Java

Kotlin

Дизайн

Отладка

Open Source

Полезные ресурсы

Сохраняем настройки

Введение

Очень часто требуется сохранить какие-то настройки. Простой пример - девочка с удовольствием пользовалась нашей программой Счётчик ворон и вертела головой в поисках очередной птицы. Но вот незадача - когда девочка вечером закрывала своё приложение, то на следующий день

Впрочем, это уже совсем другая история.

Самый простой способ, который приходит в голову - сбросить данные в файл, а при запуске приложения считывать необходимые данные из файла. Второй вариант - работать с базой данных и хранить настройки там.

Рассмотрим сначала второй вариант. Хранить данные в базе данных не всегда оправдано, если данных не слишком много, они простые и нам не нужно анализировать данные на предмет, сколько мышек поймал котик в прошлом году и сколько часов он спал в январе и феврале.

Первый вариант с файлами хорош, например, для сохранения больших текстовых заметок. Естественно, и для простых данных мы тоже можем использовать файлы. Записали что-то в файл, а потом открыли его и считали данные.

На самом деле нет необходимости изобретать свой велосипед и придумывать свою структуру для хранения данных. В Android существует класс SharedPreferences (Общие настройки), разработанный специально для этих целей. Приложение автоматически создаёт файл в своей папке и хранит простые данные в виде «ключ — значение». Весь процесс создания, открытия, чтения файла оптимизирован и избавляет вас от головной боли.

Общие настройки поддерживают базовые типы **boolean**, **String**, **float**, **long** и **int**, что делает их идеальным средством для быстрого сохранения значений по умолчанию, переменных экземпляра класса, текущего состояния UI и пользовательских настроек. Они чаще всего используются для обеспечения постоянства данных между пользовательскими сессиями и доступа к ним компонентов приложения.

Сохранение значений параметров

Если у вас сохранился старый проект по подсчёту ворон, то можете снова его открыть и добавить новый код. Либо заново создайте проект по памяти, заодно проверите, как усвоили урок и сможете ли вы самостоятельно создать проект с нуля, не заглядывая на сайт за подсказкой.

Для удобства создадим константу для имени файла настроек, например:

```
// это будет именем файла настроек
public static final String APP_PREFERENCES = "mysettings";
```

Создадим параметр, который мы хотим сохранять в настройках. Нас интересуют показания счётчика.

Создаём переменную, представляющую экземпляр класса **SharedPreferences**, который отвечает за работу с настройками:

```
private SharedPreferences mSettings;
```

Внутри метода **onCreate()** вы инициализируете эту переменную::

```
mSettings = getSharedPreferences(APP_PREFERENCES, Context.MODE_PRIVATE);
```

Вы передаёте в указанный метод **getSharedPreferences()** название вашего файла (он будет создан автоматически) и стандартное разрешение, дающее доступ только компонентам приложения.

Немного опередим события и представим, что приложение запущено, и мы считаем ворон весь день. Когда мы закрываем приложение, то нам необходимо сохранить информацию в Общих настройках. Обычно для этих целей используют методы **onPause()** или **onStop()**.

Чтобы внести изменения в настройки, нужно использовать класс **SharedPreferences.Editor**. Получить объект **Editor** можно через вызов метода **edit()** объекта **SharedPreferences**. После того, как вы внесли все необходимые изменения, вызовите метод **apply()**, чтобы изменения вступили в силу.

```
@Override
protected void onPause() {
    super.onPause();
    // Запоминаем данные
    SharedPreferences.Editor editor = mSettings.edit();
    editor.putInt(APP_PREFERENCES_COUNTER, mCounter);
    editor.apply();
}
```

Теперь при закрытии программы значение счётчика автоматически запишется в файл. При повторном запуске приложения нам уже не нужно инициализировать счётчик со значением 0. Мы можем прочитать сохранённое значение и использовать его для счётчика, чтобы продолжить подсчёт. Сделаем это в методе **onResume()**.

```
protected void onResume() {
    super.onResume();

    if (mSettings.contains(APP_PREFERENCES_COUNTER)) {
        // Получаем число из настроек
        mCounter = mSettings.getInt(APP_PREFERENCES_COUNTER, 0);
        // Выводим на экран данные из настроек
        mInfoTextView.setText("Я насчитал "
                + mCounter + " ворон");
    }
}
```

Мы проверяем сначала наличие ключа **APP_PREFERENCES_COUNTER**, а затем извлекаем из ключа его значение.

Вот и всё. Небольшие изменения в коде сделали программу продвинутой. Теперь вы можете спокойно закрывать и открывать программу, ваши данные не будут потеряны. При желании вы можете добавить кнопку для сброса счётчика. Это вам в качестве домашнего задания.

В теории показаны дополнительные примеры и даны подробные сведения об использовании Общих настроек. Вам следует хорошенько разобраться в этом механизме, так как он часто используется на практике. Более того, некоторые программисты предпочитают использовать Общие настройки вместо базы данных, если это позволяет логика программы, так как это работает быстрее и потребляет меньше ресурсов. Выбор за вами.

Вместо послесловия

Не волнуйтесь, с котёнком всё в порядке. Девочка подобрала его и принесла домой. И добрая девочка по-прежнему пользуется нашей программой "Счётчик ворон". Наверное, биологом станет или ветеринаром.

При написании статьи использовались иллюстрации Рины З..

Исходный код

Разметка

```
<?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">

    <Button
        android:id="@+id/buttonCrowsCounter"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:onClick="onClick"
        android:text="Считаем ворон" />

    <TextView
        android:id="@+id/textViewInfo"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="" />

</LinearLayout>
```

Код класса активности

```
// и если нет, то не знаю, кто его писал.

package ru.alexanderklimov.counter;

import android.content.Context;
import android.content.SharedPreferences;
import android.support.v7.app.ActionBarActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.TextView;

public class MainActivity extends ActionBarActivity {

    private int mCounter;
    private TextView mInfoTextView;

    // имя файла настройки
    public static final String APP_PREFERENCES = "mysettings";
    public static final String APP_PREFERENCES_COUNTER = "counter";
    private SharedPreferences mSettings;

    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        setContentView(R.layout.activity_main);

        mSettings = getSharedPreferences(APP_PREFERENCES, Context.MODE_PRIVATE);
        mInfoTextView = (TextView) findViewById(R.id.textViewInfo);
    }

    public void onClick(View v) {
        // Выводим на экран
        mInfoTextView.setText("Я насчитал " + ++mCounter + " ворон");
    }

    @Override
    protected void onResume() {
        super.onResume();

        if (mSettings.contains(APP_PREFERENCES_COUNTER)) {
            // Получаем число из настроек
            mCounter = mSettings.getInt(APP_PREFERENCES_COUNTER, 0);
            // Выводим на экран данные из настроек
            mInfoTextView.setText("Я насчитал "
                    + mCounter + " ворон");
        }
    }
```

```
    protected void onPause() {
        super.onPause();
        // Запоминаем данные
        SharedPreferences.Editor editor = mSettings.edit();
        editor.putInt(APP_PREFERENCES_COUNTER, mCounter);
        editor.apply();
    }
}
```

Дополнительное чтение

Обсуждение урока на форуме

Реклама

Реклама