JSでAndroidアプリを作ろう

PHP祭2012 Androidハンズオン
Nov. 3rd, 2012 @adamrocker

INDEX:

このハンズオンで完成するAndroidプロジェクト→ http://goo.gl/oMqv7

0. 環境構築

0-1. Eclipseをダウンロード

0-2. EcipseのADT(Android Development Tools)プラグインをインストール

Help > **Install New Software**->[Add]

https://dl-ssl.google.com/android/eclipse/

[REF] http://developer.android.com/intl/ja/sdk/installing/installing-adt.html

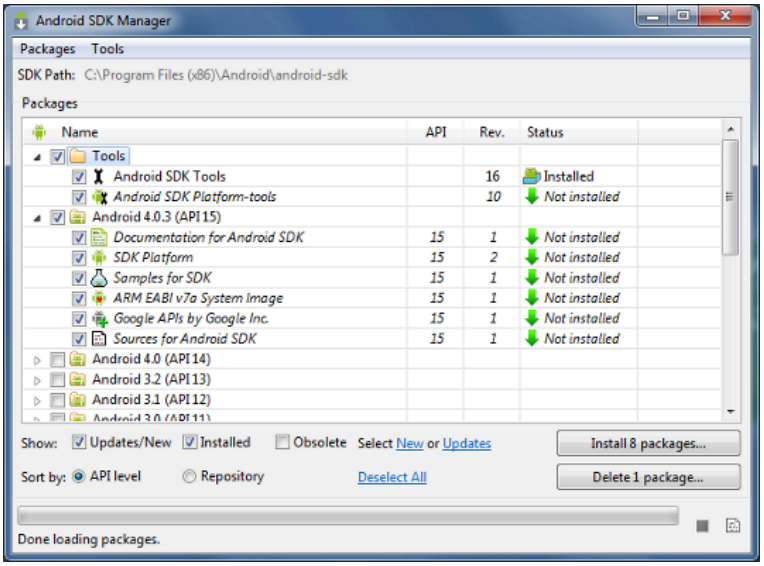
0-3. Android SDKをダウンロード

http://developer.android.com/intl/ja/sdk/index.html

zipを展開して適当な場所に配置。$SDK = /dev/android-sdk-mac_x86

0-4. Androidプラットフォームとパッケージを追加

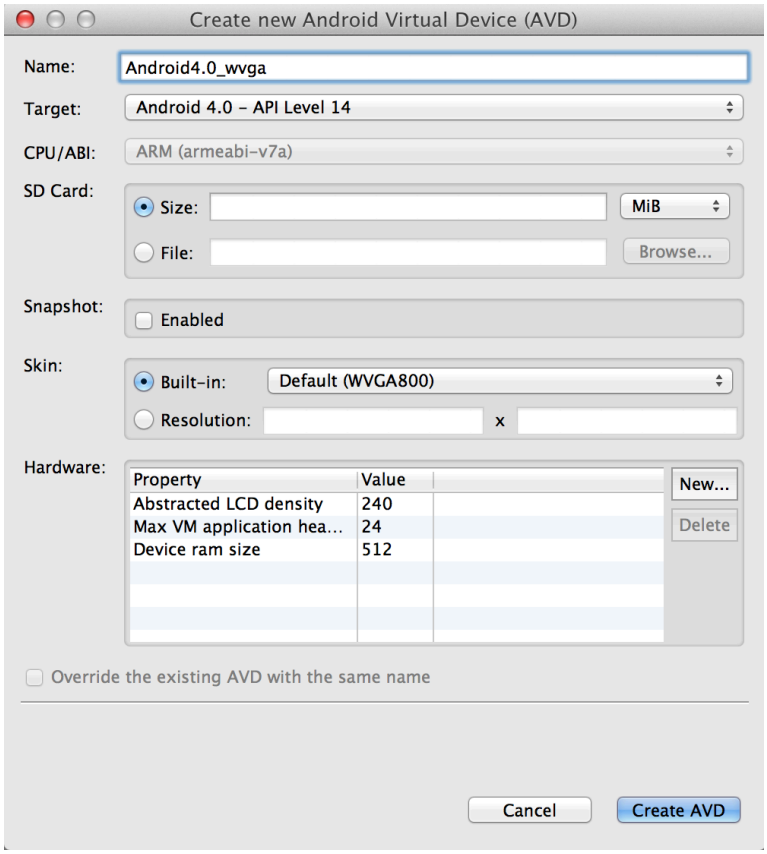
Eclipseの[Android SDK Manager]か${SDK}/tools/android_sdkで管理画面を開く。
Toolsと必要なSDKを追加する。



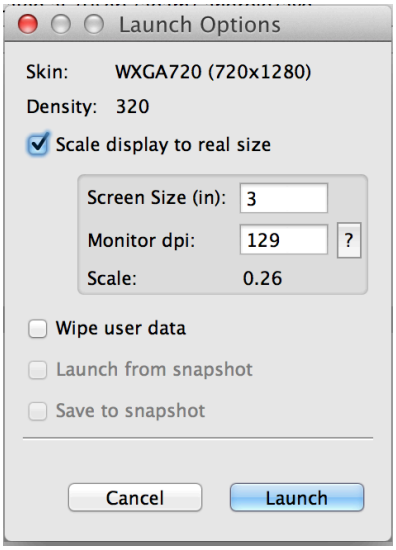
1. エミュレータでテスト環境を作る

1-1. Android4.0/WVGAなど。OSのバージョンや画面サイズ、ハードスペックを自由に作る。
実機を持っている人は特に必要ありません。

エミュレータはアプリの挙動などを解析するのに便利です。

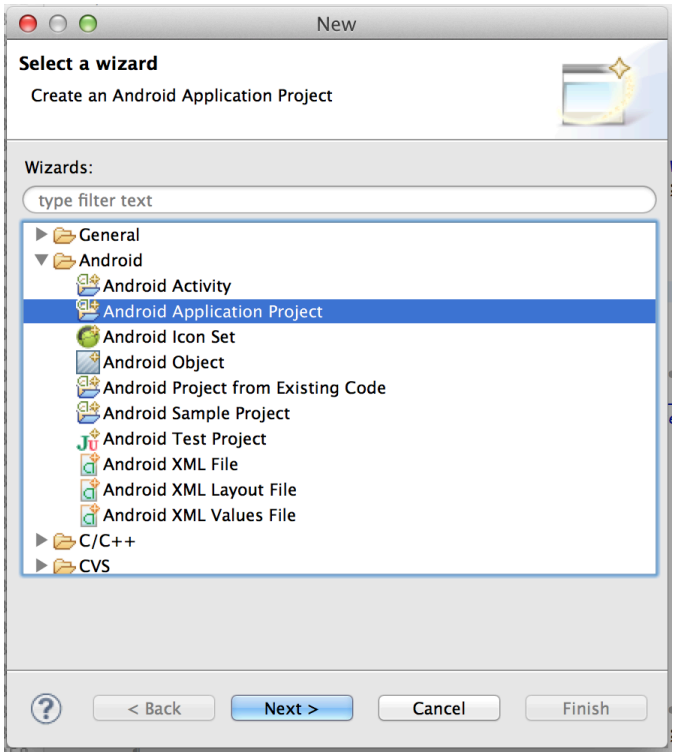


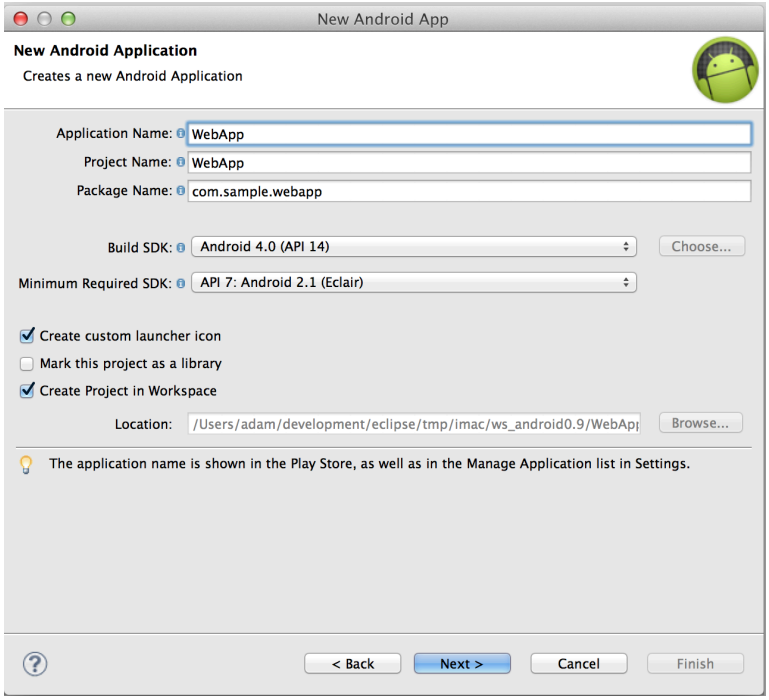
1-2. エミュレータサイズが大きすぎる時はエミュレータ起動時にスケール変更できます。



2. ネイティブなWebアプリを作る

2-1. Webアプリはブラウザベースが一般です。
つまりブラウザっぽいアプリを作ればWeb技術でネイティブっぽいアプリが作れます。
現時点でのAndroid向けFbアプリも同じ方法です。
2-2. Androidプロジェクトを作る。

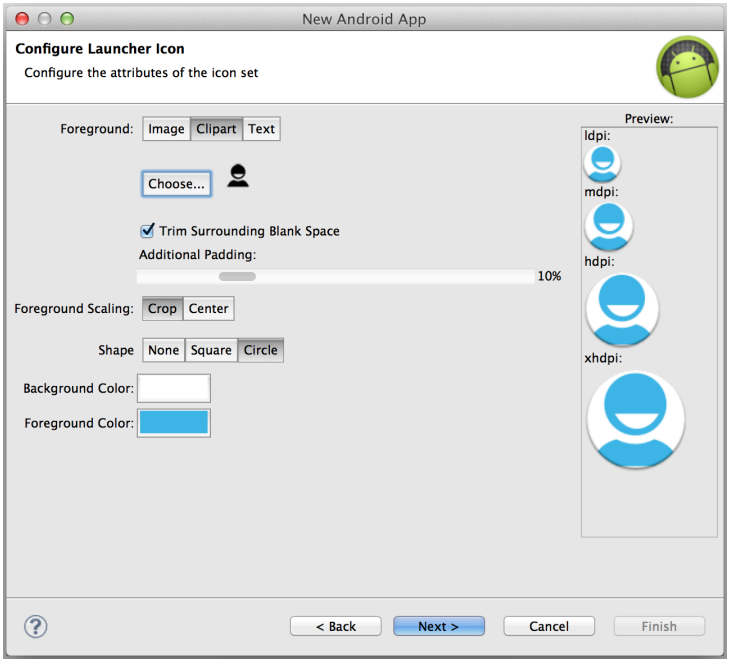




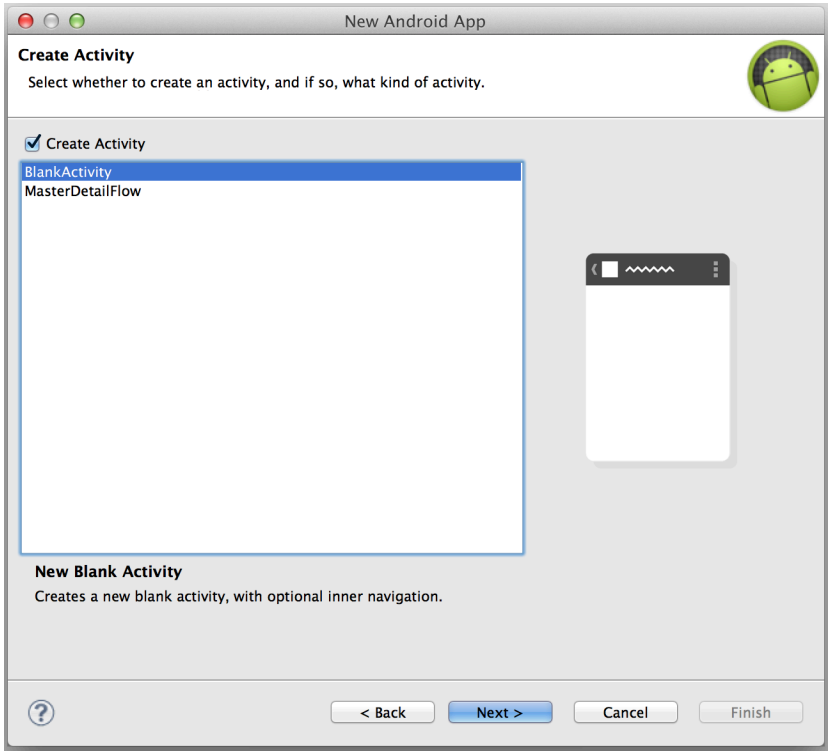
Application Name: アプリの名前（後で変更可能）
Project Name: Eclipseプロジェクトの名前（後で変更可能）
Package Name: 世界で唯一の文字列にする（たいていドメイン名。後で変更可能）

Build SDK: アプリをビルドするツールのバージョン（後で変更可能）
Minimum Required SDK:アプリの実行をサポートするOSバージョン（後で変更可能）

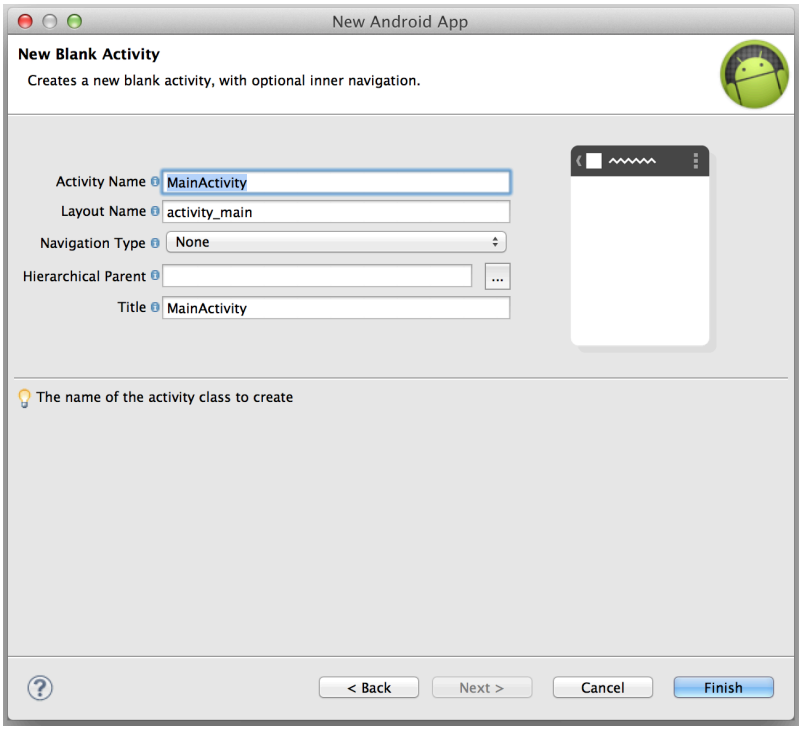
簡単なアイコンを作れる。サンプルが増えて来るとアイコンが違った方が分かりやすい。



テンプレを選択。BlankでOK。Minimum Required SDKが高いと選択肢が増える。

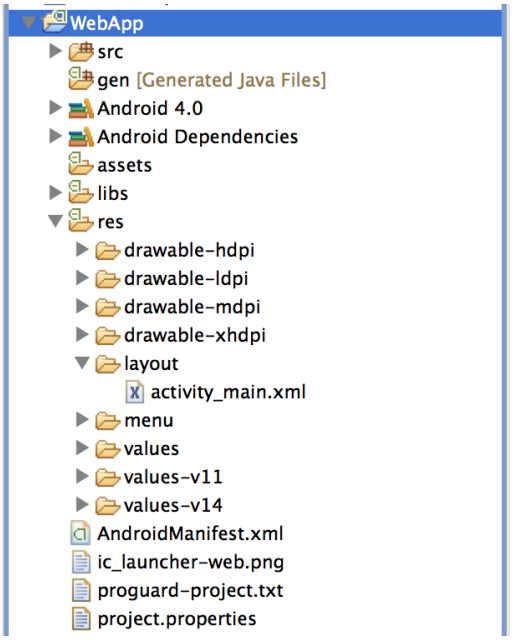


テンプレの詳細を作る。

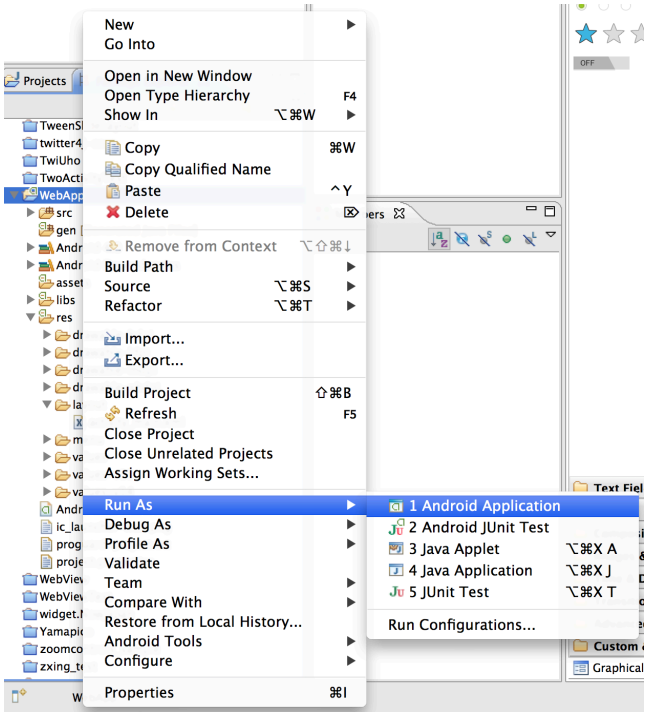


Activity Name: アクティビティの名前(クラス名)を指定（後で変更可能）
Layout Name: 画面レイアウトを定義するファイル名を指定（後で変更可能）
Navigation Type: テンプレを指定（変更不可、自前でカスタマイズする場合はNoneでOK）
Hierarchical Parent: アクティビティの親(extends)を指定
Title: アクティビティのタイトル名を指定（後で変更可能）

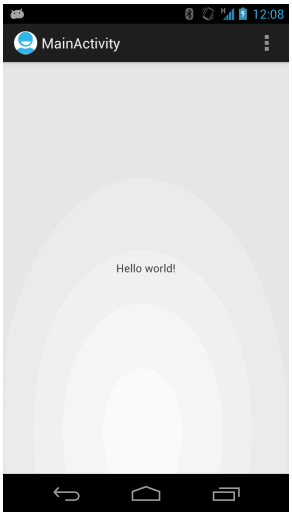
オメデトウございます！アプリが完成しました。



さっそく実行してみましょう。



実機で実行したい場合は、事前に**[設定]-[開発者向けオプション]-[USB**でバッグ]をON。
何も接続していない場合はエミュレータが起動する。



何故かビルドが失敗している。何もしていないのに動かない。昨日まで動いていたのに今日動かない。そんな時は落ち着いて

- Clean Build (menu)-[Project]-[Clean...]
- Refresh (プロジェクト右クリック)-[Refresh (F5)]
- Eclipseの再起動

どれかを試してみてください。

2-3. Webアプリのベース(WebView)を作る
画面レイアウトを定義するファイル(WebApp/res/layout/activity_main.xml)を開く。
グラフィカルなレイアウトが表示されているので、左下のタブでテキストを開く。

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
        xmlns:tools="http://schemas.android.com/tools"
        android:layout_width="match_parent"
        android:layout_height="match_parent" >

        <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"
        android:layout_centerVertical="true"
        android:text="@string/hello_world"
        tools:context=".MainActivity" />

</RelativeLayout>
```

基本はXMLでレイアウトを作ります。HTMLみたいなモノです。デフォルトでは各要素に飾り付けのattributeが付いていますが、res/values/styles.xmlにattributeを纏めるのがオススメです。HTMLとCSSを分離させるのと同じです。

このレイアウトファイルを編集してWebアプリのベースを作ります。

```
<WebView xmlns:android="http://schemas.android.com/apk/res/android"
        xmlns:tools="http://schemas.android.com/tools"
        android:id="@+id/web"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />
```

練習のためにstyles.xmlにスタイル指定を移動します。

res/layout/activity_main.xml

```
<WebView xmlns:android="http://schemas.android.com/apk/res/android"
        xmlns:tools="http://schemas.android.com/tools"
        android:id="@+id/web"
        style="@style/WebBase" />
```

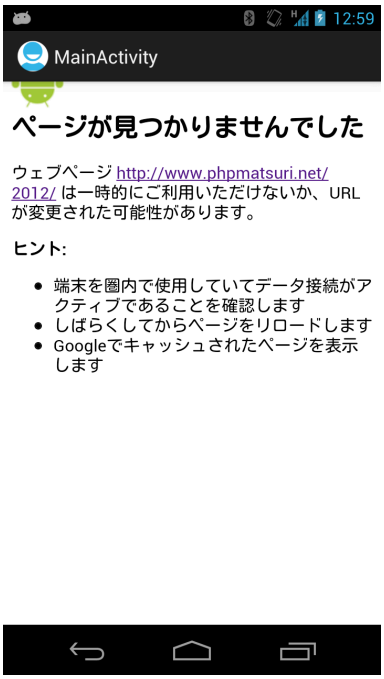
res/values/styles.xml

```
<resources xmlns:android="http://schemas.android.com/apk/res/android">

    <style name="AppTheme" parent="android:Theme.Light" />
    <style name="WebBase">
        <item name="android:layout_width">match_parent</item>
        <item name="android:layout_height">match_parent</item>
    </style>
</resources>
```

WebアプリのURLを指定する。

```
public class MainActivity extends Activity {
        private WebView mWeb;

        @Override
        public void onCreate(Bundle savedInstanceState) {
                super.onCreate(savedInstanceState);
                setContentView(R.layout.activity_main);
                mWeb = (WebView) findViewById(R.id.web);
                mWeb.loadUrl("http://www.phpmatsuri.net/2012/");
        }
}
```

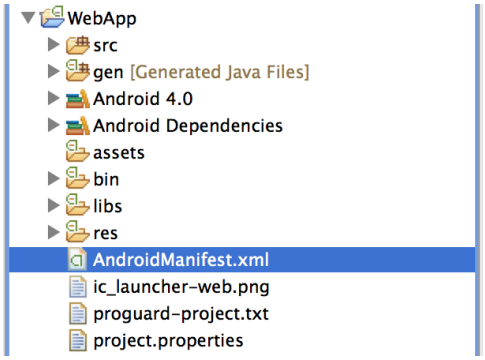
さっそく実行してみる。



エラー。インターネットにアクセスするには「パーミッション」を設定する必要がある。

3. インターネットアクセスを許可

パーミッションはユーザに公開されるアプリの持つ特性です。
パーミッションはAndroidManifest.xmlに追記する。Manifestファイルはアプリの情報を記載する。



AndroidManifest.xmlをEclipseで開くと専用エディタが開くので便利。

[Permissions]タブで[Users Permission]で<android.permission.INTERNET>を追加。

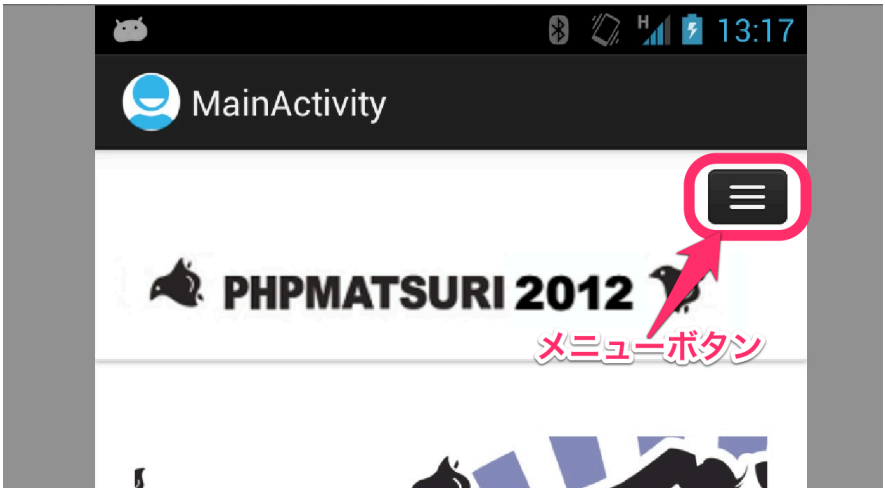
```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
        package="com.sample.webapp"
        android:versionCode="1"
        android:versionName="1.0" >
        <uses-sdk
            android:minSdkVersion="7"
            android:targetSdkVersion="15" />
        <uses-permission android:name="android.permission.INTERNET"/>
        <application
            android:icon="@drawable/ic_launcher"
            android:label="@string/app_name"
            android:theme="@style/AppTheme" >
            <activity
                android:name=".MainActivity"
                android:label="@string/title_activity_main" >
                <intent-filter>
                    <action android:name="android.intent.action.MAIN" />
                    <category android:name="android.intent.category.LAUNCHER" />
                </intent-filter>
            </activity>
        </application>
</manifest>
```

もう一回、実行！

サイトが表示されました！
これで読み込むURLを変更すればWebアプリベースのネイティブアプリが作れます。

4. JavaScriptを許可

WebViewはプレーンなwebエンジン。HTML/CSSが解釈できるだけ。
試しに、PHP祭り2012のメニューボタンを押してみてください。



何も起きません。

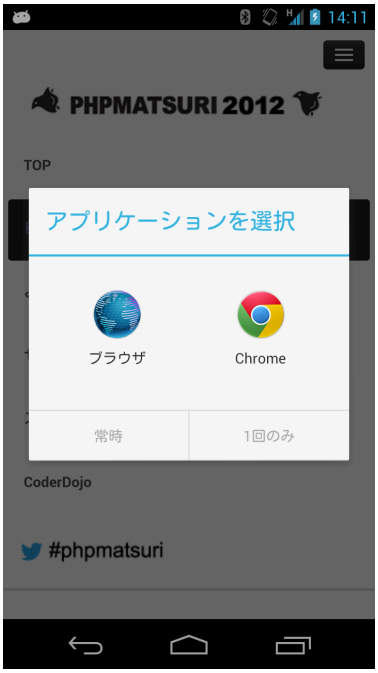
カルーセル回りません。。

何か足りない。。。
そうだJSだ！
WebViewにJavaScriptエンジンを追加する。

```
mWeb.getSettings().setJavaScriptEnabled(true);
```

もう一回実行。

完成！これでHTML/CSS/JSでアプリが作れる！何でも出来る！
ちょっとリンクタップしてみましょう。



ブラウザが起動しちゃう。アプリ内で収まって欲しい！
ページ遷移を制御する。

```
private WebViewClient mClient = new WebViewClient(){
        @Override
        public boolean shouldOverrideUrlLoading (WebView view, String url) {
            mWeb.loadUrl(url);//ページ遷移を制御
            return false;
        }
```

```
};

public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        //.....
        mWeb.setWebViewClient(mClient);
}
```

5. タイトルを消す

タイトルもHTML/CSS/JSで面倒見るのでネイティブのタイトルを消す。
AndroidManifest.xmlでActivityのテーマ(Theme)を変更する。

[Before]

```
<activity
        android:name=".MainActivity"
        android:label="@string/title_activity_main" >
```

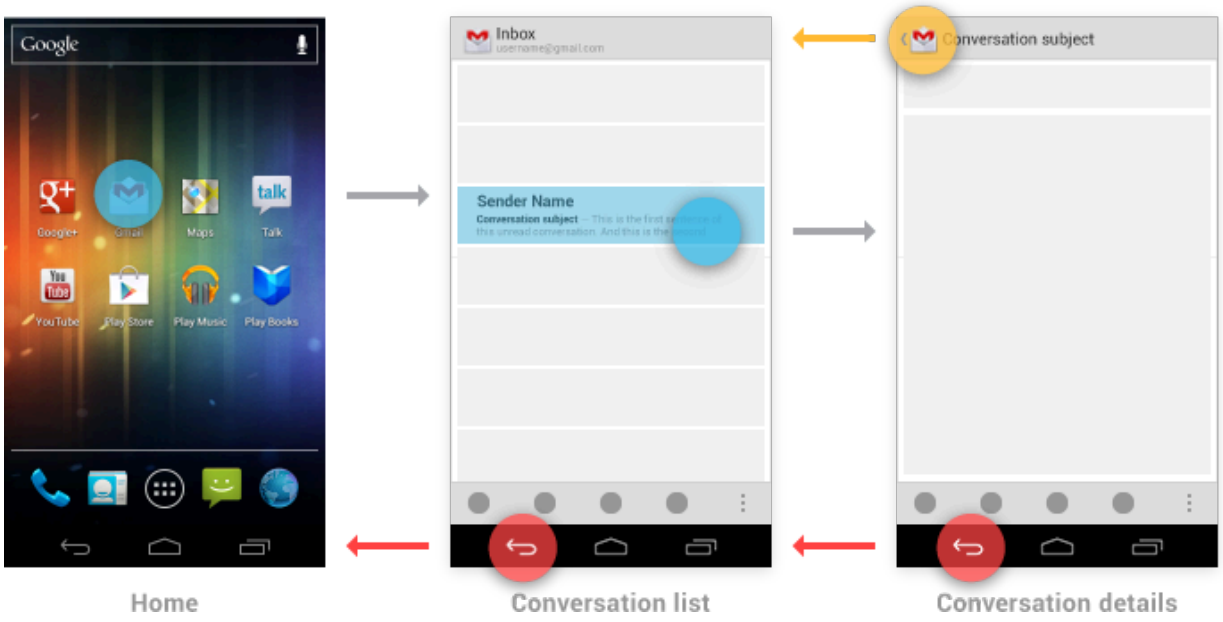
[After]

```
<activity
        android:name=".MainActivity"
        android:label="@string/title_activity_main"
        android:theme="@android:style/Theme.NoTitleBar">
```

タイトル消えてスッキリ。

6. BACKキーの処理

AndroidはiPhoneとは違います。
最大の違いはBACKキー。BACKキーは1つ前の画面に戻る動作が期待されます。
動作設計についてもドキュメントが纏められているので参考にしてください。
http://developer.android.com/design/patterns/navigation.html



Webアプリでもページが複数ある場合はBACKキーの動作を制御する。

```
public boolean dispatchKeyEvent(KeyEvent event) {
        if (event.getKeyCode() == KeyEvent.KEYCODE_BACK) {
                if (event.getAction() == KeyEvent.ACTION_UP) {
                        if ( mWeb.canGoBack() ) {
                                mWeb.goBack();
                                return true;
                        }
                }
        }
        return super.dispatchKeyEvent(event);
}
```

これでHTML/CSS/JSでアプリが作れるようになりました。皆さんは自由です。

ここから先はちょっと発展。

7. JSからNativeを操作する

Webの世界からデバイスの世界へ。
JSでNativeメソッドを実行できる。

```
private class Native {
        public String hello() {
```

```
            return "Hello, Native!";
        }
    }

    Native nat = new Native();
    mWeb.addJavascriptInterface(nat, "native");
```

natオブジェクトがJSオブジェクト(native)に変換される。
JSから以下のようにJavaメソッドをcallできる。

```
if ( native ) {
    var text = native.hello();
}
```

使い道は、アプリのバージョンを取得するとか、デバイス情報を取得するとか、色々。
これでみなさんはWeb/Device両方の世界で自由です。

A. 参考資料

- WebViewをもっとカスタマイズするための資料。
  [Android WebView Hnads-on]

- AndroidのUI/設計関係の超重要資料
  [Android Designs]

- Androidのネイティブアプリを作ってみたい
  [Androidトレーニング]

- Android開発環境
  [Android SDK Install]