Android チュートリアル 2
–データ保存と読み込み–*

愛知県立大学 山本研究室
藤浦 祥雅†

1 前回のおさらい

前回のチュートリアルで，以下のようなことを学びました．

- アプリケーションの開発方法
- アプリケーションの画面 (Activity)
- レイアウトファイルと文字列ファイル (layout ディレクトリと strings.xml)
- XML もレイアウト変更
- リソースの新規，参照の方法
- 複数 Activity の扱い方 (Intent と manifest)

Android アプリケーションの開発は，Eclipse とエミュレータで行ってきました．以降もこの二つを利用してチュートリアルを進めていきます．

また基本的に画面のレイアウトは，XML を利用して変更していきます．

今回チュートリアルは，データの保存方法についてです．データの保存方法には，Preference, DataBase, File の 3 つの方法があり，それぞれを順次，説明していきたいと思います．

2 Preference について

Preference は，キーと値による簡単なデータの保存方法です．ある一つキーについて，一つのデータのみを保存することができます．

前回チュートリアルの最後に作成した，文字を入力して出力するアプリケーションをデータが保存できるように拡張していきます．

Preference を利用してデータを保存したソースは，以下のようになります．

*URL:[http://www.aichi-pu.ac.jp/ist/lab/yamamoto/android/android-tutorial/tutorial02/tutorial02.pdf]
†mail:[qtutorial@gmail.com]

リスト 1: Preference を利用した Display

```
1  package sample.pref;
2
3  import android.app.Activity;
4  import android.content.Intent;
5  import android.content.SharedPreferences;
6  import android.os.Bundle;
7  import android.view.Menu;
8  import android.view.MenuItem;
9  import android.widget.TextView;
10
11 public class Display extends Activity {
12         private static final int EDIT = Menu.FIRST;
13         private static final int ACTIVITY_EDIT = 0;
14         private TextView tv;
15         private SharedPreferences pref;
16
17     /** Called when the activity is first created. */
18     @Override
19     public void onCreate(Bundle savedInstanceState) {
20         super.onCreate(savedInstanceState);
21         setContentView(R.layout.display);
22         tv = (TextView) findViewById(R.id.text);
23
24         pref = getSharedPreferences("Display", 0);
25         tv.setText(pref.getString("text", ""));
26     }
27
28     @Override
29     public boolean onCreateOptionsMenu(Menu menu) {
30         super.onCreateOptionsMenu(menu);
31         menu.add(0, EDIT, 0, "EDIT");
32         return true;
33     }
34
35     @Override
36     public boolean onMenuItemSelected(int featureId, MenuItem item) {
37         switch (item.getItemId()) {
38         case EDIT:
39                 edit();
40                 return true;
41         }
42         return super.onMenuItemSelected(featureId, item);
43     }
44
45     private void edit() {
46         Intent i = new Intent(this, Edit.class);
47         startActivityForResult(i, ACTIVITY_EDIT);
48     }
49
50     @Override
51     public void onActivityResult(int requestCode, int resultCode, Intent intent) {
52         super.onActivityResult(requestCode, resultCode, intent);
53
54         switch (requestCode) {
55         case ACTIVITY_EDIT:
56                 tv.setText(pref.getString("text", ""));
57                 break;
58         }
59     }
60 }
```

リスト 2: Preference を利用した Edit

```
1  package sample.pref;
2
3
4  import android.app.Activity;
5  import android.content.SharedPreferences;
6  import android.os.Bundle;
7  import android.view.View;
8  import android.widget.Button;
9  import android.widget.EditText;
10
11 public class Edit extends Activity {
12         EditText et;
13         Button confirm;
14         SharedPreferences pref;
15
16         @Override
17         protected void onCreate(Bundle savedInstanceState) {
18                 super.onCreate(savedInstanceState);
19                 setContentView(R.layout.edit);
20                 et = (EditText) findViewById(R.id.edit);
21                 confirm = (Button) findViewById(R.id.confirm);
22
23                 pref = getSharedPreferences("Display", 0);
24                 et.setText(pref.getString("text", ""));
25
26                 confirm.setOnClickListener(new View.OnClickListener() {
27                         public void onClick(View view) {
28                                 sendText();
29                         }
30                 });
31         }
32
33         private void sendText() {
34                 SharedPreferences.Editor prefEditor = pref.edit();
35                 prefEditor.putString("text", et.getText().toString());
36                 prefEditor.commit();
37
38                 setResult(RESULT_OK, getIntent());
39                 finish();
40         }
41
42
43 }
```

前回との変更点は，Bundle を利用して Activity 自体がテキストデータを受け渡ししていたのを，Preference を利用してテキストデータの受け渡しを行うようにしました．今回は，“Display” という Preference 内の “text” というキーにテキストデータを保存しています．

次からクラス別の各メソッドについて説明していきます．

Display.java

この Display.java では，Preference の読み込みを行うことで，テキストデータの取得を行っています．

onCreate() [l.19 - l.26]

ここでは，レイアウトの読み込みと TextView の読み込み以外に，Preference の取得とテキストのセットを行っています．

24 行目で，Display という Preference を取得し，25 行目で text というキーの値を TextView にセットしています．こうすることで，前回保存したデータの読み込みを行います．

onCreateOptionMenu() [l.29 - l.33]

onMenuItemSelected() [l.36 - l.43]

この二つのメソッドは，前回と変わりありません．

edit() [l.45 - l.48]

前回，このメソッドは，Bundle を利用して，Intent でテキストデータを Edit Activity に送っていましたが，今回は，このアプリケーション内で共通利用できる Preference にテキストデータを保存しておいたので，startActivityForResult() のみでよいです．

onActivityResult() [l.51 - l.62]

このメソッドでの変更点は，55 行目以降になります．前回では，帰って来た Intent の Bundle からテキストデータを取り出し，textView にセットしていましたが，今回は Preference を利用してデータを読み込むことができるので，56 行目のように pref からデータを読み込むことで，テキストの表示を行います．

Edit.java

この Edit.java では，Preference の書き込みを行うことで，テキストデータの保存を行っています．

onCreate() [l.17 - l.31]

前回は，レイアウトファイルと EditText と Bundle を読み込んで，Bundle からテキストデータを取得し，EditText にデータをセットして，表示していました．今回は，レイアウトファイルと EditText を読み込んで，Preference から取得したテキストデータを EditText にセットしています．

Preference の読み込みは，23, 24 行目で Display と同様の方法で行っています．

sendText() [l. 33 - l.40]

前回このメソッドでは，Intent と Bundle を利用して，テキストデータを Display Activity に送っていましたが，今回は，ここでテキストデータを Preference に書き込んでいます．

Preference に書き込むためには，“SharedPreferncs.Editor” というクラスを利用します．このオブジェクトは，SharedPreferences の “edit()“メソッドから取得します．ここでは，34 行目が Editor の取得部分にあたります．

35 行目では，putString() メソッドを利用して，テキストデータを書き込んでいます．

Preference の書き込みで重要なのが，36 行目の “commit()“です．この commit() を実行しないと，Preference に書き込んだデータが実際に反映されないので，Preference のデータを変更した際には，必ずこのメソッドを利用し，データの変更があったことを伝えます．

もし，二回以上の変更を行った後に，変更を反映させたい時には，最後に一度だけこのメソッドを呼ぶだけでもよいです．

2.1 Preference の利用方法

この例で Preference の読み込み，書き込みを利用してデータを扱いました．

読み込みでは，SharedPreferences クラスを利用して，Preference を読み込み，その中のキーから値を取得しました．また，書き込みでは，SharedPreferences.Editor クラスを利用して行いました．データを変更した際には，commit() を使って変更の適用を実施します．Preference を利用してデータを扱うことで，Bundle や Intent などを無駄に利用せず，データの表示や保存も簡単に行うことできます．

しかし，Preference には，今回のように直接データを扱うだけではなく，オプションや設定などのプリファレンスに利用できます．Android では，アプリケーションの設定などに利用しやすいように，XML から Preference を利用することができます．

次は，XML からの Preference 利用法を説明します．

2.2 XML から Prefernce

Android では，レイアウトファイルと同じように，Preference を扱うことができます．ここでは，XML を利用して Preference を扱ってみます．

XML を利用した Preference の画面は，次の図 1 のようになります．

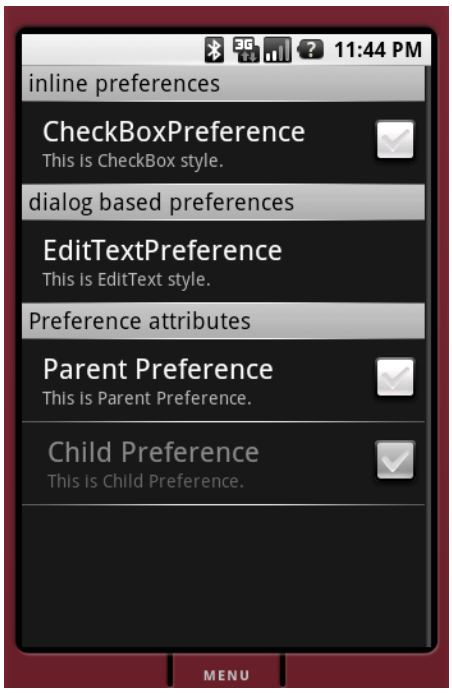

図 1: XML を利用した Preference

また，このソースと XML を以下に示します.

リスト 3: Java ファイル

```
1  package tutorial.v2;
2
3  import android.os.Bundle;
4  import android.preference.PreferenceActivity;
5
6  public class pref extends PreferenceActivity {
7      @Override
8      public void onCreate(Bundle savedInstanceState) {
9          super.onCreate(savedInstanceState);
10
11         addPreferencesFromResource(R.xml.pref);
12     }
13 }
```

リスト 4: XML ファイル

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <PreferenceScreen xmlns:android="http://schemas.android.com/apk/res/android">
3
4      <PreferenceCategory android:title="inline preferences">
5          <CheckBoxPreference
6                  android:key="checkbox_preference"
7                  android:title="CheckBoxPreference"
8                  android:summary="This is CheckBox style." />
9      </PreferenceCategory>
10
11     <PreferenceCategory android:title="dialog based preferences">
12         <EditTextPreference
13                 android:key="edittext_preference"
14                 android:title="EditTextPreference"
15                 android:summary="This is EditText style."
16                 android:dialogTitle="Please input any words." />
17     </PreferenceCategory>
18
19     <PreferenceCategory android:title="Preference attributes">
20         <CheckBoxPreference
21                 android:key="parent_checkbox_preference"
22                 android:title="Parent Preference"
23                 android:summary="This is Parent Preference." />
24
25         <CheckBoxPreference
26                 android:key="child_checkbox_preference"
27                 android:dependency="parent_checkbox_preference"
28                 android:layout="?android:attr/preferenceLayoutChild"
29                 android:title="Child Preference"
30                 android:summary="This is Child Preference." />
31     </PreferenceCategory>
32
33 </PreferenceScreen>
```

Java ファイル

リスト 3 の Java ファイルでは，レイアウトファイルと同様に，XML ファイルを読み込んでいます (l.11)．この XML を読み込む作業だけで，アプリケーションのレイアウトと Preference を表示することができます．また，このリスト 3 では，Activity の代わりに “PreferenceActivity” を継承しています．この PreferenceActivity は，Activity のサブクラスであり，Preference に特

化した Activity です．このクラスは，11 行目の “addPreferncesFromResource()“メソッドなどの Preference を扱うメソッドをもっています．

この PreferencesActivity のように，なにかに特化した Activity が多く存在します．以後，この Activity のサブクラスもチュートリアルで紹介していきたいと思っています．

XML ファイル

このアプリケーションでは，XML ファイルがメインとなっています．この XML ファイルで，画面のレイアウトと Preference の両方を定義することができます．しかし，XML ファイルで定義できるレイアウトの形式は，各 Preference で，ほぼ決まっているので，利用する Preference や表示する文字などの変更しかしません．

このリスト 4 では，いくつかのタブがあります．タブについて説明していきたいと思います．

PreferenceScreen (l.2 -) このタブは，この XML が Preference であることを示しています．この行の URL は，この XML が Android のものであることを示しているので，PreferenceScreen を利用する時は，この記述のまま利用してください．図 1 では，画面全体の部分になります．

PreferenceCategory (l.4 -) このタブは，Preference をカテゴリに分けるために利用します．図 1 では，“inline prefernces” と書かれてある部分にあたり，このカテゴリ内には，“CheckBoxPreference” を含んでいます．“android:title” という属性で，このカテゴリが表示する文字を指定しています．

CheckBoxPreference (l.5 - l.8) このタブが実際にデータを保持する部分になる Preference になります．Preference には，いくつかの種類があり，後にも説明しますが，今回紹介するのは，CheckBox, EditText です．他にも種類がありますが，他の Preference の種類は，別の機会に紹介します．

CheckBoxPreference は，“android:key” で指定したキーに対し，boolean 型の値を持ちます．また，他の属性である，“android:title” は，図 1 の白地の大きい文字の部分を指定し，“android:summary” は，その下の白地の小さな文字を指定しています．

属性も，各 Preference に対して複数の種類がありますが，この属性の紹介も別の機会にいたします．

EditTextPreference (l.12 - l.16) この Preference では，String 型のデータを保持します．データの入力方法は，EditTextPreference の部分をクリックすると，別のダイアログが開き，テキストを入力することができます．そのダイアログに表示する文字を指定する属性が，“android:dialogTitle” です．

ダイアログを開いた時の画面は，図 2 のようになります．

dependency (l.19 - l.31) 次に説明するのは，タグではなく属性になります．ここでは，新しく 2 種類の属性が利用されてます．“android:dependency” は，依存する Prefernce を指定します．指定した Preference が True である時にのみ，この Preference が利用できるようになります．

“android:layout” は，この Preference のレイアウトを指定します．この属性でレイアウトを変更することができるのですが，滅多に利用しません．今回もこの属性を利用してもしなく

てもアプリケーションは，同じ動作をします．ここで説明するのは，ここで指定されている “?” の利用法についてです．この?は，リソースの参照を示すのですが，今まで利用してきた “@” とほとんど同じ文法です．

この二つの参照の違いは，参照する先になります．“@” では，自分の開発したアプリケーションで定義したリソースから参照しており，“?” では，Android が用意しているリソースから参照しています．

なので，この Preference で指定している layout は，元々利用している layout を明示しているだけなので，この属性があってもなくても変化しないのです．

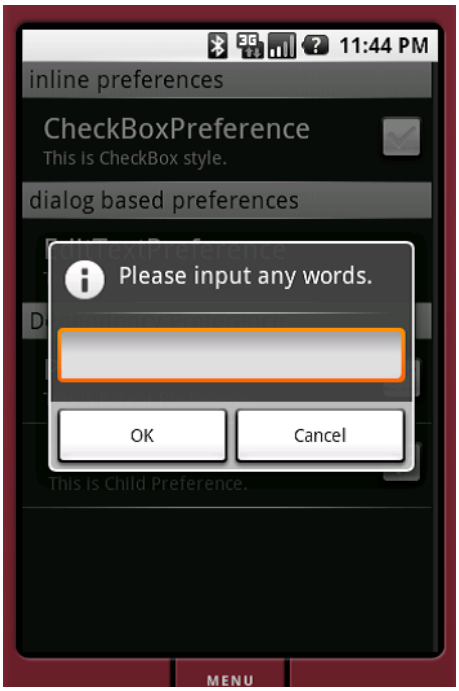

図 2: ダイアログを開いた画面

3 DataBaseについて

Android プラットフォームは，DataBase に SQLite[2] を使っています．また，Android では，DataBase を扱う API が提供されており，その API を使い DataBase を利用していきます．

SQLite を利用したアプリケーションのソースは以下のようになっています．

リスト 5: Java ファイル

```
1  package fuji.tutorial.sqlite;
2
3  import android.app.Activity;
4  import android.content.ContentValues;
5  import android.content.Context;
6  import android.content.Intent;
7  import android.database.sqlite.SQLiteCursor;
8  import android.database.sqlite.SQLiteDatabase;
9  import android.os.Bundle;
10 import android.view.Menu;
11 import android.view.MenuItem;
12 import android.widget.TextView;
13
14 public class Display extends Activity {
15     private static final int EDIT = Menu.FIRST;
16     private static final int ACTIVITY_EDIT = 0;
17     static final String DB_NAME = "file.db";
18     static final String TABLE_NAME = "text";
19     private TextView tv;
20
21     @Override
22     public void onCreate(Bundle savedInstanceState) {
23         super.onCreate(savedInstanceState);
24         setContentView(R.layout.display);
25         tv = (TextView) findViewById(R.id.text);
26         tv.setText(read());
27     }
28
29     @Override
30     public boolean onCreateOptionsMenu(Menu menu) {
31         super.onCreateOptionsMenu(menu);
32         menu.add(0, EDIT, 0, "EDIT");
33         return true;
34     }
35
36     @Override
37     public boolean onMenuItemSelected(int featureId, MenuItem item) {
38         switch (item.getItemId()) {
39         case EDIT:
40             edit();
41             return true;
42         }
43         return super.onMenuItemSelected(featureId, item);
44     }
45
46     private void edit() {
47         Intent i = new Intent(this, Edit.class);
48         startActivityForResult(i, ACTIVITY_EDIT);
49     }
50
51     @Override
52     public void onActivityResult(int requestCode, int resultCode, Intent intent){
53         super.onActivityResult(requestCode, resultCode, intent);
54
55         switch (requestCode) {
56         case ACTIVITY_EDIT:
57             tv.setText(read());
```

```
58              break;
59          }
60      }
61
62      private String read(){
63          SQLiteDatabase db = this.openOrCreateDatabase
64                                              (DB_NAME, Context.MODE_PRIVATE ,null);
65          db.execSQL("create table if not exists " + TABLE_NAME + "(info text)");
66
67          String info = "";
68
69          SQLiteCursor c =(SQLiteCursor) db.query
70               (TABLE_NAME, new String[] {"info"}, null, null, null, null, null);
71          if (c.getCount() > 0) {
72              c.moveToFirst();
73              info = c.getString(0);
74          } else {
75              ContentValues cv = new ContentValues();
76              cv.put("info", "");
77              db.insert(Display.TABLE_NAME, null, cv);
78          }
79
80          db.close();
81          return info;
82      }
83  }
```

リスト 6: XML ファイル

```
1   package fuji.tutorial.sqlite;
2
3   import android.app.Activity;
4   import android.content.ContentValues;
5   import android.content.Context;
6   import android.database.sqlite.SQLiteCursor;
7   import android.database.sqlite.SQLiteDatabase;
8   import android.os.Bundle;
9   import android.view.View;
10  import android.widget.Button;
11  import android.widget.EditText;
12
13  public class Edit extends Activity {
14      private EditText et;
15      private Button confirm;
16
17      @Override
18      protected void onCreate(Bundle savedInstanceState) {
19          super.onCreate(savedInstanceState);
20          setContentView(R.layout.edit);
21          et = (EditText) findViewById(R.id.edit);
22          confirm = (Button) findViewById(R.id.confirm);
23          et.setText(read());
24
25          confirm.setOnClickListener(new View.OnClickListener() {
26                  public void onClick(View view) {
27                      sendText();
28                  }
29              });
30      }
31
32      private void sendText() {
33          write(et.getText().toString());
34          setResult(RESULT_OK);
35          finish();
36      }
37
```

```
38      private String read() {
39          SQLiteDatabase db = this.openOrCreateDatabase
40                                  (Display.DB_NAME, Context.MODE_PRIVATE ,null);
41          String info = "";
42
43          SQLiteCursor c = (SQLiteCursor) db.query
44              (Display.TABLE_NAME, new String[] {"info"}, null, null,null,null,null);
45          if (c.getCount() > 0) {
46              c.moveToFirst();
47              info=c.getString(0);
48          }
49
50          db.close();
51          return info;
52      }
53
54      private void write(String text) {
55          SQLiteDatabase db = this.openOrCreateDatabase
56                                  (Display.DB_NAME, Context.MODE_PRIVATE, null);
57          ContentValues cv = new ContentValues();
58          cv.put("info", text);
59          db.update(Display.TABLE_NAME, cv, null, null);
60          db.close();
61      }
62  }
```

ここでは，ふたつのメソッドを追加しました．各メソッドについて解説していきます．

read()

このメソッドは，データベースからデータの取得を行っています．流れは，以下の通り．

l.63 - l.64 ここでは，データベース自体の取得と行っています．データベースが無い場合には，自動的に作成します．openOrCreateDatabase() では，データベース名とモードを指定しています．モードには，プライベート，書き込み可，読み込み可の 3 種類があります．今回は，このアプリケーションのみで利用するのでプライベートを指定しています．

l.65 execSQL() では，実際に SQLite のクエリを記述して実行します．ここでは,「取得したデータベースに指定したテーブルが無ければ，info と言うカラムを持つテーブルを作成する」というクエリを実行しています．

l.67 ここでは，取得したデータを保存する変数を定義しています．初期値は空文字列です．

l.69 データベースからデータを取得する場合には，“Cursor” というクラスを利用します．SQLiteCursor は，Cursor のサブクラスであり，SQLite では，このクラスを利用します．ここでは，テーブル名とカラム名を指定してデータの取得をしています．第一引数がテーブル名，第二引数がカラム名です．

l.71 - ここの if 文では，テーブルが作成されている時と無い時で処理を分けています．条件式では，フィールド “c” の個数で判断しています．1 つでもデータが取得できれば，72 行目に，1 つも取得できなければ，75 行目に進みます．

l.72 - l.73 ここでは，テーブルが存在するときに処理を行います．まず，72 行目で，取得したデータを先頭も位置に戻し，73 行目で文字列を取得しています．getString() の引数は，データの

インデックスを示しています．ここでは，データが 1 つのみなので，引数には 0 しか取れません．

l.75 - l.77 ここでは，テーブルが存在しないときの処理を行います．なので，テーブルの作成を行います．またデータを入力するときには，ContentValue というクラスを利用します．76 行目では，キーと値をセットしています．また，テーブルに新規データを入力するときには，insert() を利用します．

l.80 データベースを閉じます．データベースを利用したあとには，必ず閉じてください．開いたまま，別のとこで同じデータベースを開くとエラーが起こるので注意してください．

write()

ここでは，データベースにデータの保存をしています．基本的には，read() と利用しているクラスやメソッドは変わりないんですが，ここでは，新しく update() が使われています．

update() は，データを更新するときに利用するメソッドで，引数に渡したデータに書き換えます．ここでは，“info” というカラムにデータを保存しているので，info の値が書き換えられます．

4 File について

最後に，File でのデータ保存方法を紹介します．これは，おなじみのファイルでの入出力ですので，Java で利用したことのある人は，簡単な話だと思います．ここでも，今までと同じように，File を利用したアプリケーションに拡張して説明します．拡張したアプリケーションは，以下のようになります．

リスト 7: File を利用した Display

```
1  package fuji.tutorial.file;
2
3  import java.io.BufferedReader;
4  import java.io.FileInputStream;
5  import java.io.InputStreamReader;
6
7  import android.app.Activity;
8  import android.content.Intent;
9  import android.os.Bundle;
10 import android.view.Menu;
11 import android.view.MenuItem;
12 import android.widget.TextView;
13
14 public class Display extends Activity {
15         private static final int EDIT = Menu.FIRST;
16         private static final int ACTIVITY_EDIT = 0;
17         private TextView tv;
18
19     @Override
20     public void onCreate(Bundle savedInstanceState) {
21         super.onCreate(savedInstanceState);
22         setContentView(R.layout.display);
23         tv = (TextView) findViewById(R.id.text);
24         tv.setText(readFile());
25     }
26
27     @Override
28     public boolean onCreateOptionsMenu(Menu menu) {
```

```
29          super.onCreateOptionsMenu(menu);
30          menu.add(0, EDIT, 0, "EDIT");
31          return true;
32      }
33
34      @Override
35      public boolean onMenuItemSelected(int featureId, MenuItem item) {
36          switch (item.getItemId()) {
37          case EDIT:
38                  edit();
39                  return true;
40          }
41          return super.onMenuItemSelected(featureId, item);
42      }
43
44      private void edit() {
45          Intent i = new Intent(this, Edit.class);
46          startActivityForResult(i, ACTIVITY_EDIT);
47      }
48
49      @Override
50      public void onActivityResult(int requestCode, int resultCode, Intent intent) {
51          super.onActivityResult(requestCode, resultCode, intent);
52
53          switch (requestCode) {
54          case ACTIVITY_EDIT:
55                  tv.setText(readFile());
56                  break;
57          }
58      }
59
60      public String readFile() {
61          FileInputStream in = null;
62          String text = null;
63          String tmp = null;
64          try {
65                  in = this.openFileInput("data");
66                  InputStreamReader ir = new InputStreamReader(in);
67                  BufferedReader br = new BufferedReader(ir);
68
69                  text = br.readLine() + "\n";
70                          tmp = br.readLine();
71                  while (tmp != null){
72                          text = text + tmp + "\n";
73                          tmp = br.readLine();
74                  }
75                  if (text == "\n") {
76                          text = "";
77                  }
78
79                  in.close();
80                  ir.close();
81                  br.close();
82          } catch(Exception e) {
83          }
84          return text;
85      }
86  }
```

リスト 8: File を利用した Edit

```
1  package fuji.tutorial.file;
2
3
4  import java.io.BufferedReader;
5  import java.io.BufferedWriter;
6  import java.io.FileInputStream;
7  import java.io.FileOutputStream;
8  import java.io.InputStreamReader;
9  import java.io.OutputStreamWriter;
10
11 import android.app.Activity;
12 import android.content.Context;
13 import android.os.Bundle;
14 import android.view.View;
15 import android.widget.Button;
16 import android.widget.EditText;
17
18 public class Edit extends Activity {
19         EditText et;
20         Button confirm;
21
22         @Override
23         protected void onCreate(Bundle savedInstanceState) {
24                 super.onCreate(savedInstanceState);
25                 setContentView(R.layout.edit);
26                 et = (EditText) findViewById(R.id.edit);
27                 confirm = (Button) findViewById(R.id.confirm);
28
29                 et.setText(readFile());
30                 confirm.setOnClickListener(new View.OnClickListener() {
31                         public void onClick(View view) {
32                                 sendText();
33                         }
34                 });
35         }
36
37         private void sendText() {
38                 writeFile(et.getText().toString());
39                 setResult(RESULT_OK);
40                 finish();
41         }
42
43     public String readFile() {
44         FileInputStream in = null;
45         String text = null;
46         String tmp = null;
47         try {
48                 in = this.openFileInput("data");
49                 InputStreamReader ir = new InputStreamReader(in);
50                 BufferedReader br = new BufferedReader(ir);
51
52                 text = br.readLine() + "\n";
53                         tmp = br.readLine();
54                 while (tmp != null) {
55                         text = text + tmp + "\n";
56                         tmp = br.readLine();
57                 }
58
59                 if (text == "\n") {
60                         text = "";
61                 }
62
63                 in.close();
64                 ir.close();
65                 br.close();
66         } catch(Exception e) {
```

```
67          }
68          return text;
69      }
70
71      public void writeFile(String data) {
72          FileOutputStream out = null;
73          try{
74                  out = this.openFileOutput("data", Context.MODE_PRIVATE);
75                  OutputStreamWriter osw = new OutputStreamWriter(out);
76                  BufferedWriter bw = new BufferedWriter(osw);
77                  bw.write(data);
78
79                  bw.close();
80                  osw.close();
81                  out.close();
82          } catch(Exception e) {
83          }
84      }
85  }
```

Fileでの入出力で重要なのは，“openFileInput()“と “openFileOutput()“です．この二つは，FileInputStream と FileOutputStream を取得するメソッドで，この 2 種類の Stream を利用して入出力を行います．これらのメソッドは，Display.java では，65 行目に，Edit.java では，48 行目と 74 行目に出てきます．

5　保存したデータの確認方法

ここまでで，データの保存方法を 3 種類見てきましたが，実際にちゃんとデータが入っているか，確認したいものです．そこでこの節では，adb(Android Debug Bridge) を使った確認方法を紹介していきたいと思います．

5.1　adb とは

adb というのは，Android アプリケーションの開発を手助けするツールであり，自分の作ったアプリケーションをインストールしたエミュレータにアクセスすることのできるツールです．adb は，sdk と一緒に入ってます．adb 以外にも，開発を助けるツールがありますが，他のツールの紹介は，また機会があるときに．

adb でできることは，様々ですが，基本的には，以下のような使い方をします．

- アプリケーションのインストール，アンインストール
- エミュレータにアクセス
- デバッグ

5.2　エミュレータにアクセス

では，実際にエミュレータにアクセスしてみます．まず，エミュレータを起動した状態でコマンドプロンプトを開き，“adb shell” を実行します．

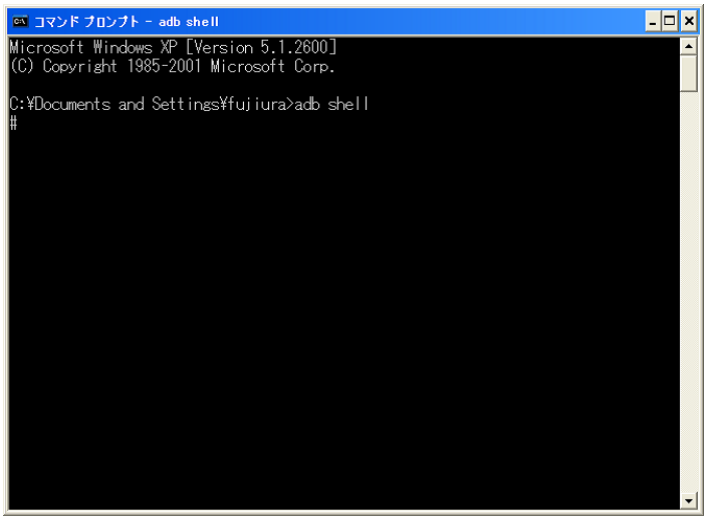

図 3: コマンドプロンプトを開く

先頭に “#” が出てきたら，成功です．うまくいかない場合は，sdk をインストールしたディレクトリの tools というディレクトリ内で実行するか，環境変数を設定してください．これで，エミュレータにアクセスでき，シェルを起動することができました．

Android では，カーネルに Linux を利用しているため，Linux 同様のコマンドで，ディレクトリの移動，ファイル操作を行うことができます．

とりあえず，“ls” すると，こんな感じに．

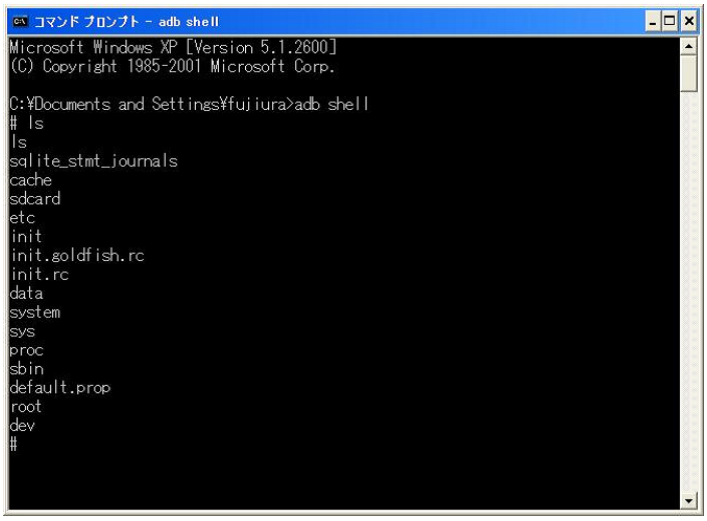

図 4: シェルで ls

このエミュレータの “/” には，15 個のファイル&ディレクトリがあります．この中でアプリケーションに関係するものとして，“data” というディレクトリがあります．“data” のディレクトリの中身は，こんな感じになってます．

今回使うのは，この中の “data” というディレクトリになります．この “/data/data/” には，各

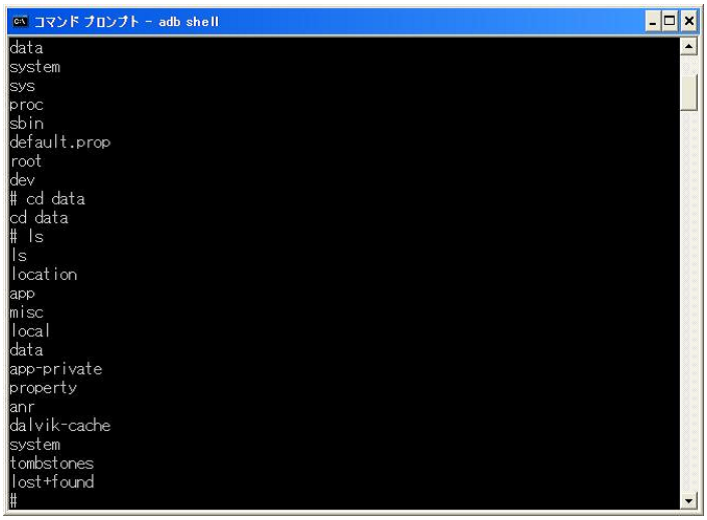

図 5: data ディレクトリの中身

アプリケーションが利用するデータを保存する先になり，今回のチュートリアルでアプリケーションが保存したデータは，全てここに置かれます．

また，“/data/app/” には，インストールしたアプリケーションが配置されます．この app ディレクトリ内のデータを削除することで，アンインストールが可能になります．

エミュレータ内で，アプリケーションを識別するための名前として，パッケージ名が利用されます．なので，開発したアプリケーションのパッケージ名と同じものを削除してください．

5.3 Preference の確認方法

では，実際に保存したデータの確認を行っていきたいと思います．この小節で利用するアプリケーションは，2 章で利用した，文字を入出力するアプリケーションの Preference 拡張です．このアプリケーションのパッケージ名は，“sample.pref” です．

Preference のデータは，エミュレータ内の “/data/data/パッケージ名/shared_prefs/” にあります．

このアプリケーションは，リスト 1 の 24 行目で，“Display” という Preference を読み込み&作成しています．なので，保存した文字は，“/data/data/sample.pref/Display.xml” に保存されています．Preference は，XML を利用してキーと値の保存を行っています．Display.xml となったのは，そのためです．

この Display.xml の中身をみるためには，cat というコマンドを利用します．

図 6: “my tutorial” を出力&保存

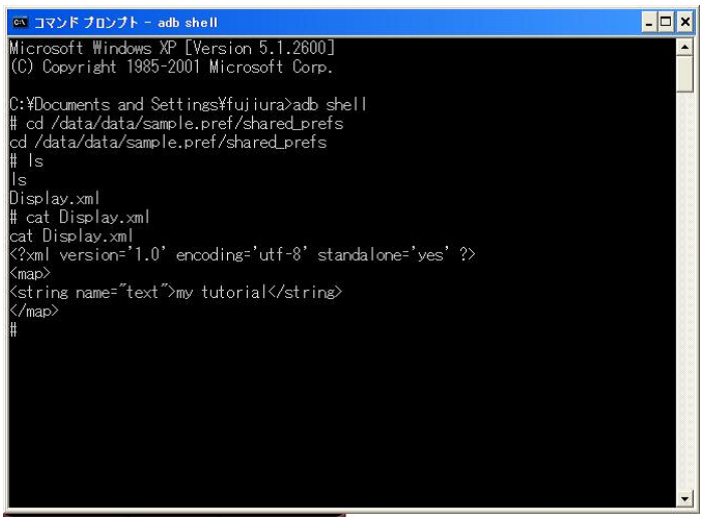

図 7: Display.xml の text タグに “my tutorial”

5.4 DataBase の確認方法

データベースでも，データの保存先は，似たようなところにあります．ここでは，3 章で扱ったアプリケーションで確認を行います．このアプリケーションのパッケージ名は，“fuji.tutorial.sqlite” です．また，データベースは，“databases” というディレクトリに保存されます．リスト 5 では，データベース名を “file.db” に，テーブル名を “text” に，カラムを “info” にしました．

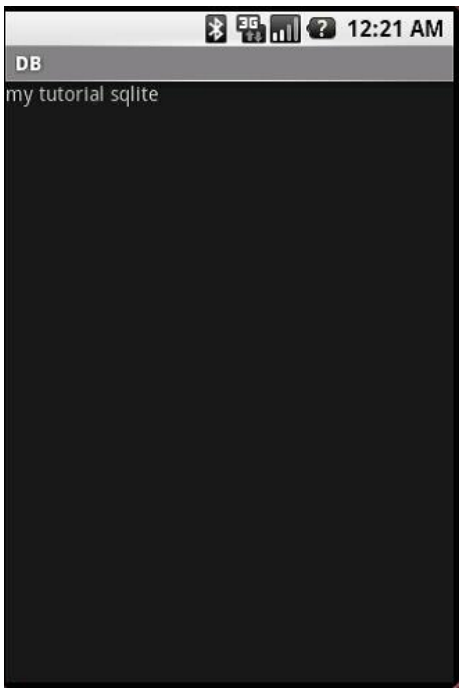

図 8: “my tutorial sqlite” を出力&保存

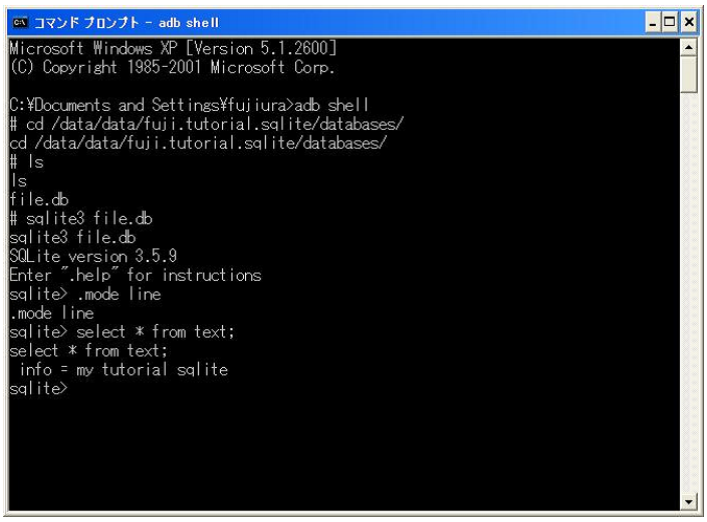

図 9: file.db の text テーブルの info カラムに “my tutorial sqlite”

データベースの確認を行う時には，sqlite3 というコマンドを利用します．sqlite3 は，シェルから SQL コマンドを実行することができます．図 9 では，2 つのコマンドを利用しています．

.mode line これは，出力モードを変更するコマンドです．line は出力を，“カラム名 = データ” という形式で出力します．

select * from text; select は，データを取得するコマンドです．これは，text というテーブルから全てのデータを取得して来ています．

この select を利用して得たデータを見てみると，カラム名 info に図 8 で表示されている文字が保存されていることが確認できます．

5.5 File の確認方法

File も他の二つと同様に，“/data/data/パッケージ名/” のディレクトリ内にある “Files” という，File を保存するディレクトリ内にあります．

ここでは，4 章で利用したアプリケーションを使って確認していきます．このアプリケーションのパッケージ名は，“fuji.tutorial.file” です．

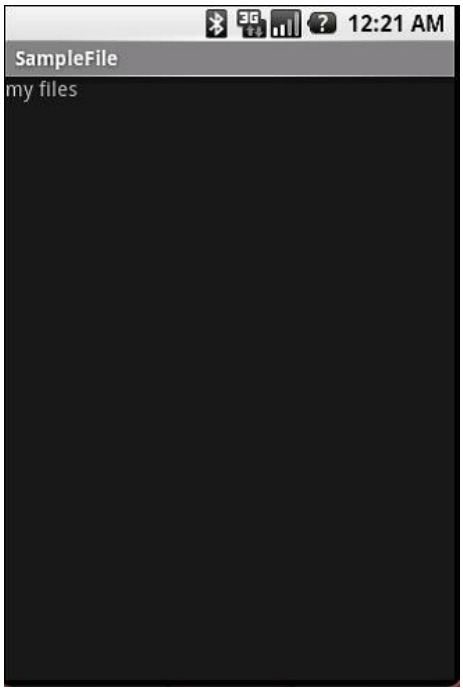

図 10: “my files” を出力&保存

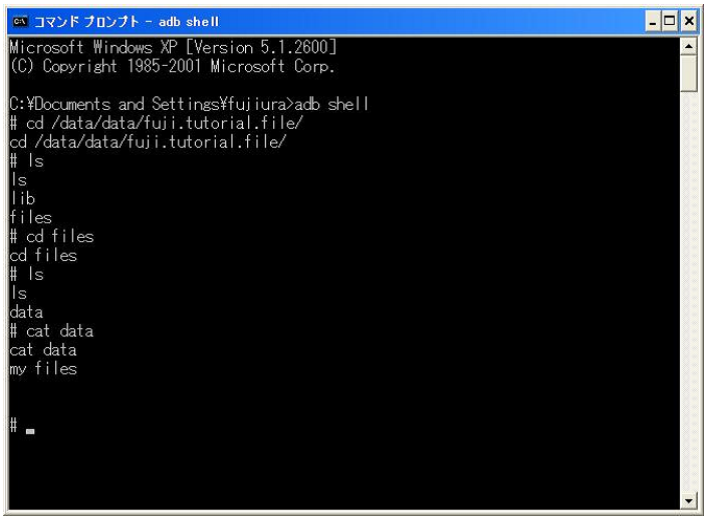

図 11: data ファイルに “my files”

6 まとめ

今回は，3 種類のデータの保存方法を紹介しました．これらのデータ保存先は，以下の表の通り．

	位置	コマンド
Preference	/data/data/パッケージ名/shared_pref/	cat
DataBase	/data/data/パッケージ名/databases/	sqlite3
File	/data/data/パッケージ名/files/	cat

表 1: データ保存先と表示方法

また，今回やったことは，こんな感じ．

- データの保存 (Preference, DataBase, File)
- adb について

- データの確認方法

ここまでで，簡単なアプリケーション作成とデータの扱い方についてチュートリアル形式で説明してきました．今回の adb のように，Android には，開発を手助けするツールがいくつもあります．その紹介や，そのほかの機能ついても紹介していきたいと思っております．

まだまだ，不十分なチュートリアルですが，これからも，変更，発行していきますので，よろしかったらお付き合いを．

参考文献

[1] Android SDK: `http://code.google.com/android/`

[2] SQLite: `http://www.sqlite.org/`

変更履歴

2008/11/05：Ver 1.0：公開開始