CHAPTER_02

はじめてのAndroidアプリ

Chapter02では、Androidアプリを開発するための環境を整備していきます。必要なソフトをインストールするとともに、デバッグの方法なども確認しておきましょう。またアプリを公開するための手続きについても紹介します。

#02-01 開発ツールのインストールについて

Chapter02では本書の開発で使用するソフトウェアをインストールして環境を構築していきます。使用するソフトウェアは以下の通りです。

・Google Chrome … Webブラウザ（HTML/JavaScriptのテストに使う）
・Aptana … HTML/JavaScriptの開発環境
・Android SDK … Androidの開発に必要な開発キット
・jsWaffle … HTML/JavaScriptをAndroidプロジェクトに変換するフレームワーク

上記、すべてのアプリケーションはオープンソースで開発されており、無償で入手することができます。これは、Androidの透明性をよく表わしていると言えます。

Google Chrome

AndroidのHTML/JavaScriptのエンジンは、パソコンのWebブラウザ「Google Chrome」と共通する部分が多く、特にHTMLのレンダリング部分には、WebKit(ウェブキット)が採用されています。そのため、実際にAndroid端末に転送する前に、Google Chrome上でだいたいの表示や実行の様子を確認することができます。これにより、効率的にアプリの開発を行うことができます。

Google Chromeのインストール

STEP_01

Google Chromeは以下のURLからダウンロードすることができます。URLにアクセスし、［Google Chromeを無料ダウンロード］のボタンをクリックします。

Google Chrome のダウンロードサイト
http://www.google.com/chrome/intl/ja/landing.html?hl=ja

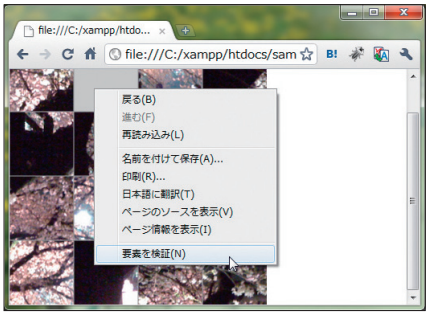
STEP_02

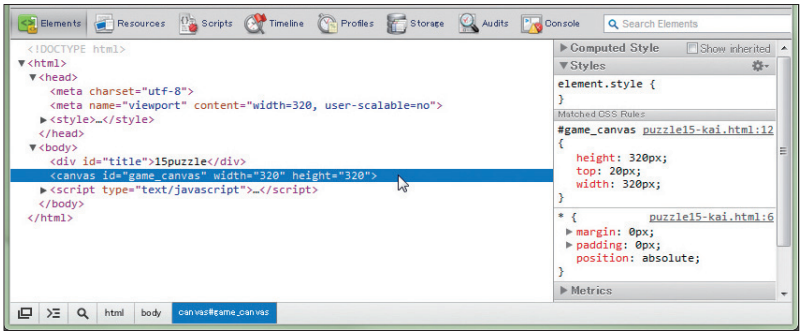
ライセンスの確認画面が表示されるので、確認したら［同意してインストール］をクリックすると、Google Chromeがインストールされます。

Google Chromeでデバッグを行うには

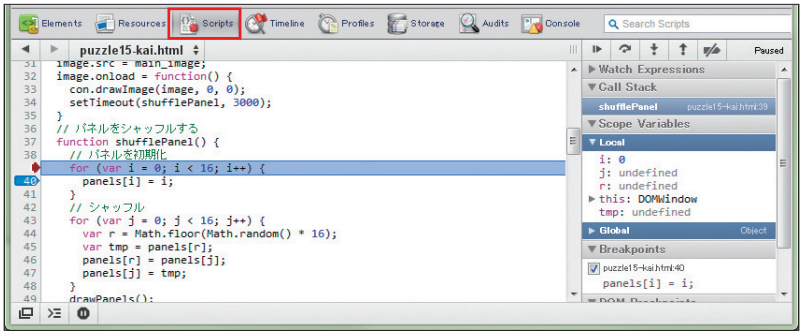
Google Chromeにはさまざまな機能があり、HTML/JavaScriptの開発に便利なデバッグ機能を備えています。デバッグしたいHTMLファイルをChrome上にドラッグ&ドロップして表示したら、右クリックしてポップアップしたメニューにある［要素を検証］をクリックします。すると、HTML要素がツリー構造で表示され、直感的にHTMLの構造を調べることができます。

| 図 02-001 | Google Chromeでのデバッグ





| 図 02-002 |



また、[Scripts]のタブをクリックすると、スクリプトのデバッグ専用画面になり、画面の左側にある行番号をクリックすることでブレークポイントと言って、途中で実行を中断して一行ずつ実行することができます。また、ブレークポイントで実行を中断すると、その時点の変数の内容を確認したり、関数の呼び出し履歴を確認することができます。こうしたデバッグ支援機能がはじめから備わっているので、HTML/JavaScriptのデバッグを非常に手軽に行うことができます。

Aptana Studio 2と必要プラグイン

Androidの開発においては、Javaの統合環境Eclipseを使うのが一般的ですが、本書では、HTMLやJavaScriptを主に扱いますので、EclipseをWeb開発向けにチューニングしたAptana Studioを利用してAndroidの開発を行っていきます。もちろん、EclipseにAptanaプラグインをインストールする方法もありますが、ここでは、便利なAptana Studioを使ったセットアップの方法を紹介します。

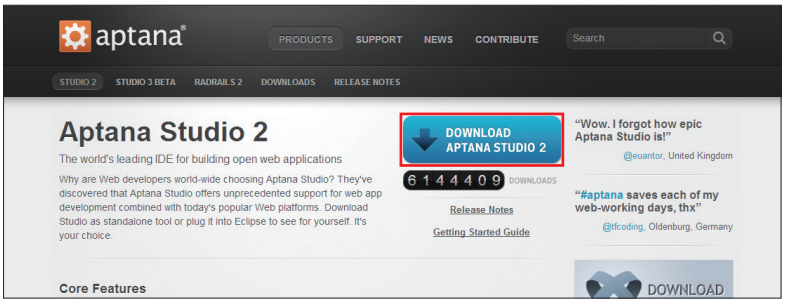
Aptana Studio 2のインストール

STEP_01

Aptana Studio 2は、下記AptanaのWebサイトからダウンロードすることができます。Windows/Mac OS X/Linuxで動作可能です。

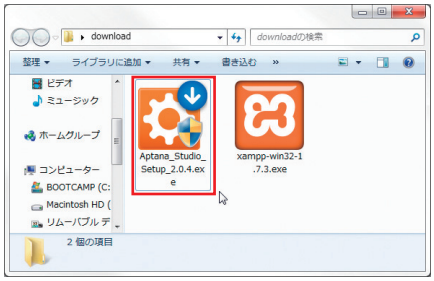
「DOWNLOAD APTANA STUDIO 2」ボタンをクリックしてインストーラーをダウンロードします。

Aptana Studio 2のWebサイト
http://www.aptana.org/products/studio2



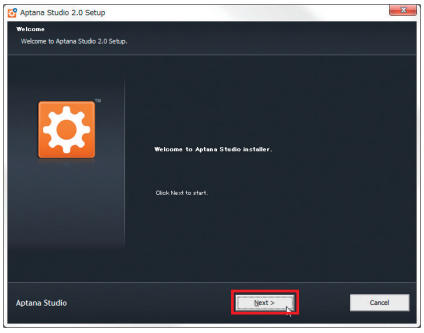
STEP_02

親切なインストーラーが用意されていますので、ダウンロードしたインストーラーをダブルクリックして実行します。



STEP_03

指示に従って［Next］ボタンを数回クリックすることでインストールを完了させることができます。

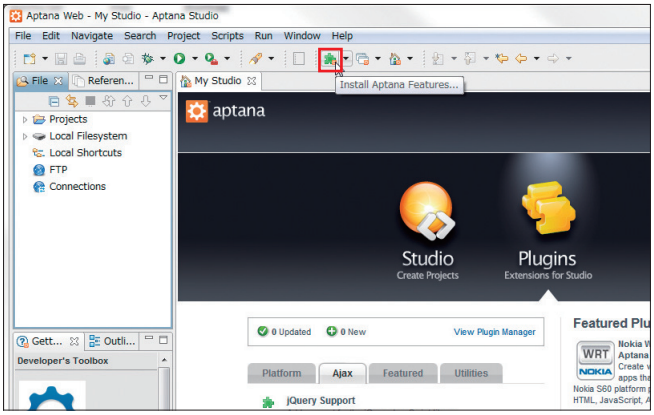


必要なプラグインのインストール

STEP_01

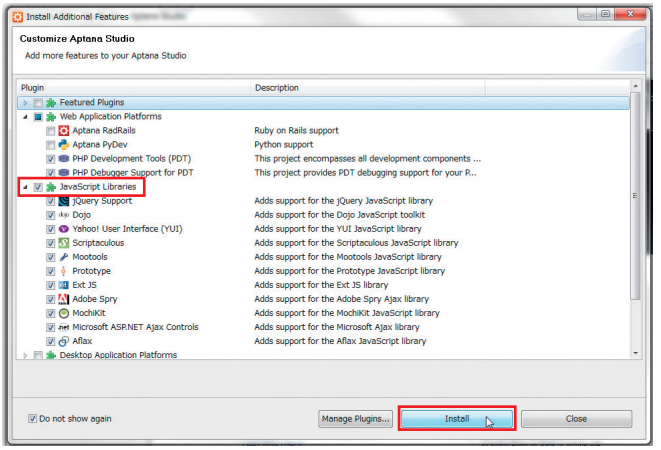
Aptana Studioは素の状態でも、HTML/CSS/JavaScript/XMLの基本的な機能を備えているのですが、プラグインを追加することで、より強力な開発環境を手に入れることができます。

プラグインを追加するには、Aptana Studioのツールバーから、パズルのピースのアイコンを持つ「Install Aptana Features...」をクリックします。



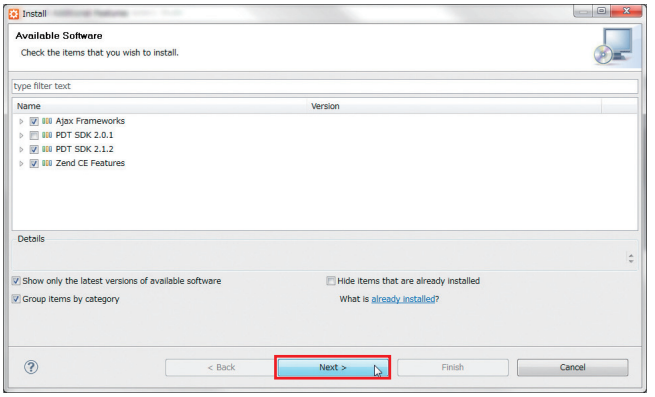
STEP_02

すると、Customize Aptana Studioのダイアログが表示されます。このダイアログで、必要と思える項目にチェックしたら［Install］ボタンをクリックします。ここでは「JavaScript Libraries」にチェックを入れています。



STEP_03

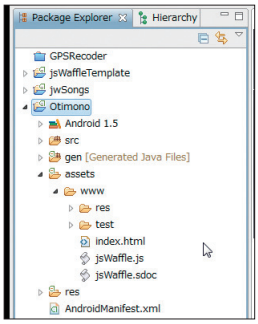
その後、インストールに必要なパッケージが列挙されますので、内容を確認して、チェックを入れて［Next］ボタンをクリックします（例えば、バージョン違いのプラグインがいくつか列挙されたりしますので、同じ名前のパッケージが列挙されたなら、最新のパッケージの方にチェックを入れます）。



TIPS // 02-001

Aptanaのプロジェクト管理について

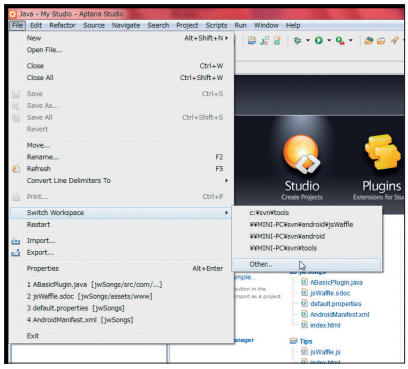
Aptanaでは、アプリケーションを作る時、はじめにプロジェクト（Project）を作ることになります。プロジェクトでは、アプリケーションを作る上で必要となるソースコードや素材ファイルなどをまとめて管理することができるのでとても便利です。



Package Explorerを使ってプロジェクト内のファイルを管理できます

ワークスペースについて

さて、Aptanaでプロジェクトを作成すると、プロジェクトは、ワークスペース（Workspace）と呼ばれる指定ディレクトリ以下に配置されていきます。そのままプロジェクトを作り続けると、ワークスペース内にプロジェクトがたくさんできて、管理が大変になってしまいます。



そこで、ワークスペースを切り替えて使うことになります。ワークスペースを切り替えるには、メニューから［File > Switch Workspace］で行うことができます。新しいワークスペースを作るには、［File > Switch Workspace > Other...］をクリックします。必要がなければ、切り替えずにそのまま使用してください。

ワークスペースを扱う上での注意点

ワークスペース内のプロジェクトは、Aptanaによって管理されます。そのため、Windowsのエクスプローラー上で勝手にプロジェクト用のフォルダを作成したり、既存プロジェクトを削除しても、AptanaのPackage Explorerには反映されません。かえって面倒なことになるので、プロジェクトの作成、削除、変更は、Aptana上で行うようにしましょう。

加えて、本書の手順に沿って、外部ソースコードからプロジェクトを作成する場合（Create project from existing sourceをチェックした場合）、プロジェクトで使用するソースコードは、ワークスペース以下にコピーされませんので、プロジェクトを作ったからといって既存のソースコードを削除しないように注意しましょう。

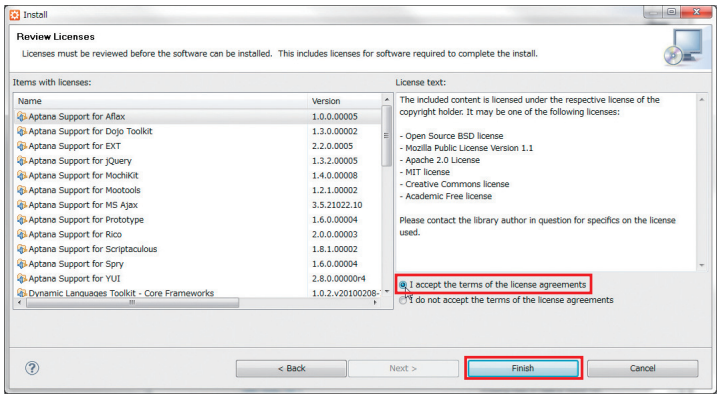
プロジェクトのインポートについて

上述の注意点の通り、Windowsのエクスプローラーを使って、自分のワークスペースに他人が作ったプロジェクトをコピーしたとしても、プロジェクトの存在を自動では認識してくれません。

例えば本書のサンプルプロジェクトを使おうと思う場合も同じです。P.10で紹介している手順に沿って、既存のソースからプロジェクトを作成する必要があります。

STEP_04

すると、必要なファイルがダウンロードされますので、紅茶を片手に音楽を聴きながら、しばらく待ちましょう。その後、インストール前にプラグインのライセンスが表示されますので、「I accept the terms of the license agreements」（ライセンスに同意する）にチェックして、[Finish] ボタンをクリックします。以上でAptanaのセットアップは完了です。



Android SDK

Androidの開発に必要なのは、Androidの開発者サイトで提供されている、Android SDKです。これには、コンパイラからデバッグ用のエミュレーターまで開発に必要なツールが一式用意されています。ここではそのインストール方法を紹介します。

JDKのインストール

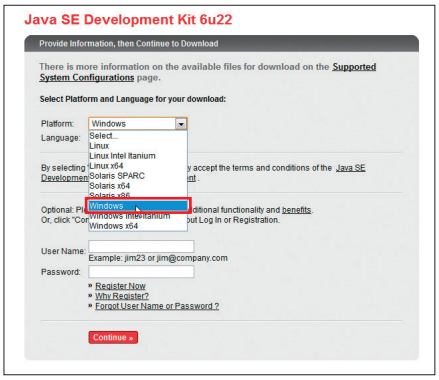
STEP_01

Android SDKを利用するには、Javaの開発環境であるJDK（Java Development Kit）が必要となります。Android SDKをセットアップする前に、JDKをインストールします。
下記JDKのダウンロードページを開き、[JDKダウンロード] のリンクをクリックします。

JDKのダウンロードページ
http://java.sun.com/javase/ja/6/download.html

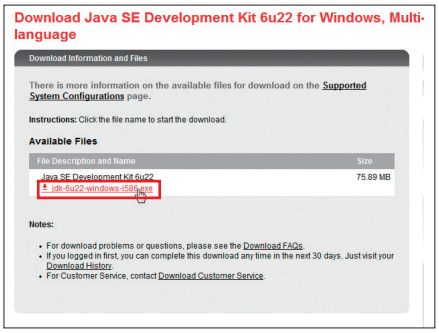
STEP_02

次にプラットフォームの選択画面が出るので、[Platform]のプルダウンメニューからインストールするプラットフォームを選択して[Continue]ボタンをクリックします。



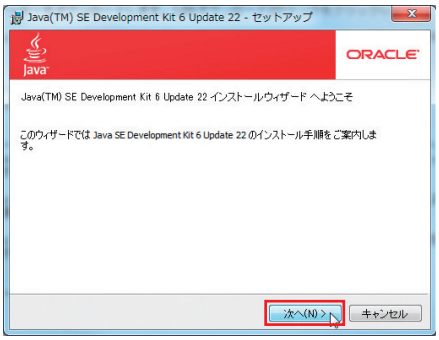
STEP_03

続いて、ファイルが表示されるのでリンクをクリックしてダウンロードを行います。ファイル名はバージョンアップにつれ変更になるので、画面と異なる場合があります。



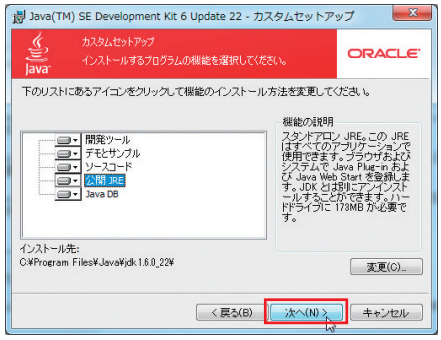
STEP_04

ダウンロードできたら、ダウンロードしたJDKのインストーラーをダブルクリックして実行します。



STEP_05

インストールする機能やインストール先の確認画面です。特にこだわりがなければそのまま［次へ］をクリックします。他の画面も特に希望がなければそのまま［次へ］をクリックして進みます。



STEP_06

JDKのインストールが完了した場面です。

Android SDKのインストール

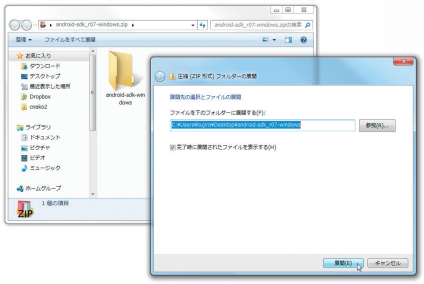
STEP_01

Android SDKは複数OSをサポートしています。ここでは、WindowsにAndroid SDKをインストールする手順を紹介しますが、Mac/Linuxでもほとんど同じ流れになります。
はじめに、Android SDKをダウンロードしておきましょう。このSDKは、ツール（エミュレータなど）やドキュメント、サンプルなどを含んでいます。下記サイトにアクセスし、ここではZIPファイルをダウンロードします。

Android SDK のダウンロード
http://developer.android.com/intl/ja/sdk/index.html

STEP_02

ダウンロードしたら、解凍ソフトでZIPファイルを解凍します（Windowsであればファイルを右クリックして［すべてを展開］を選択するなど）。そして、Windowsなら、分かりやすくCドライブなどに「android-sdk-windows」などの名前でコピーしておきます。ここでは便宜的に、c: ¥android-sdk- windowsにコピーしたものとして進めます。



TIPS // 02-002

Android SDKがサポートするOS

Windows XP(32-bit)、Vista(32/64ビット)、Windows 7（32/64ビット）
Mac OS X 10.5.8 以降（Intel Macのみ）
Linux（Linux Ubuntu Hardy Heronで動作確認済みとのこと）

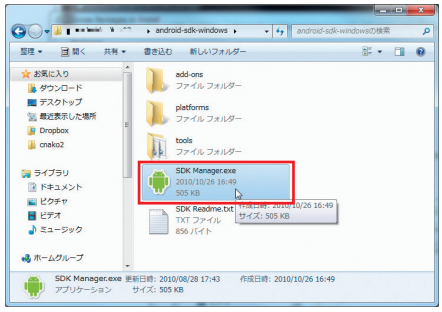
TIPS // 02-003

解凍したファイルのコピー先

コピー先までのパスに日本語や空白スペースが入るとうまく動かない場合があるようです。なるべくドライブの直下などにコピーしてください。

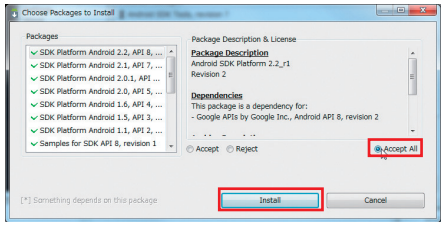
STEP_03

そして、解凍したアーカイブの中に含まれる、「SDK Manager.exe」をダブルクリックして実行します。このSDK Managerを使うと、Androidの各バージョンをセットアップすることができます。



STEP_04

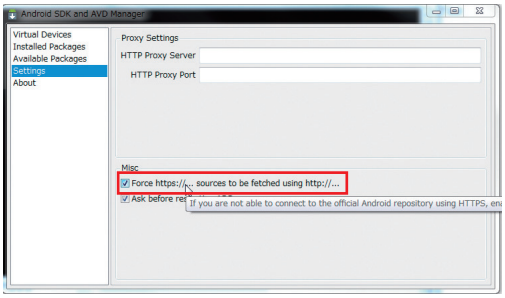
SDK Managerのパッケージのインストール画面が出たら、右側［Install］ボタンの右上にある［Accept All］（すべてに同意する）を選んで、［Install］ボタンをクリックします。



TIPS // 02-004

SDK Managerでエラーが出る場合

何かしらの問題で、SDK Managerで正しくインストールが完了しない場合があります。その場合、一度インストールをキャンセルして、SDK Managerの左側のリストから「Settings」を選び、「Force https://... sources to be fetched using http://...」にチェックを入れます。



Androidプラグインの導入

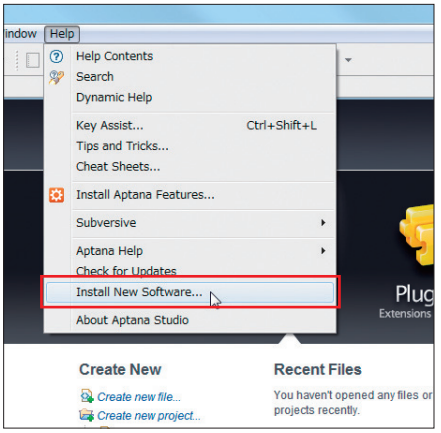
次に、Android プラグイン（ADT）をインストールします。ここでは、Aptanaにプラグインを導入したものとして説明します（Eclipseを使った場合でも手順はほとんど同じです）。

STEP_01

はじめに、Aptanaを起動します。
Windowsでは、[すべてのプログラム > Aptana > Aptana Studio 2.0]をクリックして起動します。

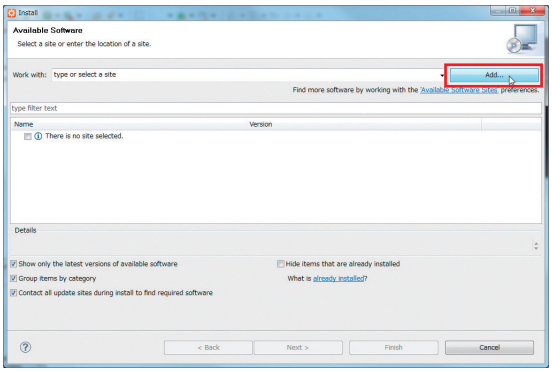
STEP_02

メニューの[Help] より[Install New Software...]をクリックします。



STEP_03

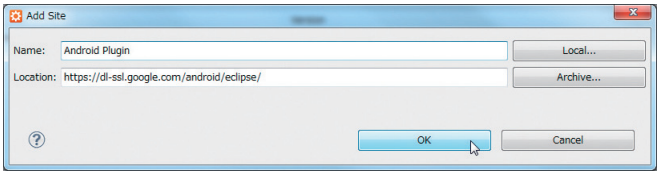
Installダイアログが表示されたら、画面右上の［Add...］ボタンをクリックします。



STEP_04

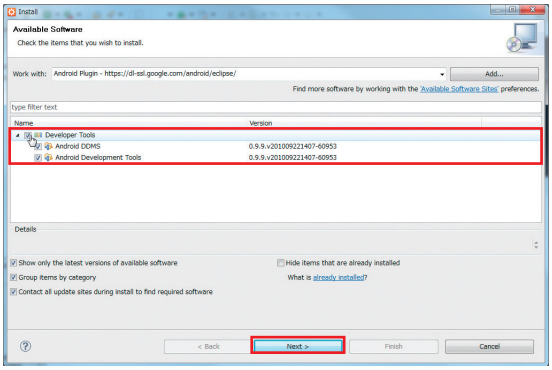
Add Siteのダイアログが表示されたら、そこにAndroidプラグインのインストール用サイトを記入します。右の値を記入してください。

Name：Android Plugin
Location：https://dl-ssl.google.com/android/eclipse/



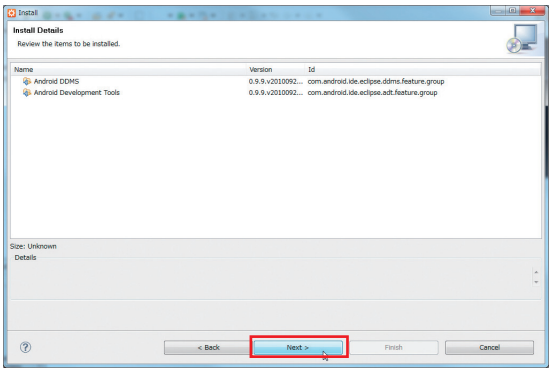
STEP_05

サイトにAndroidプラグインを追加すると、インストール可能なプラグインの一覧が表示されます。ここでは、［Developer Tools］にチェックを入れます。そうすると、その下位に位置する、Android DDMSとAndroid Development Toolsにもチェックが入ります。チェックしたら［Next］ボタンをクリックします。



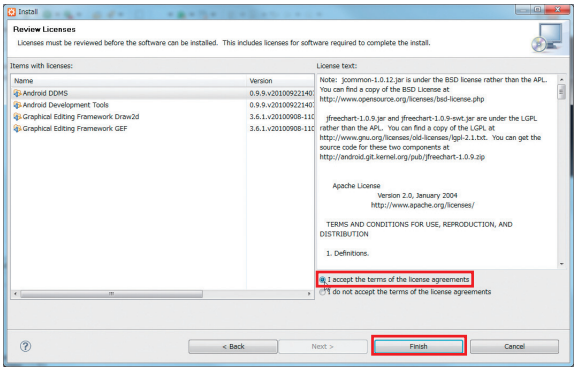
STEP_06

インストールするアイテムが表示されますので確認して［Next］ボタンをクリックします。



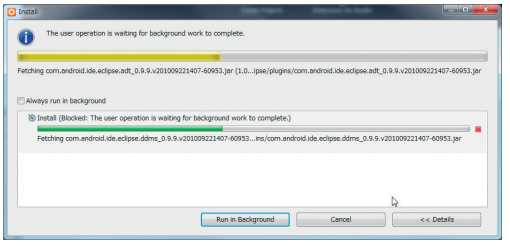
STEP_07

プラグインのライセンスが表示されますので［I accept the term of the license agreements］（ライセンスに同意する）をチェックして［Finish］ボタンをクリックします。



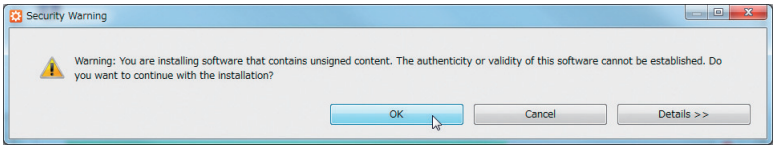
STEP_08

インストールが開始されますので、しばらく作業が終わるのを待ちます。



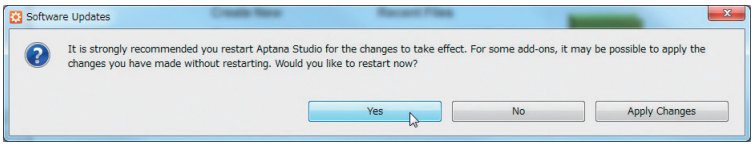
STEP_09

途中で、セキュリティの警告（Security Warning）が出ますが、今回は問題がありませんので、［OK］をクリックして処理を継続します。



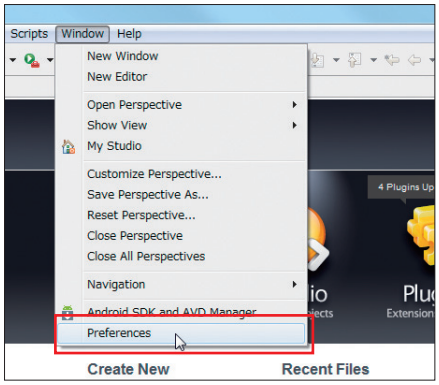
STEP_10

インストールが完了すると再起動を求められますので、［Yes］をクリックして、Aptanaを再起動します。



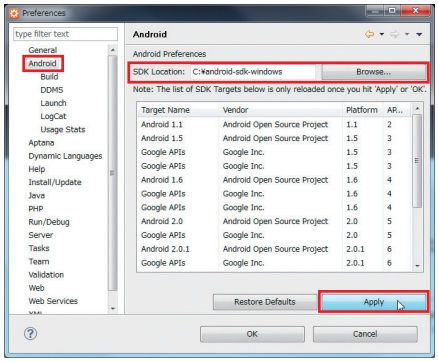
STEP_11

次に、Android SDKとADTプラグインを結びつける作業を行います。メニューの［Window > Preferences］をクリックします。



STEP_12

設定画面（Preferences）が表示されたら、左側のリストから［Android］をクリックします。すると、右の図のような画面が表示されるので、右上にある［SDK Location］に、Android SDKのインストールパス（ここではC:¥android-sdk-windows）を入力します。入力したら右下の［Apply］ボタンをクリックします。すると、インストールされているAndroidのバージョンが表示されます。

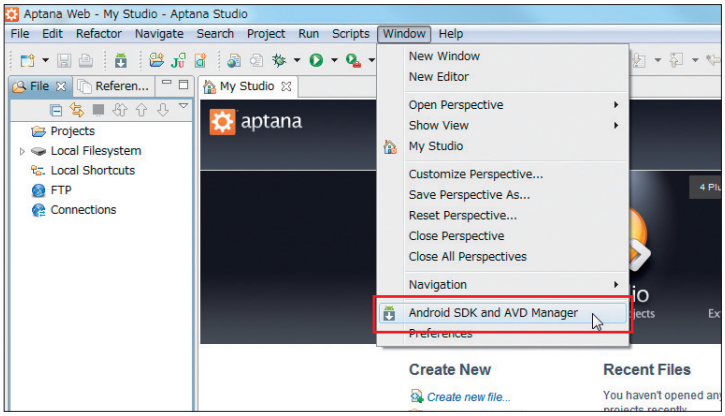


仮想デバイスの作成

Android SDKには、Androidのエミュレーターが含まれています。ここでは、エミュレーター上で動作する仮想デバイスの作り方を紹介します。仮想デバイスではAndroidのプラットフォームバージョンや画像サイズなど自由に設定することが可能です。そのために、AVD Managerを使います。AVDとはAndroid Virtual Deviceの略で、Android仮想デバイス管理ツールを意味しています。

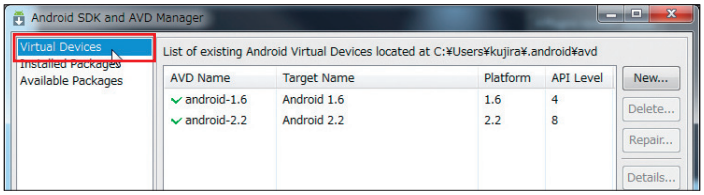
STEP_01

Aptana Studioを起動して、メニューから［Window > Android SDK and AVD Manager］をクリックします（あるいは、Android SDKに含まれる「SDK Manager.exe」を起動します）。



STEP_02

そして、画面の左側から［Virtual Devices］をクリックします。



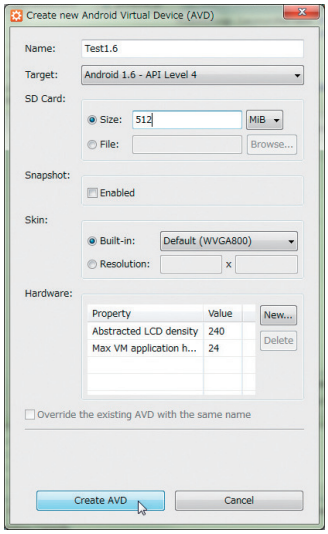
STEP_03

画面の右上にある［New...］ボタンをクリックします。

STEP_04

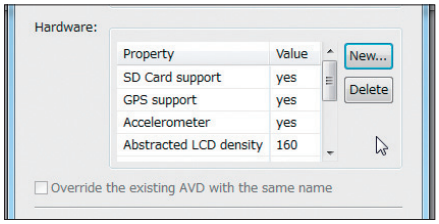
以下を参考に仮想デバイスの設定を記入します。

パラメータ	パラメータの意味	ここでの記入例
Name	AVDの名前	Test1.6
Target	Androidのバージョン	Android 1.6 - API Level 4
SD Card	SDカードのサイズを指定	Size: 512 [MiB]
Skin	画面のサイズ	Built-in: Default (WVGA800)
Hardware	搭載するハードウェア	(STEP_05を参照)



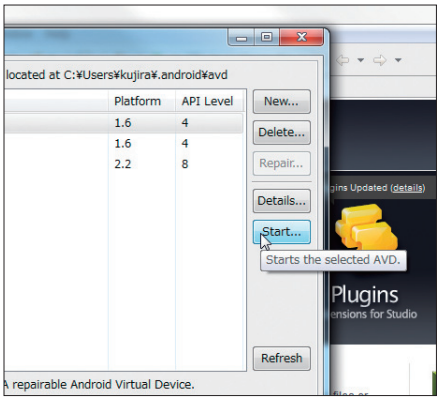
STEP_05

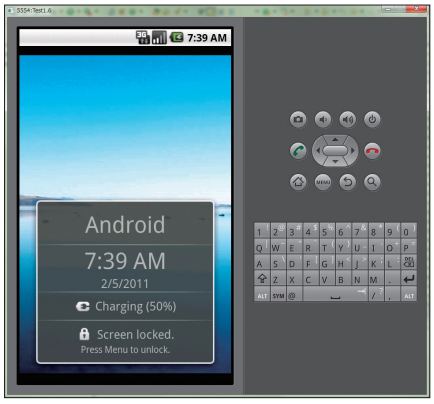
一般的なスマートフォン端末では、ハードウェアとして、カメラやGPSをサポートしています。Hardwareの項目にある［New］ボタンをクリックして、必要と思える機能を追加します。項目が記入できたなら、［Create AVD］のボタンをクリックします。



STEP_06

作成した［Test1.6］を選択したら、右側の［Start...］ボタンをクリックします。すると、Androidのエミュレータが起動します。





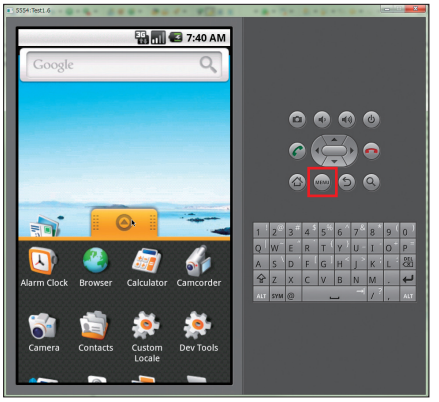
TIPS // 02-005

エミュレータの起動に時間がかかる場合

マシンの性能によりますが、仮想デバイスの初めての起動時には、非常に時間がかかります。[Start] ボタンを押してから根気強く起動を待ちましょう（Windowsのタスクマネージャーで［プロセス］のタブを確認すると、emulator.exeという名前のプロセスが起動しています）。

STEP_07

メニュー画面が表示されたら、画面右側にあるキーボード上部の［MENU］ボタンをマウスでクリックすると、Androidのホーム画面が表示されます。



STEP_08

Android端末を終了させるには、キーボードの上方、右側にある赤い終話のボタンを長押しし、［Power off］あるいは［電源を切る］のメニューをタップします。

仮想エミュレーターでは、通常のウィンドウと同じで右上の［x］ボタンをクリックして終了させることができます（エミュレーターの便利なところです）。

TIPS // 02-006

仮想マシンの言語を日本語にする

作ったばかりの仮想マシンは、言語の設定が「英語」になっています。そこで、日本語に設定しておきましょう。

ホーム画面で、[MENU]ボタンを押して、[Settings > Locale & Text > Select locale > Japanese]をタップします。

jsWaffleのインストール

jsWaffleは、HTML/JavaScriptを使ってAndroidアプリケーションを作るためのフレームワークで、簡単にAndroidプロジェクトのひな型を生成することができます。jsWaffleは、Webブラウザから簡単にインストールが可能です。jsWaffleのダウンロードページへ行き、インストールボタンをクリックするだけです。では、一つずつ手順を確認していきましょう。

STEP_01

まずは、jsWaffleのダウンロードページを開きます。[Download]ボタンを押して、次のページへ進みます。

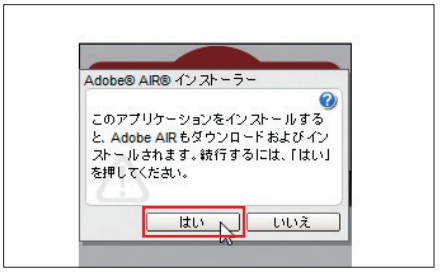
jsWaffle ダウンロードページ
http://d.aoikujira.com/jsWaffle/download.php

STEP_02

そして、画面中央にある［INSTALL NOW］というボタンをクリックします。

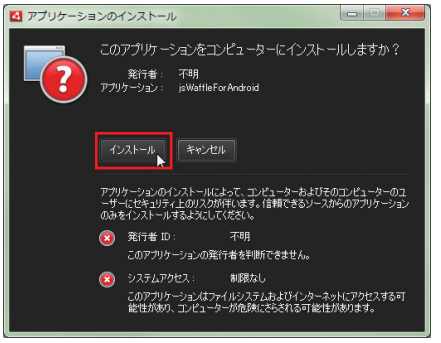
STEP_03

もし、Adobe AIRのランタイムがインストールされていなければ、右のような画面が出ます。［はい］をインストールしてください。既にインストールされていれば、「このファイルを開く、または保存しますか?」という確認ダイアログが出るので、「開く」ボタンをクリックしてください。



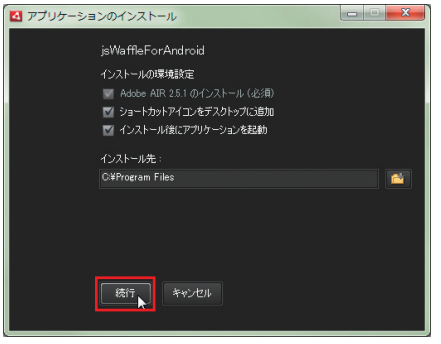
STEP_04

すると、右のような確認画面が表示されます。発行者が不明になっているかもしれませんが、アプリケーションが「jsWaffleForAndroid」と表示されていることを確認したら、［インストール］ボタンをクリックします。



STEP_05

ここで、インストールの環境設定の確認画面が出ます。問題がなければ、そのまま［続行］をクリックしてください。

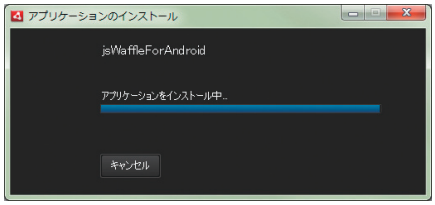


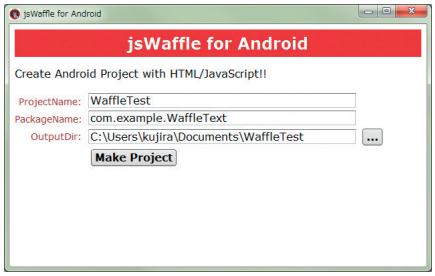
STEP_06

はじめて、Adobe AIRをインストールする際には、ライセンスへの同意ダイアログが表示されます。ここでは、ライセンスを一読の上、[同意する]のボタンをクリックします。

STEP_07

以上の手順で、jsWaffleのインストールが行われます。少しの間、待っているとインストールが行われ、終了すると、jsWaffleが起動します。





#02-02 開発の手順を確認しよう

開発のワークフローを概観する

Aptana/Android SDK/jsWaffleのインストールが終わっているとして、ここでは、開発手順を紹介します。jsWaffleは、Androidの開発において、HTML/JavaScriptを動かすためのフレームワークとなっています。そのため、はじめに、jsWaffleで Androidプロジェクトのひな型を作り、それを、Aptanaにインポートすることで開発を始めることができます。

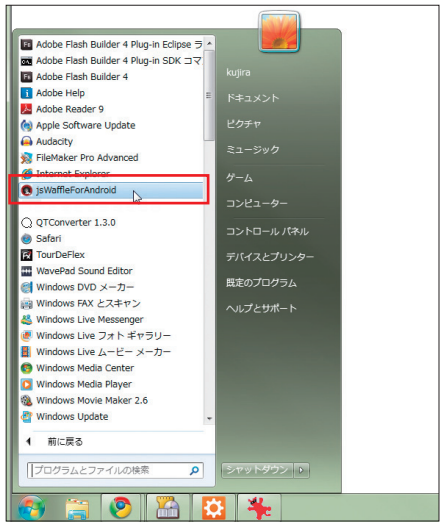
Aptanaにプロジェクトをインポートしたなら、assetsというフォルダに含まれるHTMLファイルを書き換えたり素材を足したりして編集を行います。そして、Aptanaでプロジェクトを実行すると、実機やエミュレータ上で動作を確認できます。

1. jsWaffleでAndroidプロジェクトのひな型を作る
2. Aptanaでプロジェクトのひな型をインポートする
3. assetsフォルダの中にあるHTMLを編集する
4. Androidアプリとして実行する
5. 動作を確認する
6. 手順3に戻ってアプリが完成するまで繰り返す

具体的に開発手順を確認していこう

STEP_01

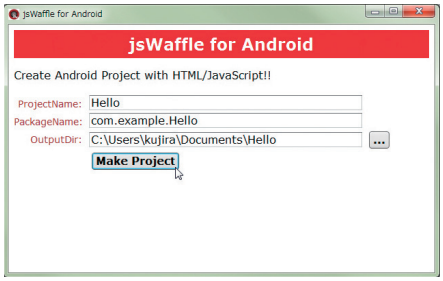
では、ここから、実際に手を動かして開発手順を確認していきましょう。OSのアプリケーション一覧から、jsWaffleを起動します。Windowsなら、スタートメニューから［すべてのプログラム ＞ jsWaffle］をクリックします。前の項目でインストール後、起動したままの人はSTEP_02へ進んでください。



STEP_02

jsWaffleが起動したら、プロジェクト名やパッケージ名を記入します。ここでは、下のように値を入力してください。ProjectNameにはアプリケーションの名前を、PackageNameには他と重複しない名前（通常はWebサイトのドメイン名を逆さにしたもの+アプリ名）を指定します。OutputDirはマイドキュメントなどを指定します。

ProjectName：Hello
PackageName：com.example.Hello
OutputDir：<マイドキュメントのパス>Hello



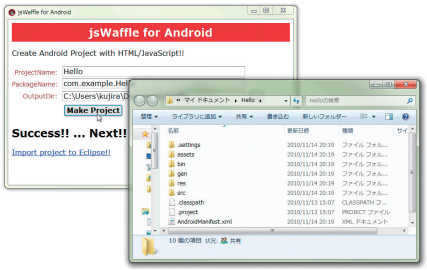
TIPS // 02-007

OutputDirのパス

このOutputDirには、Aptana（Eclipse）のWorkspace以外のパスを指定してください。Workspace内にひな型を生成した場合には、操作の手順が異なってしまいます。

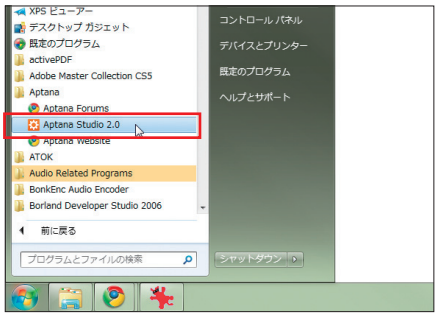
STEP_03

記入が終わったら、［Make Project］ボタンをクリックします。すると、フォームに記入した値に応じた、Androidプロジェクトのひな型が生成されます。生成が終わると、そのディレクトリが起動します。
jsWaffleと開いたディレクトリは閉じてしまって構いません。



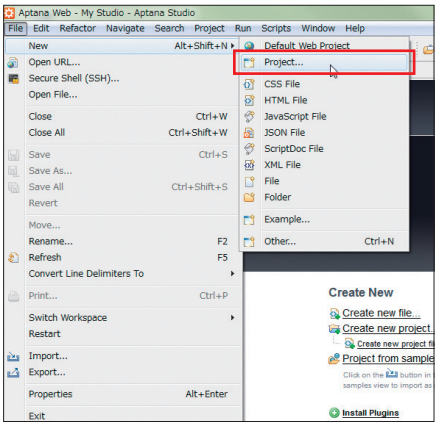
STEP_04

次に、Aptana（あるいはEclipse）を起動します。Windowsなら、スタートメニューから、［すべてのプログラム ＞ Aptana ＞ Aptana Studio 2.0］をクリックします。



STEP_05

そして、新規プロジェクトを作成します。Aptanaが起動したら、メニューから［File > New > Project...］をクリックします。



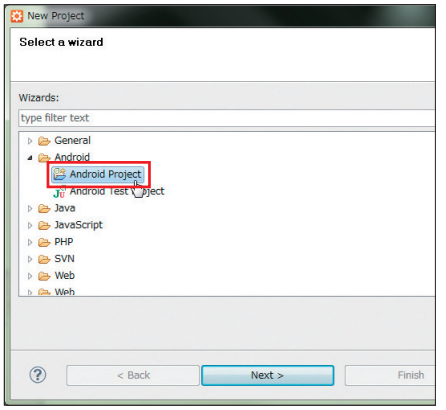
TIPS // 02-008

Aptanaのメニューに関して

Aptanaの設定により、若干メニューの項目が変わる可能性があります。メニューの［File］以下に表示される［Aptana Project］をクリックすれば、STEP_05 ～ 06と同じ動作になります。

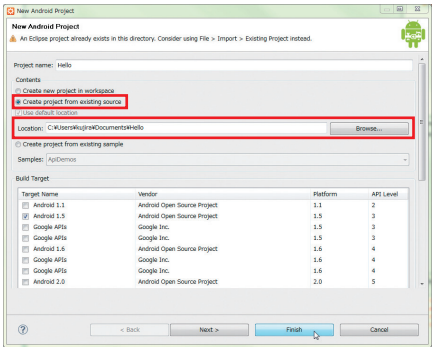
STEP_06

［New Project］のダイアログが表示されたら、［Android > Android Project］を選択状態にして、［Next］ボタンをクリックします。すると、［New Android Project］のダイアログが表示されます。



STEP_07

ここで、先ほど作成した、jsWaffleのひな型を元にして新規プロジェクトを作成します。Project nameの下側、Contentsの囲みから、［Create project from existing source］をチェックします。それから、Locationの部分に、jsWaffleで生成したプロジェクトのパスを選択します。上部に警告が出ますが、無視して進めてください。プロジェクト名が自動的に設定されたら、［Finish］ボタンをクリックします。



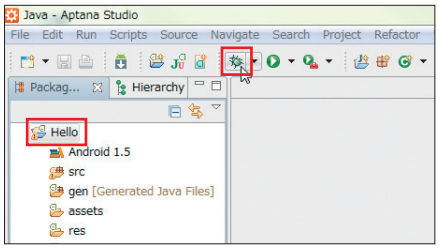
TIPS // 02-009

Android SDKのパスに注意

この作業の前に、Workspaceに、Android SDKのパスを設定してある必要があります（P.39参照）。

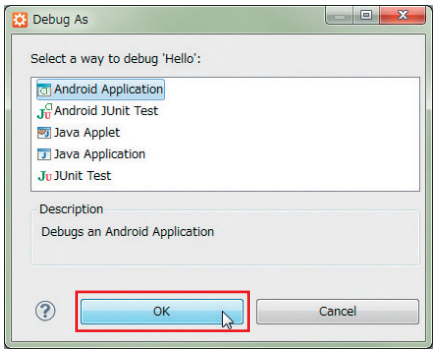
STEP_08

Package Explorerでプロジェクトを選択し、メニューのデバッグボタンをクリックして、jsWaffleのデモを実行してみましょう。ここでエラーが表示される場合は、この後の「コンパイラバージョンの確認」を試してみてください。



STEP_09

続いて、[Debug As]のダイアログが表示されるので［Android Application］を選択して［OK］ボタンをクリックします。
すると、Androidのエミュレータが起動します。はじめてエミュレータを起動させる時にはしばらく時間がかかります。Androidのロゴが光っている間は起動中です。じっと待ちましょう。



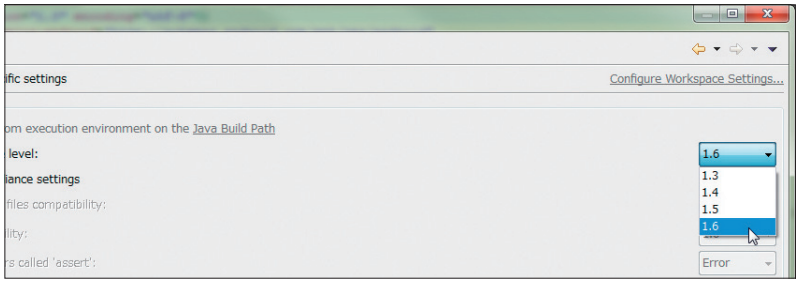
STEP_10

jsWaffleのデモが実行されたら、ボタンをクリックして動作を確認してみましょう。エミュレータ上では体験できない機能もありますが、雰囲気がつかめると思います。もし、パソコンとAndroid端末の実機をUSBでつないでいるなら、エミュレータの代わりに、実機でアプリケーションが実行されます（詳しくはChapter02末尾のコラムをご覧ください）。

TIPS // 02-010

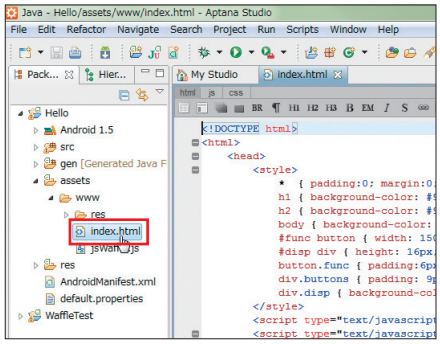
コンパイラバージョンの確認（エラーが出る場合）

もし、エラーが出てデモが実行できない場合は、コンパイラのバージョンが問題になっている場合があります。その場合は、Package Explorerでプロジェクトを右クリックして［Properties］を選択し、［Java Compiler］にあるコンパイラのバージョンを1.6に変更してみてください。



STEP_11

デモを表示しているHTMLを、自分のHTMLファイルに差し替えたり、編集することで独自のアプリケーションを作ることができます。ここでは、画面にHelloと表示するアプリケーションを作ってみます。AptanaのPackage Explorerから、ファイル「assets/www/index.html」をダブルクリックして開きます。

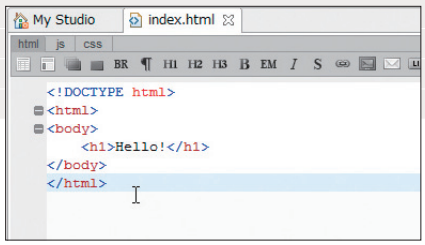


STEP_12

ファイル「assets/www/index.html」はアプリケーションが実行された時に表示される、メインアプリケーションHTMLファイルです。このファイルを、以下のように書き換えて保存してください。

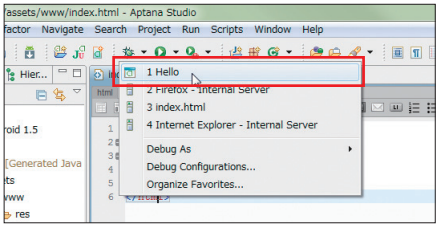
| スクリプト 02-001 | サンプルファイル:assets/www/index.html

```
01 <!DOCTYPE html>
02 <html>
03 <body>
04   <h1>Hello!</h1>
05 </body>
06 </html>
```



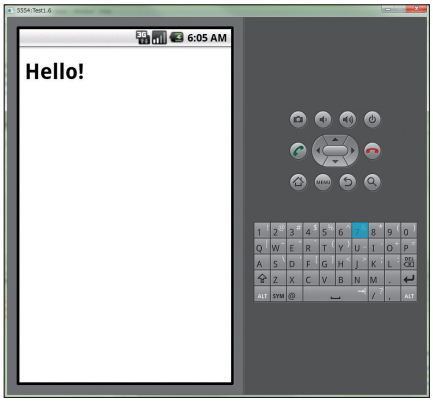
STEP_13

書き換えた内容を実行するには、デバッグボタンをクリックします。二度目に実行する場合には、デバッグボタンの右側の▼ボタンから選んで実行することができます。



STEP_14

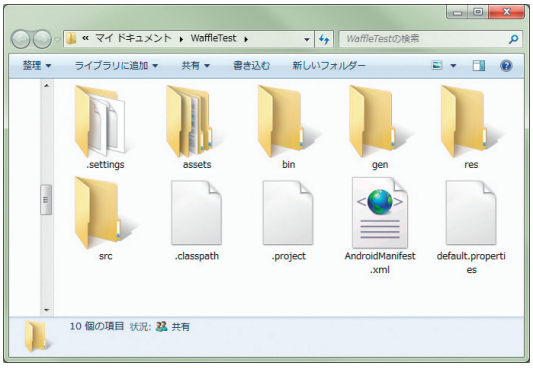
そして、無事にAndroidエミュレータで実行されると、右記のように表示されます。



jsWaffleのフォルダ構成を確認しよう

jsWaffleを使って作ったAndroidプロジェクトのひな型は、以下のようなフォルダ構成になっています。ここでは、その役割を簡単に紹介します。

| 図 02-003 |



jsWaffleで生成したプロジェクトのひな型

プロジェクトのルートディレクトリは次のような意味を持っています。これは、Javaで作ったAndroidプロジェクトの一般的な構成となっています。