非推奨メソッド等の掲載について

「スマートにプログラミング Android 入門編　第 2 版　SDK 4/2.3 対応」（2012 年 6 月 11 日発行　第 2 版第 1 刷）において、一部現在では非推奨となったレイアウト及びメソッドの利用が掲載されておりました。お詫びと共に補足・訂正させて頂きます。下記にその対応について示します。

株式会社リックテレコム
書籍出版部
（情報更新日：2012.8.14）

非推奨レイアウト・コマンド	掲載箇所	推奨レイアウト・コマンド[1]
AbsoluteLayout	Chapter 4　レイアウトとビュー （132 頁）	**FrameLayout** あるいは **RelativeLayout** といった、自分でカスタマイズしたレイアウトを代わりに使用するよう推奨されています。
onStart メソッド	Chapter 7　サービス 263 頁以降数カ所	**onStartCommand メソッド**が推奨されています。二つのメソッドの比較についてはこちらをご参照下さい。 （本文の訂正版 PDF は現在作成中です。詳しくは book-q@ric.co.jp までお問い合わせ下さい。）
managedQuery メソッド	Chapter 9　コンテントプロバイダ 380 頁以降数カ所	**ContentResolver の query メソッド**が推奨されています。 （本文の訂正版 PDF は現在作成中です。詳しくは book-q@ric.co.jp までお問い合わせ下さい。）
TabActivity	Chapter 10　ダイアログ、メニュー、タブ 「10-3　タブ」全体	**Action Bar** 及び **Fragment** という機能を使用するよう推奨されています。（追加となる参照 PDF はこち

[1] 2012 年 7 月時点の推奨です。

		らです。)

●Service クラスの onStart()／onStartCommand()メソッド

１．二つのメソッド

サービスは、アクティビティなどから startService()メソッドで起動されます。対象のサービスが起動していない場合、onCreate()メソッドが呼び出されます。その後に、**onStart()メソッド**、または **onStartCommand()**メソッドが呼び出されます。

前者の onStart()メソッドは、Android 1.6(API レベル 4)までで使用されていたメソッドで、API レベル 5 以降は非推奨となり、代わりに onStartCommand()メソッドを使用するように推奨されました。したがって API レベル 5 以降を対象とする場合は、onStart()メソッドではなく、onStartCommand()メソッドを使用するようにしましょう。

開発対象が API レベル 5 以降の場合、onStart()がオーバライドされていて、onStartCommand()がオーバライドされていない場合、onStart()メソッドが呼び出されます。onStart()と onStartCommand()の両方がオーバライドされている場合、onStartCommand()メソッドのみが呼び出されます。

API レベル 4 までを対象とする場合、onStart()メソッドと onStartCommand()メソッドの両方を実装します。

****

・onStart メソッド：

void onStart(Intent intent, int startId)

・onStartCommand()メソッド：

int onStartCommand(Intent intent, int flags, int startId)

****

２．onStartCommand()メソッドの戻り値

onStart()メソッドには、戻り値がありませんでしたが、onStartCommand()メソッドには int 型の戻り値があります。onStartCommand()メソッドをオーバライドするときに、以下の定数のうちどれかを return することでサービスが強制終了されたあとのサービスの動作を指定できます。

・START_NOT_STICKY

サービスが強制終了した場合、サービスは再起動しません。

不必要にサービスの再起動がされないため、サービスを安全に制御できます。

・START_STICKY

サービスが強制終了した場合、サービスが再起動されます。onStartCommand()メソッ

ドが再度呼び出されますが、引数の Intent には null が渡されます。サービスの停止をしない限りは、サービスの実行を終了したくない音楽再生などの処理に使用します。

・START_REDELIVER_INTENT

サービスが強制終了した場合、サービスが再起動されます。onStartCommand()メソッドが再度呼び出されて、引数の Intent には直前に動作していたサービスに引き渡された Intent が渡されます。直前の処理を再開したいファイルダウンロードのような処理に使用します。

以下は、START_STICKY を使用した場合の例です。

```
****
public class SampleService extends Service {

(省略)・・・・
@Override
public int onStartCommand(Intent intent, int flags, int startId) {
    (省略)・・・・

    return START_STICKY;
}

}
****
```

なお onStart()メソッドは、START_STICKY と同様の動作となります。onStart()メソッドを使用した場合、または、onStartCommand()メソッドで START_STICKY を return した場合、サービスが再起動したときに引数の Intent が null になります。

そのため、onStart()メソッド、もしくは onStartCommand()メソッドの処理で Intent を使用する処理がある場合、Intent が null か否かの判定処理を記述したほうが良いでしょう。

３．onStartCommand()メソッドの引数

onStartCommand()メソッドの第 1 引数には、Intent が渡されます。これは、onStart()メソッドの第 1 引数と同様です。

第 2 引数には、そのサービスが通常起動か、再起動されたのかなどを示す以下の整数が引き渡されます（表 1)。

表 1

第 2 引数の値	状　態
0	通常起動したとき
1	onStartCommand()メソッドの戻り値が START_REDELIVER_INTENT で、サービスが強制終了後の再起動したとき
2	onStartCommand()メソッドから正しく return がなかった場合に、サービスの再呼び出しをしたとき

Service には、これらの判定用に以下の定数が用意されており、onStartCommand()メソッドの処理内で使用することが可能です。

・**START_FLAG_REDELIVERY**

定数値は 1 で、START_REDELIVER_INTENT で、サービスが 強制終了後の再起動したのかの判定に使用

・**START_FLAG_RETRY**

定数値は 2 で、サービスの再呼び出しをしたときの判定に使用

第 3 引数は、onStart()メソッドの第 2 引数と同様で、サービスを開始するリクエストを一意に判別するための ID で、stopSelfResult()メソッドの引数に指定することで、該当のサービスを停止することができます。

Chapter 10　ダイアログ、メニュー、タブ

10-4　タブ（Android3.0 以降の場合）

本書の「10-3　タブ」のサンプルアプリケーションで使用した TabActivity は、2012 年 7 月現在では非推奨となっています。Android 3.0（API レベル 11）以降を対象とする場合、TabActivity の代わりにアクションバー（Action Bar）とフラグメント（Fragment）という機能を使用するよう推奨されています。

アクションバーは、画面上部に表示されるバーでオプションメニューやタブ機能を提供します。フラグメントは、画面レイアウトを分割する機能で、タブレットのような画面の大きい端末に有効です。１つのアクティビティが管理するレイアウトを複数のフラグメントに分割することにより、より複雑な画面構成および画面制御を実現できるようになります。

■1　　サンプルアプリケーションの作成

それでは、「3-3　タブ」で説明したサンプルアプリケーションを、アクションバーとフラグメントを使用して再度作成しましょう。（図 10.4.1）。

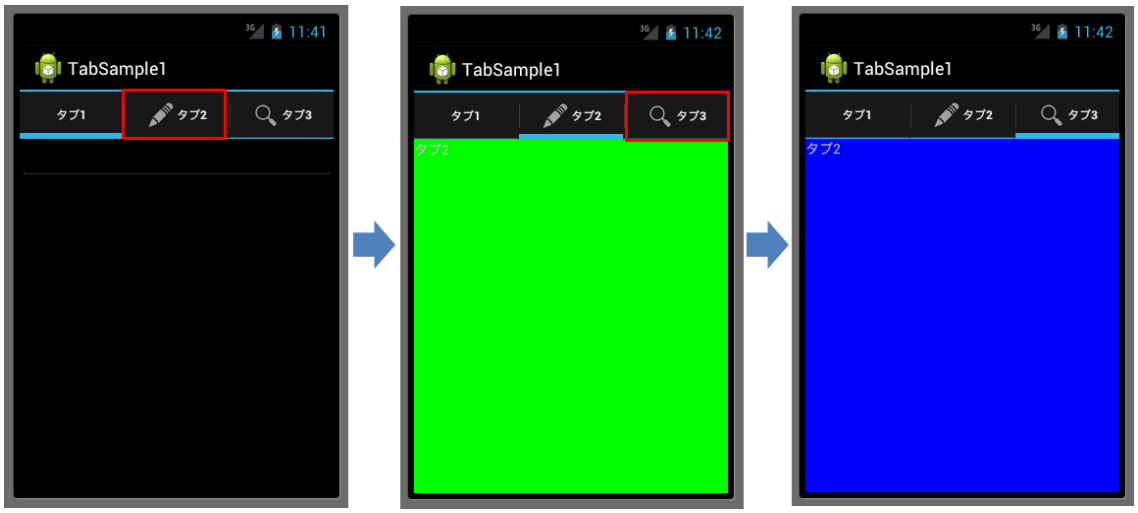

図 10.4.1　TabSample1

最初に「TabSample1」プロジェクトを次の設定で作成します（表 10.4.1）。「Build Target」は、Android 4.0（Android 3.0 以降）で設定します。

表 10.4.1　「TabSample1」プロジェクトの設定

入力項目名	入力内容
Project name:	TabSample1
Build Target:	Android 4.0

Application name:	TabSample1
Package name:	example.android.tabsample1
Create Activity:	TabSample1Activity

次にレイアウトとビューの設定ファイル「tabsample1.xml」を作成します（リスト 10.4.1）。レイアウトは、「LinearLayout」を選択します。

リスト 10.4.1　tabsample1.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:id="@+id/tabsample1" >


</LinearLayout>
```

レイアウトとビューの設定ファイルと同じ「layout」フォルダに、レイアウト設定ファイルを作成するときと同じ手順、3 つのフラグメント設定ファイルを作成します（リスト 10.4.2、10.4.3、10.4.4）。レイアウトは、「FrameLayout」を選択します。

これらの設定ファイルが、各タブが選択されたときに表示する画面になります。

リスト 10.4.2　fragment1.xml

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

    <EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:textSize="18sp"
        android:inputType="text"/>

</FrameLayout>
```

リスト 10.4.3　fragment2.xml

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

    <TextView
        android:layout_width="wrap_content"
```

```
        android:layout_height="wrap_content"
        android:text="@string/tab2" />

</FrameLayout>
```

リスト 10.4.4　fragment3.xml

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

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/tab3" />

</FrameLayout>
```

文字列定義ファイルに、タブ 2 とタブ 3 が選択されたときに表示する文字列を定義します（リスト 10.4.5)。

リスト 10.4.5　strings.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>

    <string name="hello">Hello World, TabSample1Activity!</string>
    <string name="app_name">TabSample1</string>

    <string name="tab2">タブ 2</string>
    <string name="tab3">タブ 3</string>

</resources>
```

アクティビティクラスは、次の通りに記述します（リスト 10.4.6)。

リスト 10.4.6　TabSample1Activity.java

```
package example.android.tabsample1;

import android.app.ActionBar;
import android.app.ActionBar.Tab;
import android.app.Activity;
import android.app.Fragment;
import android.app.FragmentTransaction;
import android.os.Bundle;

public class TabSample1Activity extends Activity {
    // onCreate メソッド(画面初期表示イベントハンドラ)
    @Override
```



```
public void onCreate(Bundle savedInstanceState) {
    // スーパークラスの onCreate メソッド呼び出し
    super.onCreate(savedInstanceState);
    // レイアウト設定ファイルの指定
    setContentView(R.layout.tabsample1);

    // ActionBar にタブ表示モード設定
    getActionBar().setNavigationMode(ActionBar.NAVIGATION_MODE_TABS);

    // タブ 1 の設定
    ActionBar.Tab tab1 = getActionBar().newTab();
    tab1.setText("タブ 1");
    tab1.setTabListener(new TabListener1(new Fragment1()));

    // タブ 2 の設定
    ActionBar.Tab tab2 = getActionBar().newTab();
    tab2.setText("タブ 2");
    tab2.setIcon(android.R.drawable.ic_menu_edit);
    tab2.setTabListener(new TabListener1(new Fragment2()));

    // タブ 3 の設定
    ActionBar.Tab tab3 = getActionBar().newTab();
    tab3.setText("タブ 3");
    tab3.setIcon(android.R.drawable.ic_menu_search);
    tab3.setTabListener(new TabListener1(new Fragment3()));

    // 各タブを ActionBar に設定
    getActionBar().addTab(tab1);
    getActionBar().addTab(tab2);
    getActionBar().addTab(tab3);
}

// タブ選択リスナー定義
private class TabListener1 implements ActionBar.TabListener {
    private Fragment fragment;

    // コンストラクタ定義
    public TabListener1(Fragment fragment) {
        // 対象フラグメント取得
        this.fragment = fragment;
    }

    // onTabReselected メソッド(タブ再選択時イベントハンドラ)
    public void onTabReselected(Tab tab, FragmentTransaction ft) {
    }

    // onTabSelected メソッド(タブ選択時イベントハンドラ)
    public void onTabSelected(Tab tab, FragmentTransaction ft) {
        // 対象フラグメント追加
        ft.add(R.id.tabsample1, fragment);
```

```
        }

        // onTabUnselected メソッド(タブ選択解除時イベントハンドラ)
        public void onTabUnselected(Tab tab, FragmentTransaction ft) {
            // 対象フラグメント除去
            ft.remove(fragment);
        }
    }
}
```

アクティビティクラスと同じパッケージに、各フラグメントの画面を制御する 3 つのフラグメントクラスを、次の通りに記述します（リスト 10.4.7、10.4.8、10.4.9）。

リスト 10.4.7　Fragment1.java

```
package example.android.tabsample1;

import android.app.Fragment;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;

public class Fragment1 extends Fragment {
    // onCreateView メソッド(フラグメント初期表示イベントハンドラ)
    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
            Bundle savedInstanceState) {
        // レイアウト設定ファイルからビューオブジェクト取得
        return inflater.inflate(R.layout.fragment1, container, false);
    }
}
```

リスト 10.4.8　Fragment2.java

```
package example.android.tabsample1;

import android.app.Fragment;
import android.graphics.Color;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;

public class Fragment2 extends Fragment {
        // onCreateView メソッド(フラグメント初期表示イベントハンドラ)
        @Override
        public View onCreateView(LayoutInflater inflater, ViewGroup container,
                Bundle savedInstanceState) {
```

```
                    // レイアウト設定ファイルからビューオブジェクト取得
                    View view = inflater.inflate(R.layout.fragment2, container, false);
                    // ビューに背景色を設定
                    view.setBackgroundColor(Color.GREEN);
                    return view;
            }
}
```

リスト 10.4.9　Fragment3.java

```
package example.android.tabsample1;

import android.app.Fragment;
import android.graphics.Color;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;

public class Fragment3 extends Fragment {
    // onCreateView メソッド(フラグメント初期表示イベントハンドラ)
    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
            Bundle savedInstanceState) {
        // レイアウト設定ファイルからビューオブジェクト取得
        View view = inflater.inflate(R.layout.fragment2, container, false);
        // ビューに背景色を設定
        view.setBackgroundColor(Color.BLUE);
        return view;
    }
}
```

これで設定とプログラミングについては一通り完了です。一度実行し、先ほどの画面が表示され、想定どおりに処理が行われるか確認しておきましょう。

次項では、このサンプルアプリケーションを参考にアクションバーとフラグメントを使用したタブについて解説します。

■2　アクションバーのタブ

アクションバーにタブを表示するには、最初にアクションバーをタブ表示モードに設定します（リスト 10.4.10）。

リスト 10.4.10

```
public class TabSample1Activity extends Activity {
    // onCreate メソッド(画面初期表示イベントハンドラ)
    @Override
    public void onCreate(Bundle savedInstanceState) {
        // スーパークラスの onCreate メソッド呼び出し
```



```
        super.onCreate(savedInstanceState);
        // レイアウト設定ファイルの指定
        setContentView(R.layout.tabsample1);

        // ActionBar にタブ表示モード設定
        getActionBar().setNavigationMode(ActionBar.NAVIGATION_MODE_TABS);
```

Activity クラスの getActionBar メソッドで ActionBar クラスのインスタンスを取得し、setNavigationMode メソッドで表示モードを設定します。タブを表示するモードを指定する場合は、ActionBar クラスの定数 NAVIGATION_MODE_TABS を引数にします。

各タブの設定のための記述をリスト 10.4.11 に示します。ActionBar.Tab クラスのインスタンスを取得して、タブに表示する文言や画像、タブが選択されたときに呼ばれるリスナーなどを設定していきます。

リスト 10.4.11

```
// タブ 1 の設定
ActionBar.Tab tab1 = getActionBar().newTab();
tab1.setText("タブ 1");
tab1.setTabListener(new TabListener1(new Fragment1()));

// タブ 2 の設定
ActionBar.Tab tab2 = getActionBar().newTab();
tab2.setText("タブ 2");
tab2.setIcon(android.R.drawable.ic_menu_edit);
tab2.setTabListener(new TabListener1(new Fragment2()));

// タブ 3 の設定
ActionBar.Tab tab3 = getActionBar().newTab();
tab3.setText("タブ 3");
tab3.setIcon(android.R.drawable.ic_menu_search);
tab3.setTabListener(new TabListener1(new Fragment3()));
```

ActionBar の newTab メソッドで ActionBar.Tab オブジェクトを生成します。また、ActionBar.Tab の setText メソッドでタブに表示する文言、setIcon メソッドでタブに表示するアイコン画像を設定します。また、setTabListener メソッドでタブが選択されたときのリスナーを関連付けます。

各タブの設定が完了したら、ActionBar.Tab オブジェクトを ActionBar に addTab メソッドで設定します（リスト 10.4.12）。

リスト 10.4.12

```
// 各タブを ActionBar に設定
getActionBar().addTab(tab1);
getActionBar().addTab(tab2);
getActionBar().addTab(tab3);
```


タブが選択されたときのイベントをハンドリングする ActionBar.TabListener を実装したリスナークラスの定義です（リスト 10.4.13）。この ActionBar.TabListener を使用した場合、タブが選択された場合に呼ばれる onTabSelected メソッド、タブが再選択された場合に呼ばれる onTabReselected メソッド、タブの選択が外れた場合に呼ばれる onTabUnselected メソッドを実装します。

リスト 10.4.13

```
// タブ選択リスナー定義
private class TabListener1 implements ActionBar.TabListener {
    private Fragment fragment;

    // コンストラクタ定義
    public TabListener1(Fragment fragment) {
        // 対象フラグメント取得
        this.fragment = fragment;
    }

    // onTabReselected メソッド(タブ再選択時イベントハンドラ)
    public void onTabReselected(Tab tab, FragmentTransaction ft) {
    }

    // onTabSelected メソッド(タブ選択時イベントハンドラ)
    public void onTabSelected(Tab tab, FragmentTransaction ft) {
        // 対象フラグメント追加
        ft.add(R.id.tabsample1, fragment);
    }

    // onTabUnselected メソッド(タブ選択解除時イベントハンドラ)
    public void onTabUnselected(Tab tab, FragmentTransaction ft) {
        // 対象フラグメント削除
        ft.remove(fragment);
    }
}
```

■3　フラグメント

各タブが選択されたときに表示する画面は、フラグメントを使用して定義します。フラグメントの画面は、通常のレイアウトとビューの設定ファイルと同様に設定することができます。

また、これらの画面を制御するフラグメントクラスを Fragment クラス、またはそのサブクラスを継承して作成します。フラグメントクラスは、アクティビティクラスと同じように画面を制御する機能を持ち、onCreate メソッド、onStart メソッド、onPause メソッ

ドなどのコールバックメソッドがあります。

リスト 10.4.14 が、タブ 1 が選択されたときに表示する画面を制御するフラグメントクラスの定義です。

リスト 10.4.14

```
public class Fragment1 extends Fragment {
    // onCreateView メソッド(フラグメント初期表示イベントハンドラ)
    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
            Bundle savedInstanceState) {
        // レイアウト設定ファイルからビューオブジェクト取得
        return inflater.inflate(R.layout.fragment1, container, false);
    }
}
```

onCreateView メソッドは、フラグメントの画面が最初に表示されたときに呼ばれます。サンプルでは、第 1 引数に引き渡された LayoutInflater を使用して、レイアウト設定ファイルをビューオブジェクトとして取得し、これを戻り値としています。この戻り値となったビューオブジェクトが、フラグメントを追加する画面で表示されます。

タブ 2 とタブ 3 のフラグメントクラスも同様に、レイアウト設定ファイルをビューオブジェクトとして取得し、戻り値としています。タブ 2 とタブ 3 のフラグメントクラスでは、View クラスの setBackgroundColor メソッドで背景色も設定しています。

フラグメントの定義ができたら、アクティビティにフラグメントを追加します。フラグメントをアクティビティに追加する方法は、2 通りあります。

1 つ目は、アクティビティが制御しているレイアウト設定ファイルに fragment タグを使用してフラグメントをビューのように設定する方法です。レイアウト設定ファイルにフラグメントを定義した例です（リスト 10.4.15）。

リスト 10.4.15

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:id="@+id/tabsample1" >

    <fragment
        android:name="example.android.tabsample1.Fragment1"
        android:id="@+id/fragment1"
        android:layout_width="wrap_content"
        android:layout_height="match_parent"
        android:layout_weight="1" />
```



```
</LinearLayout>
```

2 つ目は、アクティビティでフラグメントを追加する処理を行う方法です。タブを使用したときには、タブを選択するたびに動的に表示するフラグメントを切り換える必要がありますので、サンプルではこの方法で実装しています。

アクティビティでフラグメントを追加したり、置き換えたり、除去するには、FragmentTransaction クラスを使用します。FragmentTransaction の add メソッドで追加、remove メソッドで除去、replace メソッドで置き換えができます。

サンプルでは、アクティビティのタブ選択リスナークラスの onTabSelected メソッドと onTabUnselected メソッドの引数に FragmentTransaction のインスタンスが引き渡されますので、これらのメソッド内でフラグメントの追加および除去を行っています（リスト 10.4.16）。

リスト 10.4.16

```
// onTabSelected メソッド(タブ選択時イベントハンドラ)
public void onTabSelected(Tab tab, FragmentTransaction ft) {
    // 対象フラグメント追加
    ft.add(R.id.tabsample1, fragment);
}

// onTabUnselected メソッド(タブ選択解除時イベントハンドラ)
public void onTabUnselected(Tab tab, FragmentTransaction ft) {
    // 対象フラグメント除去
    ft.remove(fragment);
}
```

onTabSelected メソッドや onTabUnselected メソッドのように FragmentTransaction が引数として引き渡されていない場合、FragmentManager クラスのインスタンスを Activity の getFragmentManager メソッドで取得し、FragmentTransaction のインスタンスを FragmentManager の beginTransaction メソッドで取得します。また、フラグメントの追加や除去の後に、FragmentTransaction の commit メソッドで変更を確定します（リスト 10.4.17）。

リスト 10.4.17

```
// FragmentManager インスタンス取得
FragmentManager fm= getFragmentManager();

// FragmentTransaction インスタンス取得
FragmentTransaction ft = fm.beginTransaction();

// 対象フラグメント追加
```



```
ft.add(R.id. tabsample1, fragment);

// コミット
ft.commit();
```

onTabSelected メソッドや onTabUnselected メソッドでは、FragmentTransaction の commit メソッドの実行は必要ありません。

