保存

データを保存する方法として，主に，3 種類ある．

[1] SharedPreference

[2] ファイル

[3] データベース

これらのうち，よく使いそうな，[1]と[2]の方法を説明する．

(1) SharedPreference

SharedPreference は，変数名とその値のペアをアプリごとに記憶しておくことができる機能である．たとえば，アプリを起動した回数を記憶しておいたり，前回起動した日時を覚えておいたりすることができる．さらに，Java のコレクションである Set を使って，一連のデータ列を記憶しておくこともできる．

SharedPreference を使用する例として，起動回数を保存する例を示す．

(a) MainActivityのフィールド

onCreate メソッド内で重複して利用する変数をリスト 1 に定義する．

リスト 1　MainActivity のフィールド

```
1  private static final String PREF_COUNT_KEY = "ExecuteCount";
2  SharedPreferences sharedPref;
```

(b) Preferenceの読み込みと書き込み

MainActivity の onCreate メソッドに，読み込みと書き込みをする処理を追加する．追加した後のコードをリスト 2 に示す．

リスト 2 MainActivity の onCreate メソッド

```
1  protected void onCreate(Bundle savedInstanceState)
2  {
3      super.onCreate(savedInstanceState);
4      setContentView(R.layout.activity_main);
5
6      //Preference を取得する
7      sharedPref = this.getPreferences(Context.MODE_PRIVATE);
8      //ExecuteCount として保存されている整数を取得する．存在しない場合は 0 を返す
9      int executeCount = sharedPref.getInt(PREF_COUNT_KEY, 0);
10     //今回の実行回数分を追加する
11     executeCount++;
12
13     //TextView を作って，表示する文字列を定義する
14     TextView tv =  new TextView(this);
15     tv.setTextSize(40);
16     tv.setText(executeCount + "回目です");
17
18     //TextView をレイアウト上に表示する
19     setContentView(tv);
20
21     //Preference を取得する
22     sharedPref = this.getPreferences(Context.MODE_PRIVATE);
23     //取得した Preference を編集するためのエディタを作る
24     SharedPreferences.Editor editor = sharedPref.edit();
25     //ExecuteCount として，実行した回数の数値を書き込む
```

```
26      editor.putInt(PREF_COUNT_KEY, executeCount);
27      //変更した Preference を確定させる
28      editor.commit();
29  }
```

(c) 実行の様子

実装して実行した様子を図 1 と図 2 に示す．一度実行するごとに，カウントが増えていく様子がわかる．

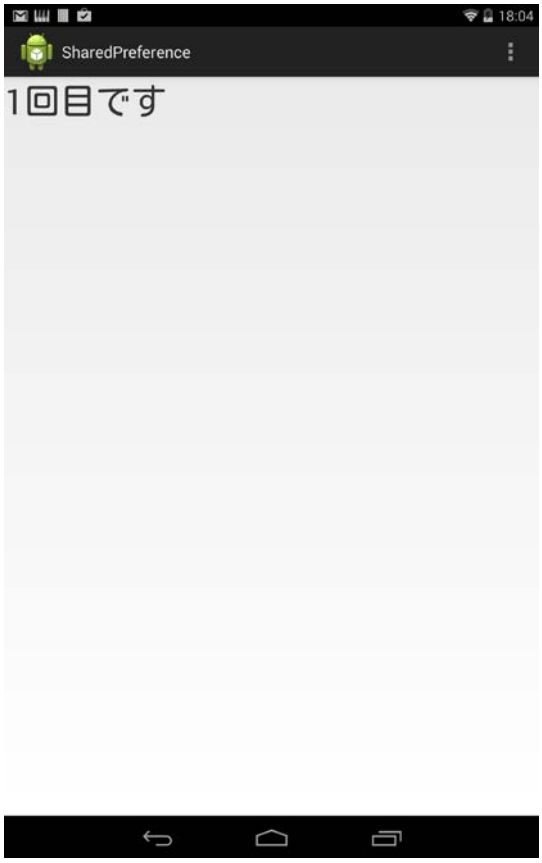


図 1　初回起動時の表示

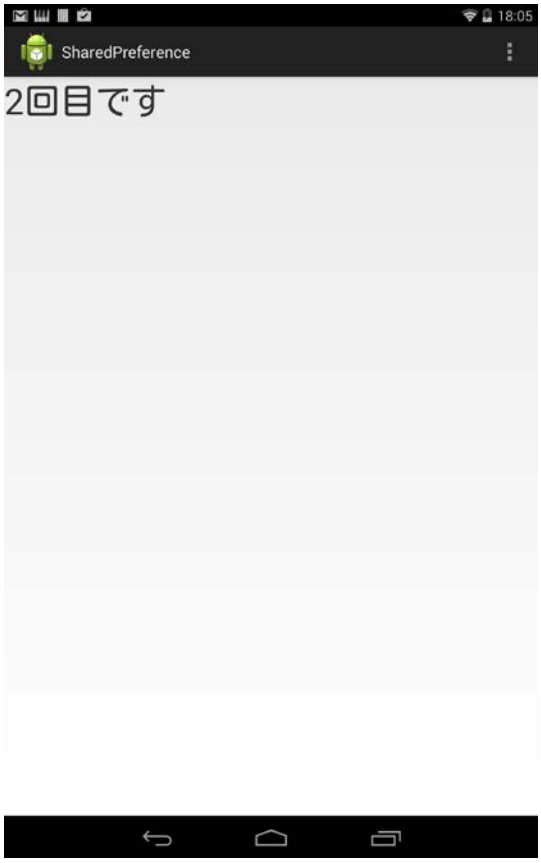


図 2　2 回目起動時の表示

(2) ストレージ

(a) ソースコードでの処理

音楽ファイルや写真ファイルを取得したいときには，リスト 3 のように処理する．

リスト 3　共有ディレクトリの取得とファイルリスト取得

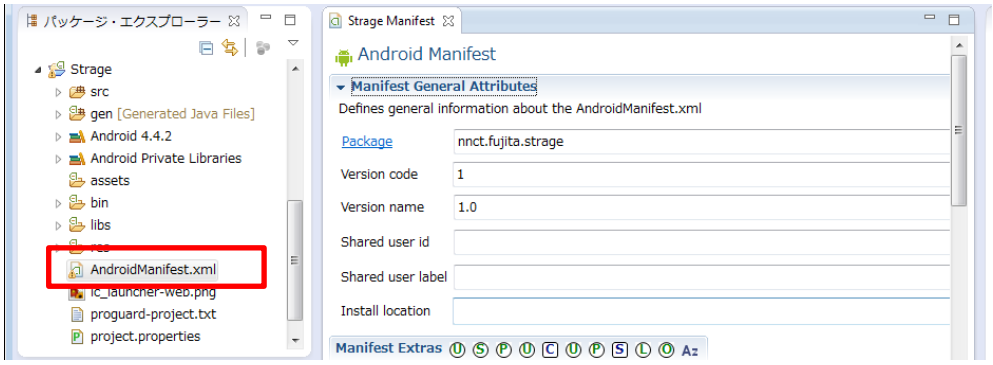
```
1   //共有ディレクトリの中から Music フォルダを取得する
2   File dir = Environment.
                    getExternalStoragePublicDirectory(Environment.DIRECTORY_MUSIC);
3   //Music フォルダ内のファイルリストを取得する
4   File files[] = dir.listFiles();
5
6   //レイアウト上の TextView に表示する
7   TextView tv = (TextView)findViewById(R.id.textView1);
8   for(inct i=0;i< files.length;i++)
9   {
10      tv.setText(tv.getText() + "¥n" +files[i].getPath());
11  }
```

注意として，フォルダにアクセスするには，アプリの設定として，パーミッションを設定する必要がある．

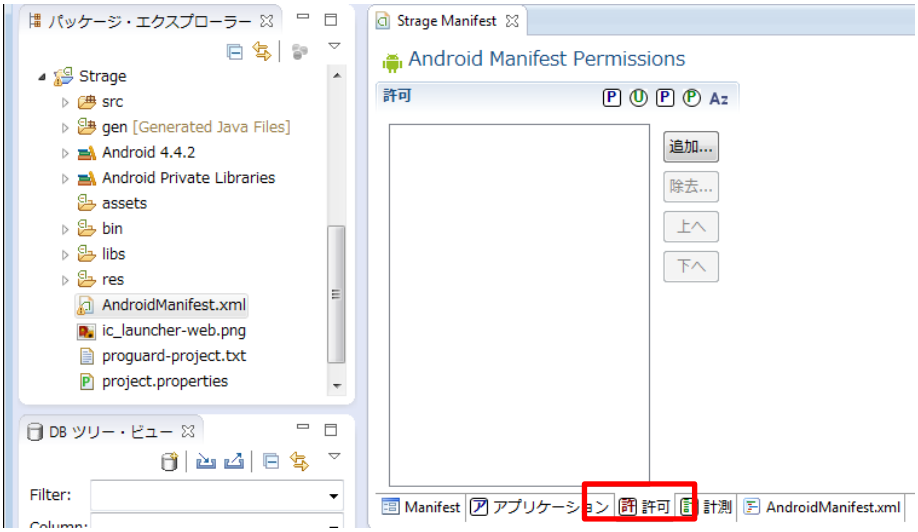
(b) ストレージにアクセスする許可

1） AndroidManifst.xml ファイルを開く

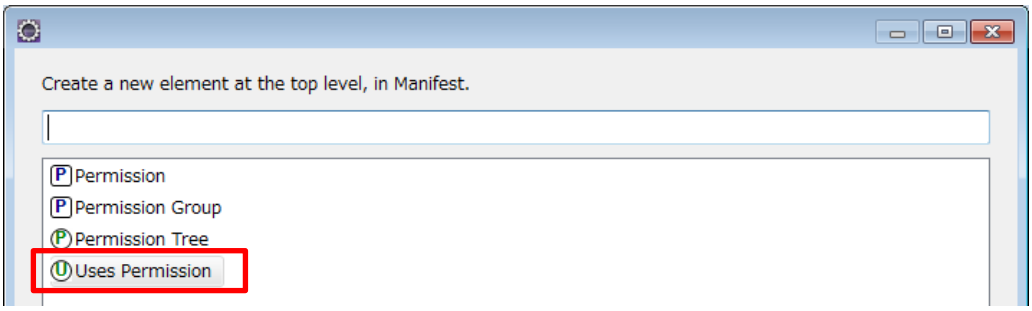
プロジェクトフォルダにある AndroidManifest.xml ファイルをダブルクリックする．



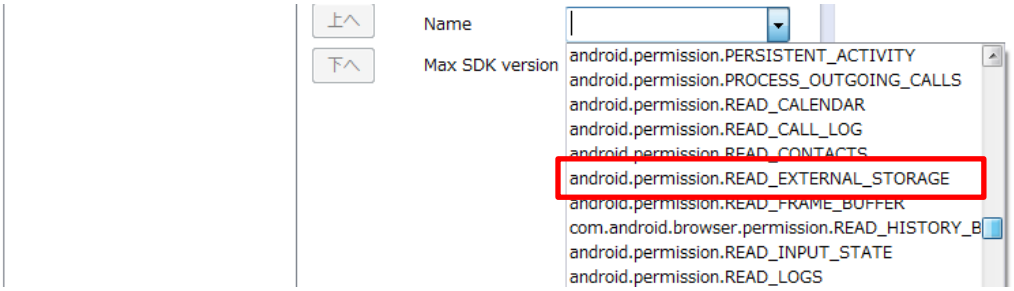
2） 「許可」のタブを開く



3） 追加する種類を選択する



4） 「Name」の選択タブから，「android.permission.READ_EXTERNAL_STRAGE」を選ぶ．



(c) 実行

(3) ファイル

そのアプリで使う設定ファイルやデータファイルを保存，読み込みする方法を示す．

アプリに関連するファイルを保存する場所は複数ある．それぞれ，適切な場所を選んで，利用する．

(a) アプリごとに与えられたデータフォルダ

getFilesDir()メソッドで得られるフォルダに保存する場合である．

ユーザが触れることができない場所にある．NEXUS7 では，具体的には，

/data/data/nnct.fujita.file/files

のようになる．「nnct.fujita.file」の部分はパッケージ名であるので，それぞれのアプリごとに異なるので，注意してほしい．この場所にアクセスするには，openFileInput()メソッドを利用すると便利である．

リスト 4　データファイル書き込み処理

```
 1  try
 2  {
 3      //ファイルに書き込む文字列を準備する
 4      String writeString = "string1¥nstring2";
 5
 6      // /data/data/nnct.fujita.file/files/data.txt を開く
 7      FileOutputStream fileOutputStream =
                                        openFileOutput("data.dat", MODE_PRIVATE);
 8      //ファイルストリームに文字列を書き込む
 9      fileOutputStream.write(writeString.getBytes());
10      //ファイルストリームを閉じる
11      fileOutputStream.close();
12  }
13  catch (FileNotFoundException e)
14  {
15      //ファイルがなかったときの例外処理
16  }
17  catch (IOException e)
18  {
19      // write できなかったときの例外処理
20  }
```

リスト 5　データファイル読み込み処理

```
 1  try
 2  {
 3      // /data/data/nnct.fujita.file/files/data.txt を読み込み用に開く
 4      FileInputStream fileInputStream = openFileInput("data.dat");
 5      //読み込んだファイル内容を格納するための配列をデータバイト分用意する
 6      byte readBytes[] = new byte[fileInputStream.available()];
 7      //ファイルストリームからファイルデータを読み込む
 8      fileInputStream.read(readBytes);
 9      //読み込んだデータを文字列に変換する
10      String readString = new String(readBytes);
11
12      //レイアウト上のテキストビューに表示する
13      TextView tv = (TextView)findViewById(R.id.textView1);
14      tv.setText(readString);
15  }
16  catch (FileNotFoundException e)
17  {
18      //ファイルがなかったときの例外処理
19  }
20  catch (IOException e)
21  {
22      //read できなかったときの例外処理
23  }
```

(b) ユーザのデータフォルダ

getExternalFilesDir ()メソッドで得られるフォルダに保存する場合である．ユーザおよび端末ごとに異なる可能性があるが，NEXUS7 で一人のみユーザがいる場合，

/storage/emulated/0/Android/data/nnct.fujita.file/files

となる．「nnct.fujita.file」の部分はパッケージ名であるので，それぞれのアプリごとに異なるので，注意してほしい．

このフォルダを利用する場合には，FileOutputStream クラスをインスタンス化する

ときに，ディレクトリとファイル名からなるファイルパスを指定する．ファイルストリームの操作については，openFileInput()メソッドを使うときと同様にできる．

リスト 6　ファイル書き込みの処理

```
 1 try
 2 {
 3     FileOutputStream fileOutStream = new FileOutputStream(dir5 + "/data.dat");
 4     String writeString = "string1¥nstring2¥nstring3";
 5     fileOutStream.write(writeString.getBytes());
 6     fileOutStream.close();
 7 }
 8 catch (FileNotFoundException e)
 9 {
10     e.printStackTrace();
11 }
12 catch (IOException e)//write の Exception
13 {
14     e.printStackTrace();
15 }
```

リスト 7　ファイル読み込み処理

```
 1 try
 2 {
 3     FileInputStream fileInputStream = new FileInputStream(dir5 + "/data.dat");
 4     //読み込んだファイル内容を格納するための配列をデータバイト分用意する
 5     byte readBytes[] = new byte[fileInputStream.available()];
 6     //ファイルストリームからファイルデータを読み込む
 7     fileInputStream.read(readBytes);
 8     //読み込んだデータを文字列に変換する
 9     String readString = new String(readBytes);
10
11     //レイアウト上のテキストビューに表示する
12     TextView tv = (TextView)findViewById(R.id.textView1);
13     tv.setText(readString);
14 }
15 catch (FileNotFoundException e)
16 {
17     e.printStackTrace();
18 }
19 catch (IOException e)//write の Exception
20 {
21     e.printStackTrace();
22 }
```