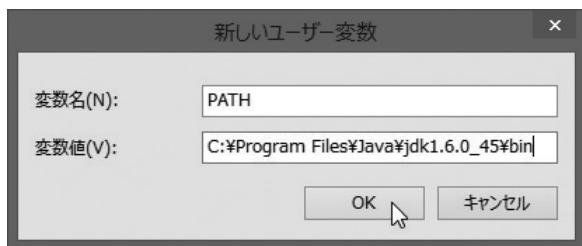


図11:「変数名」と「変数値」を確認して「OK」をクリックする。以下順に「OK」をクリックしてダイアログを閉じていく

環境変数を設定したら、それが正しく機能していることを確かめておきましょう。もういちど、「スタート」を右クリックしてメニューを表示させ、こんどは「コマンドプロンプト」を選びます。すると「コマンドプロンプト」というウインドウが表示されます。ここで、「java -version」とタイプ入力してから「Enter」キーを押しましょう。画面上に「java version "1.6.0\_45"」から始まるバージョン情報が表示されれば、環境変数は正しく設定されていることが分かります(図12)。

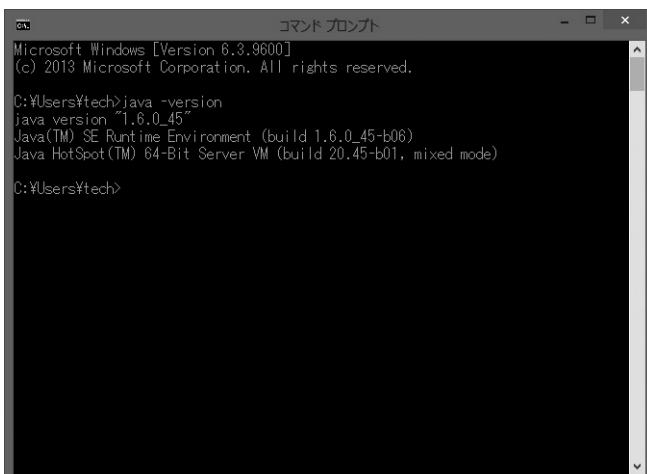


図12:コマンドプロンプトを開き、「java -version」とタイプ入力して、JDKへのパスが正しく設定されていることを確認する

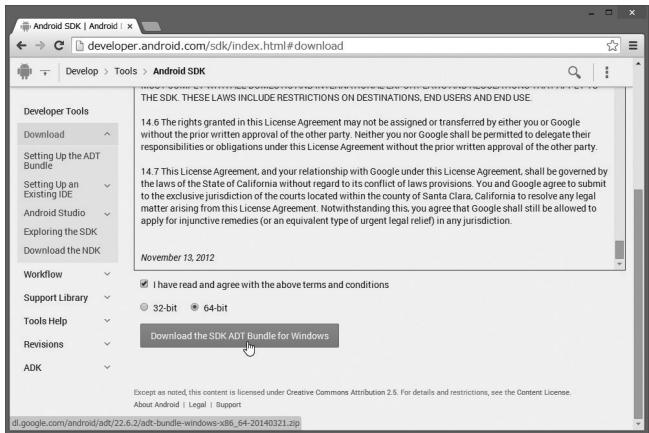
ここでエラーが発生した場合は、これまでの設定をもう一度最初から見直してください。

## ADT Bundleのダウンロードとインストール

この講座では「ADT Bundle」として提供されているEclipse+ADTを標準的なAndroidアプリの開発環境として使用します。

ADT Bundleのインストーラーは、すでに示したAndroidの開発者サイトの「Develop」→「Tools」のページにある「download the Android SDK」のリンクをクリックすると開く「Get the Android SDK」ページ(4-1の図1)からダウンロードします。そのページで「Download the SDK」ボタンをクリックすると、「Terms and Conditions」の文面が表示されます。その条件に同意したことを示すために、「I have read and agree with ...」のチェックボックスをオンにし、「32-bit」または「64-bit」のラジオボタンから、いずれかを選び、「Download the SDK ADT Bundle for Windows」ボタンをクリックすれば、選択した条件の最新版が自動的にダウンロードされます(図13)。ここでは、もちろん「64-bit」を選

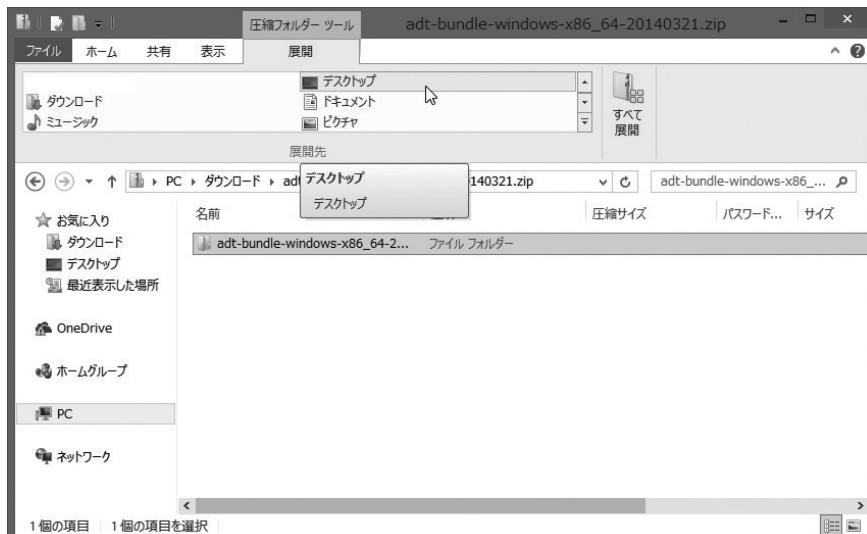
択します。



**図13:** ADT Bundleをダウンロードするには、「Terms and Conditions」の文面に同意し、32または64ビットのいずれかを選択する(本講座では64ビットを使用)

ここからは、Windowsの「ダウンロード」フォルダーに「adt-bundle-windows-x86\_64-20140321.zip」が保存されているものとして話を進めます。ダウンロードされたADT Bundleは、実行ファイルではなく、zip形式の圧縮ファイルになっています。

Windowsのエクスプローラーで「ダウンロード」フォルダーを開き、このzipファイルを展開します。zipファイルを選択すると、エクスプローラーのリボンに、展開先の候補と「すべて展開」というボタンが表示されます(図14)。ここでは、展開先として「デスクトップ」を選んで展開しました。もちろん、その場所を憶えていることができれば、別の場所でもかまいません。



**図14:** ダウンロードしたADT Bundleのzipファイルは、Windowsのエクスプローラーを使って、デスクトップなどに「すべて展開」する

## ADT Bundleの起動チェック

この中には、非常に多くのフォルダ／ファイルが含まれているため(1万項目以上)、展開にはそれなりの時間がかかります。この講座の標準パソコンでも10分弱は見ておく必要があるでしょう。

展開後のフォルダーは、元のzipファイルと同じ「adt-bundle-windows-x86\_64-20140321」という名前になります。それを開くと、その中に「eclipse」というフォル

XPなど32bitバージョンのWindowsを使用している場合は「32-bit」でもかまいません。

Windows 7以降のPCでは、ファイルを右クリックして「すべて展開」としてもOKです。

ダーがあるので、さらにそれを開きます(図15)。その中の「eclipse.exe」をダブルクリックすれば、ADT BundleとしてのEclipseを起動することができます。

Eclipseを起動すると、まず「Workspace Launcher」という名前のダイアログが開き、Eclipseのワークスペースの場所をどこに設定するのか聞いてきます(図16)。デフォルトはユーザーごとのフォルダーの下の「workspace」フォルダーです。通常はそのままで良いでしょう。

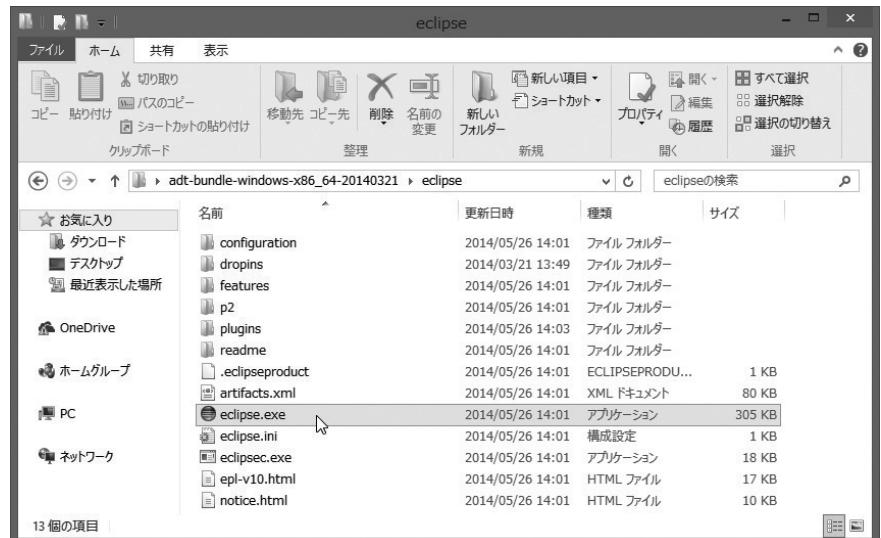


図15:展開したADT Bundleのフォルダーに含まれる「eclipse」フォルダーの中にある「eclipse.exe」をダブルクリックして起動する

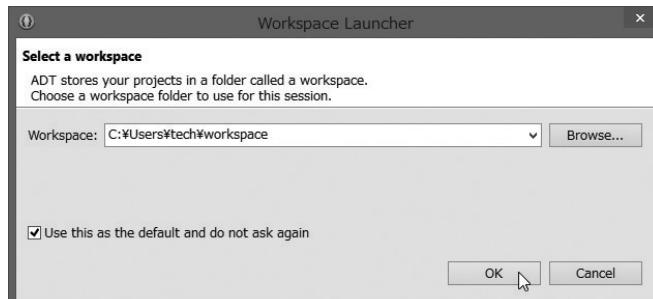


図16:Eclipseを起動する  
と「Workspace Launcher」ダイアログが開くので  
ワークスペースの位置を  
確認して「OK」をクリック  
する

次にEclipseは、というよりも、その中に含まれているADTが「Welcome to Android Development」というダイアログを表示し、「Contribute Usage Statistics?」と聞いてきます(図17)。これは、Android SDKの利用状況に関する情報をグーグルが匿名で収集することを許可するかどうかを設定するものです。



この情報は、SDKを進化させるために利用されることが多いですが、どちらを選ぶかは任意です。

図17:Eclipseの起動途中で表示される「Welcome to Android Development」ダイアログでは、統計情報の収集に協力するかどうかを選ぶ

「Yes」または「No」のラジオボタンを選んでから「Finish」ボタンをクリックすれば、Eclipseが起動します。

最初にEclipseを起動した際には、「Android IDE」というタブがついたビュー(画面)が1つだけ表示されています(図18)。

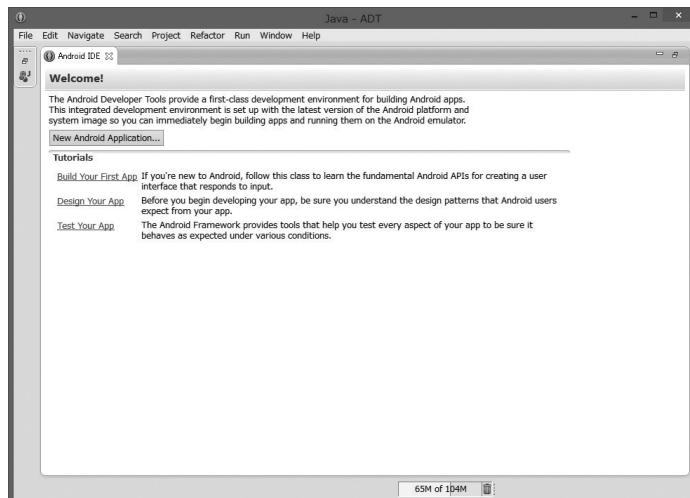


図18:Eclipseの起動が完了すると、「Android IDE」というタブの付いた「Welcome!」ビューが表示される。これが表示されれば環境はほぼ整ったと言える

の中には「Welcome!」と表示されていて、EclipseとADTを使うのが初めての人ための簡単なガイドが示されています。

これが表示されれば、JDKを含め、ADT Bundleのインストールがうまくいったことが分かります。ここでは、「Android IDE」タブの右にある「×」マークをクリックして、このビューを閉じておきましょう。



## 4-2-2 SDKマネージャーで追加のインストールをしよう

### SDKマネージャーの起動とパッケージのインストール

SDKには、Androidのアプリにリンクして使える実行可能なプログラムだけでなく、そのソースコードやドキュメントも含まれています。

ADT Bundleには、Eclipse本体とADTだけでなく、「Android SDK (Software Development Kit)」と呼ばれる開発環境の重要なコンポーネントが含まれています。このSDKでは、Androidならではの機能を実現するためのプログラムがライブラリーとして含まれています。このSDKによって、Androidアプリと、一般的なJavaアプリの違いが生まれると考えれば良いでしょう。

SDKは、OSとしてのAndroid本体のバージョンに対応したものが常に用意されます。したがって、開発ツールそのものとは独立して、Androidのバージョンが更新されれば、SDKも更新されます。ADT Bundleに含まれている開発ツールの中には、そのSDKを管理する専用のものも含まれています。それが「SDKマネージャー (SDK Manager)」です。

SDKマネージャーは、Eclipse内から起動することができます。そのための方法は2つあります。1つは、Eclipseのツールバーに配置された「Android SDK Manager」ボタンをクリックする方法(図19)、もう1つは、Eclipseの「Window」メニューから「Android SDK Manager」を選択する方法です(図20)。

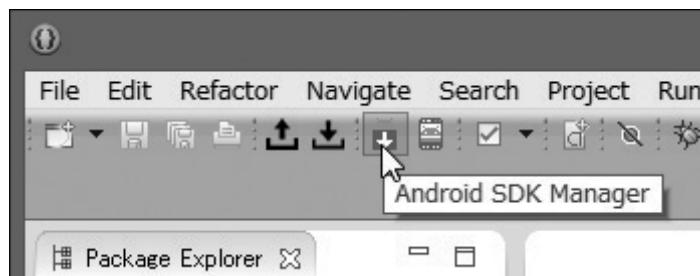


図19:SDKマネージャーを起動する1つ目の方法。Eclipseのツールバーにある「Android SDK Manager」ボタンをクリックする

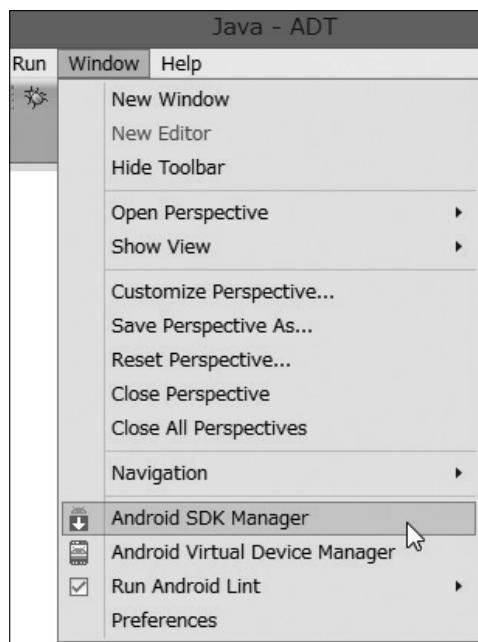


図20:SDKマネージャーを起動するもう1つの方法。Eclipseの「Window」メニューから「Android SDK Manager」を選択する

結果は、もちろんどちらも同じで「Android SDK Manager」というウインドウが開きます(図21)。

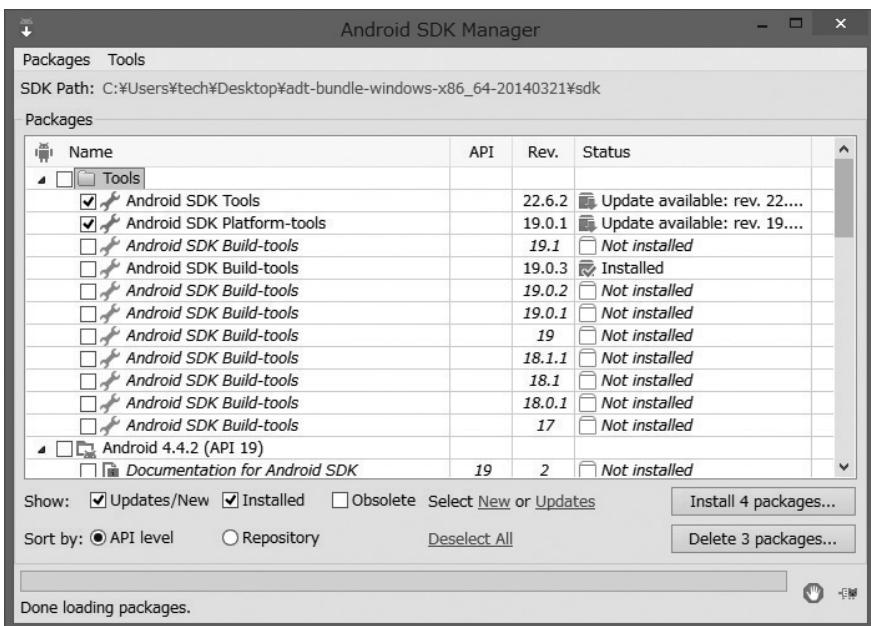
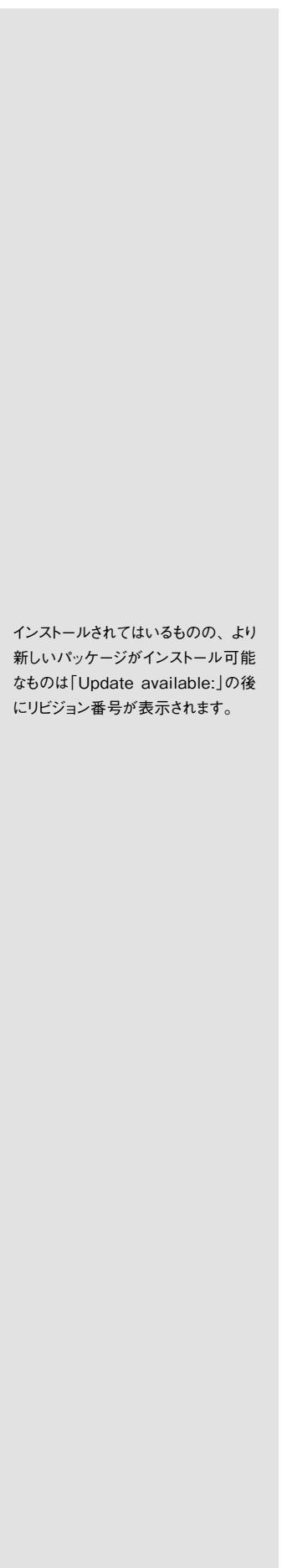


図21:SDKマネージャーの初期画面。インストールが推奨されるパッケージの前には、自動的にチェックマークが付いている

このウインドウには、SDKとしてインストール可能なパッケージがリストアップされていて、それぞれのバージョン(リビジョン番号)や状態も表示されています。すでにインストールされているものは「Installed」、まだインストールされていないものは「Not installed」という状態になっています。

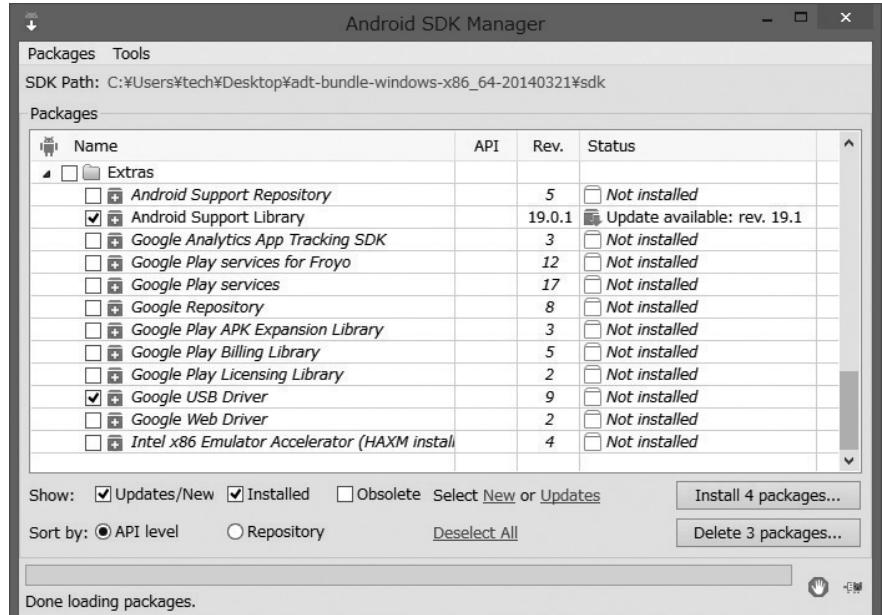
いずれにしても、パッケージの前のチェックボックスをオンにして「Install...」ボタンをクリックすることで、必要なパッケージをインストールできます。SDKマネージャーを起動するたびにインストールが推奨されるパッケージには、自動的にチェックマークが付きます。

ここで示すデフォルトの状態では、4つのパッケージに自動的にチェックマークが付いているので、インストール用のボタンには「Install 4 packages...」と表示されています。その下のボタンはパッケージを削除するためのもので、チェックされた4つのうちの3つが削除可能なので、「Delete 3 packages...」と表示されています。ただし、削除可能だからといって削除してしまうと、Androidアプリ開発環境が正常に動作してなくなってしまう場合がほとんどでしょう。この削除ボタンは、自分が何をしようとしているかを十分に理解し、注意して使う必要があります。



インストールされてはいるものの、より新しいパッケージがインストール可能なものは「Update available:」の後にリビジョン番号が表示されます。

図に示した状態では、まず「Android SDK Tools」と「Android SDK Platform-tools」の2つのパッケージに新しいリビジョン(バージョン)があって、インストール可能な状態になっていることが分かります。また、このリストをスクロールして下方を見ると、「Extras」というグループもあります。その中では「Android Support Library」がアップデート可能であり、「Google USB Driver」はまだインストールされていませんが、新規インストールが推奨されている状態です(図22)。



この例では「Install 4 package...」ボタンが「4」という数字になっています。この部分は、パッケージのチェック数に応じて変わります。

図22:SDKマネージャーでは、「Extras」のグループに含まれるパッケージも、2つがチェックされていてインストール推奨状態になっている

チェックマークの付いたパッケージをインストールするには、そのまま「Install 4 package...」ボタンをクリックします。

すると「Choose Packages to Install」というダイアログが表示され、ライセンス条項を承諾するかどうかを聞いてきます(図23)。インストールを実行するには、「Accept License」のラジオボタンを選択します。すると、インストールしようとしているパッケージの一覧の前にあるマークが緑色のチェックマークに変わるので「Install」ボタンをクリックします(図24)。

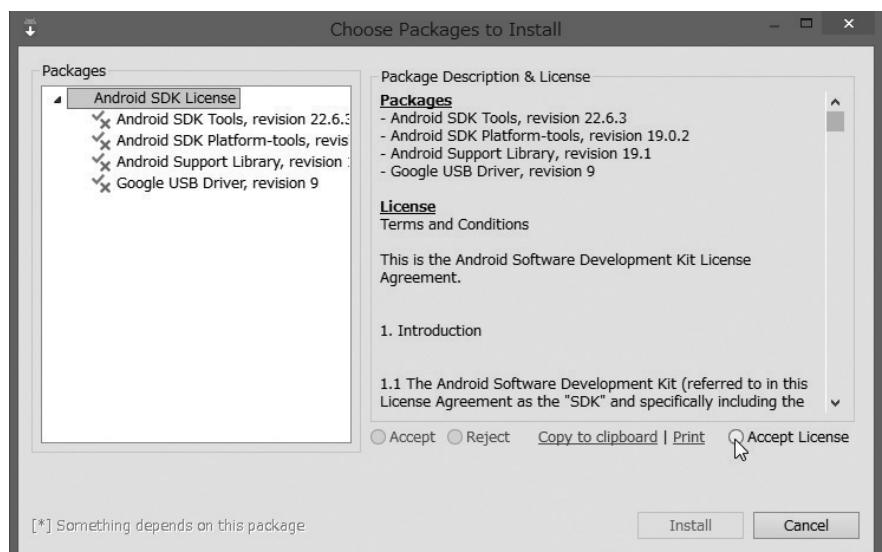


図23:選択したパッケージをインストールするには、ダイアログの右下近くにある「Accept License」のラジオボタンを選択状態にする

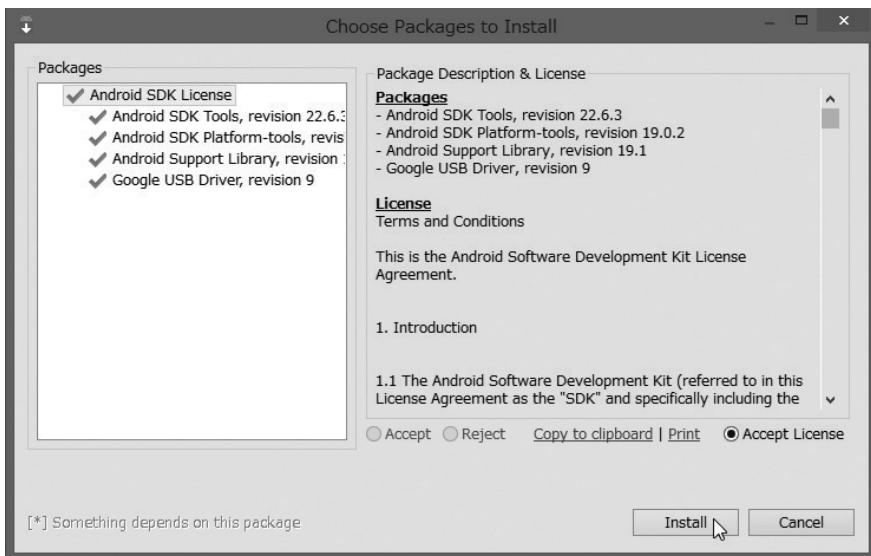


図24:ダイアログ左上の選択したパッケージリストの前に緑のチェックマークが付いたら、「Install」をクリックしてインストールできる

その後、選択されたパッケージをダウンロードしながらインストール作業が始まります。インストールが終了すると、SDKがアップデートされた場合は、SDKマネージャーを再起動するように促すダイアログが表示されるので、「OK」ボタンをクリックして閉じておきます(図25)。

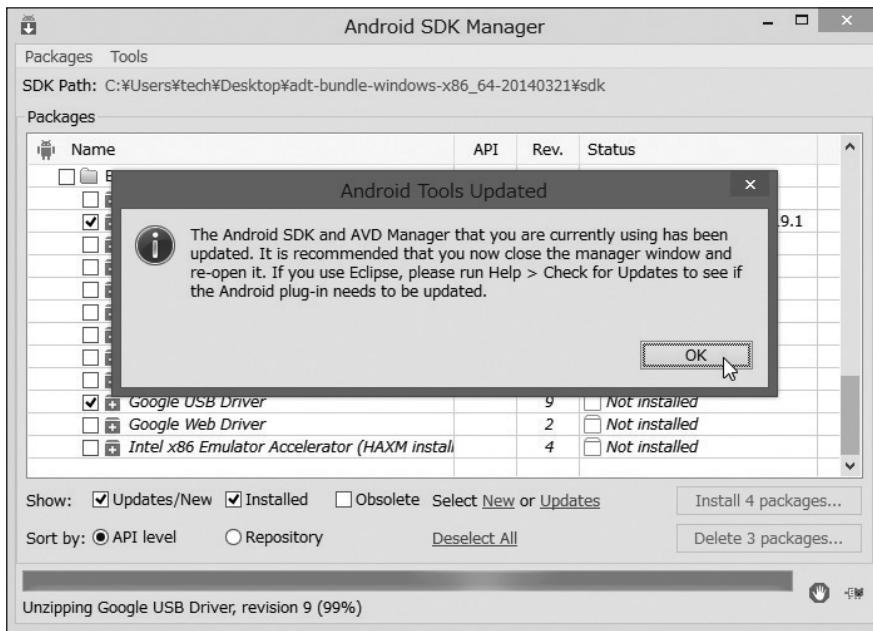


図25:SDKマネージャーによるパッケージのアップデートには、多少の時間がかかる。終了すると、必要に応じてSDKマネージャーを再起動するように促される

選択した項目の数や内容にもよりますが、しばらく時間がかかる場合もあります。



### プラグインのアップデート

このSDKマネージャーでアップデートできないプラグインなどについては、Eclipseの「Help」メニューにある「Check for Upda

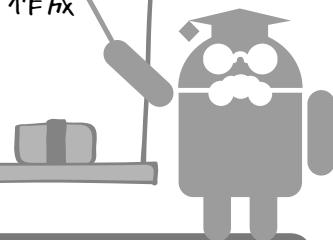
tes」を選ぶことで、可能なアップデートがあるかどうかをチェックし、必要に応じてアップデートすることができます。

# 4-3

# 開発環境を動かしてみたい！

著：柴田文彦

早速ですが、Androidアプリ開発環境を実際に動かして、簡単なアプリケーションを作成してみましょう！さらに、作成したアプリケーションをテストしてみてください。



## 4-3-1 アプリ制作の第一歩「プロジェクト」の作成

### 新規Androidプロジェクトを作ってみよう

Androidのアプリケーションは、Eclipseから見ると1つのプロジェクトとして作成します。言い換えば、1つのプロジェクトには、Androidアプリケーションを構成するのに必要なすべてのファイル、あるいはリソースが含まれていることになります。Eclipseのプロジェクトには、色々な種類がありますが、Androidアプリケーションを作成するためには、そのためのプロジェクトの種類「Android Application Project」が用意されています。これは、Eclipse本体に由来するものではなく、ADTによって追加されたものです。

新規のAndroidアプリケーションプロジェクトを作成するには、Eclipseの「File」メニューから「New」サブメニューにある「Android Application Project」を選択します（図1）。

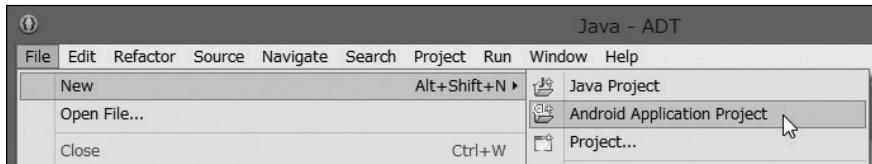


図1：新たなAndroidアプリのプロジェクトを作成するために、「File」メニューの「New」サブメニューから「Android Application Project」を選択する

このメニュー操作には、「Alt+Shift+N」のショートカットが割り振られています。それほど頻繁に使うメニュー項目でもありませんが、覚えておくと素早く操作できます。いずれにしても、この結果「New Android Application」というタイトルの、ウィザード（対話形式で設定できる）状のダイアログが表示されます（図2）。



図2:新規のAndroidアプリのプロジェクトの作成するために、アプリの名前、サポートするSDKのバージョンなどを入力するステップ

いちばん上の「Application Name:」欄には、これから作成するアプリケーションの名前を入力します。この例ではアプリの名前として「My First App」と入力しました。その結果「Project Name:」欄には、そこからスペースを取り除いた「MyFirstApp」が自動的に入力されました。また。「Package Name:」には、そのプロジェクト名をすべて小文字にした「myfirstapp」を「com.example.」の後ろに付けたものが自動的に入力されます。

このパッケージ名は、アプリケーションを構成するJavaプログラムとしてのパッケージ名のことです。これによって、そのプログラムが一意に識別されるものでなければなりません。またプロジェクトの中では、この文字列からディレクトリ構造が生成され、ソースコードなどは、その中に格納されます。

このパッケージ名としては、通常はいわゆる「逆ドメイン記法(reverse domain notation)」で記述します。一般的にはアプリを開発する会社、組織のインターネットのドメイン名を逆から書いたものとします。個人で開発する場合には、自分で作るアプリケーションの中で一貫性があり、他の開発者と混同する恐れのない文字列であれば、どのようなものでも良いでしょう。

次に1行分のスペースが空いて、SDKのバージョンに関する設定があります。その中のいちばん上の「Minimum Required SDK:」は、このアプリケーションがサポートする最も古いAPIのバージョンを指定します。それによりこのアプリは、そのバージョン以降のAndroidでの動作が可能となります。これは、このアプリの中では、こ

アプリケーションの名前を入力すると、それに応じてその下にある「Project Name:」と「Package Name:」欄も、自動的に入力されます。

このパッケージ名の例では、アプリ名を表す文字列の前は「jp.techinstitute.ti12345.」とします。この「ti12345」の部分には、この講座の受講者番号と対応する文字列を各自指定してください。

「Minimum Required SDK:」について、デフォルトでは「API 8: Android 2.2 (Froyo)」という、かなり古いバージョンが指定されています。それによってこのアプリが動作可能なAndroidデバイスの種類、数は多くなりますが、指定したバージョン以降に新しく付加された機能は使えません。

で指定したAPIよりも新しいバージョンのAPIは使用できないことを意味します。その下の「Target SDK:」には、上の「Minimum SDK」とは逆に、このアプリケーションが動作する最新のAndroidバージョンを指定します。デフォルトでは、現時点で最新の「API 19: Android 4.4 (KitKat)」が指定されています。その下の「Compile With:」には、アプリケーションをコンパイルし、ビルドするSDKのバージョンを指定します。これは、開発環境にインストールされているSDKの中の最新のものが自動的に選択されます。ここでも「API 19: Android 4.4 (KitKat)」がデフォルトで指定されています。以上の設定によって、「Minimum Required SDK:」で指定したバージョン(Android 2.2)から「Target SDK:」で指定したバージョン(Android 4.4)の間(両者を含む)のAndroidプラットフォームで動作するアプリが、「Compile With:」で指定したSDK(Android 4.4)を使って作成されることになります。

さらにその下には「Theme:」というポップアップメニューがあり、デフォルトでは「Holo Light with Dark Action Bar」が指定されています。これは、アプリ画面の見栄えに関する基本的な「テーマ」を設定するものです。この場合、白っぽい背景に濃いグレーの文字、黒っぽいアクションバー、といったテーマとなっています。この画面での設定ができたら「Next >」ボタンをクリックして、次の設定に移ります。こんどは「Configure Project」というサブタイトルの設定画面で、複数のチェックボックスが並んだものとなっています(図3)。

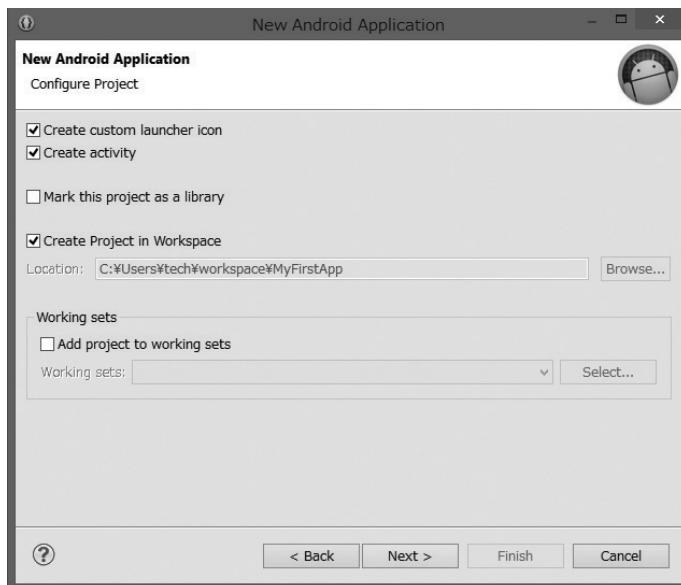


図3:作成するAndroidアプリのプロジェクトについて、細かなオプションを設定する「Configure Project」のステップ

いちばん上の「Create custom launcher icon」は、これから作るアプリに固有のカスタムなアイコンを作成する場合にオンにします。その場合、実際のアイコンは、次のステップで指定することになります。その下の「Create activity」は、このプロジェクトの中に、最初からアクティビティを作成する場合にはオンにします。アクティビティは、Androidアプリが動作するために不可欠なオブジェクトです。後で手動で追加することもできます。その場合にはオフでもかまいませんが、通常はオンのままで良いでしょう。

その下には、1行分のスペースを空けて「Mark this project as a library」があります。一般的Androidアプリを作成する場合には、そのままオフにしておきます。さらに1行空けて下にある「Create Project in Workspace」は、Eclipseのワークスペースの中にプロジェクトを作成する場合にオンにします。このEclipseのワークスペースとは、Eclipseの起動時に場所を確認してくる「workspace」のことです。デフォルトはオンです。これをオフにすれば、プロジェクトの保存場所を、ワークスペース以外に指定できます。

いちばん下の「Add project to working sets」は、このプロジェクトを特定のワーキングセットに追加する場合にオンにします。ワーキングセットとは、Eclipseがプロジェクトを分類するために使うものです。これをオンにすると、次の行で指定するワーキングセットに、このプロジェクトを追加できます。既存のワーキングセットに追加する場合には、「Select...」ボタンをクリックして、あらかじめ作成してあるワーキングセットから選択することができます。

以上の設定が済んだら「Next >」ボタンをクリックします。カスタムアイコンを作成することが指定してある場合には、そのためのステップとなります(図4)。

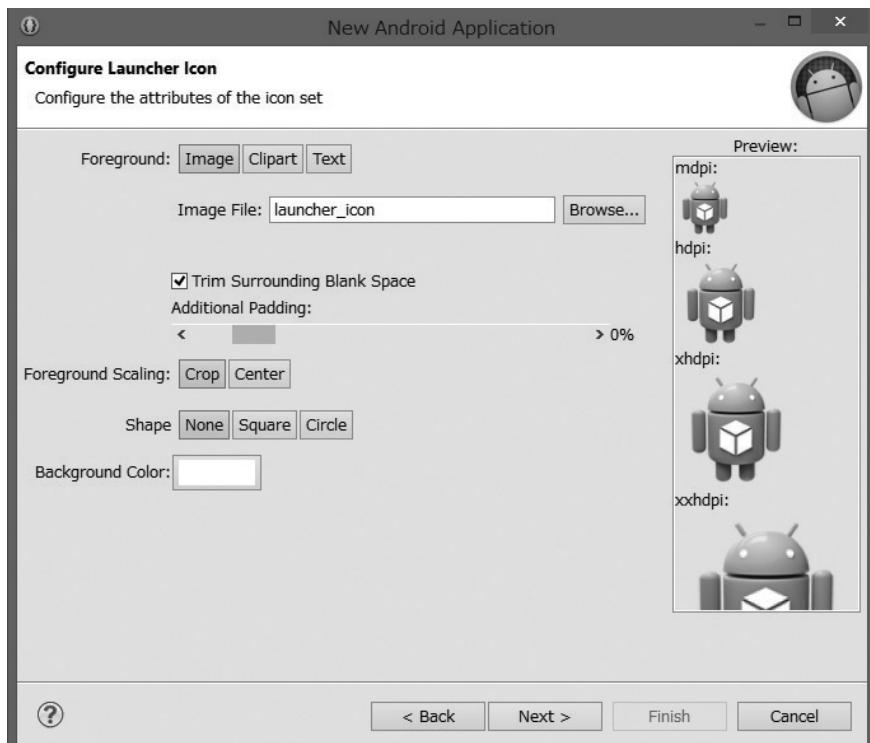


図4: アプリのアイコンをカスタマイズする「Configure Launcher Icon」ステップ

ここでは、アプリと起動するためのアイコンを3種類の方法で指定することができます。1つは、あらかじめ用意したアイコン用の画像を読み込む方法。2つ目はクリップアートから選択する方法。3つ目はテキストを入力してアイコンをその場で生成する方法です。これらは「Foreground:」で選びます。また、いずれの場合についても「Trim Surrounding Blank Space」のチェックボックスと、その下の「Additional Padding:」のスライダー、「Foreground Scaling:」では、アイコンのトリミングや、スケーリング(拡大、縮小)の方法を指定できます。また「Shape:」では、アイコンの外

「Mark this project as a library」は、このプロジェクトにライブラリーとしての印を付けるという意味です。

デフォルトのまま、「Next >」をクリックして先に進むことで、Androidのマスコット、いわゆる「ドロイド君」の画像が、ランチャーアイコンとしてセットされます。

形線の形状を、「None」(なし)、「Square」(正方形)、「Circle」(円)から選択できます。さらに「Background Color:」ではアイコンの背景色も指定できます。

ここではすべてデフォルトのまま、「Next >」をクリックして先に進みます。

次のステップは「Create Activity」です(図5)。これは、以前のステップでアクティビティを作成することを選択した場合に有効となるものです。



図5:アプリケーションのアクティビティの種類を選択する「Create Activity」ステップ

すでに述べたように、通常のAndroidアプリには、少なくとも1つのアクティビティが必要です。最初からアクティビティを作成するには、ここで「Create Activity」のチェックボックはオンのまま、その下のリストから作成するアクティビティの種類を選びます。現状の選択肢には、「Blank Activity」、「Fullscreen Activity」、「Master/Detail Flow」の3種類があります。

「BlankActivity」は空のアクティビティという意味ですが、ここでは画面の左上に「Hello world!」という文字列を表示する、いわゆる「ハローワールド」アプリを実現するアクティビティを作成します。「Fullscreen Activity」では、Androidデバイスの画面全体を使って動作するアプリを作成します。「Master/Detail Flow」は、リスト状に複数の項目を並べた画面(マスター)を表示し、ユーザーがその中から選んだ項目に関する情報を別の画面(ディテール)に表示するタイプのアプリを実現します。

ここでも、デフォルトの「Blank Activity」を選んだまま「Next >」をクリックして次のステップに進みます。このステップの名前は、「Blank Activity」となっています。これは、前のステップで「Blank Activity」を選んだからです(図6)。

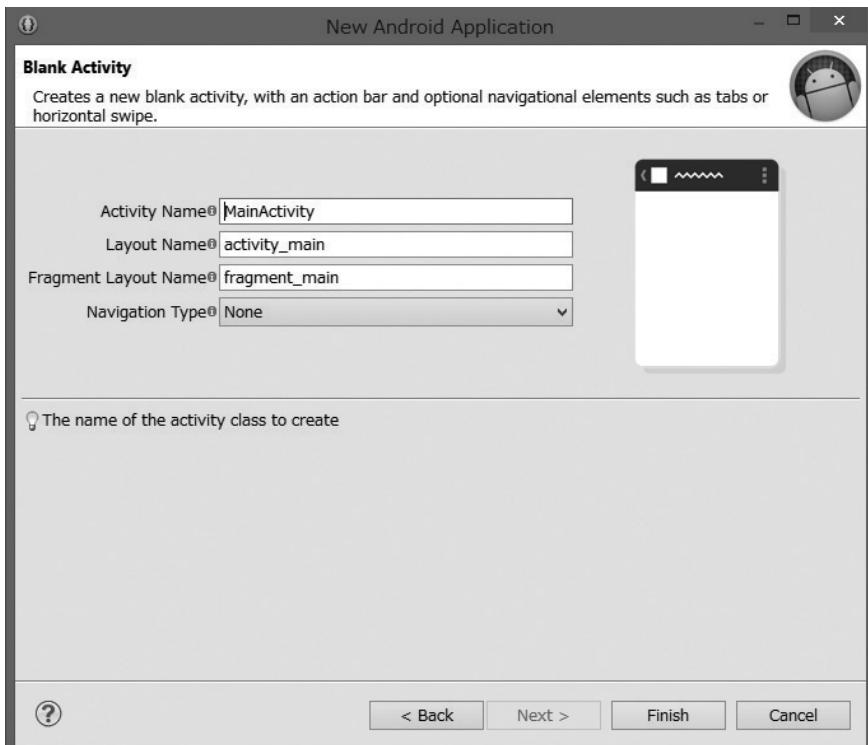


図6:アクティビティとレイアウトXMLファイルの名前、ナビゲーションタイプを設定する「Blank Activity」ステップ

ここでは、「Activity Name」で作成するアクティビティの名前、「Layout Name」と「Fragment Layout Name」で、このアクティビティに付随するレイアウトのXMLファイルの名前を指定します。また「Navigation Type」そしてナビゲーション機能のタイプを設定します。ここでは、各項目についての説明を割愛します。

以上の操作で、「MyFirstApp」という名前の新しいプロジェクトが作成され、Eclipseの「Package Explorer」というビューの中に現れるはずです(図7)。

図6以降は、すべての設定をデフォルトまま、「Finish」をクリックして、Androidアプリ作成のウィザードを完了させてください。

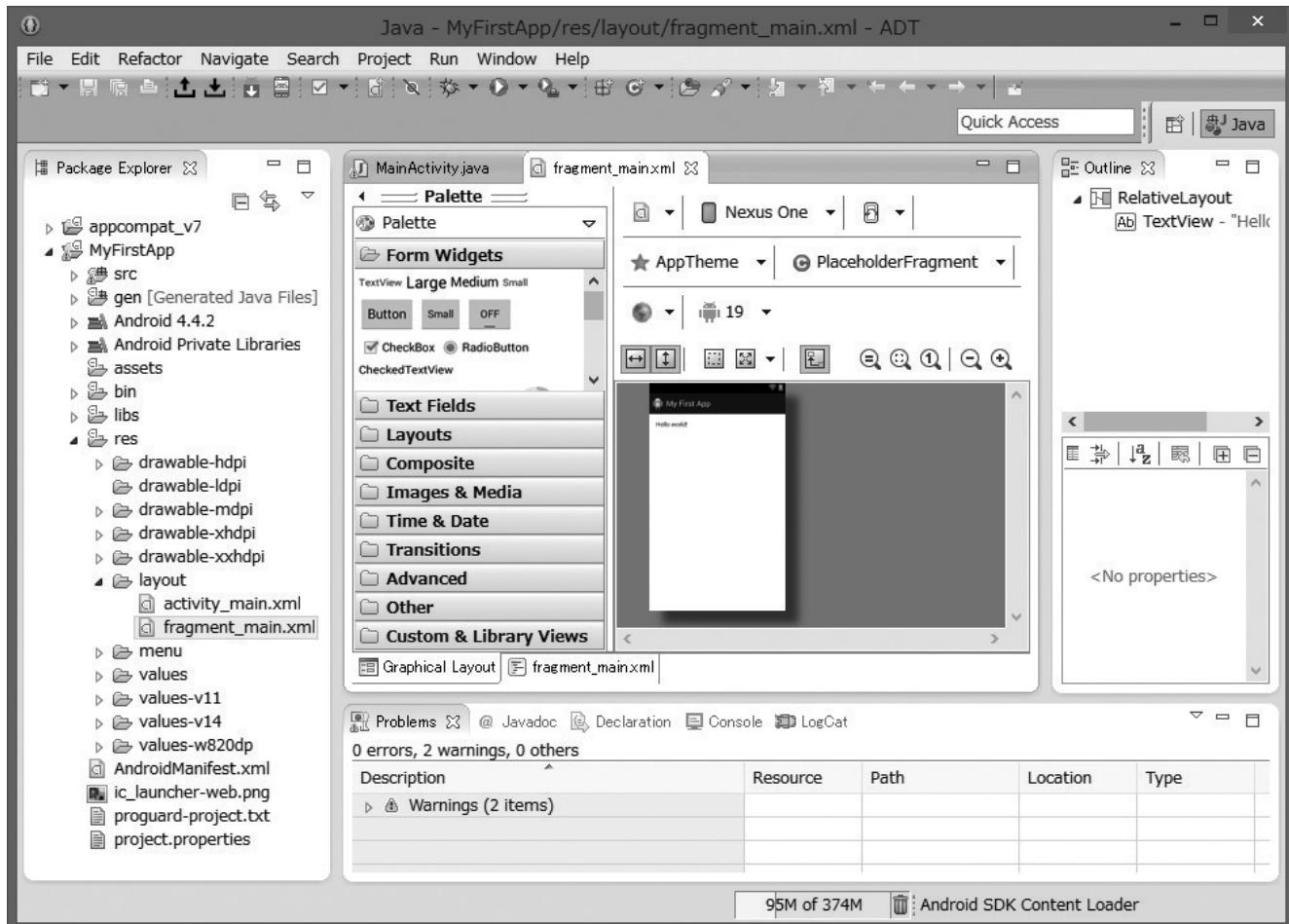


図7:新しいAndroidアプリのプロジェクト「MyFirstApp」が作成され、「Package Explorer」の中に表示される

この「Package Explorer」の中には、もう1つ「appcompat\_v7」というプロジェクトも見えます。これは最初からあったものではなく、新たなプロジェクトを作成したことによって自動的に作成されたものです。

## Androidアプリプロジェクトの構成

Windowsのエクスプローラーでフォルダーを操作するのと同じやり方で「Package Explorer」の中で「MyFirstApp」プロジェクトを開き、中身をざっと確認しておきましょう(以下、図7を見ながら読んでください)。

まず、プロジェクトの中でいちばん上にあるのは「src」のグループです。この中には、アプリケーションを構成するソースコードが格納されています。初期状態では、プロジェクトの作成過程で指定したアクティビティの名前に一致するJavaソースコードが自動的に追加されています。「Package Explorer」のビューの右側にあるエディターのビューの中に「MainActivity.java」というタブが見えますが、これがそのソースコードに対応するものです。

「src」の下の「gen」には、コンパイルによってソースコードから作成された「R.java」ファイルなどが入ります。また「Android4.4.2」にはAndroid SDKを構成するライブラリーが格納されています。このグループの名前は、このプロジェクトをビルドするのに使うAndroidのバージョンに対応したものです。

そこから少し離れて下の方にある「res」は、リソース(Resource)の略です。この中にはJavaのソースコード以外に、アプリケーションを構成する各種のリソースが含まれています。その中にはまず「drawable-」で始まる5つのフォルダーがあります。これらはアイコン画像をはじめとして、このプロジェクトで使用する画像ファイルを格納するためのものです。それが5つもあるのは、5種類の画面の解像度に対応しているからです。いずれのフォルダーにも、初期状態ではランチャーアイコンと、検索機能を表すボタンの画像が入っています。アプリで使用する画像は、必要に応じて後でここに追加したり、入れ替えたりします。

その下の「layout」には、このアプリケーションのユーザーインターフェースの画面レイアウトを定義するXMLファイルが入ります。この例では「activity\_main.xml」と「fragment\_main.xml」ファイルです。後者は初期状態でエディターのビューが開いて、これがこのアプリのメインの画面を形成していることが分かります。

「menu」には、アプリで使うメニューを定義するXMLファイルを格納します。また「values」には、Javaソースコードとは独立して文字列を定義するファイルを格納します。初期状態では「strings.xml」というファイルが入っていて、画面に表示する「Hello world!」の文字列を定義しています。

「AndroidManifest.xml」は、Androidアプリのプロジェクトの構成要素の中で最も重要なファイルの1つで、個々のプロジェクト全体の設計図に相当するものです。どのアプリケーションにも、すべてこの同じ名前で含まれています。

5種類の解像度とフォルダ名の関係は、「hdpi」が高、「mdpi」が中、「ldpi」が低の各解像度表しています。「xhdpi」は「hdpi」より高い解像度、「xxhdpi」は、それよりさらに高い解像度を表しています。



## 4-3-2 エミュレーター上でのアプリの起動

### Androidアプリの起動

ここまでに示した手順で作成したAndroidアプリのプロジェクトは、すでに述べたように画面に「Hello world!」と表示するだけのアプリとして完成した状態になっています。とりあえずそれを動かしてみましょう。最初はパソコン上で動作するエミュレーターを使って起動します。

Eclipseのプロジェクトとして作成中のアプリを起動するには、まずそのプロジェクトを「Package Explore」のビューの中で選択します。Eclipseを使って複数のアプリを開発する際には、このPackage Explorerの中にそれらのプロジェクトが並びます。どのプロジェクトのアプリを起動するか指示するために、プロジェクトの選択が必要なのです。

このようにわざわざAndroidアプリを選択するのは、当たり前のことなので不要ではないかと思われるかもしれません、EclipseではAndroidアプリ以外のプログラムも開発できるので、プロジェクトをどのように起動するのかを指示することが必要となります。

次に「Run」メニューの「Run As」サブメニューから「Android Application」を選びます(図8)。

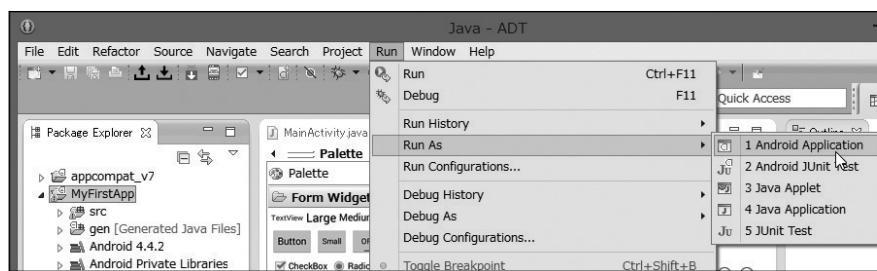


図8:起動するアプリのプロジェクトを「Package Explore」で選択した状態で「Run」メニューの「Run As」サブメニューから「Android Application」を選択する

「Run」メニューから単に「Run」を選んだ場合、あるいはツールバーにある「Run」ボタンをクリックして起動しようとすると、「Run As」というダイアログが表示され、このアプリを起動する方法、言い換えればどんなプログラムとみなして起動するかを聞いてきます(図9)。その場合には、ダイアログにあるリストの中から「Android Application」を選択してから「OK」をクリックします。



図9:プログラムの種類を指定しないで起動すると「Run As」ダイアログが表示されるので、ここで「Android Application」を選んでもいい

なお、同じアプリの2回目以降の起動時には、すでに「Android Application」として起動するものであることを、Eclipseが記憶してくれるので、ツールバーの「Run」ボタンをクリックするだけで起動できるようになります。

## 仮想デバイスの作成と設定

Eclipse上で初めてAndroidアプリを起動しようとする場合、なおかつAndroidデバイスの実機が開発環境のPCに接続されていない場合には、必ず「Android AVD Error」というタイトルのダイアログが表示されます(図10)。これは、Androidアプリを起動可能なターゲットとなるデバイスがみつからないというエラーです。それと同時に、アプリをエミュレーター上で起動するために、AVD(Android Virtual Device:エミュレーターのこと)を新たに作成するかと聞いています。

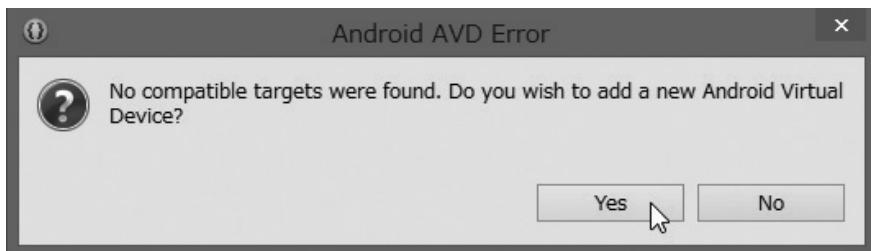


図10:初めてAndroidアプリを起動しようとする際に表示される「Android AVD Error」のダイアログ

ここでは、単に「Yes」をクリックすれば良いのです。それによって、新たなAVDを作成することができます。そのために「Android Virtual Device Manager」というダイアログが表示され、さらにその上に「Android Device Chooser」というダイアログも表示されます(図11)。

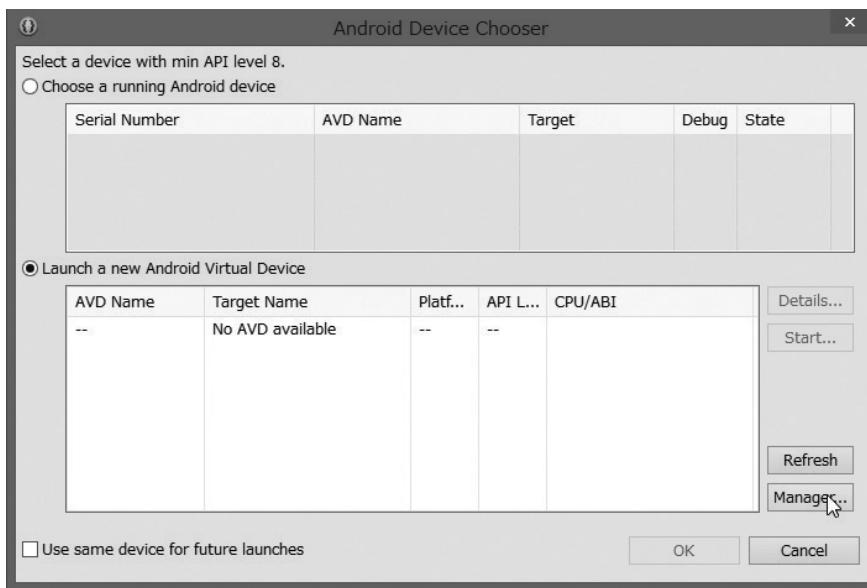


図11:アプリを起動するAndroidデバイス、またはAVDを選択する「Android Device Chooser」ダイアログ

ここでは「Cancel」ボタンをクリックすることで、この直前に開いた「Android Virtual Device Manager」に戻ることもできます。

もし開発用として有効なAndroidデバイスが開発環境に接続されているか、すでにAVDが作成してあれば、その中からアプリ起動用の環境を選択することができます。どちらも存在しなければ、新たにAVDを作成してから、再びこのダイアログで起動用の環境を選択することになります。ここでは「Manager...」ボタンをクリックして、改めて「Android Virtual Device Manager」を開きます。

「Android Virtual Device Manager」のダイアログでは、まず「New...」ボタンをクリックして、新たなAVDの作成を開始します(図12)。

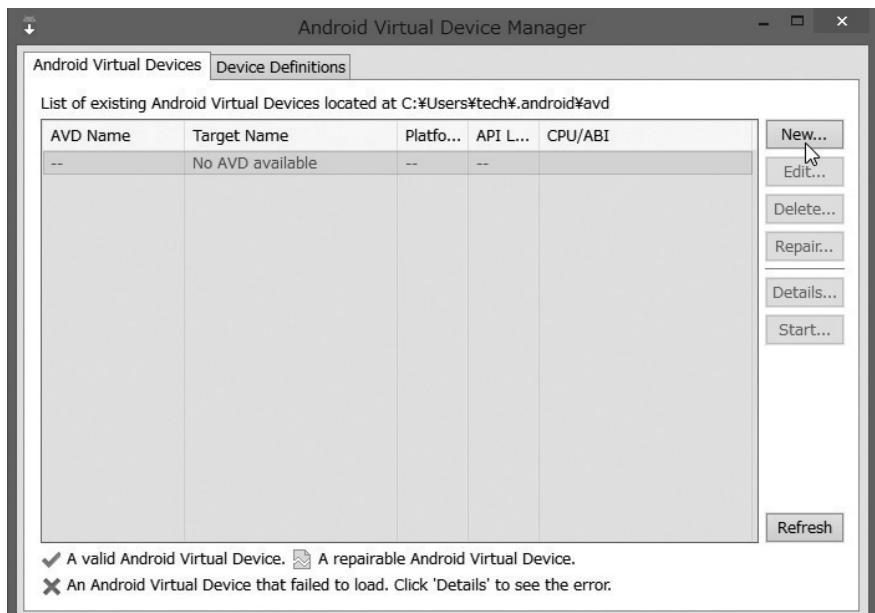


図12:「Android Virtual Device Manager」を使って新たなAVDを作成するために「New...」をクリックする

すると、「Create new Android Virtual Device (AVD)」というダイアログが表示されます(図13)。ここでは、AVDの名前と仮想的なデバイスの仕様を設定します。

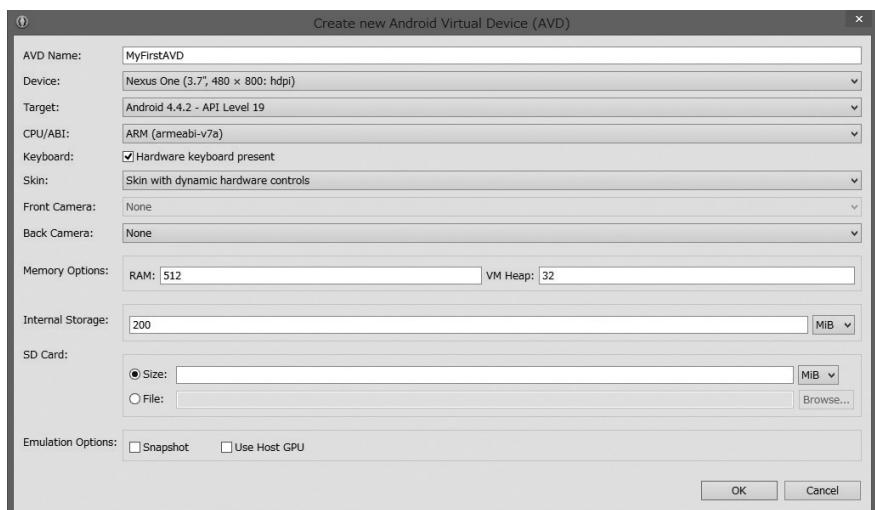


図13:新たに作成するAVDの名前と仕様を設定する「Create new Android Virtual Device (AVD)」ダイアログ

まず「AVD Name:」には「MyFirstAVD」と入力しました。これは何でもかまいません。「Device:」欄はメニューになっているので、ここでは小さめのサイズの画面を備えた「Nexus One (3.7", 480 x 800: hdpi)」を選んでいます。「Target:」欄ではAndroidのバージョンを選択しますが、この例の開発環境には1種類のSDK

しかインストールしていないので、そのバージョン「Android 4.4.2 - API Level 19」しか選べません。「Skin:」欄には「Skin with dynamic hardware controls」を選択しています。

とりあえず、その他の設定はそのままで、「OK」ボタンをクリックしてこのダイアログを閉じ、AVDの作成を完了させます。

この結果、「MyFirstAVD」という名前の新たなAVDが作成され、「Android Virtual Device Manager」ウィンドウの「Android Virtual Devices」に、AVDの1つの選択肢として表示されます(図14)。ここでは、そのAVDを選択して、「Start...」ボタンをクリックします。

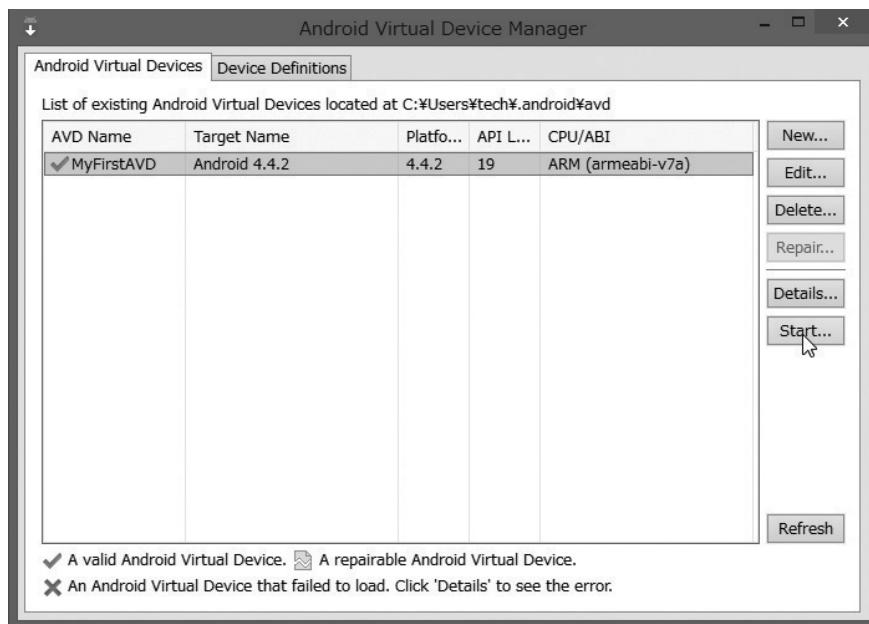


図14:新たに作成したAVDが「Android Virtual Device Manager」ダイアログのリストに表示された。それを選択して「Start...」ボタンをクリックする

するとこんどは「Launch Options」という小さなダイアログが表示されます(図15)。この状態では、「Scale display to real size」と「Wipe user data」という2つのオプションが選択可能になっています。

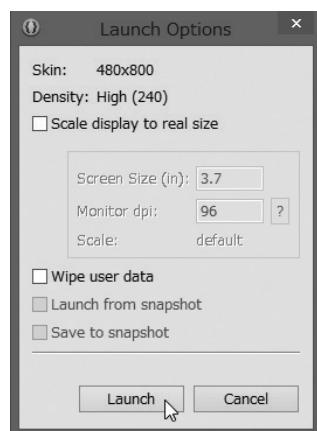


図15:「Launch Options」のダイアログでは、起動するAVDのオプションを設定できる。ここではそのまま「Launch」ボタンをクリックして仮想デバイスを起動する

前者は、開発環境の画面仕様を考慮して、AVDの画面の表示サイズを実際のデバイスの画面サイズに近付けるというものです。後者はAVDに記録されているユーザーごとのデータを消去してから起動するのですが、初めて起動する際

には意味がありません。そのまま「Launch」ボタンをクリックして起動しましょう。

本来は、これでAVDが起動し、その上でAndroidアプリが動作するはずです。しかし、新たに作成したAVDを起動すると、途中でフリーズしてしまうことが多いようです。またEclipseからAVDへのAndroidアプリの転送がうまくいかないこともあります。AVD自体の起動が途中で止まってしまった場合には、AVDのウインドウ右上角の「×」をクリックして、いったんこのAVDを閉じ、Androidアプリの起動をやり直すしかありません。

その結果、AVDがうまく起動すれば、通常のAndroidデバイスと同様のロック画面が表示されるはずです。ここでは錠のマークを右にスライドして、ロックを解除します(図16)。



図16:AVDがうまく起動すると、ロック画面が表示されるので、ロックを解除してホームを表示する

この時点では、EclipseからAVDにAndroidアプリの転送、インストールも済んでいるので、すぐに目的のAndroidアプリが起動します(図17)。

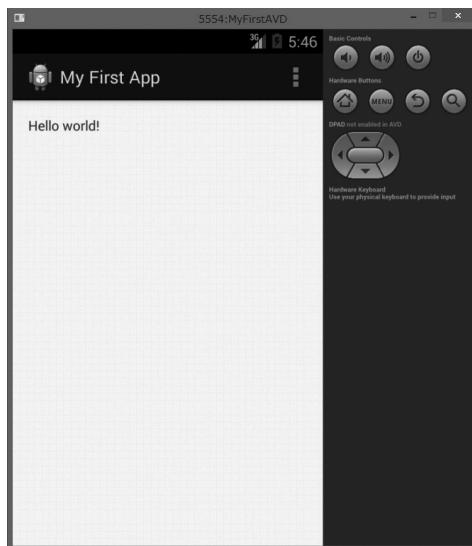


図17:AVD上に転送され、自動的にインストールされたAndroidアプリ「My First App」が起動した

この例は、Androidアプリの新規プロジェクトとして作成したままのアプリなので、白い画面に「Hello world!」と表示されるだけのものとなります。

Eclipseの動作はAVDとは独立していて、この時点では、Eclipse側は通常の状態に戻っているはずです。アプリを構成するソースコードの編集や、別のアプリプ



## 4-3-3 Android デバイス側の準備

### Androidアプリの起動

AndroidアプリをAndroidデバイスの実機上で動作させるには、開発環境のパソコンとAndroidデバイスをUSBケーブルで接続し、パソコン側からデバイスをコントロールする必要があります。ただし、ただ単にAndroidデバイスを用意してUSBケーブルで接続すれば良いというものではありません。

そうするためには、まずAndroidデバイス上での準備が必要です。ここではサムスン製のAndroidデバイス(スマートフォン)「SC-02F」を使うと仮定して、デバイス側の準備について解説します。異なるデバイスであっても、Androidのバージョンが同じであれば、そのために必要な操作はだいたい同じです。ここではAndroid 4.3を使用していますが、4.4でも大差はないでしょう。

まず、スマートフォンのAndroidの「設定」アプリを起動し、「一般」タブにある「端末情報」を開きます(図18)。その中には様々な情報が並んでいますが、いちばん下には「ビルド番号」が表示されているはずです。このビルド番号の表示の上を連続7回タップします。それにより「デベロッパーモード」が有効になります。



図18:「設定」→「一般」→「端末情報」にある「ビルド番号」を7回連続タップして、「デベロッパーモード」を有効にする

その後設定アプリの画面を1つ戻って、「一般」タブの内容が表示される状態になります(図19)。デベロッパーモードを有効にすることで、その中に「開発者向けオプション」が追加されます。



図19:「デベロッパーモード」を有効にすることで、「一般」タブの中に「開発者向けオプション」が追加された

その項目をタップして「開発者向けオプション」の一覧を開きます。すると、その中には「デバッガ」というセクションがあり、そのいちばん上に「USBデバッガ」という項目が配置されています(図20)。



図20:「開発者向けオプション」を開いて、「デバッガ」セクションにある「USBデバッガ」のオプションをオンにする

その右端にあるチェックボックスにタップして、このオプションを有効にします。すると「USBデバッガを許可しますか?」というダイアログが表示され、本当にこのオプションを有効にしても良いかどうかを確認してきます(図21)。ここではもちろん「OK」をタップして、この設定をオンにします。

Androidデバイス側の準備は、ひとまずこれで完了です。



図21:「USBデバッグ」のオプションをオンにしようとすると、ダイアログを表示して確認してくるので「OK」をタップする

## パソコンとAndroidデバイスの接続

Androidデバイス側の準備ができたら、開発環境のパソコンとデバイスをUSBケーブルで接続します。初めて接続する際には、USBドライバーが自動的にインストールされるのが正常な動作です。そのためには、パソコンがインターネットに接続していることが必要です。

ドライバーが正常にインストールされると、パソコンの画面に「タップして、このデバイスに対して行う操作を選んでください。」という大きなボタンが表示されます(図22)。これは通常は無視してかまいません。

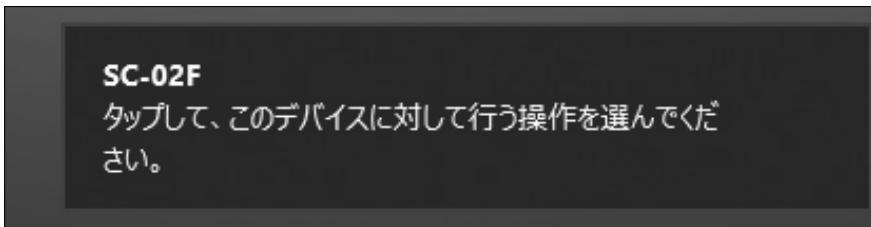


図22:Androidデバイスを開発環境のパソコンに接続すると、ドライバーが自動的にインストールされた後、接続操作を選択するためのボタンが表示される

この大きなボタンをタップ、またはクリックすると、通常のAndroidデバイスを、一般的なUSBデバイスとして接続する際の方法を選択するメニューが表示されます(図23)。Androidアプリを実行するための接続は、それらのどれでもないので、この表示も無視するか、「何もしない」を選んでおけば良いでしょう。



図23:接続方法の選択メニューが表示されたら「何もしない」を選ぶか、表示が消えるまで操作せずに待てば良い

もしUSBドライバーのインストールや、その後の接続がうまくいかない場合には、Windowsのデバイスマネージャを使って、ドライバーを更新することで解決する可能性もあります。そのためには、デスクトップの左下にある「スタート」ボタンを右クリックし、表示されるメニューから「デバイスマネージャー」を選択します(図24)。

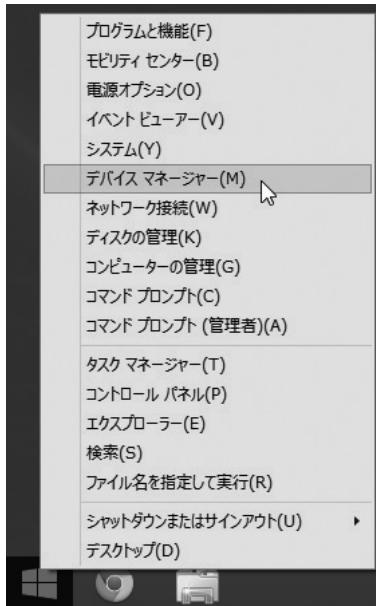


図24:USBドライバーを更新する場合には、「スタート」ボタンを右クリックすると表示するメニューから「デバイスマネージャー」を選ぶ

すると「デバイスマネージャー」のウインドウが開きます。その中の「ユニバーサル シリアル バス コントローラー」の下層にあるAndroidデバイス上で右クリックしてメニューを表示させ、「ドライバーソフトウェアの更新(P)...」選びます(図25)。

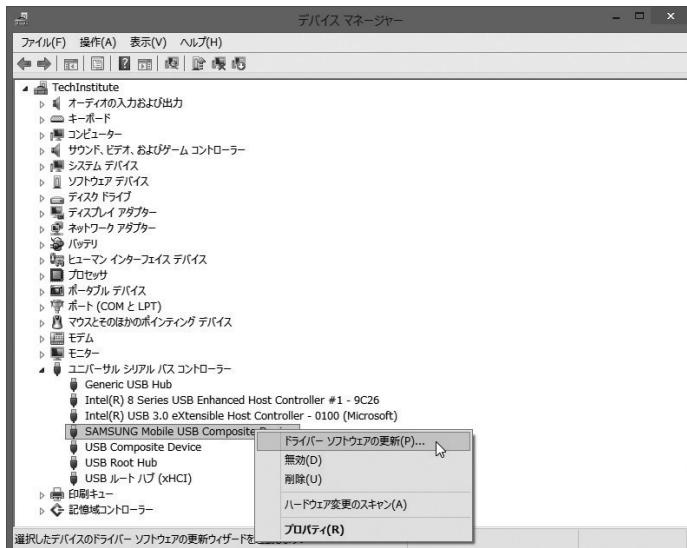


図25:「デバイスマネージャー」では、USBに接続されているデバイスの中からAndroidデバイスを探して右ボタンクリックし、メニューから「ドライバーソフトウェアの更新...」を選ぶ

もし、複数のデバイスが接続されている場合には、その中から目的のものを1つ選択し、「OK」ボタンをクリックします。

その結果、「ドライバー ソフトウェアの更新」というダイアログが表示されるので、その中にある「ドライバー ソフトウェアの最新版を自動検索します」という大きいボタンをクリックします(図26)。



図26:「ドライバー ソフトウェアの更新」ダイアログで「自動検索」を選択するとインターネット上で最新ドライバー検索する

すると、必要なドライバーの最新版をインターネット上で検索し、インストールされるものより新しいドライバーが見つかれば、自動的にインストールします。すでに最新版がインストールされていれば、そのように表示するダイアログが現れます(図27)。



図27:最新ドライバーを自動検索した場合には、その結果がダイアログに表示される

## 実機を選択して実行

Androidデバイス上で開発者モードが有効にされ、そのデバイスが開発環境にUSB接続され、有効なドライバーがインストールされれば、すべての準備が完了します。その状態でEclipseからAndroidアプリを起動すると、「Android Device Chooser」というダイアログが表示されます(図28)。ここでは「Choose a running Android device」が選択され、その下のリストには実機がシリアル番号とともに表示されているはずです。

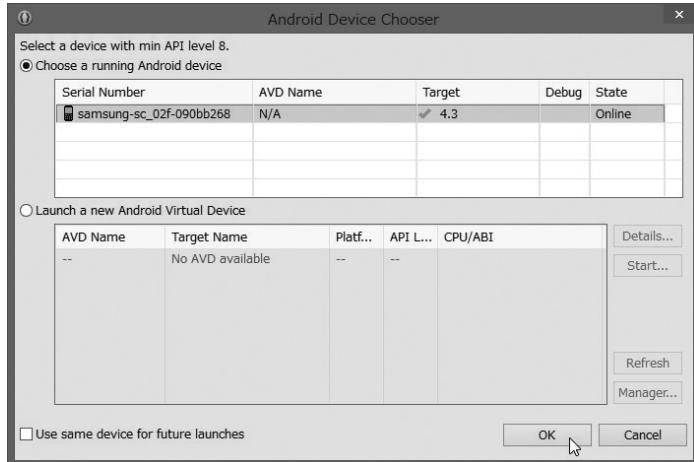


図28:Androidデバイスの実機を接続した状態でアプリを起動すると「Android Device Chooser」が開く。その上でアプリを起動するデバイスを選んで「OK」をクリックする

ここでは、その状態を確かめて「OK」をクリックするだけで良いのです。すると、Androidデバイスの画面には、「USBデバッグを許可しますか?」というダイアログが表示されます(図29)。



図29:Androidデバイス上でUSB経由でアプリを起動しようとすると表示される確認のダイアログ。チェックボックスをオンにして「OK」をタップする

このダイアログに対しては、「OK」をタップすれば良いのですが、その前に「このパソコンからのUSBデバッグを常に許可する」をチェックしておけば、以降同じ確認をしてこなくなるので便利です。

その後、さらに「インストールされたアプリが不正な動作をしないかどうかをGoogleが定期的に確認します。」というダイアログが表示されます(図30)。これは自分で作成したアプリを動かす場合にはどちらでもかまいません。とりあえず「同意する」を選んでおきましょう。設定は後でも変更できます。

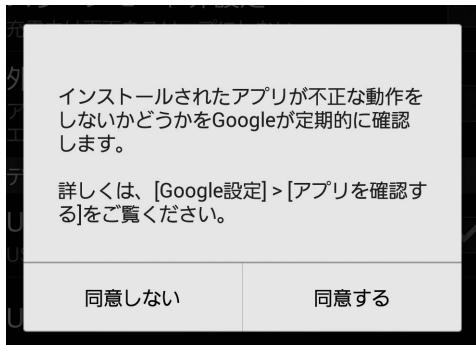


図30:外部からインストールしたアプリをグーグルが定期的にチェックすることを確認するダイアログ。どちらを選んでも良いいる

以上のような過程を経ると、ようやくAndroidデバイスの実機上でアプリが動作します(図31)。ここではその方法については述べませんが、これで実機を使ったデバッグ作業もできるようになりました。

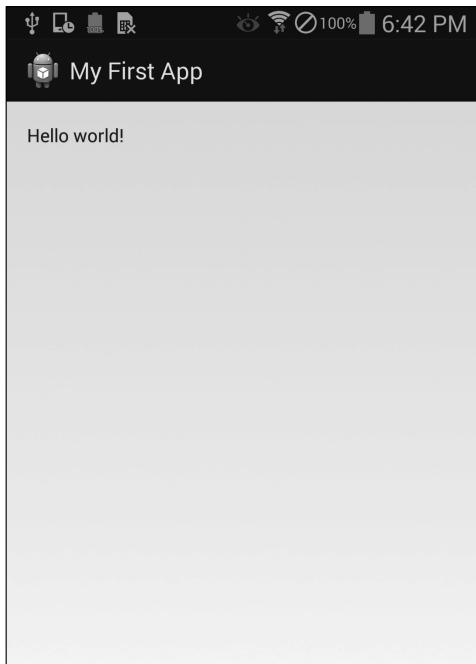


図31:Eclipse上で作成したアプリがUSBケーブル経由でAndroidデバイスの実機に転送されて動作した

