Android 6.0 Marshmallow アプリ開発入門 Vol.3

IT研究所

第 10 章　インテントの基礎

10.1 インテントとは・・ 10-1
10.2 インテントと AndroidManifest.xml・・・ 10-2
10.3 明示的インテント・・ 10-4
10.4 明示的インテントの利用方法 1・・・ 10-5
10.5 明示的インテントの利用方法 2・・・ 10-9
10.6 暗黙的インテント・・ 10-10
10.7 暗黙的インテントの利用方法 1・・・ 10-11
10.8 暗黙的インテントの利用方法 2・・・ 10-14

第 11 章　インテントの活用

11.1 インテントの活用方法 1・・・ 11-1
11.2 インテントの活用方法 2・・・ 11-4
11.3 起動先からインテントを受け取る方法・・・ 11-7
11.4 インテントの活用方法 3・・・ 11-8

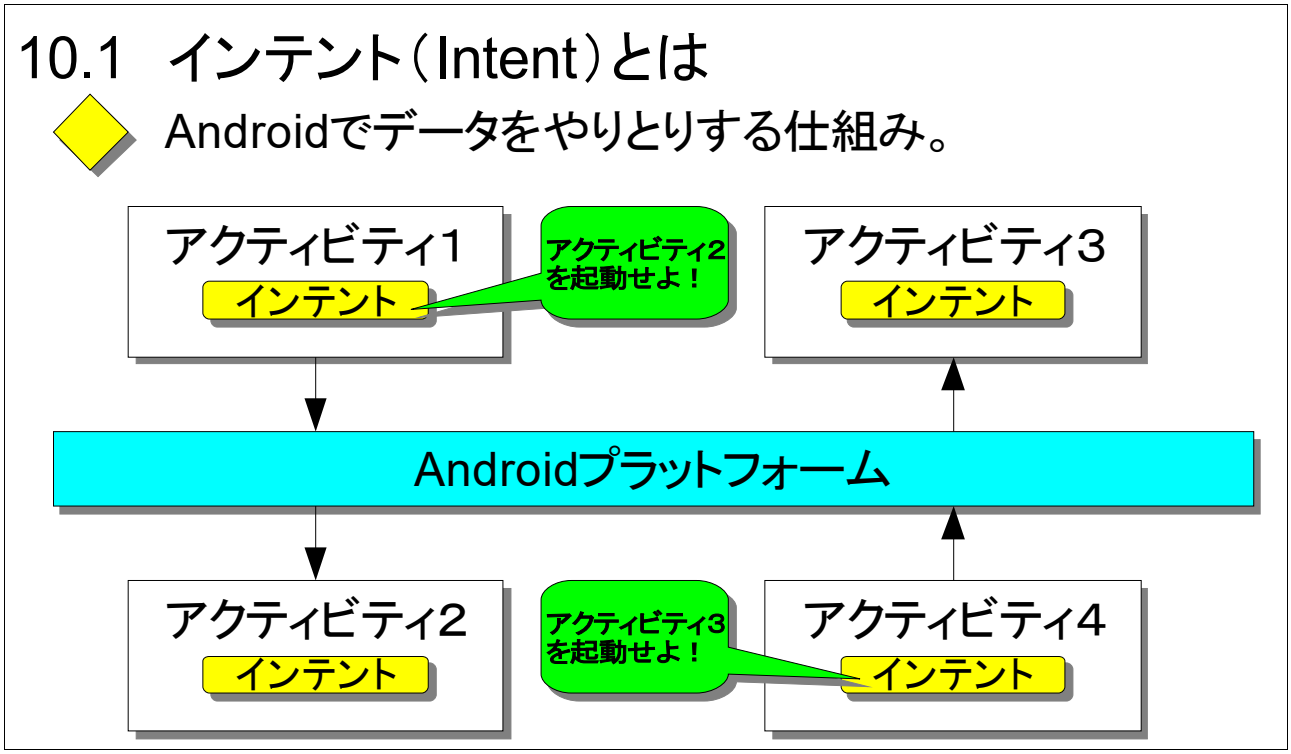
第 12 章　データ管理（前編）

12.1 データ管理・・ 12-1
12.2 ファイル・アクセス・・・ 12-2
12.3 ファイルアクセスの利用方法 1・・・ 12-3
12.4 ファイルアクセスの利用方法 2・・・ 12-5
12.5 プリファレンス・・ 12-7
12.6 プリファレンスの利用方法 1・・・ 12-9
12.7 プリファレンスの利用方法 2・・・ 12-11

第 13 章　データ管理（後編）

13.1 データベースについて・・ 13-1
13.2 SQLite の操作方法・・・ 13-2
13.3 SQLite の操作手順・・・ 13-3
13.4 データベースの利用方法・・・ 13-8

第 10 章
インテントの基礎



Android は、マルチタスクで複数のアプリケーションを同時にいくつも起動し動かすことができます。この特性を活かすために考え出されたのがインテントです。

インテントとは、「意図」「目的」という意味があり、Android においては、アプリケーションの中の 1 つ 1 つの機能であり、アプリケーション同士や、アプリケーションと部品、アプリケーションとシステムを橋渡しする仕組みのことです。通常の Android アプリケーションは、複数の画面から構成され画面間を行き来します。このように複数の画面間で画面遷移を行う場合にインテントという仕組みを利用します。また、インテントを利用することで同一アプリケーション内での画面遷移だけでなく、他のアプリケーションとの連携も可能になります。

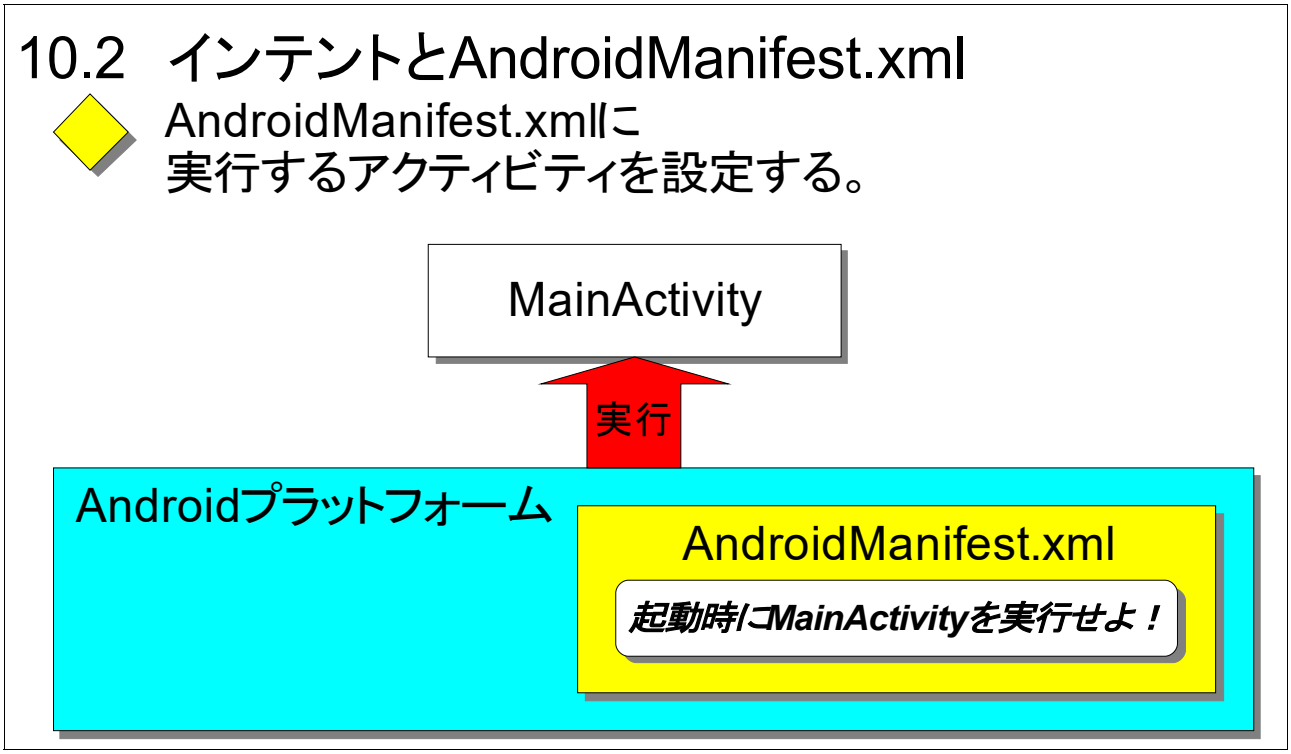
インテントによって呼び出し可能なコンポーネントは「アクティビティ」、「サービス」、「ブロードキャストレシーバ」です。アクティビティからアクティビティを呼び出したり、バックグラウンドで起動しているサービスをアクティビティから呼び出したりすることが可能です。

インテントは、起動したいアクティビティやサービスの基本情報と引き渡したい付加情報を保持します。呼び出し元のアプリケーションは「何をどのように起動するのか」という情報を持ったインテントを Android プラットフォームに送り、インテントに保持されている情報を元に Android プラットフォームが、該当するアクティビティやサービスを起動します。

◆インテントによる起動の種類

インテントには、「明示的インテント」と「暗黙的インテント」の２種類がある。

種類	説明
明示的インテント	どの機能を呼び出すかを予め指定して、直接それを起動する。
暗黙的インテント	何をしたいかは明示せずに 漠然とそのデータに対して実行できる機能の一覧を要求する。



インテントは、プログラムの実行に関する情報を管理するメッセージです。Android は実行するプログラムが発生すると、その情報をインテントという形で発行します。それぞれのプログラムには、自身がどのようなインテントを受け取ることができるかの情報が記述されており、発行されたインテントによって特定のプログラムを実行できるような仕組みになっています。

Android アプリケーションを起動時に MainActivity が必ず実行されるのは、「AndroidManifest.xml」ファイルに記述されているからです。

◆「AndroidManifest.xml」の内容

```
<activity
   android:name=".MainActivity"
   android:label="@string/app_name" >
   <intent-filter>
      <action android:name="android.intent.action.MAIN" />
      <category android:name="android.intent.category.LAUNCHER" />
   </intent-filter>
</activity>

※一部抜粋
```

●intent-filter タグ内の項目の説明

タグ	説明
intent-filter	アクティビティが受け取ることができるインテントに関する情報を記述する。
action	実行する内容を記述する。
category	インテントを分類する。
data	URLを指定する。
type	dataを補完する。MIMEタイプなどを設定する。

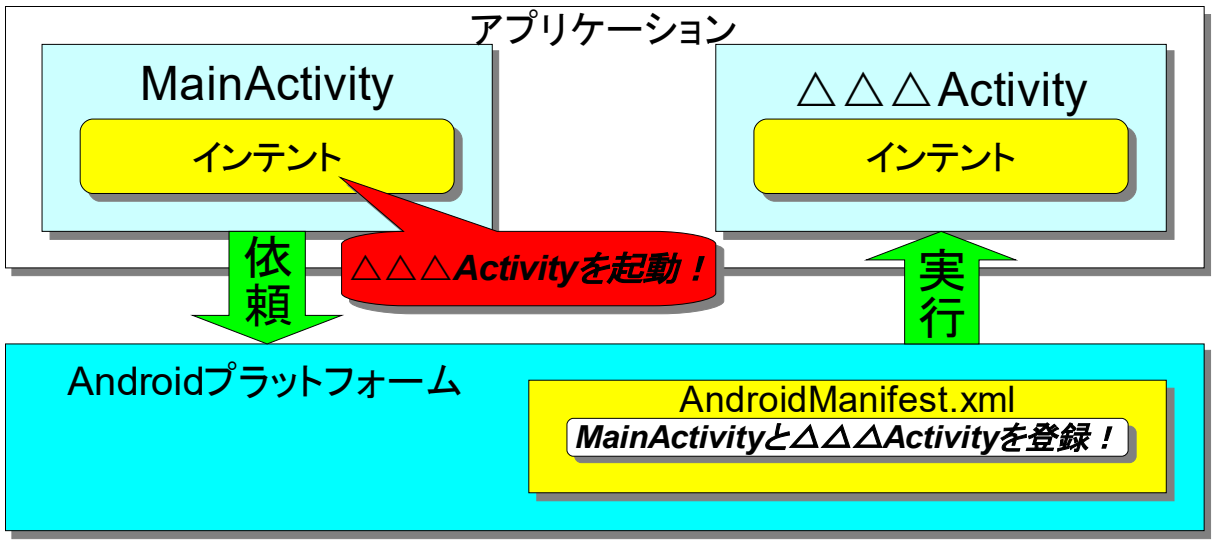
よって、「AndroidManifest.xml」に記述されているデフォルトの intent-filter の意味するところはランチャー（起動）向けのアプリケーション設定です。

action タグの android.intent.action.MAIN はその Activity がアプリケーションであり、引数もなく単独で最初に実行される Activity だという設定です。

category の android.intent.category.LAUNCHER はシステムの LAUNCHER（起動）に登録できることを設定しています。

10.3 明示的インテント

アクティビティを切り替える。



明示的インテントの主な利用方法は、アクティビティの切り替えです。このような目的で使用する場合の明示的インテントでは、アプリケーション起動時に呼び出されるアクティビティと、それから呼び出されるアクティビティをマニフェストファイル「AndroidManifest.xml」に登録します。

◆明示的インテントでの「AndroidManifest.xml」の設定方法

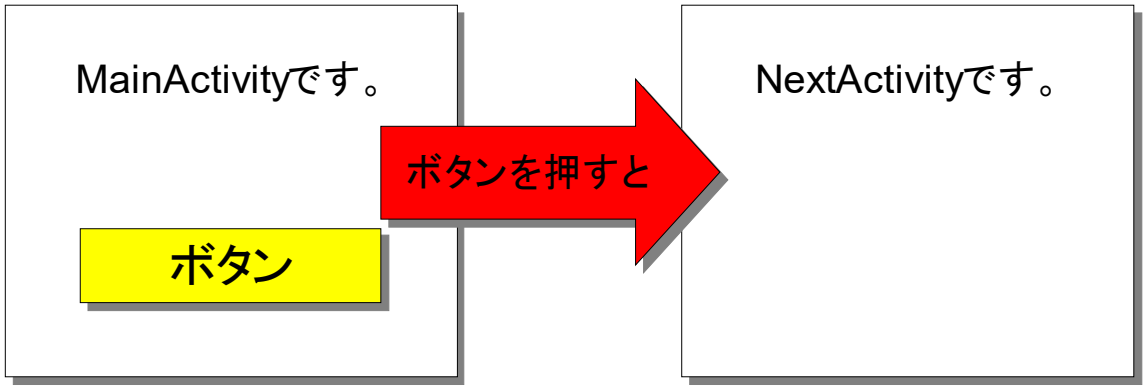
起動時に呼び出されるアクティビティを「MainActivity」、それから呼び出されるアクティビティを「△△△Activity」とした場合の例として記述します。

```
<activity
    android:name=".MainActivity"
    android:label="@string/app_name" >
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</activity>

<activity
    android:name=".△△△Activity"
    android:label="ラベル" >
</activity>

※一部抜粋
```

10.4　明示的インテントの利用方法1

アプリケーション名：IntentApp01

動作概要



◆作成手順

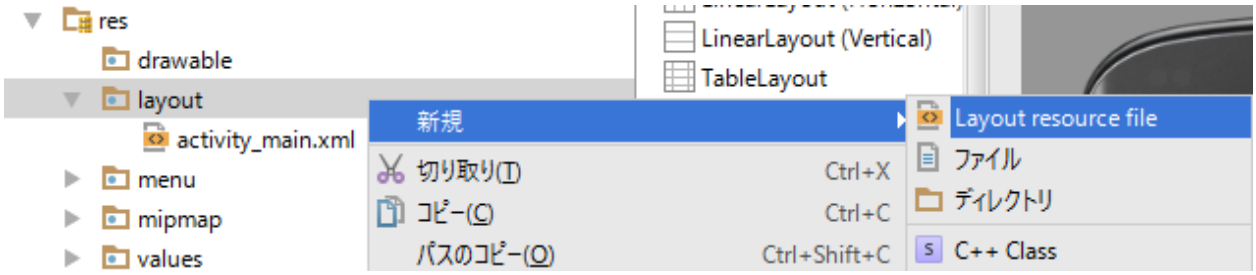
①新規アプリケーション「IntentApp01」を作成する。

②「activity_main.xml」で「TextView（Plain TextView）」と「Button」を配置する。

③「res」フォルダ内の「layout」フォルダに「activity_next.xml」ファイルを作成し「TextView」を配置する。

●「res」フォルダ内の「lauout」フォルダ上で右クリックしメニューを表示し

[新規]→[Layout resource file]を選択する。



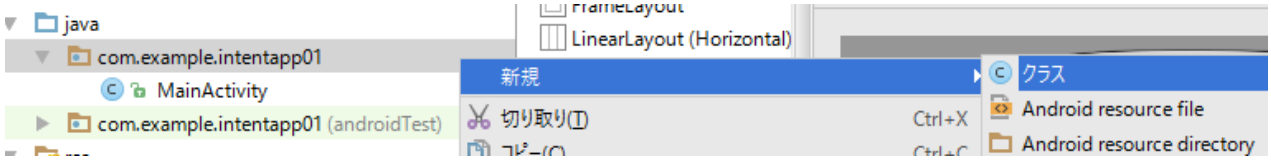
●File name に「activity_next」と入力し、Root element に「RelativeLayout」と入力し、[OK]ボタンを押す。

●「activity_next.xml」で「TextView（Plain TextView）」を配置し、「Next Activity です。」と文字設定を行う。

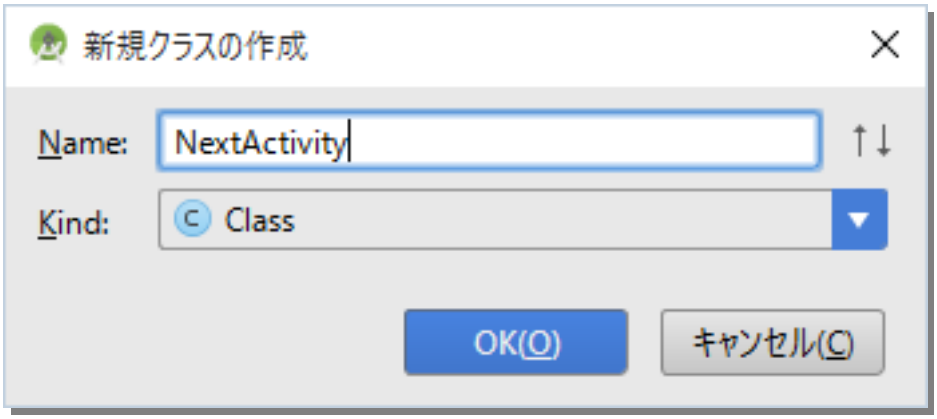
④「java」フォルダ内の「com.example.intentapp01」フォルダに「NextActivity.java」ファイルを作成する。

●「java」フォルダ内の「com.example.intentapp01」フォルダ上で右クリックしメニューを表示し

[新規]→[クラス]を選択する。



●Name に「NextActivity」と入力し[OK]ボタンを押す。



⑤「AndroidManifest.xml」を変更する。

ファイル名：AndroidManifest.xml

```
    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:name=".MainActivity"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>

        <activity
            android:name=".NextActivity"
            android:label="別のアクティビティです。" >
        </activity>

    </application>
```

※一部抜粋

⑥「NextActivity.java」のプログラムを行う。

ファイル名：NextActivity.java

```
package com.example.intentapp01;

import android.app.Activity;
import android.os.Bundle;

public class NextActivity extends Activity {

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_next);
    }
}
```

⑦「MainActivity.java」でボタンをクリックしたときに画面遷移するプログラムを設定する。

●Intent クラス

Intent クラスは、android.content パッケージで定義されている。

●Intent クラスのコンストラクタ

```
【書式】
public Intent (Context context, Class<?> cls)

【引数】
第1引数には、Intentを作成するContextのオブジェクトを指定する。
（ActivityはContextのサブクラスであるため、Activityを設定することができる。）
第2引数には、起動するクラスを指定する。

【機能】
Intentを生成する。
```

●プログラム例

ファイル名：MainActivity.java

```
package com.example.intentapp01;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;

public class MainActivity extends Activity implements OnClickListener {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        Button button = (Button)findViewById(R.id.button);
        button.setOnClickListener(this);
    }

    @Override
    public void onClick(View view) {
        Intent intent = new Intent(this, com.example.intentapp01.NextActivity.class);
        startActivity(intent);
    }
}
```

※一部抜粋

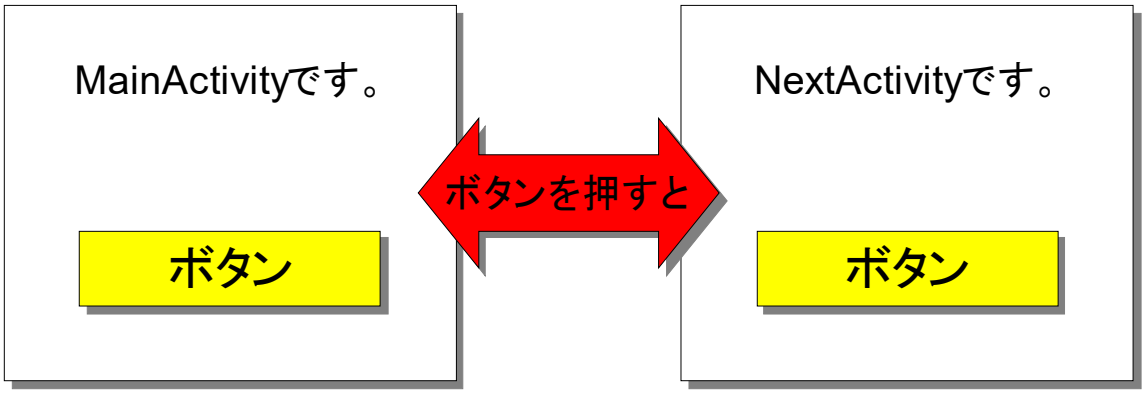
★startActivity メソッド（Activity クラスに定義）

【書式】
public void startActivity(Intent intent)

【機能】
指定のアクティビティを起動するためのインテントを実行する。

10.5 明示的インテントの利用方法2

- アプリケーション名：IntentApp01
- 動作概要



◆作成手順

①「IntentApp01」を変更する。

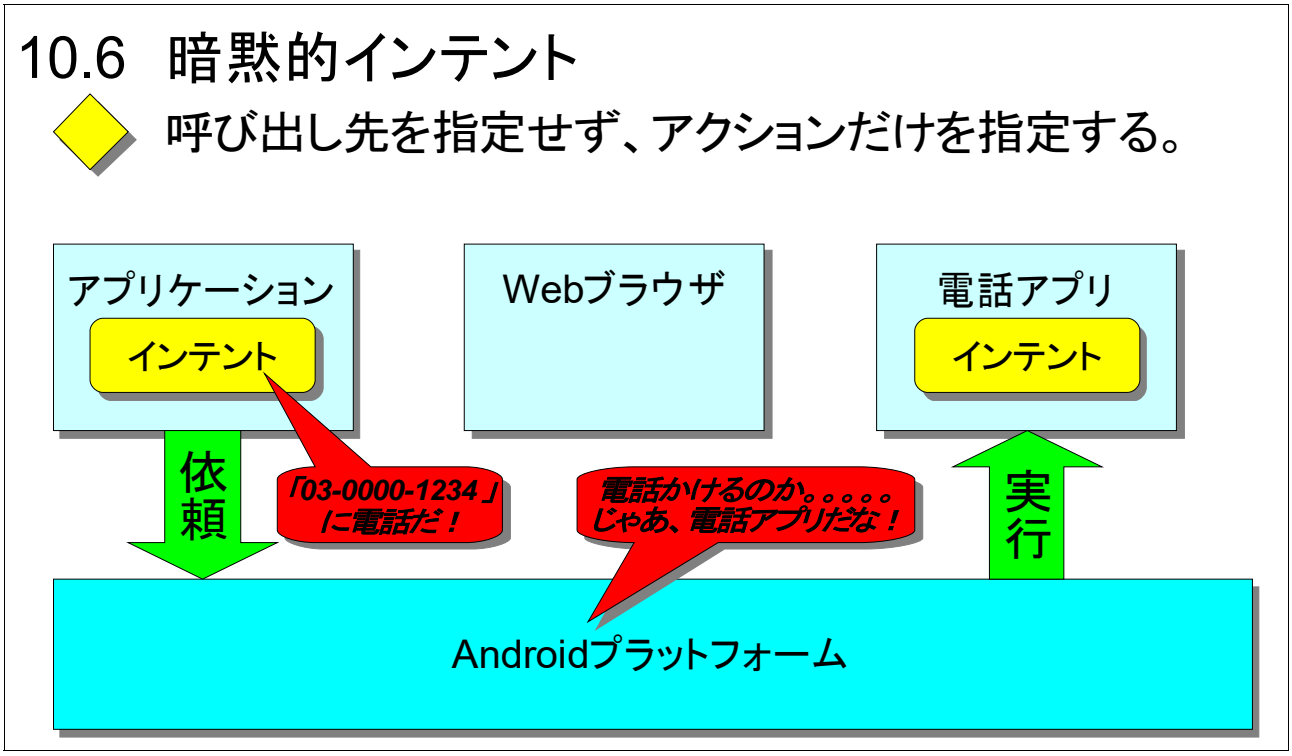
②「activity_next.xml」で「Button」を追加配置する。

③「NextActivity.java」でボタンをクリックしたときに画面遷移するプログラムを設定する。

●プログラム例

ファイル名：NextActivity.java

```
package com.example.intentapp01;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;

public class NextActivity extends Activity implements OnClickListener {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_next);

        Button button = (Button)findViewById(R.id.button2);
        button.setOnClickListener(this);
    }

    @Override
    public void onClick(View view) {
        Intent intent = new Intent(this, com.example.intentapp01.MainActivity.class);
        startActivity(intent);
    }
}
```



暗黙的インテントの利用方法は、呼び出し先のコンポーネントを指定せず、アクションだけを指定する方法です。起動するコンポーネントは各アプリケーションの設定情報から Android プラットフォームが選定します。

◆暗黙的インテントの利用方法

●暗黙的インテントを使用すると Android に含まれているアプリケーションと連携して、以下の事ができる。

・電話アプリケーション・・・指定した電話番号のダイヤルウィンドウを開く。

・Web ブラウザ・・・指定した URL の Web サイトをブラウザで表示する。

・アドレス帳・・・アドレス帳のリストを表示する。

●暗黙的インテントは、システム全体に向けてブロードキャスト（同報通信）することができる。

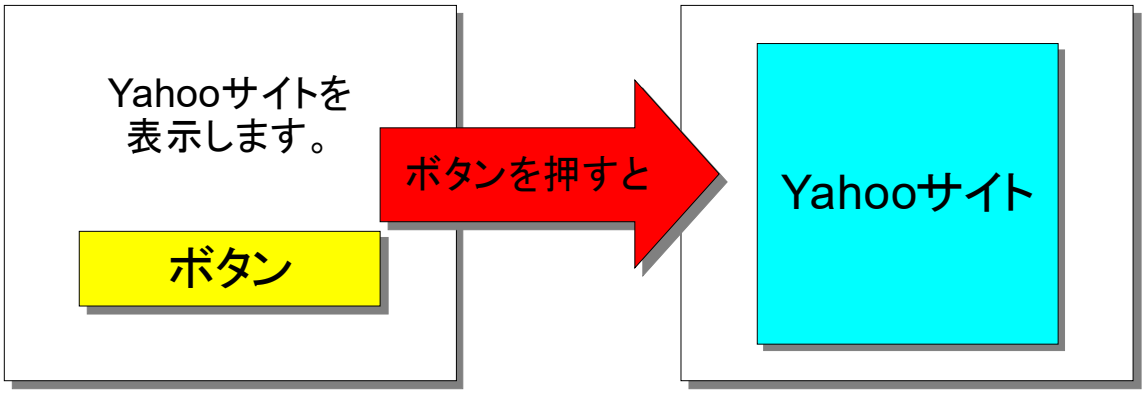
●ブロードキャストされるインテントは、「システムの起動完了」や「電池残量の変化が発生した」等の状況の変化を伝える手段として使用している。

●ブロードキャストされるインテントは、ブロードキャストレシーバだけが受け取ることができる。

10.7 暗黙的インテントの利用方法1

アプリケーション名：IntentApp02

動作概要



◆作成手順

①新規アプリケーション「IntentApp02」を作成する。

②「activity_main.xml」で「TextView（Plain TextView）」と「Button」を配置する。

③「MainActivity.java」でボタンをクリックしたときに Web サイトを表示するプログラムを設定する。

●Intent クラス

Intent クラスは、android.content パッケージで定義されている。

●Intent クラスのコンストラクタ

【書式】
public Intent (String action, Uri uri)

【引数】
第1引数には、何かを表示させるためのアクション（アクティビティ・アクション）を指定する。
第2引数には、URI（URLを表すオブジェクト）を指定する。

【機能】
Intentを生成する。

●アクション

アクションには「アクティビティ・アクション」と「ブロードキャスト・アクション」がある。

アクティビティ・アクション	他のアクティビティを起動する際に、何をしたいのかを示すためのアクション
ブロードキャスト・アクション	何らかのイベントが発生したことを 他のオブジェクトに知らせる際に指定するアクション

（1）主なアクティビティ・アクション

アクション	説明
ACTION_MAIN	メインエントリーポイントとしてアクションを起動する。 データを戻すことができない。
ACTION_VIEW	データをユーザに表示するためにアクションを起動する。
ACTION_ATTACH_DATA	別の場所にデータが添付されていることを示す。
ACTION_EDIT	データを編集するためにアクションを起動する。
ACTION_PICK	データの中からアイテムを選択するためにアクションを起動する。 選択されたアイテムを受け取る。
ACTION_CHOOSER	起動するActivityを選択するチューザーをユーザに示し、選択してもらう。
ACTION_GET_CONTENT	指定した種類のデータをユーザに選択してもらう。
ACTION_DIAL	データで指定した番号を電話番号として設定する。 データを指定しなければ、ユーザがその場で設定する。 電話をかける操作はユーザに任せる。
ACTION_CALL	データで指定した人に電話をかける。 データを指定しなければ、ユーザがその場で指定した番号に電話をかける。
ACTION_SEND	別の人にデータを送信する。データの宛先は指定しない。 このIntentを受け取ったアクションが、ユーザに宛先を問う。
ACTION_SENDTO	データによって指定された人にメッセージを送信する。
ACTION_ANSWER	かかってきた電話に対処するアクションを起動する。
ACTION_INSERT	データに含まれるURIが示すディレクトリに空のデータを挿入する。
ACTION_DELETE	データに含まれるURIが示すデータをコンテナから削除する。
ACTION_RUN	データを起動する。その意味はデータの種類によって異なる。
ACTION_SYNC	データの同期を実行する。
ACTION_PICK_ACTIVITY	Intentが与えられるActivityを選択し、そのクラスを返す。
ACTION_SEARCH	検索を実行する。
ACTION_WEB_SEARCH	WEB検索を実行する。

（2）主なブロードキャスト・アクション

アクション	説明
ACTION_BATTERY_CHANGED	充電中かどうかや、電池残量が変わった時に送られる。
ACTION_BATTERY_LOW	電池残量が少なくなった場合に送られる。
ACTION_BOOT_COMPLETED	システムの起動が完了したことを示す。
ACTION_PACKAGE_ADDED	新しいアプリケーションパッケージが追加されたことを示す。
ACTION_PACKAGE_CHANGED	存在するアプリケーションのパッケージが変更されたことを示す。
ACTION_PACKAGE_REMOVED	アプリケーションパッケージを削除したことを示す。
ACTION_TIMEZONE_CHANGED	タイムゾーンが変更されたことを示す。
ACTION_TIME_CHANGED	時刻が変更されたことを示す。
ACTION_TIME_TICK	現在時刻が変わった時に毎分送られるアクション。
ACTION_HEADSET_PLUG	ハンドセットのプラグが抜き差しされたことを示す。

●プログラム例

ファイル名：MainActivity.java

```
package com.example.intentapp02;

import android.app.Activity;
import android.content.Intent;
import android.net.Uri;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;

public class MainActivity extends Activity implements OnClickListener {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        Button button = (Button)findViewById(R.id.button);
        button.setOnClickListener(this);
    }

    @Override
    public void onClick(View view) {
        String urlStr = "http://www.yahoo.co.jp";

        Intent intent = new Intent(Intent.ACTION_VIEW, Uri.parse(urlStr));
        startActivity(intent);
    }
}
```

※一部抜粋

★parse メソッド（Uri クラスに定義）

【書式】
public static Uri parse(String url)

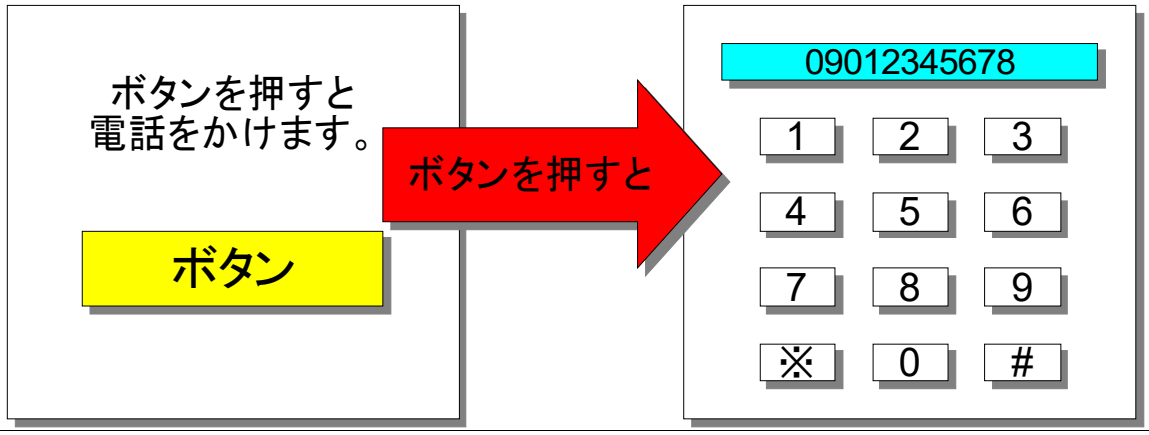
【戻り値】
android.net.Uriクラスのオブジェクトを返す。

【機能】
urlで指定したUriオブジェクトを生成する。

10.8 暗黙的インテントの利用方法2

アプリケーション名：IntentApp03

動作概要



◆作成手順

①新規アプリケーション「IntentApp03」を作成する。

②「activity_main.xml」で「TextView」と「Button」を配置する。

③「MainActivity.java」でボタンをクリックしたときに電話アプリを表示するプログラムを設定する。

●プログラム例

ファイル名：MainActivity.java

```
package com.example.intentapp03;

public class MainActivity extends Activity implements OnClickListener {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        Button button = (Button)findViewById(R.id.button);
        button.setOnClickListener(this);
    }

    @Override
    public void onClick(View view) {
        String telNo = "tel:09012345678";

        Intent intent = new Intent(Intent.ACTION_DIAL, Uri.parse(telNo));
        startActivity(intent);
    }
}
```

※一部抜粋

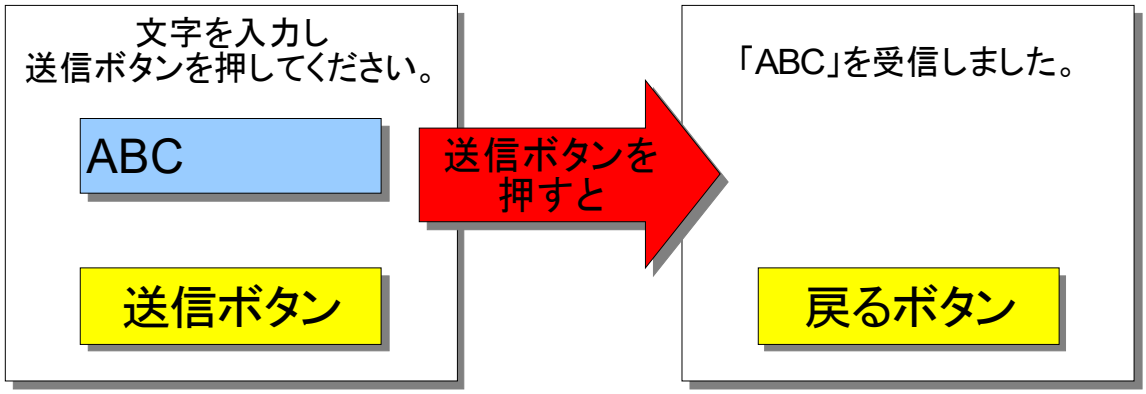
メモ

第 11 章
インテントの活用

11.1 インテントの活用方法1

アプリケーション名：IntentApp04

動作概要



◆作成手順

①新規アプリケーション「IntentApp04」を作成する。

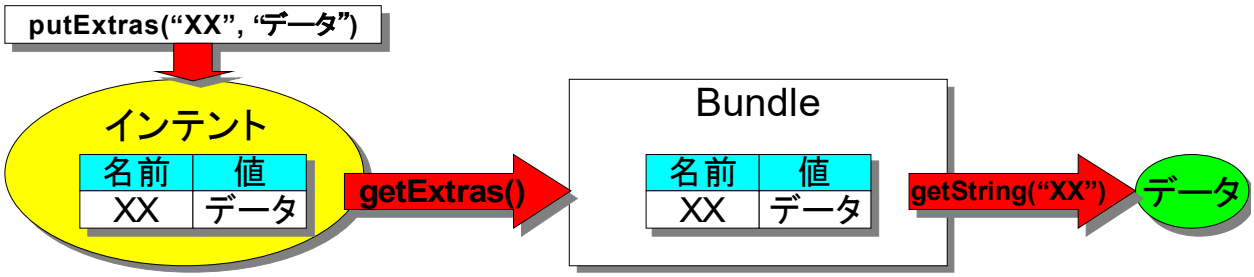
②「activity_main.xml」で「TextView（Plain TextView）」と「EditText（Plain Text）」と「Button」を配置する。

③「activity_next.xml」で「TextView」と「Button」を配置する。

④「AndroidManifest.xml」で「NextActivity」を Activity 設定する。

⑤「MainActivity.java」と「NextActivity.java」のプログラムを設定する。

●データ送受信の流れ



●データ送受信に関するメソッド

メソッド	説明
putExtra （Intentクラスに定義）	【書式】 public Intent putExtra(String name, String value) 【機能】 画面遷移先で必要な付加情報をインテントに設定する。
getExtras （Intentクラスに定義）	【書式】 public Bundle getExtras() 【機能】 インテントが保持している付加情報を取得する。
getString （Bundleクラスに定義）	【書式】 public String getString(String name) 【機能】 Bundleに格納されている付加情報を名前を指定し値を取得する。

● 「MainActivity.java」のプログラム例

ファイル名：MainActivity.java

```
public class MainActivity extends Activity implements OnClickListener {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        Button button = (Button)findViewById(R.id.button);
        button.setOnClickListener(this);
    }

    @Override
    public void onClick(View view) {
        EditText edit = (EditText)findViewById(R.id.editText);
        String str = edit.getText().toString();

        Intent intent = new Intent(this, NextActivity.class);
        intent.putExtra("SEND_DATA", str);

        startActivity(intent);
    }
}
```

※一部抜粋

● 「NextActivity.java」のプログラム例

ファイル名：NextActivity.java

```
public class NextActivity extends Activity implements OnClickListener {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_next);

        Button button = (Button)findViewById(R.id.button2);
        button.setOnClickListener(this);

        Intent intent = getIntent();
        Bundle bundle = intent.getExtras();
        CharSequence receiveStr = bundle.getCharSequence("SEND_DATA");

        TextView text = (TextView)findViewById(R.id.textView2);
        text.setText(receiveStr + "を受信しました。");
    }

    @Override
    public void onClick(View view) {
        Intent intent = new Intent(this, MainActivity.class);
        startActivity(intent);

        finish();
    }
}
```

※一部抜粋

★getIntent メソッド（Activity クラスに定義）

【書式】
public Intent getIntent()

【機能】
Activity起動時に送られたインテントを取得する。

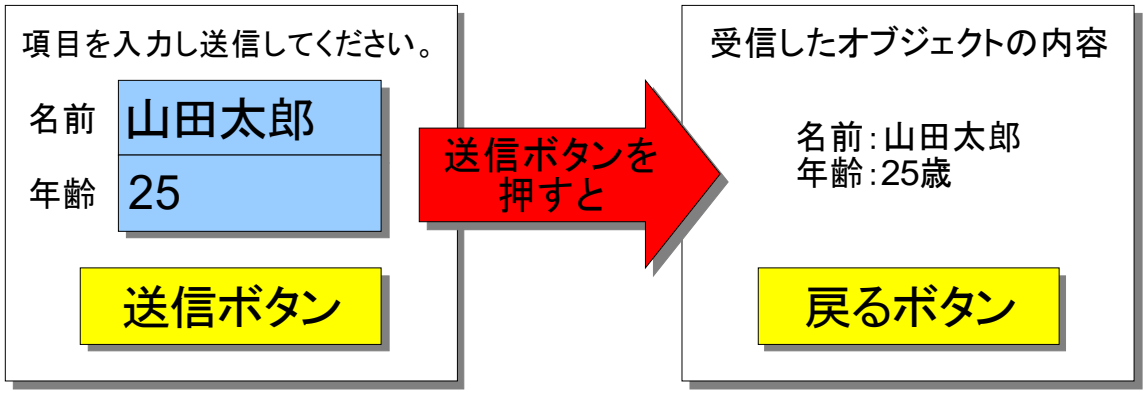
★finish メソッド（Activity クラスに定義）

【書式】
public void finish()

【機能】
Activityを終了する。

11.2 インテントの活用方法2

- アプリケーション名：IntentApp05
- 動作概要



◆作成手順

①新規アプリケーション「IntentApp05」を作成する。

②「activity_main.xml」で「TextView」と「EditText」と「Button」を配置する。

③「activity_next.xml」で「TextView」と「Button」を配置する。

④「AndroidManifest.xml」で「NextActivity」を Activity 設定する。

⑤送信するオブジェクト用のクラス（PersonalData.java）を定義する。

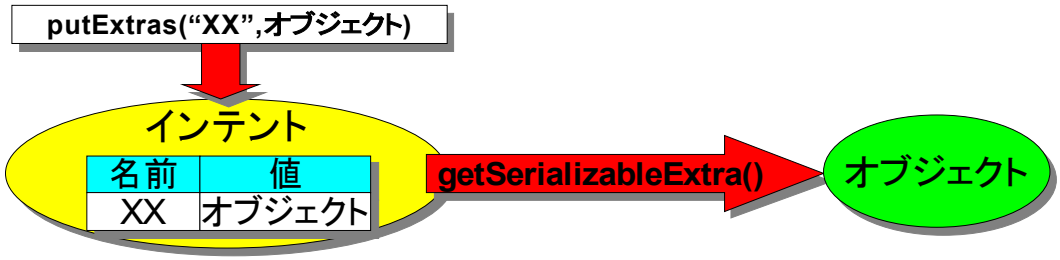
ファイル名：PersonalData.java

```
package com.example.intentapp05;

public class PersonalData implements java.io.Serializable {
    private String name;
    private int age;

    public PersonalData() {}

    public PersonalData(String name, int age) {
        this.name = name;
        this.age = age;
    }

    public String getName() {
        return name;
    }

    public int getAge() {
        return age;
    }
}
```

※セッタを省略して記述する。

⑥「MainActivity.java」と「NextActivity.java」のプログラムを設定する。

●オブジェクト送受信の流れ



※オブジェクトは、必ずSerializableインターフェースを実装したものでなければいけない。

●オブジェクト送受信に関するメソッド

メソッド	説明
putExtra （Intentクラスに定義）	【書式】 public Intent putExtra(String name, Serializable value) 【機能】 画面遷移先で必要な付加情報をインテントに設定する。
getSerializableExtra （Intentクラスに定義）	【書式】 public Serializable getSerializableExtra(String name) 【機能】 インテントが保持している付加情報を名前を指定し取得する。

●「MainActivity.java」のプログラム例

ファイル名：MainActivity.java

```
public class MainActivity extends Activity implements OnClickListener {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        Button button = (Button)findViewById(R.id.button);
        button.setOnClickListener(this);
    }

    @Override
    public void onClick(View view) {
        EditText edit1 = (EditText)findViewById(R.id.editText);
        EditText edit2 = (EditText)findViewById(R.id.editText2);

        String name = edit1.getText().toString();
        int age = Integer.parseInt(edit2.getText().toString());

        PersonalData data = new PersonalData(name, age);

        Intent intent = new Intent(this, NextActivity.class);
        intent.putExtra("PERSONAL_DATA", data);

        startActivity(intent);
    }
}
```

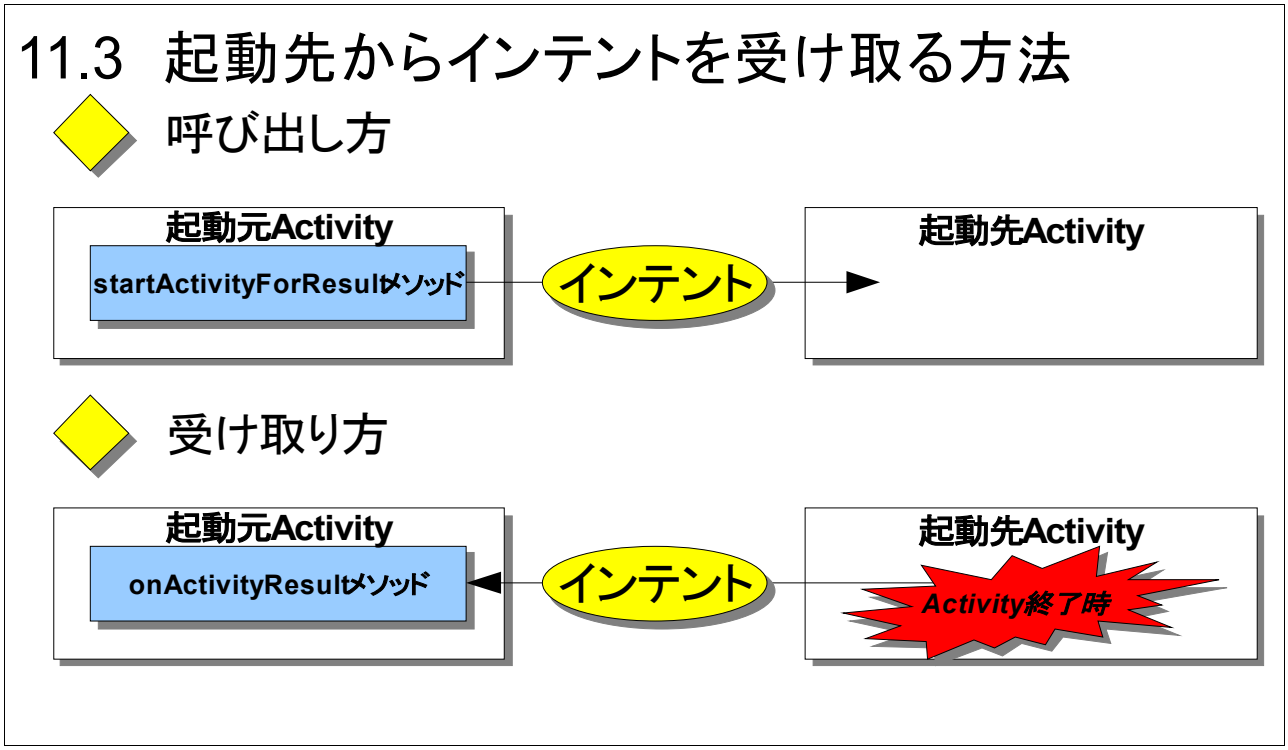
※一部抜粋

● 「NextActivity.java」のプログラム例

ファイル名：NextActivity.java

```
public class NextActivity extends Activity implements OnClickListener {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_next);

        Button button = (Button)findViewById(R.id.button2);
        button.setOnClickListener(this);

        Intent intent = getIntent();
        PersonalData data = (PersonalData)intent.getSerializableExtra("PERSONAL_DATA");

        String str = "";
        str += "名前：" + data.getName() + "\n";
        str += "年齢：" + data.getAge() + "\n";

        TextView text = (TextView)findViewById(R.id.textView5);
        text.setText(str);
    }

    @Override
    public void onClick(View view) {
        Intent intent = new Intent(this, MainActivity.class);
        startActivity(intent);

        finish();
    }
}
```

※一部抜粋



Activity を起動する場合、起動元の Activity から startActivity メソッドを使用し、生成したインテントを渡すことで、起動先の Activity を起動させることができます。しかし、起動先の Activity から戻り値となるインテントを受け取らせる処理を必要とする場合は、startActiovityForResult メソッドを使用します。

startActivityForResult により起動させた Activity が、finish メソッドにより終了した時に、起動元に Intent オブジェクトを送ります。このインテントは、起動元の onActivityResult メソッドに渡されるので、このメソッドで返ってきたデータの処理を行います。

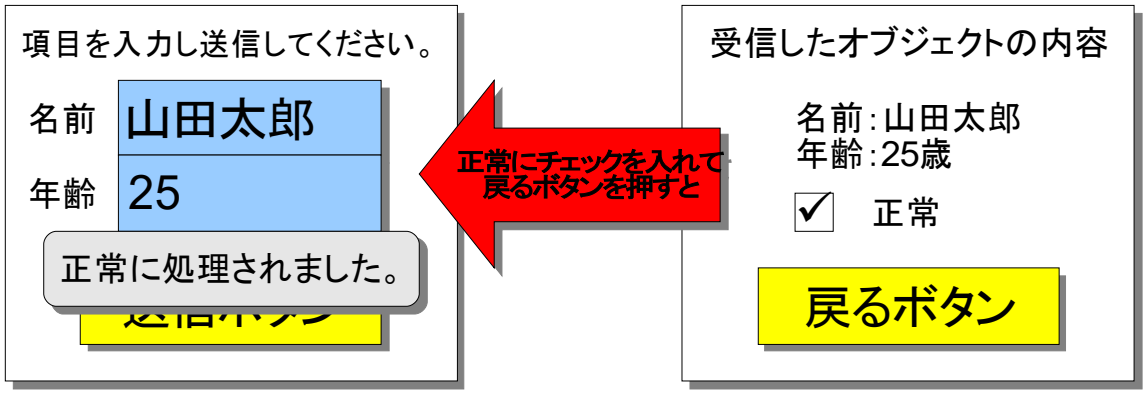
◆起動先から戻り値を取得する際に使用するメソッド

メソッド	説明
startActivityForResul （Activityクラスに定義）	【書式】 public void startActivityForResult(Intent intent, int requestCode) ※第1引数には、実行するインテントを指定する。 ※第2引数には、リクエストコードを指定する。 【機能】 起動先Activityが終了時に、 戻り値となるインテントを取得できる形でアクティビティを起動する。
onActivityResult （Activityクラスに定義）	【書式】 public void onActivityResult(int requestCode, int resultCode, Intent intent) ※第1引数には、取得するリクエストコードを指定する。 ※第2引数には、取得する結果コードを指定する。 ※第3引数には、取得するインテントを指定する。 【機能】 起動先Activityが終了時に呼び出されるメソッド。

11.4 インテントの活用方法3

アプリケーション名：IntentApp05

動作概要



◆作成手順

①「IntentApp05」を変更する。

②「activity_next.xml」で「CheckBox」を追加配置する。

③「MainActivity.java」と「NextActivity.java」のプログラムを設定する。

●「MainActivity.java」のプログラム例

ファイル名：MainActivity.java

```
public class MainActivity extends Activity implements OnClickListener {
    private static int REQUEST_CODE = 0;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        Button button = (Button)findViewById(R.id.button);
        button.setOnClickListener(this);
    }

    @Override
    public void onClick(View view) {
        EditText edit1 = (EditText)findViewById(R.id.editText);
        EditText edit2 = (EditText)findViewById(R.id.editText2);

        String name = edit1.getText().toString();
        int age = Integer.parseInt(edit2.getText().toString());
        PersonalData data = new PersonalData(name, age);

        Intent intent = new Intent(this, NextActivity.class);
        intent.putExtra("PERSONAL_DATA", data);
        startActivityForResult(intent, REQUEST_CODE);
    }
```

```
    @Override
    protected void onActivityResult(int requestCode, int resultCode, Intent intent) {
        super.onActivityResult(requestCode, resultCode, intent);

        if (requestCode != REQUEST_CODE) {
            return;
        }

        String str = "";
        Bundle bundle = intent.getExtras();

        if (resultCode == Activity.RESULT_OK) {
            str = bundle.getString("MESSAGE");
        } else if (resultCode == Activity.RESULT_CANCELED) {
            str = bundle.getString("ERROR_MESSAGE");
        }

        Toast toast = Toast.makeText(this, str, Toast.LENGTH_SHORT);
        toast.show();
    }
}
```
※一部抜粋

● 「NextActivity.java」のプログラム例

ファイル名：NextActivity.java

```
public class NextActivity extends Activity implements OnClickListener {

    @Override
    public void onClick(View view) {
        CheckBox checkBox = (CheckBox)findViewById(R.id.checkBox);

        Intent intent = new Intent();

        if (checkBox.isChecked()) {
            intent.putExtra("MESSAGE", "正常に処理されました。");
            setResult(Activity.RESULT_OK, intent);
        } else {
            intent.putExtra("ERROR_MESSAGE", "正常に処理されませんでした。");
            setResult(Activity.RESULT_CANCELED, intent);
        }

        finish();
    }
}
```

※一部抜粋

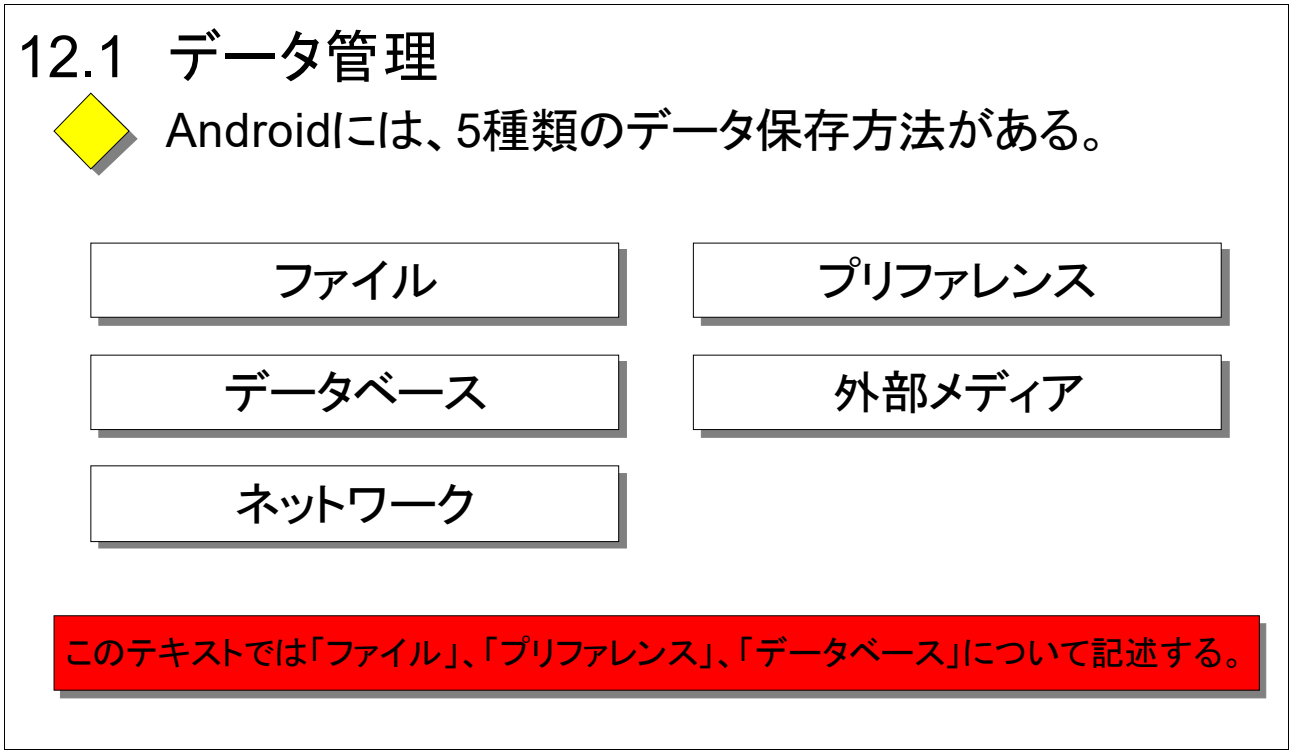
★setResult メソッド（Activity クラスに定義）

【書式】
public void setResult(int resultCode, Intent intent)
※第1引数には、成功時はActivity.RESULT_OK、キャンセル時はActivity.RESULT_CANCELEDを設定する。
※第2引数には、インテントを設定する。

【機能】
戻り値となる結果コードとインテントを設定する。


メモ

第 12 章
データ管理（前編）



Android アプリケーションで扱うデータは、アプリケーション終了時に消滅します。電話帳などのように永続的にデータを保存したい場合には、ファイルやデータベース等を利用する必要があります。Android には、データの保存方法として 5 種類あります。

◆Android のデータ保存方法

●ファイル

Android 端末の内部の記憶媒体にファイル形式でデータを保存する。

●プリファレンス

Android 端末の内部の記憶媒体に名前と値を組み合わせて XML ファイルにデータを保存する。

●データベース

Android には、SQLite というデータベースがデフォルトで搭載されており、データの検索、登録、更新、削除などのデータ操作を効率よく行うことができる。

●外部メディア

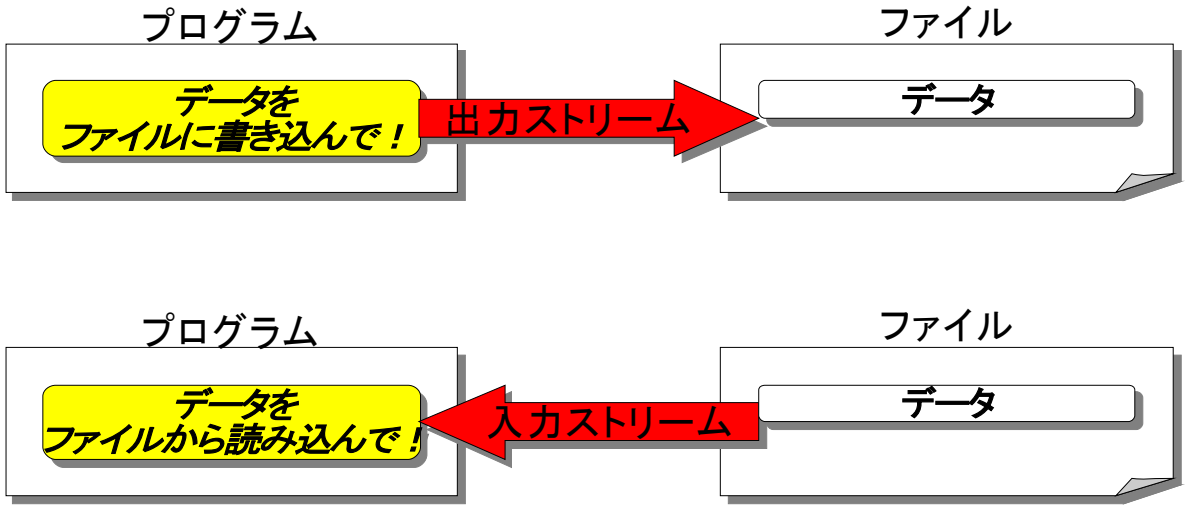
Android では、SD カードなどの外部の記憶媒体の利用が可能であり、ファイル形式でデータを保存することができる。

●ネットワーク

Android では、Web によるネットワークを使用しデータを保存したり、データを取得することができる。

12.2 ファイル・アクセス

ファイルを使用しデータを読み書きする。



Android 端末の内部の記憶媒体にファイル形式で、データをアクセス（データの読み書き）することが可能です。それは、プログラムとファイル間のストリーム（通信経路）を作成することにより実現することができます。また、Android ではファイルアクセス用のメソッドが用意されています。

◆ファイルアクセスの流れ

ファイルへの書き込み	ファイルからの読み込み
①出力ストリームを生成 ②ファイルへデータを書き込む ③出力ストリームを閉じる	①入力ストリームを生成 ②ファイルからデータを読み込む ③入力ストリームを閉じる

◆ファイルアクセス用メソッド

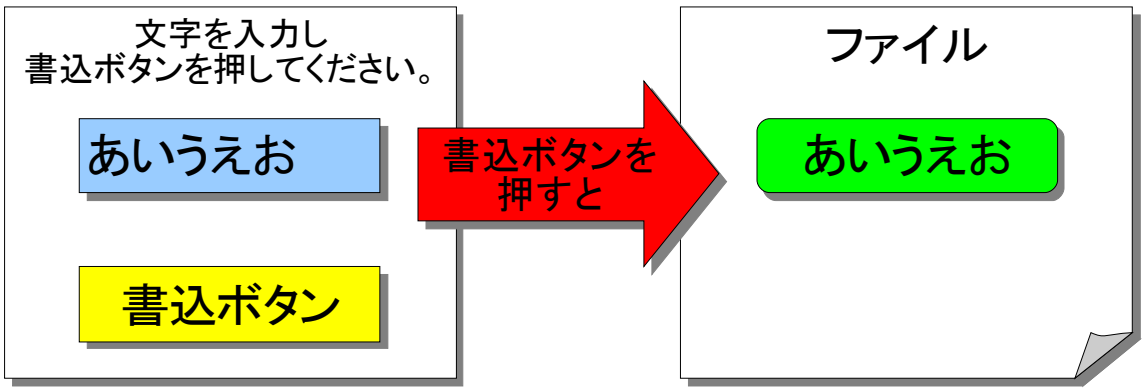
Activity には、ファイルにアクセスするための専用ストリームを取得するメソッドが定義されている。

<table>
<tr><th>メソッド</th><th>説明</th></tr>
<tr><td>openFileInput
（Contextクラスに定義）</td><td>【書式】 public FileInputStream openFileInput(String fileName)
【機能】 第1引数で指定したファイルからFileInputStreamオブジェクトを生成する。</td></tr>
<tr><td>openFileOutput
（Contextクラスに定義）</td><td>【書式】 public FileOutputStream openFileOutput(String fileName, int mode)
【機能】 第1引数で指定したファイルからFileOutputStreamオブジェクトを生成する。
第2引数には、処理モード定数を指定する。
【処理モード定数】
<table>
<tr><th>定数</th><th>説明</th></tr>
<tr><td>MODE_PRIVATE</td><td>他のアプリケーションからのアクセスはできない。</td></tr>
<tr><td>MODE_WORLD_READABLE</td><td>他のアプリケーションからの読み込みができる。</td></tr>
<tr><td>MODE_WORLD_WRITEABLE</td><td>他のアプリケーションからの書き込みができる。</td></tr>
<tr><td>MODE_APPEND</td><td>既存ファイルに追加書き込みができる。</td></tr>
</table>
</td></tr>
</table>

12.3 ファイル・アクセスの利用方法1

アプリケーション名：FileAccessApp

動作概要



◆作成手順

①新規アプリケーション「FileAccessApp」を作成する。

②「activity_main.xml」で「TextView」と「EditText」と「Button」を配置する。

③「MainActivity.java」でプログラムを設定する。

●プログラム例

ファイル名：MainActivity.java

```
package com.example.fileaccessapp;

import java.io.FileOutputStream;

import android.os.Bundle;
import android.app.Activity;
import android.view.Menu;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

public class MainActivity extends Activity implements OnClickListener {

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        Button button = (Button)findViewById(R.id.button);
        button.setOnClickListener(this);
    }
```

```
    @Override
    public void onClick(View v) {
        Toast toast = null;
        EditText edit = (EditText)findViewById(R.id.editText);

        String result = "正常に書込ができました。";
        String str = edit.getText().toString();

        try {
            FileOutputStream stream = openFileOutput("data.txt", MODE_APPEND);
            BufferedWriter out = new BufferedWriter(new OutputStreamWriter(stream));
            out.write(str + "\n");
            out.close();
        } catch (Exception e) {
            result = e.getMessage();
        } finally {
            edit.setText("");

            toast = Toast.makeText(this, result, Toast.LENGTH_SHORT);
            toast.show();
        }
    }
}
```

※一部抜粋

●ファイルの保存先

ディレクトリ	/data/data/パッケージ名/files
ファイル名	プログラムで指定したファイル名

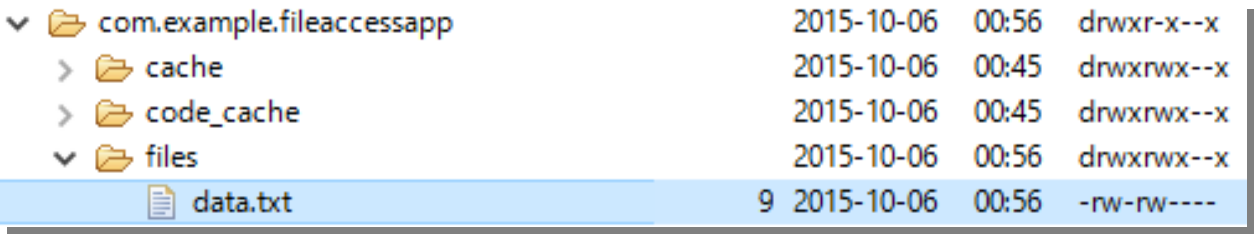
【実行確認】

①エミュレータを起動し、EditText に文字列を入力後、書込を行う。

②書き込み後、[ツール] → [Android] → [Android Device Monitor]を選択し

「File Explorer」タブで/data/data/com.example.fileaccessapp/files フォルダ内に

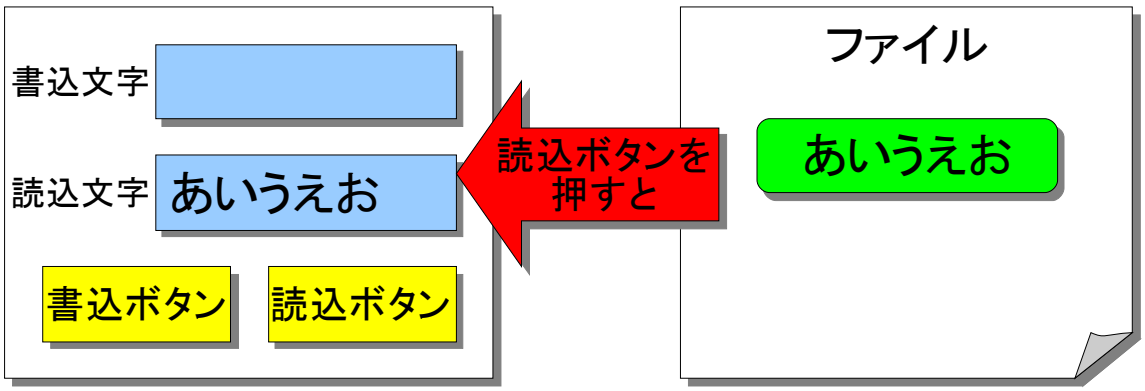
「data.txt」ファイルが作成されているのを確認する。



12.4 ファイル・アクセスの利用方法2

アプリケーション名：FileAccessApp

動作概要



◆作成手順

①「FileAccessApp」を変更する。

②「activity_main.xml」で「TextView」と「EditText」と「Button」を追加配置する。

③「MainActivity.java」でプログラムを設定する。

●プログラム例

ファイル名：MainActivity.java

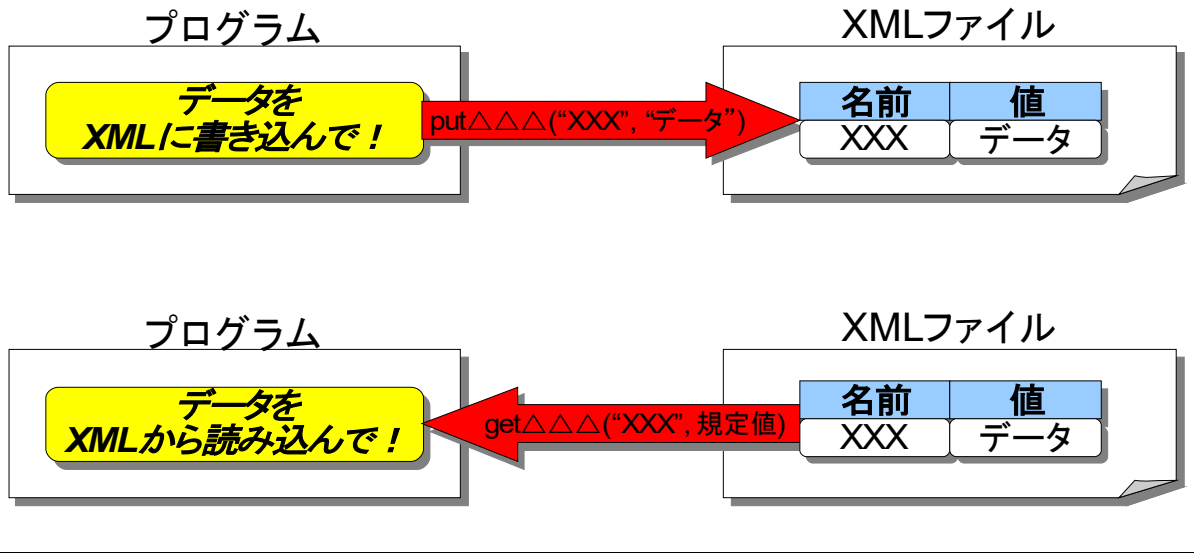
```
package com.example.fileaccessapp;

public class MainActivity extends Activity implements OnClickListener {

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        Button button1 = (Button)findViewById(R.id.button);
        button1.setOnClickListener(this);

        Button button2 = (Button)findViewById(R.id.button2);
        button2.setOnClickListener(this);
    }
```

```
    @Override
    public void onClick(View view) {
        Button button = (Button)view;
        Toast toast = null;
        String fileName = "data.txt";

        if (button.getId() == R.id.button) {
            EditText edit = (EditText)findViewById(R.id.editText);

            String result = "正常に書込ができました。";
            String str = edit.getText().toString();

            try {
                saveText(fileName, str);
            } catch (Exception e) {
                result = e.getMessage();
            } finally {
                edit.setText("");

                toast = Toast.makeText(this, result, Toast.LENGTH_SHORT);
                toast.show();
            }
        } else if (button.getId() == R.id.button2) {
            EditText edit = (EditText)findViewById(R.id.editText2);

            String str = "";
            String result = "正常に読込ができました。";

            try {
                str = readText(fileName);
                edit.setText(str);
            } catch (Exception e) {
                result = e.getMessage();
            } finally {
                toast = Toast.makeText(this, result, Toast.LENGTH_SHORT);
                toast.show();
            }
        }
    }

    private void saveText(String fileName, String str) throws IOException {
        FileOutputStream stream = openFileOutput(fileName, MODE_APPEND);
        BufferedWriter out = new BufferedWriter(new OutputStreamWriter(stream));

        out.write(str + "\n");
        out.close();
    }

    private String readText(String fileName) throws IOException {
        FileInputStream stream = openFileInput(fileName);
        BufferedReader in = new BufferedReader(new InputStreamReader(stream));

        String str = "";
        String line = "";

        while ((line = in.readLine()) != null) {
            str += line + "\n";
        }

        in.close();

        return str;
    }
}
```

※一部抜粋

12.5 プリファレンス

XMLファイルを使用しデータを読み書きする。



目覚まし時計のように、アラームを鳴らす時間を設定したりする場合がある。このように保持したいデータが数個の場合には、データベースを使用するよりもプリファレンスを使用したほうが便利である。

プリファレンスは、名前と値の組み合わせによりデータを XML ファイルに保存する。

◆プリファレンス利用の手順

①プリファレンスにアクセスするための SharedPreferences オブジェクトを取得する。

メソッド	説明
getSharedPreferences （Contextクラスに定義）	【書式】 public SharedPreferences getSharedPreferences(String fileName, int mode) 【機能】 第1引数で指定したﾌｧｲﾙにｱｸｾｽするSharedPreferences ｵﾌﾞｼﾞｪｸﾄを生成する。 第2引数には、処理モード定数を指定する。 【処理モード定数】 定数 / 説明 MODE_PRIVATE / 他のアプリケーションからのアクセスはできない。 MODE_WORLD_READABLE / 他のアプリケーションからの読み込みができる。 MODE_WORLD_WRITEABLE / 他のアプリケーションからの書き込みができる。

②データの登録、変更、削除をするための SharedPreferences.Editor オブジェクトを取得する。

メソッド	説明
edit （SharedPreferences インターフェースに定義）	【書式】 public SharedPreferences.Editor edit() 【機能】 指定したSharedPreferences のEditorオブジェクトを生成する。 このオブジェクトによりデータの登録、変更、削除が行える。

③データを保存または取得する。（保存する場合は、コミットすることでプリファレンスに書き込まれる。）

●データ保存用メソッド（SharedPreferences.Editor インターフェースに定義）

メソッド	説明
putBoolean	【書式】 public abstract SharedPreferences.Editor putBoolean(String key, boolean value) 【機能】 boolean値を設定する。
putFloat	【書式】 public abstract SharedPreferences.Editor putFloat(String key, float value) 【機能】 float値を設定する。
putInt	【書式】 public abstract SharedPreferences.Editor putInt(String key, int value) 【機能】 int値を設定する。
putLong	【書式】 public abstract SharedPreferences.Editor putLong(String key, long value) 【機能】 long値を設定する。
putString	【書式】 public abstract SharedPreferences.Editor putStringString key, String value) 【機能】 String値を設定する。

●データ取得用メソッド（SharedPreferences インターフェースに定義）

メソッド	説明
getBoolean	【書式】 public abstract boolean getBoolean(String key, boolean defValue) 【機能】 boolean 値を取得する。
getFloat	【書式】 public abstract float getFloat(String key, float defValue) 【機能】 float値を取得する。
getInt	【書式】 public abstract int getInt(String key, int defValue) 【機能】 int値を取得する。
getLong	【書式】 public abstract long getLong(String key, long defValue) 【機能】 long値を取得する。
getString	【書式】 public abstract String getStringString key, String defValue) 【機能】 String値を取得する。

※第 2 引数の「defValue」は、データを取得できなかった時の、規定値を設定する。

④処理を完結させる。（データの登録、変更、削除の場合に使用する。）

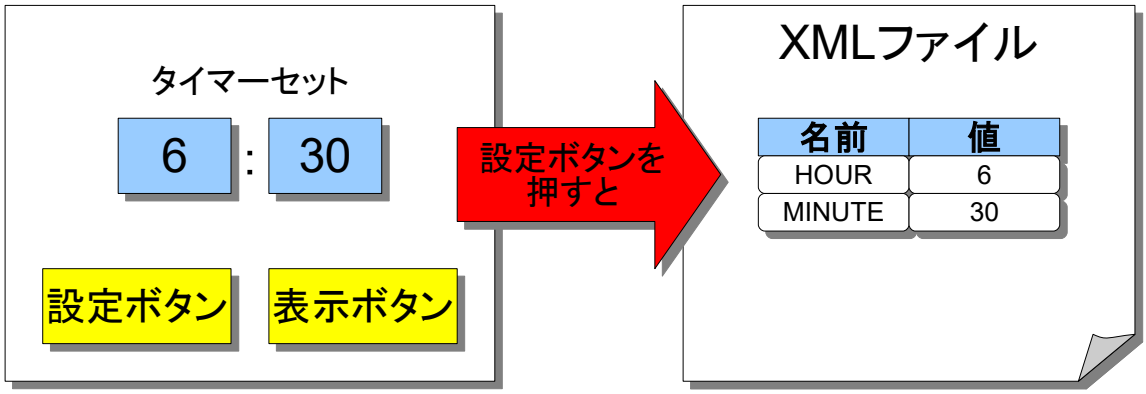
●commit メソッド（SharedPreferences.Editor インターフェースに定義）

メソッド	説明
commit	【書式】 public abstract boolean commit() 【機能】 処理を完結させる。

12.6 プリファレンスの利用方法1

アプリケーション名：PreferencesApp

動作概要



◆作成手順

①新規アプリケーション「PreferencesApp」を作成する。

②「activity_main.xml」で「TextView」と「EditText」と「Button」を配置する。

③「MainActivity.java」でプログラムを設定する。

●プログラム例

ファイル名：MainActivity.java

```
package com.example.preferencesapp;

import android.os.Bundle;
import android.app.Activity;
import android.content.SharedPreferences;
import android.view.Menu;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

public class MainActivity extends Activity implements OnClickListener {
    private EditText hourEdit;
    private EditText minuteEdit;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        hourEdit = (EditText)findViewById(R.id.editText);
        minuteEdit = (EditText)findViewById(R.id.editText2);
        Button saveButton = (Button)findViewById(R.id.button);
        saveButton.setOnClickListener(this);
    }
```

```
    @Override
    public void onClick(View view) {
        String fileName = "data";

        saveRecord(fileName);

        Toast toast = Toast.makeText(this, "正常にタイマー設定ができました。", Toast.LENGTH_SHORT);
        toast.show();

    }

    private void saveRecord(String fileName) {
        SharedPreferences preference = getSharedPreferences(fileName, MODE_PRIVATE);
        SharedPreferences.Editor editor = preference.edit();

        int hour = Integer.parseInt(hourEdit.getText().toString());
        int minute = Integer.parseInt(minuteEdit.getText().toString());

        editor.putInt("HOUR", hour);
        editor.putInt("MINUTE", minute);

        editor.commit();
    }
}
```

※一部抜粋

●ファイルの保存先

ディレクトリ	/data/data/パッケージ名/shared_prefs
ファイル名	プログラムで指定したファイル名

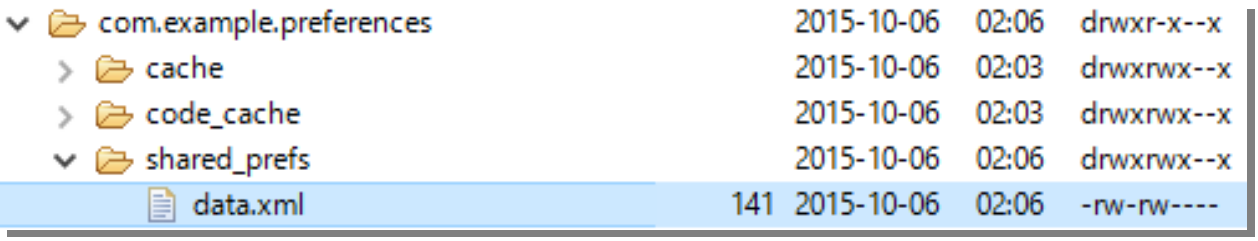
【実行確認】

①エミュレータを起動し、EditText に「時」と「分」を入力後、設定を行う。

②設定後、[ツール] → [Android] → [Android Device Monitor]を選択し

「File Explorer」タブで/data/data/com.example.preferencesapp/shared_prefs フォルダ内に

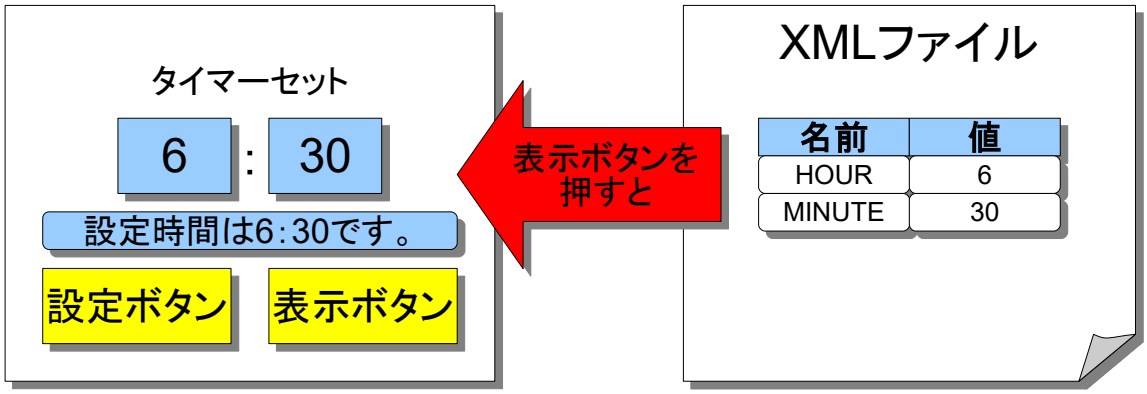
「data.xml」ファイルが作成されているのを確認する。



12.7 プリファレンスの利用方法2

アプリケーション名：PreferencesApp

動作概要



◆作成手順

①「PreferencesApp」を変更する。

②「MainActivity.java」でプログラムを設定する。

●プログラム例

ファイル名：MainActivity.java

```
package com.example.preferences;

public class MainActivity extends Activity implements OnClickListener {
    private EditText hourEdit;
    private EditText minuteEdit;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        hourEdit = (EditText)findViewById(R.id.editText);
        minuteEdit = (EditText)findViewById(R.id.editText2);

        Button saveButton = (Button)findViewById(R.id.button);
        saveButton.setOnClickListener(this);

        Button loadButton = (Button)findViewById(R.id.button2);
        loadButton.setOnClickListener(this);
    }
```

```
    @Override
    public void onClick(View view) {
        Button button = (Button)view;

        String str = "";
        String fileName = "data";

        if (button.getId() == R.id.button) {
            str = saveRecord(fileName);
        } else {
            str = loadRecord(fileName);
        }

        Toast toast = Toast.makeText(this, str, Toast.LENGTH_SHORT);
        toast.show();
    }

    private String saveRecord(String fileName) {
        String str = "";

        SharedPreferences preference = getSharedPreferences(fileName, MODE_PRIVATE);
        SharedPreferences.Editor editor = preference.edit();

        int hour = Integer.parseInt(hourEdit.getText().toString());
        int minute = Integer.parseInt(minuteEdit.getText().toString());

        editor.putInt("HOUR", hour);
        editor.putInt("MINUTE", minute);

        if (editor.commit()) {
            str = "正常にタイマー設定ができました。";
        } else {
            str = "タイマー設定に失敗しました。";
        }

        return str;
    }

    private String loadRecord(String fileName) {
        String str = "";

        SharedPreferences preference = getSharedPreferences(fileName, MODE_PRIVATE);

        int hour = preference.getInt("HOUR", -1);
        int minute = preference.getInt("MINUTE", -1);

        if (hour == -1 || minute == -1) {
            str = "タイマーが設定されていません。";
        } else {
            str = "設定時間は" + hour + ":" + minute + "です。";
        }

        return str;
    }
}
```

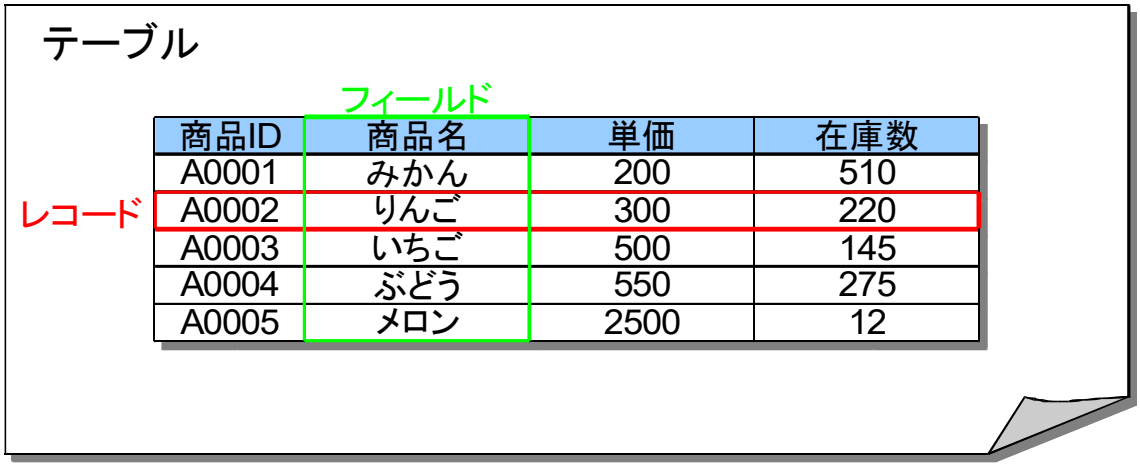
※一部抜粋

メモ

第 13 章
データ管理（後編）

13.1 データベースについて

様々な情報を格納するデータの集合体



データベースとは、様々な情報を格納するデータの集合体のことです。データベースには、データを簡単に利用するための機能が組み込まれており、大量のデータを高速に処理することができます。

現在のデータベースと言えば、２次元の表で表現するリレーショナルデータベース（RDB）が主流で 、Oracle、MySQL、PostgreSQL、Access などの種類があります。

また、ファイル処理と比較しても、データベースを利用するほうのメリットが大きいため、アプリケーション製作には、なくてはならない存在です。

Android には SQLite というオープンソースのデータベースが同梱されています。これは、アプリケーションに組み込んで利用できる軽量のデータベースです。RDBMS とは異なりサーバ型ではないので、ログイン ID やパスワードなども不要です。また、プリファレンスと同様にファイルでデータベースを管理しているため、バックアップも容易です。

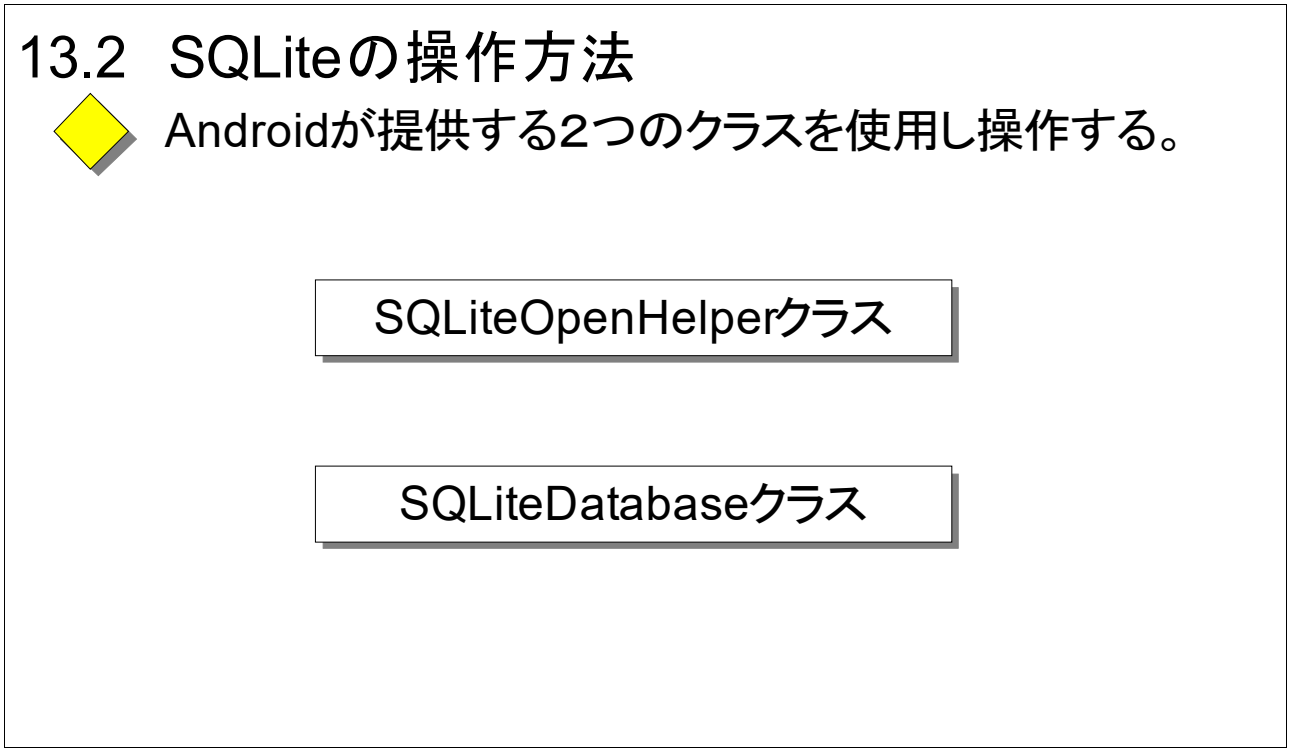
◆データベースのメリット

●同時アクセスをしても、データの整合性が保たれる。
（データベースの機能として自動的に排他制御が行われる）

●不正なデータを登録できないようにする機能が備わっている。

●トランザクションを管理する機能が備わっている。

●バックアップやリカバリ機能が備わっている。



Android では SQLite がサポートされており、Android が提供する２つのクラスを使用することにより SQLite を操作することができます。

◆Android が提供する SQLite 操作用クラス

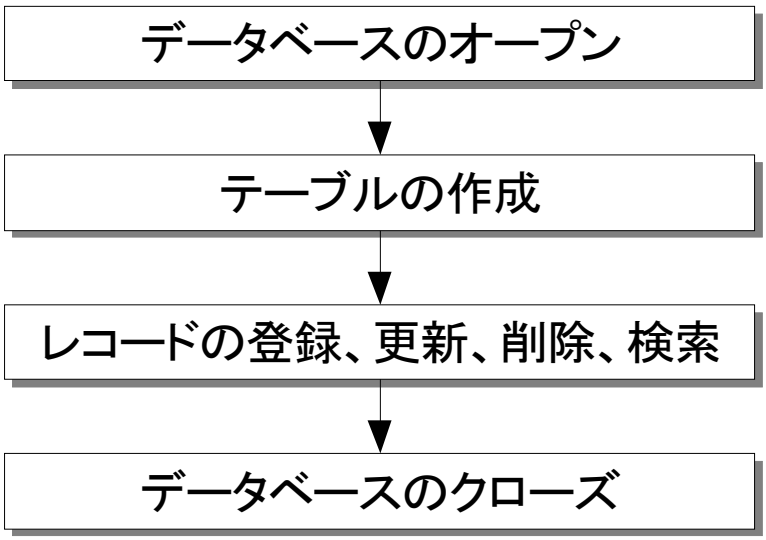
●SQLiteOpenHelper クラス

データベースの作成､テーブルの作成､データベースとの接続（オープン）､切断（クローズ）などの処理を行うための抽象クラスである。

●SQLiteDatabase クラス

テーブルの検索、登録、更新、削除を行うメソッドを提供するクラスである。

13.3 SQLiteの操作手順



Android で SQLite の操作を行うには、以下の手順で行います。

◆SQLite の操作手順

①データベースのオープン

SQLiteOpenHelper クラスのコンストラクタで、データベースを作成しオープンする。SQLiteOpenHelper クラスは抽象クラスであるため、このクラスを継承したクラスを定義する。

●SQLiteOpenHelper クラス（一部抜粋）

```
public abstract class SQLiteOpenHelper {
   //コンストラクタ
   public SQLiteOpenHelper(Context context, String name, SQLiteDatabase.CursolFactory factory, int version) {    ・・}

   //メソッド
   public SQLiteDatabase getReadableDatabase() {  ・・}
   public SQLiteDatabase getWritableDatabase() {  ・・}
   public abstract void onCreate(SQLiteDatabase db);
   public abstract void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion);
   public  void onOpen(SQLiteDatabase db){  ・・}
   public synchronized void close() {  ・・}
}
```

●SQLiteOpenHelper クラスのコンストラクタ

```
【書式】
public SQLiteOpenHelper(Context context, String name, SQLiteDatabase.CursolFactory factory, int version)

【引数】
第1引数には、データベースを所有するContextオブジェクトを指定する。
第2引数には、データベースファイルの名前を指定する。
第3引数には、CursorFactoryオブジェクトを指定する。通常はnullを指定する。
第4引数には、データベースのバージョンを指定する。
```

●SQLiteOpenHelper クラスのメソッド

メソッド	説明
onCreate	【書式】 public abstract void onCreate(SQLiteDatabase db) 【機能】 データベースが作成されたタイミングで呼び出されるメソッド。
onUpgrade	【書式】 public abstract void onCreate(SQLiteDatabase db, int oldVer, int newVer) 【機能】 データベースのバージョンを変更したタイミングで呼び出されるメソッド。
getWritableDatabase	【書式】 public SQLiteDatabase getWritableDatabase() 【機能】 データの作成、登録、更新、削除処理を行うSQLiteDatabaseｵﾌﾞｼﾞｪｸﾄを取得する。
getReadableDatabase	【書式】 public SQLiteDatabase getReadableDatabase() 【機能】 データの検索など読み込み専用処理を行うSQLiteDatabaseｵﾌﾞｼﾞｪｸﾄを取得する。

●SQLiteOpenHelper クラスを継承したクラスを定義する。

```
public class △△△Helper extends SQLiteOpenHelper {
    //コンストラクタ
    public △△△Helper(Context context) {
        super(context, "databaseName", null, 1);
    }

    @Override
    public void onCreate(SQLiteDatabase db) {

    }

    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {

    }
}
```

②テーブルの作成

データベースが作成されると SQLiteOpenHelper クラスの onCreate メソッドが呼び出され、そのメソッドの引数である SQLiteDatabase オブジェクトを使用し、テーブルを作成する SQL 文を実行する。

●SQLiteDatabase クラス（一部抜粋）

```
public class SQLiteDatabase {
    //メソッド
    public void execSQL(String sql) { ・・}
    public void beginTransaction() { ・・}
    public void setTransactionSuccessful() { ・・}
    public void endTransaction() { ・・・}
    public long insert(String table, String nullColumnHack, ContentValues values) {    ・・}
    public int update(String table, ContentValues values, String whereClause, String[] whereArgs) {    ・・}
    public int delete(String table, String whereClause, String[] whereArgs) {    ・・}
    public Cursor query(String table, String[] columns, String selection, String[] selectionArgs, String groupBy,
                                                                                    String having, String orderBy) { ・・}
}
```

●SQL 文実行メソッド（SQLiteDatabase クラスに定義）

メソッド	説明
execSQL	【書式】 public void execSQL(String sql) 【機能】 検索のような結果を返すSQL文ではないSQL文を実行する。

●テーブル作成の SQL 文

```
【書式】
CREATE TABLE [テーブル名] (
    [フィールド名1] [データ型] [制約],
    [フィールド名2] [データ型] [制約],・・・・・・・・・・
);
```

★SQLite のデータ型

データ型	説明
NULL	NULL値
INTEGER	符号付整数
REAL	浮動小数
TEXT	テキスト（UTF-8、UTF-16BE、UTF-16-LEのいずれかにより格納する。）
BLOB	入力データを、そのまま格納する。（Binary Large Object の略）

★SQLite の制約

制約	説明
PRIMARY KEY	主キー設定
NOT NULL	NULLを許可しない設定
DEFAULT	レコード追加時に、規定値が自動的に格納される設定
UNIQUE	重複する値を格納できなくする設定
AUTOINCREMENT	最後に格納された値に1加算した値を使用する設定

●onCreate メソッドでテーブルを作成するプログラムを行う。

```
@Override
public void onCreate(SQLiteDatabase db) {
    db.execSQL("CREATE TABLE ・・・・・・・・・;");
}
```

③レコードの登録、更新、削除、検索

アクティビティで、例えばボタンが押された時に、SQLiteDatabase クラスに定義されている登録用の insert メソッド、更新用の update メソッド、削除用の delete メソッド、SQL 文実行用の execSQL メソッドなどを使用し、レコードの登録、更新、削除を行う。また、query メソッドを使用しレコードの検索を行う。

●登録、更新、削除、検索メソッド（SQLiteDatabase クラスに定義）

メソッド	説明
insert	【書式】 public long insert(String table, String nullColumnHack, ContentValues values) ※第1引数には、テーブル名を指定する。 ※第2引数には、null値が許されない列に値が指定されていない場合、代わりに利用される値を指定する。 ※第3引数には、登録データを格納したContentValuesオブジェクトを指定する。 【機能】 レコードを登録する。
update	【書式】 public int update(String table, ContentValues values, String whereClause, String[] whereArgs) ※第1引数には、テーブル名を指定する。 ※第2引数には、登録データを格納したContentValuesオブジェクトを指定する。 ※第3引数には、更新する条件式を指定する。 ※第4引数には、更新条件に「？」が含まれる場合に置き変わる値を指定する。不要な場合はnull。 【機能】 レコードを更新する。
delete	【書式】 public int delete(String table, String whereClause, String[] whereArgs) ※第1引数には、テーブル名を指定する。 ※第2引数には、削除する条件式を指定する。nullにすると全レコードが削除対象となる。 ※第3引数には、削除条件に「？」が含まれる場合に置き変わる値を指定する。不要な場合はnull。 【機能】 レコードを削除する。
query	【書式】public Cursor query(String table, String[] columns, String selection, String[] selectionArgs, String groupBy, String having, String orderBy) ※第1引数には、テーブル名を指定する。 ※第2引数には、取得する列名の配列を指定する。 ※第3引数には、選択する条件式を指定する。 ※第4引数には、選択条件に「？」が含まれる場合に置き変わる値を指定する。不要な場合はnull。 ※第5引数には、集計条件を指定する。（GROUP BY） ※第6引数には、集計関数を含む選択条件を指定する。（HAVING） ※第7引数には、並べ替えの条件を設定する。（ORDER BY） 【機能】 レコードを検索する。

●トランザクション制御用メソッド（SQLiteDatabase クラスに定義）

メソッド	説明
beginTransaction	【書式】 public void beginTransaction() 【機能】 排他モードでトランザクションを開始する。
setTransactionSuccessful	【書式】 public void setTransactionSuccessful() 【機能】 コミットする。
endTransaction	【書式】 public void endTransaction() 【機能】 該当のトランザクションを終了する。

④データベースのクローズ

データベースの処理が終了したら、データベースのクローズを行う。

●データベースのｸﾛｰｽﾞﾒｿｯﾄﾞ（SQLiteDatabase のスーパークラスである SQLiteClosable クラスに定義）

メソッド	説明
close	【書式】 public void close() 【機能】 データベースをクローズする。

●アクティビティでデータベースを使用するプログラムを行う。

＜例 1＞ボタンを押したら、登録あるいは更新あるいは削除を行う場合

```
public class △△△Activity extends Activity implements OnClickListener {
    private △△△Helper helper;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        ・・・・・・・・・・
        Button button = (Button)findViewById(R.id.button1);
        button.setOnClickListener(this);

        helper = new △△△Helper(this);
    }

    @Override
    public void onClick(View view) {
        SQLiteDatabase db = helper.getWritableDatabase();

        //登録あるいは更新あるいは削除処理を行う。

        db.close();
    }
}
```

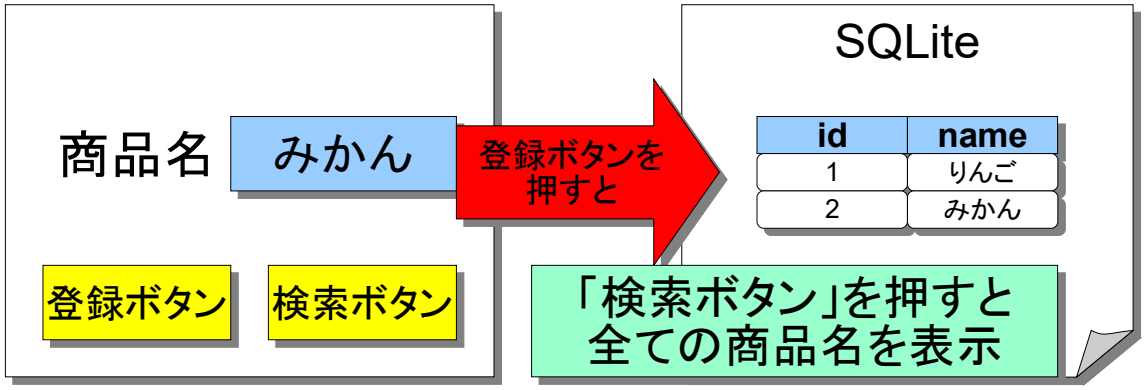
＜例 2＞ボタンを押したら、検索を行う場合

```
public class △△△Activity extends Activity implements OnClickListener {
    private △△△Helper helper;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        ・・・・・・・・・・
        Button button = (Button)findViewById(R.id.button1);
        button.setOnClickListener(this);

        helper = new △△△Helper(this);
    }

    @Override
    public void onClick(View view) {
        SQLiteDatabase db = helper.getReadableDatabase();

        //検索処理を行う。

        db.close();
    }
}
```

※データベースが作成されるタイミングは、getWritableDatabase または getReadableDatabase メソッドによりデータベースにアクセスした時である。

13.4 データベースの利用方法

- アプリケーション名：DatabaseApp
- 動作概要



◆作成手順

①新規アプリケーション「DatabaseApp」を作成する。

②「activity_main.xml」で「TextView」と「EditText」と「Button」２つを配置する。

③「DatabaseHelper.java」でプログラムを構成する。

ファイル名：DatabaseHelper.java

```
package com.example.databaseapp;

import android.content.Context;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;

public class DatabaseHelper extends SQLiteOpenHelper {
    private static String DB_NAME = "android_sqlite";
    private static int DB_VERSION = 1;

    public DatabaseHelper(Context context) {
        super(context, DB_NAME, null, DB_VERSION);
    }

    @Override
    public void onCreate(SQLiteDatabase db) {
        String sql = "CREATE TABLE Products(";
        sql += "id INTEGER PRIMARY KEY AUTOINCREMENT,";
        sql += "name TEXT NOT NULL";
        sql += ");";

        db.execSQL(sql);
    }

    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {

    }
}
```

④「MainActivity.java」でプログラムを構成する。

ファイル名：MainActivity.java

```
package com.example.databaseapp;

import android.app.Activity;
import android.content.ContentValues;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.view.View.OnClickListener;
import android.widget.Toast;

public class MainActivity extends Activity implements OnClickListener {
    private DatabaseHelper helper;
    private EditText nameEdit;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        helper = new DatabaseHelper(this);

        nameEdit = (EditText)findViewById(R.id.editText);

        Button registButton = (Button)findViewById(R.id.button);
        registButton.setOnClickListener(this);

        Button searchButton = (Button)findViewById(R.id.button2);
        searchButton.setOnClickListener(this);
    }

    @Override
    public void onClick(View view) {
        Button button = (Button)view;

        String str = null;

        switch (button.getId()) {
            case R.id.button : str = insertRecord(); break;
            case R.id.button2 : str = searchRecords(); break;
        }

        Toast toast = Toast.makeText(this, str, Toast.LENGTH_SHORT);
        toast.show();
    }

    private String insertRecord() {
        SQLiteDatabase db = helper.getWritableDatabase();

        String name = nameEdit.getText().toString();

        ContentValues values = new ContentValues();
        values.put("name", name);

        String str = db.insert("Products", null, values) != -1 ? "登録成功" : "登録失敗";

        return str;
    }
```

```
    private String searchRecords() {
        String str = "";
        SQLiteDatabase db = helper.getReadableDatabase();

        String[] columns = {"name"};

        Cursor cursor = db.query("Products", columns, null, null, null, null, null);

        while (cursor.moveToNext()) {
            str += cursor.getString(0) + "\n";
        }

        return str;
    }
}
```


メモ