

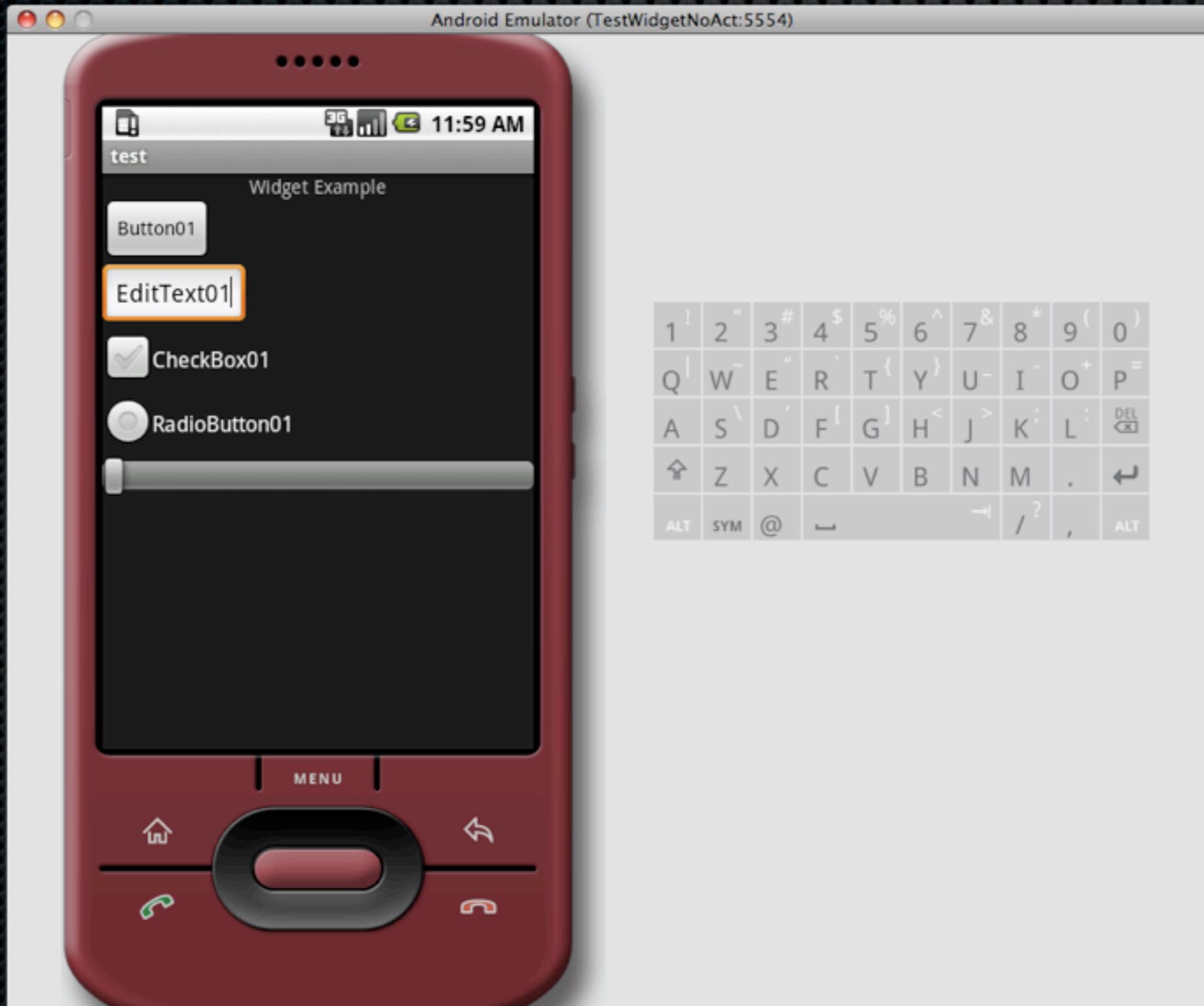
# AppWidgetで Widgetを作ろう！

日本Androidの会

宮川 彬央

# Widgetとは？

- 特定の機能をもった、小さなアプリケーション

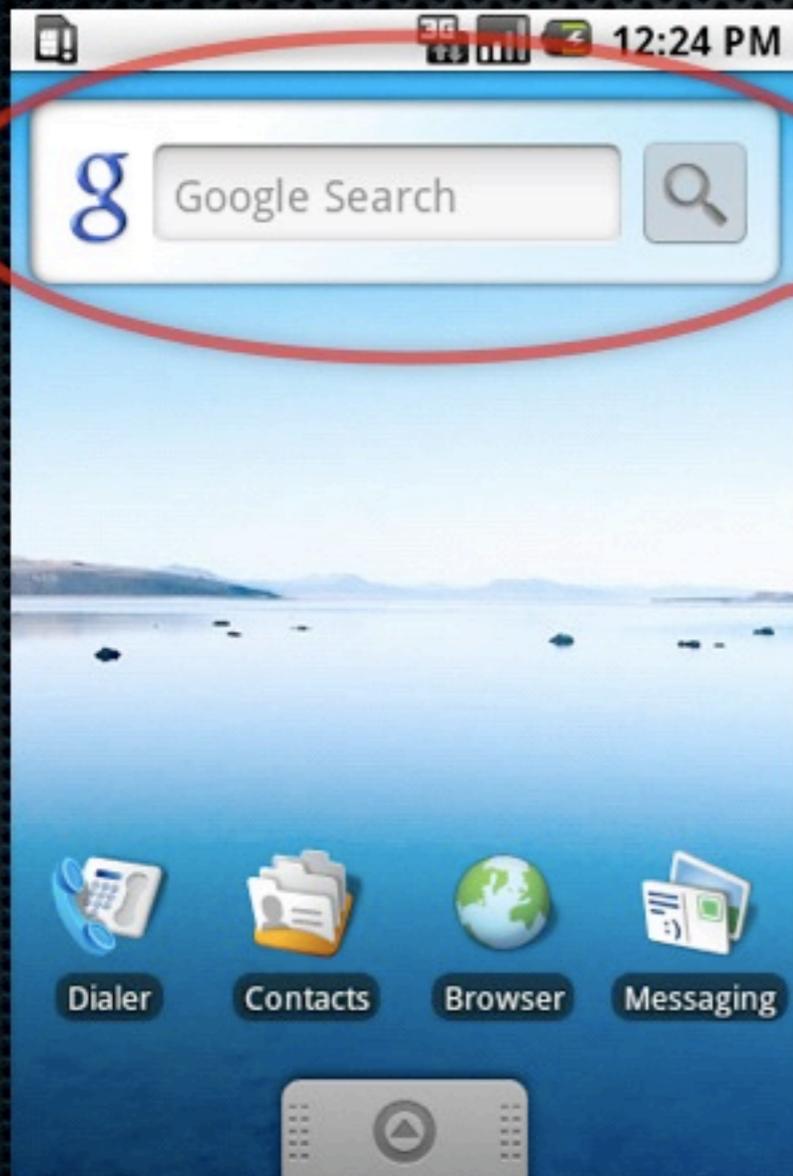


# App Widgetとは？

- Widgetを作るためのフレームワーク

好きな機能を実装したWidgetを作ることができる！

(※Android SDK 1.5から可能)

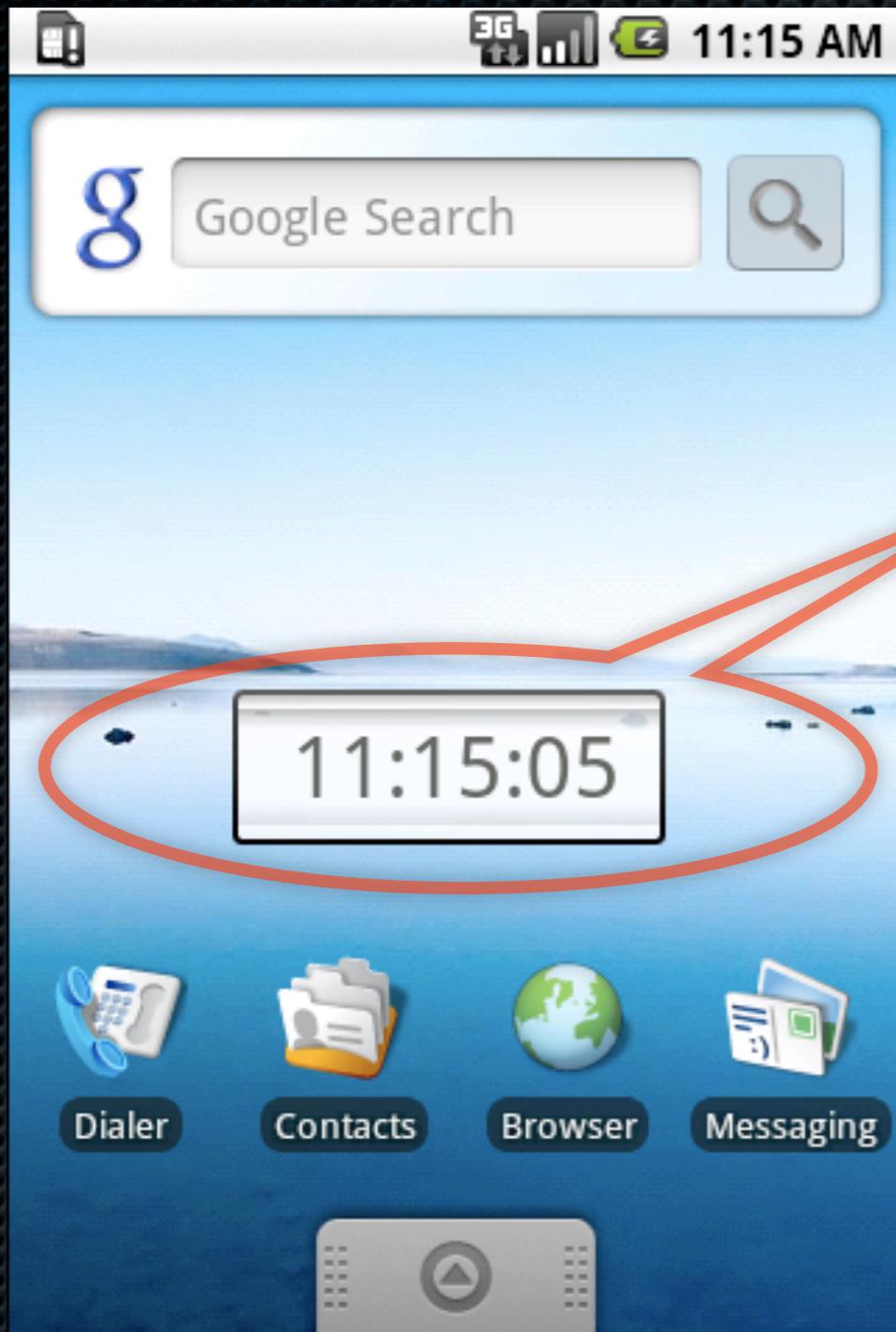


- Google Search

- ・ (自分で)ブラウザを立ち上げずに、google検索ができる

~~ブラウザ起動~~ > 入力 > 結果表示

# 時計を作ってみよう！



## ●ClockWidget

- ・テキストで現在の時間を表示する
- ・1秒毎に更新(・・・)
- ・20秒ごとに背景が変わります  
(※意味はないです)
- ・時計、右上にあるやん  
(※禁句です)

# 作成手順

- 1.新規プロジェクトをAndroid 1.5で作成
- 2.App Widgetのレイアウトを作成
- 3.App Widgetの特徴を決める
- 4.App Widgetの処理(時刻を表示)を作成
- 5.AndroidManifest.xmlにreceiverを追記

# 1. 新規プロジェクトを作成

New Android Project

Creates a new Android Project resource.

Project name: ClockWidget

Contents

- Create new project in worksp...
- Create project from existing sou...
- Use default locati...

Location: Users/miyagawaakio/Documents/workspace/ClockWidget

Build Target

Target Name	Vendor	Platform	A...vel
<input type="checkbox"/> Android 1.1	Android Open Source Project	1.1	2
<input checked="" type="checkbox"/> Android 1.5	Android Open Source Project	1.5	3
<input type="checkbox"/> Google APIs	Google Inc.	1.5	3

Standard Android platform 1.5

Properties

Application name: ClockWidget

Package name: jp.android.ClockWidget

Create Activity:

Min SDK Version: 3

< Back Next > Cancel Finish

Android 1.5をチェック

jp.android.ClockWidget

チェックをはずす  
(Activityなし)

# 2. App Widgetのレイアウトを作成

- ClockWidget/res/layout/main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    >

    <TextView
        android:id="@+id/widget_0"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:background="@android:drawable/spinner_background"
        android:textColor="#555555"
        android:padding="5dp"
        android:textSize="30.0dp"
        android:gravity="center"
        android:visibility="gone"
    />

    <TextView
        android:id="@+id/widget_1"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:background="@android:drawable/dark_header"
        android:textColor="#FFFFFF"
        android:padding="5dp"
        android:textSize="30.0dp"
        android:gravity="center"
        android:visibility="gone"
    />

    <TextView
        android:id="@+id/widget_2"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:background="@android:drawable/divider_horizontal_textfield"
        android:textColor="#555555"
        android:padding="5dp"
        android:textSize="30.0dp"
        android:gravity="center"
        android:visibility="gone"
    />
</LinearLayout>
```

## 3-1. App Widgetの特徴を決める

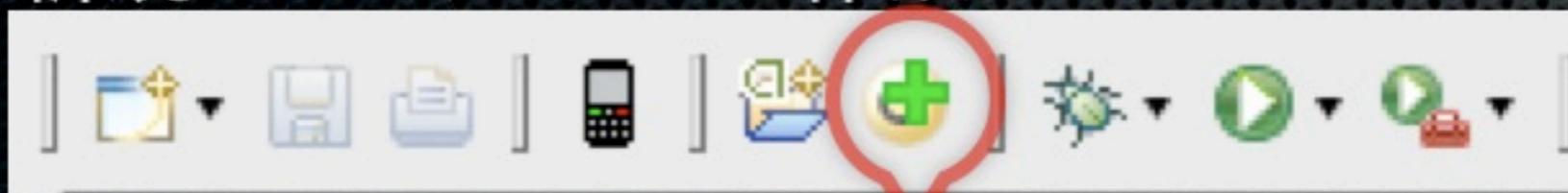
App Widgetの

- ・ 幅
- ・ 高さ
- ・ 更新間隔
- ・ 参照するレイアウト

等をxmlで定義する。

## 3-2. App Widgetの特徴を決める

- 新規xmlファイルを作成



↑ toolbarのアイコンを選択

File : widget.xml

AppWidget Provi...  
をチェック

New Android XML File

Creates a new Android XML file.

Project: ClockWidget

File: widget.xml

What type of resource would you like to create:

- Layout
- Values
- Menu
- AppWidget Provi...
- Preferen...
- Searcha...
- Animation

What type of resource configuration would you like:

Available Qualifiers	Chosen Qualifiers
Country Code	
Network Code	
Language	
Region	
Orientation	
Pixel Density	
Touch Screen	
Keyboard	
Text Input	
Navigation	

Folder: /res/xml

Select the root element for the XML file: appwidget-provider

Cancel Finish

## 3-3. App Widgetの特徴を決める

- ClockWidget/res/xml/widget.xml

```
<?xml version="1.0" encoding="utf-8"?>
<appwidget-provider xmlns:android="http://schemas.android.com/apk/res/android"
    android:minWidth="146dp"
    android:minHeight="72dp"
    android:updatePeriodMillis="1000"
    android:initialLayout="@layout/main"
/>
```

android:minWidth	App Widgetを表示するのに最低限必要な幅
android:minHeight	App Widgetを表示するのに最低限必要な高さ
android:updatePeriodMills	App Widgetが更新される間隔(ミリ秒)
android:initialLayout	App Widgetのレイアウト

# 4-1. App Widgetの処理を作成

- 新規javaファイルを作成

パッケージ名を右クリック > New > Class

**Package :**  
jp.android.ClockWidget

**Name :**  
ClockWidgetProvider

**Superclass :**  
android.appwidget.AppWidgetProvider

Java Class  
Create a new Java class.

Source folder: ClockWidget/src Browse...

Package: jp.android.ClockWidget Browse...

Enclosing type: Browse...

Name: ClockWidgetProvider

Modifiers:  public  default  private  protected  
 abstract  final  static

Superclass: android.appwidget.AppWidgetProvider Browse...

Interfaces: Add... Remove

Which method stubs would you like to create?

public static void main(String[] args)  
 Constructors from superclass  
 Inherited abstract methods

Do you want to add comments? (Configure templates and default value [here](#))

Generate comments

? < Back Next > Cancel Finish

# 4-2. App Widgetの処理を作成

- ClockWidget/src/パッケージ名/ClockWidgetProvider.java

```
public class ClockWidgetProvider extends AppWidgetProvider {
    private static int[] view_list = {
        R.id.widget_0,
        R.id.widget_1,
        R.id.widget_2,
    };

    @Override
    public void onUpdate(Context context, AppWidgetManager appWidgetManager, int[] appWidgetIds) {
        super.onUpdate(context, appWidgetManager, appWidgetIds);

        // 現在の時間を取得
        Time t = new Time();
        t.setToNow();

        // 20秒毎に背景を変える
        int view_index = 2;
        if( t.second < 20 ) {
            view_index = 0;
        }
        else if( t.second < 40 ) {
            view_index = 1;
        }
        int target_id = view_list[view_index];

        // Widget更新処理を呼ぶ
        final int length = appWidgetIds.length;

        for(int i = 0; i < length; i++) {
            int appWidgetId = appWidgetIds[i];
            updateAppWidget(context, appWidgetManager, appWidgetId,
                R.layout.main, target_id, t.format("%H:%M:%S"));
        }
    }
}
```

↓続きは次のページ

# 4-3. App Widgetの処理を作成

- ClockWidget/src/パッケージ名/ClockWidgetProvider.java

```
void updateAppWidget(Context context, AppWidgetManager appWidgetManager,
    int appWidgetId, int widgetLayoutId, int viewId, String setValue) {

    // Widgetのレイアウトを取得
    RemoteViews rv = new RemoteViews(context.getPackageName(), widgetLayoutId);

    // ターゲットのTextViewだけ見えるようにし、(VISIBLE)
    // それ以外のTextViewは見えなくする(GONE)
    rv.setInt(viewId, "setVisibility", View.VISIBLE);

    // 取得したレイアウト内のView(TextView)に値を設定
    rv.setCharSequence(viewId, "setText", setValue);

    final int length = view_list.length;
    for(int i = 0; i < length; i++) {
        int check_view = view_list[i];
        if( check_view == viewId ) continue;
        rv.setInt(check_view, "setVisibility", View.GONE);
    }
    // Widget更新
    appWidgetManager.updateAppWidget(appWidgetId, rv);
}

/**
 * App Widgetを消しても、onDeletedが呼ばれない(1.5のバグらしい・・・)
 * intentは飛んでくるので(APPWIDGET_DELETED)、自分でonDeletedを呼ぶ
 * @see http://developer.android.com/guide/topics/appwidgets/index.html#AppWidgetProvider
 */
@Override
public void onReceive(Context context, Intent intent) {
    final String action = intent.getAction();
    Bundle extras = intent.getExtras();

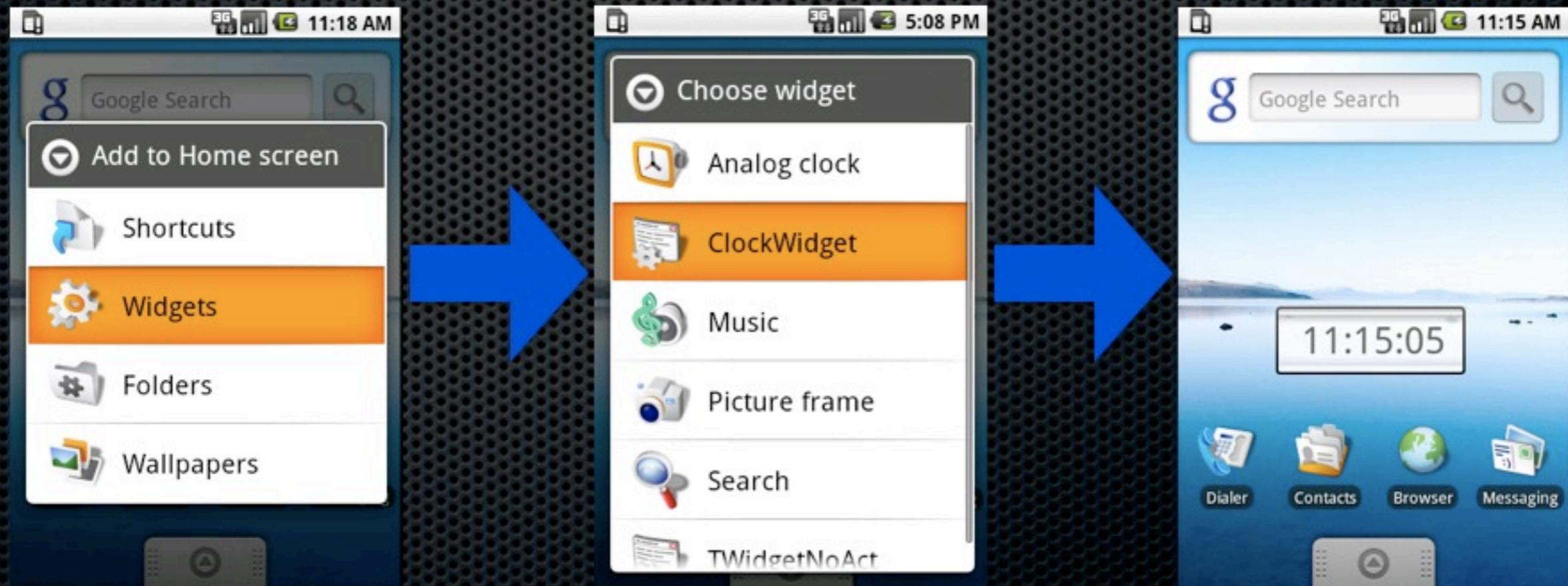
    if( extras != null && AppWidgetManager.ACTION_APPWIDGET_DELETED.equals(action) ) {
        final int appWidgetId = extras.getInt(AppWidgetManager.EXTRA_APPWIDGET_ID,
            AppWidgetManager.INVALID_APPWIDGET_ID);
        if( appWidgetId != AppWidgetManager.INVALID_APPWIDGET_ID ) {
            this.onDeleted(context, new int[]{appWidgetId});
        }
    }
    else {
        super.onReceive(context, intent);
    }
}
```

# 5. AndroidManifest.xmlにreceiverを追記

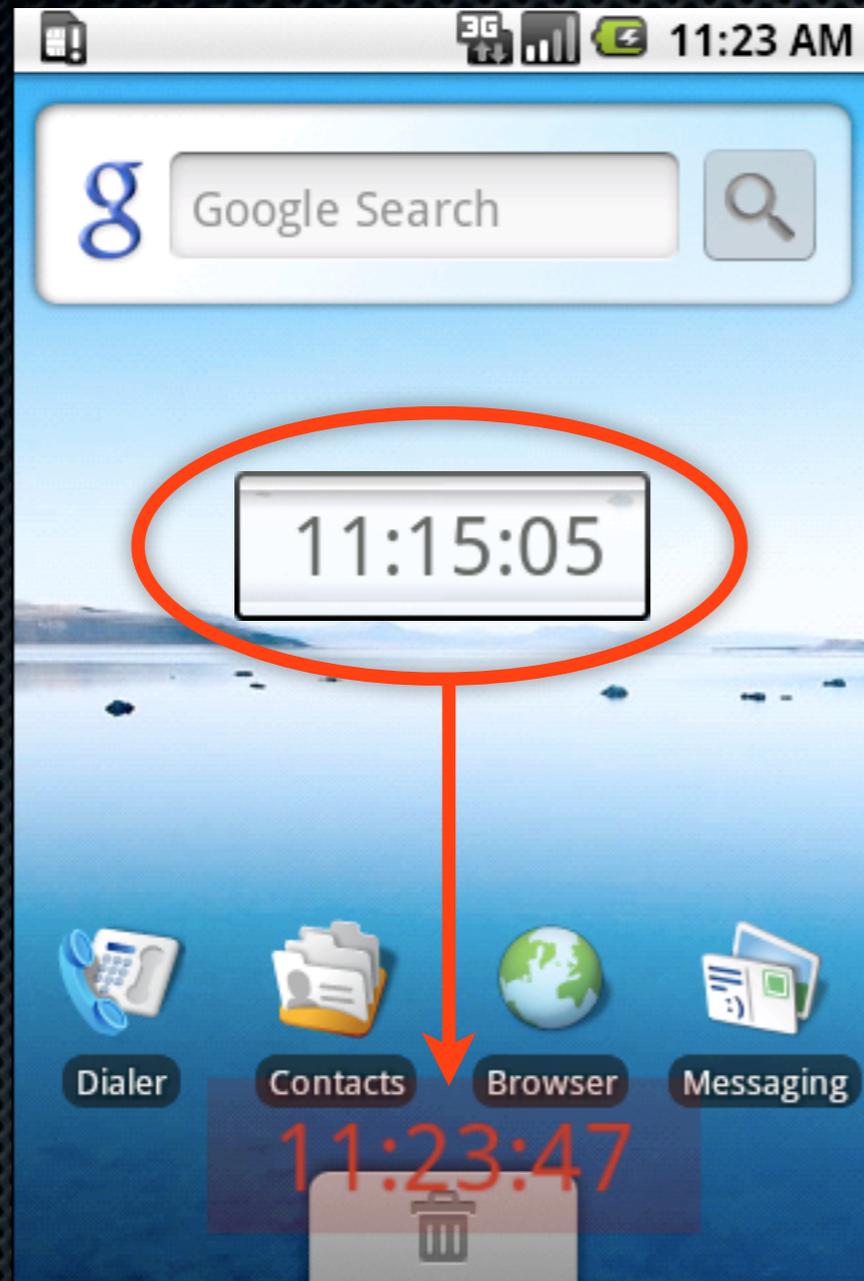
```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="jp.android.ClockWidget"
    android:versionCode="1"
    android:versionName="1.0">
    <application android:icon="@drawable/icon" android:label="@string/app_name">
        <receiver android:name=".ClockWidgetProvider">
            <intent-filter>
                <action android:name="android.appwidget.action.APPWIDGET_UPDATE" />
            </intent-filter>
            <meta-data android:name="android.appwidget.provider"
                android:resource="@xml/widget" />
        </receiver>
    </application>
    <uses-sdk android:minSdkVersion="3" />
</manifest>
```

# App WidgetをHome Screenに追加する

1. Run > Run or Debugで実行する
2. Home Screenで何も表示されていない箇所を長押し  
(MENUボタン > AddでもOK)
3. リストからWidgetsを選択
4. 作成したWidgetを選択



# App WidgetをHome Screenから削除する



消したいWidgetを長押しして選択した状態で、ドラッグ&ドロップして消します。

# ソース解説-1

・ ClockWidget/src/パッケージ名/ClockWidgetProvider.java

## onUpdate(・・・):

App Widgetが更新されると呼ばれる。

Widgetに一意的なIDが振られ、**appWidgetIds**に格納される。

同じWidgetが複数個配置されても、全てのWidgetに更新処理を行う必要あり。

```
public class ClockWidgetProvider extends AppWidgetProvider {  
(省略)  
    @Override  
    public void onUpdate(Context context, AppWidgetManager appWidgetManager, int[] appWidgetIds) {  
        super.onUpdate(context, appWidgetManager, appWidgetIds);  
  
(省略)  
        // Widget更新処理を呼ぶ  
        final int length = appWidgetIds.length;  
  
        for(int i = 0; i < length; i++) {  
            int appWidgetId = appWidgetIds[i];  
            updateAppWidget(context, appWidgetManager, appWidgetId,  
                R.layout.main, target_id, t.format("%H:%M:%S"));  
        }  
    }  
}
```

# ソース解説-2

・ ClockWidget/src/パッケージ名/ClockWidgetProvider.java

App Widgetのレイアウトは、Viewではなく、**RemoteViews**で取得。

更新しないと、Widgetの見た目が変わらない。

Widgetが消されても呼ばれないので、**onDeleted()**は自分で呼ぶ

```
void updateAppWidget(Context context, AppWidgetManager appWidgetManager,
    int appWidgetId, int widgetLayoutId, int viewId, String setValue) {
    // Widgetのレイアウトを取得
    RemoteViews rv = new RemoteViews(context.getPackageName(), widgetLayoutId);
    (省略)
    // Widget更新
    appWidgetManager.updateAppWidget(appWidgetId, rv);
}

@Override
public void onReceive(Context context, Intent intent) {
    final String action = intent.getAction();
    Bundle extras = intent.getExtras();

    if( extras != null && AppWidgetManager.ACTION_APPWIDGET_DELETED.equals(action) ) {
        final int appWidgetId = extras.getInt(AppWidgetManager.EXTRA_APPWIDGET_ID,
            AppWidgetManager.INVALID_APPWIDGET_ID);
        if( appWidgetId != AppWidgetManager.INVALID_APPWIDGET_ID ) {
            this.onDeleted(context, new int[]{appWidgetId});
        }
    }
    else {
        super.onReceive(context, intent);
    }
}
}
```

# ハマったところ

## ●RemoteViewsクラスのset\*メソッド

```
public void setInt (int viewId, String methodName, int value)
```

Call a method taking one int on a view in the layout for this RemoteViews.

### Parameters

<i>viewId</i>	The id of the view whose text should change
<i>methodName</i>	The name of the method to call.
<i>value</i>	The value to pass to the method.

Q1. 引数methodNameに何を指定するの？

A. viewIdで指定されたクラスのメソッドを指定する

viewIdは、TextViewのID。  
TextViewクラスのメソッド  
setTextを指定できる。

```
void updateAppWidget(省略) {  
    // Widgetのレイアウトを取得  
    RemoteViews rv = new RemoteViews(context.getPackageName(), widgetLayoutId);  
    // 取得したレイアウト内のView(TextView)に値を設定  
    rv.setCharSequence(viewId, "setText", setValue);  
    (省略)  
}
```

※画像はAndroid developersサイトからキャプチャーしました。

[http://developer.android.com/reference/android/widget/RemoteViews.html#setInt\(int,%20java.lang.String,%20int\)](http://developer.android.com/reference/android/widget/RemoteViews.html#setInt(int,%20java.lang.String,%20int))

# ハマったところ(続き)

## ●RemoteViewクラスのset\*メソッド(続き)

Q2. TextViewクラスの親クラス(View)のメソッド  
setBackgroundColorを呼ぶとエラー・・・

こんなエラーが出ます。

```
android.widget.RemoteViews
$ActionException: view:
android.widget.TextView can't use method
with RemoteViews: setBackgroundColor(int)
```

```
void updateAppWidget(省略) {
    // Widgetのレイアウトを取得
    RemoteViews rv = new RemoteViews(context.getPackageName(), widgetLayoutId);

    // 取得したレイアウト内のView(TextView)に値を設定
    rv.setCharSequence(viewId, "setBackgroundColor", 0xFFFFFFFF);
    (省略)
}
```

A2. 次のannotationがないメソッドは、  
呼び出せない。  
(RemotableViewMethodがあればOK)

- @RemotableViewMethod
- @android.view.RemotableViewMethod

# ハマったところ(続き)

## ●RemoteViewクラスのset\*メソッド(続き)

frameworks > base > core > java > android > widget > RemoteView.java(432行目あたり)

呼び出すメソッドのannotationに、  
RemotableViewMethodクラスが含ま  
れるか？

```
Class klass = view.getClass();  
Method method = null;  
try {  
    method = klass.getMethod(this.methodName, getParameterType());  
}  
catch (NoSuchMethodException ex) {  
    throw new ActionException("view: " + klass.getName() + " doesn't have method: "  
        + this.methodName + "(" + param.getName() + ")");  
}  
  
if (!method.isAnnotationPresent(RemotableViewMethod.class)) {  
    throw new ActionException("view: " + klass.getName()  
        + " can't use method with RemoteViews: "  
        + this.methodName + "(" + param.getName() + ")");  
}
```

### Tips

Viewクラスでは、  
setVisibilityのみ

@RemotableViewMethodがついている。  
→呼び出しOK！

# ハマったところ(続き)

## ●RemoteViewクラスのset\*メソッド(続き)

frameworks > base > core > java > android > view > View.java(2684行目あたり)

```
/**
 * Set the enabled state of this view.
 *
 * @param visibility One of {@link #VISIBLE}, {@link #INVISIBLE}, or {@link #GONE}.
 * @attr ref android.R.styleable#View_visibility
 */
@RemotableViewMethod
public void setVisibility(int visibility) {
    setFlags(visibility, VISIBILITY_MASK);
    if (mBGDrawable != null) mBGDrawable.setVisible(visibility == VISIBLE, false);
}
```

frameworks > base > core > java > android > widget > TextView.java(2474行目あたり)

```
/**
 * Sets the string value of the TextView. TextView <em>does not</em> accept
 * HTML-like formatting, which you can do with text strings in XML resource files.
 * To style your strings, attach android.text.style.* objects to a
 * {@link android.text.SpannableString SpannableString}, or see the
 * <a href="{@docRoot}guide/topics/resources/available-resources.html#stringresources">
 * Available Resource Types</a> documentation for an example of setting
 * formatted text in the XML resource file.
 *
 * @attr ref android.R.styleable#TextView_text
 */
@android.view.RemotableViewMethod
public final void setText(CharSequence text) {
    setText(text, mBufferType);
}
```

終わり