**Упражнение 1**

**к теме 7 Работа с файлами**

В Android есть несколько способов хранения данных:

**Preferences** - в качестве аналогии можно привести виндовые INI-файлы

**Oбычные файлы** (**Internal** и **Externel Storage**)- внутренние и внешние   
 (на SD карте)

**Assets** – файлы произвольных типов в ресурсах

**Ресурсы –** данные различных типов, зарегистрированные в проекте

[**Хранение данных Preferences.**](http://startandroid.ru/ru/uroki/vse-uroki-spiskom/73-urok-33-hranenie-dannyh-preferences.html)

  Мы начнем с самого простого – **Preferences**. Значения сохраняются в виде пары: **имя**, **значение**.

Разработаем приложение. В нем будет поле для ввода текста и две кнопки – **Save**и **Load**. По нажатию на **Save**мы будем **сохранять**значение из поля, по нажатию на **Load**– **загружать**.

Создадим проект.

Откроем **main.xml** и создадим такой экран:

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

<LinearLayout

    xmlns:android="http://schemas.android.com/apk/res/android"

    android:layout\_width="fill\_parent"

    android:layout\_height="fill\_parent"

    android:orientation="vertical">

        <EditText

            android:id="@+id/etText"

            android:layout\_width="match\_parent"

            android:layout\_height="wrap\_content">

                <requestFocus>

                </requestFocus>

        </EditText>

        <LinearLayout

            android:id="@+id/linearLayout1"

            android:layout\_width="match\_parent"

            android:layout\_height="wrap\_content"

            android:orientation="horizontal">

                <Button

                    android:id="@+id/btnSave"

                    android:layout\_width="wrap\_content"

                    android:layout\_height="wrap\_content"

                    android:text="Save">

                </Button>

                <Button

                    android:id="@+id/btnLoad"

                    android:layout\_width="wrap\_content"

                    android:layout\_height="wrap\_content"

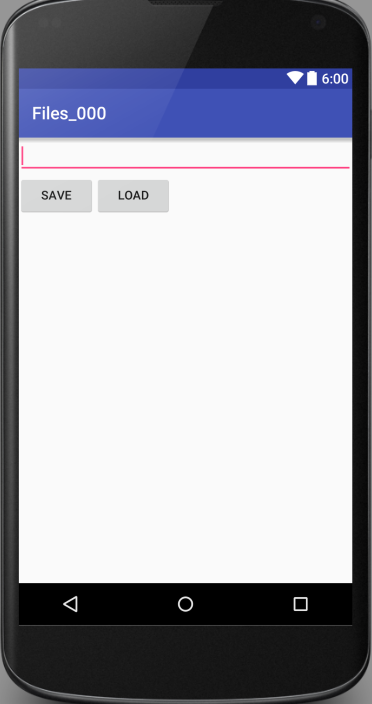
                    android:text="Load">

                </Button>

        </LinearLayout>

</LinearLayout>

Поле ввода и две кнопки.



Теперь пишем код в **MainActivity.java**:

package ru.startandroid.develop.p0331sharedpreferences;

import android.app.Activity;

import android.content.SharedPreferences;

import android.content.SharedPreferences.Editor;

import android.os.Bundle;

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 {

  EditText etText;

  Button btnSave, btnLoad;

  SharedPreferences sPref;

  final String SAVED\_TEXT = "saved\_text";

    /\*\* Called when the activity is first created. \*/

    @Override

    public void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);

        setContentView(R.layout.main);

        etText = (EditText) findViewById(R.id.etText);

        btnSave = (Button) findViewById(R.id.btnSave);

        btnSave.setOnClickListener(this);

        btnLoad = (Button) findViewById(R.id.btnLoad);

        btnLoad.setOnClickListener(this);

    }

  @Override

  public void onClick(View v) {

    switch (v.getId()) {

    case R.id.btnSave:

      saveText();

      break;

    case R.id.btnLoad:

      loadText();

      break;

    default:

      break;

    }

  }

  void saveText() {

    sPref = getPreferences(MODE\_PRIVATE);

    Editor ed = sPref.edit();

    ed.putString(SAVED\_TEXT, etText.getText().toString());

    ed.commit();

    Toast.makeText(this, "Text saved", Toast.LENGTH\_SHORT).show();

  }

  void loadText() {

    sPref = getPreferences(MODE\_PRIVATE);

    String savedText = sPref.getString(SAVED\_TEXT, "");

    etText.setText(savedText);

    Toast.makeText(this, "Text loaded", Toast.LENGTH\_SHORT).show();

  }

}

Определение элементов экрана, присвоение обработчиков и реализация onClick – тут все понятно и как обычно. Нам интересны **методы**, которые мы вызываем в onClick

Процедура **saveText**– сохранение данных.

Сначала с помощью метода [getPreferences](http://developer.android.com/reference/android/app/Activity.html" \l "getPreferences(int)" \t "_blank) получаем объект **sPref**класса **SharedPreferences**, который позволяет **работать**с **данными**(читать и писать). Константа [MODE\_PRIVATE](http://developer.android.com/reference/android/content/Context.html#MODE_PRIVATE) используется для настройки **доступа**и означает, что после сохранения, данные будут видны только **этому**приложению.

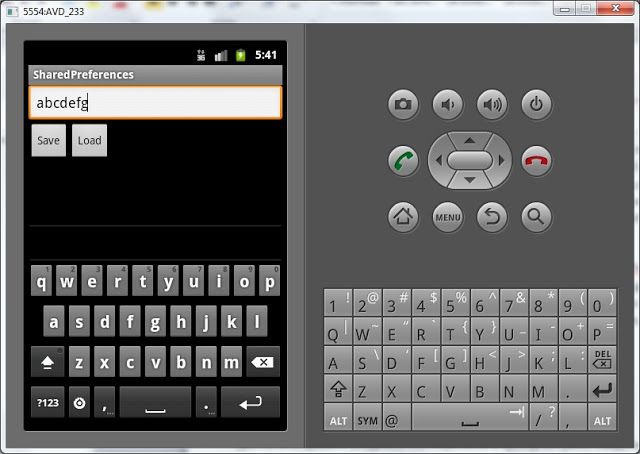
Далее, чтобы редактировать данные, необходим объект **Editor**– получаем его из sPref. В метод [putString](http://developer.android.com/reference/android/content/SharedPreferences.Editor.html" \l "putString(java.lang.String, java.lang.String)" \t "_blank) указываем **наименование**переменной – это константа **SAVED\_TEXT**, и **значение**– содержимое поля **etText**. Чтобы данные сохранились, необходимо выполнить [commit](http://developer.android.com/reference/android/content/SharedPreferences.Editor.html" \l "commit()" \t "_blank). И для наглядности выводим **сообщение**, что данные сохранены.

Процедура **loadText**– загрузка данных.

Так же, как и saveText, с помощью метода **getPreferences**получаем объект sPref класса **SharedPreferences**. **MODE\_PRIVATE**снова указывается, хотя и используется только при записи данных. Здесь Editor мы не используем, т.к. нас интересует только чтение данных. Читаем с помощью метода [getString](http://developer.android.com/reference/android/content/SharedPreferences.html" \l "getString(java.lang.String, java.lang.String)" \t "_blank) – в параметрах указываем константу - это **имя,**и **значение**по умолчанию (пустая строка). Далее пишем значение в поле ввода **etText**и выводим **сообщение**, что данные считаны.

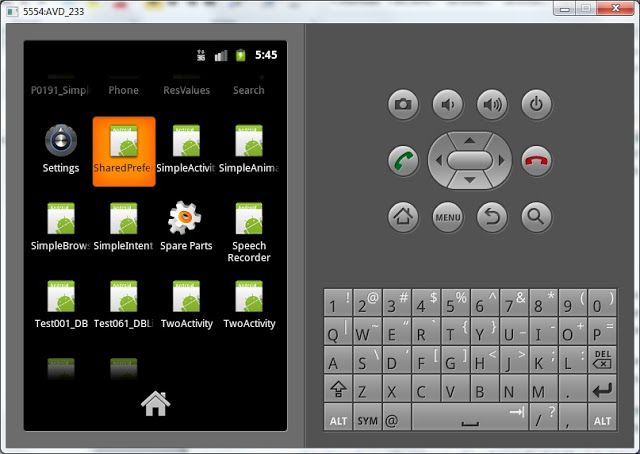
 Все сохраняем, запускаем приложение.

 Для начала, давайте убедимся, что сохранение в принципе нужно. Введите какой-нить текст в поле ввода



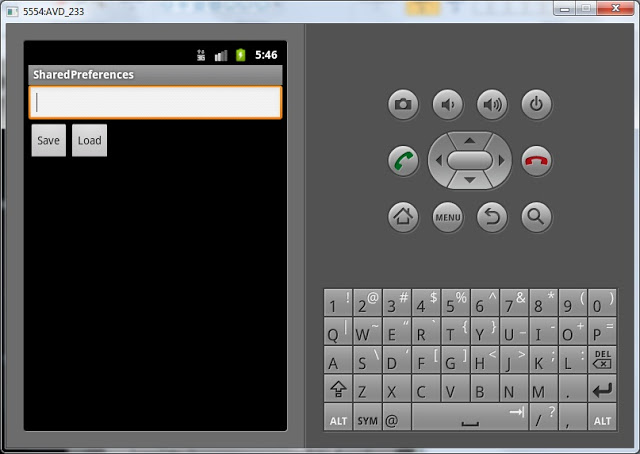
и не нажимая кнопку **Save**закройте приложение кнопкой **Назад**.

Теперь найдите приложение в общем списке приложений эмулятора

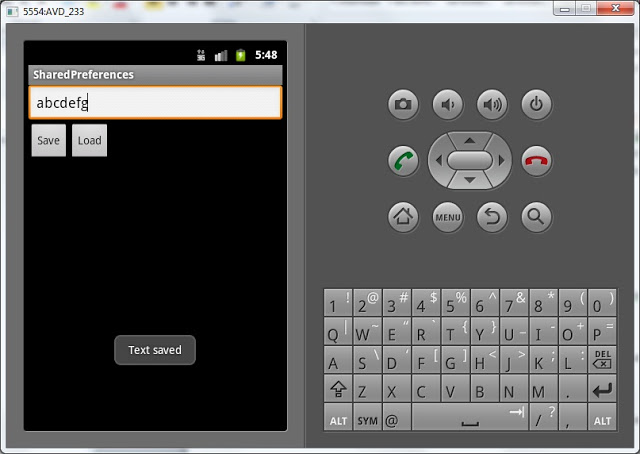


и запустите снова.

Поле ввода пустое. То, что мы вводили – пропало при закрытии программы. Нажатие на **Load**тоже ничего не даст – мы ничего не сохраняли.

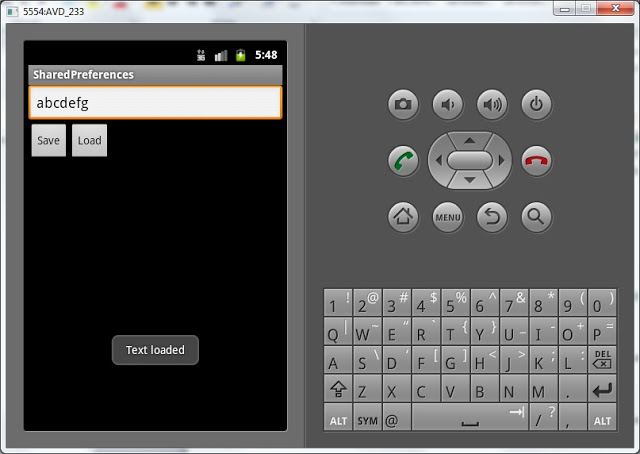


Давайте попробуем сохранять. Снова введите значение и нажмите **Save**.



Значение сохранилось в системе.

Теперь закроем приложение (**Назад**), снова откроем и нажмем **Load**. Значение считалось и отобразилось.



  Давайте сделаем так, чтобы сохранение и загрузка происходили **автоматически**при **закрытии**и **открытии**приложения и не надо было жать кнопки.  Для этого метод **loadText**будем вызывать в **onCreate**.

public void onCreate(Bundle savedInstanceState) {

    ...

    btnLoad = (Button) findViewById(R.id.btnLoad);

    btnLoad.setOnClickListener(this);

**loadText();**

}

(Добавляете только последнюю строку)

а метод saveText - в onDestroy

@Override

protected void onDestroy() {

  super.onDestroy();

  saveText();

}

Все сохраним, запустим. Теперь можно вводить данные, закрывать приложение, снова открывать и данные не потеряются. Кнопки **Save**и **Load**также работают. В какой момент сохранять данные в ваших приложениях – решать только вам. По **нажатию**кнопки, при **закрытии**программы или еще по какому-либо событию. Главное – теперь вы это умеете.

**Где хранятся данные?**

Preferences - данные сохраняются в файлы и вы можете посмотреть их. Для этого в Android Studio откройте меню **Tools**> Android> **Android Device Monitor**и выберите **Android**> **File Explorer**. Отобразилась файловая система эмулятора.   
Открываем папку

*data/data/ru.startandroid.develop.p0331sharedpreferences/shared\_prefs* 

и видим там файл **MainActivity.xml**. Если его выгрузить на комп и открыть - увидим следующее:

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

<map>

        <string name="saved\_text">abcdefg</string>

</map>

Все верно, имя - *saved\_text* и значение - *abcdefg*.

Обратите внимание, что в пути к файлу используется наш package: *ru.startandroid.develop.p0331sharedpreferences*.

Теперь разберемся, откуда взялось наименование файла **MainActivity.xml**. Кроме метода **getPreferences**, который мы использовали, есть метод [getSharedPreferences](http://developer.android.com/reference/android/content/ContextWrapper.html#getSharedPreferences(java.lang.String, int)). Он выполняет абсолютно те же функции, но позволяет указать имя файла для хранения данных. Т.е., например, если бы мы в **saveText**использовали для получение SharedPreferences такой код:

sPref = getSharedPreferences("MyPref", MODE\_PRIVATE);

То данные сохранились бы в файле **MyPref.xml**, а не в **MainActivity.xml**.

Теперь если мы посмотрим исходники метода **getPreferences**, то видим следующее:

public SharedPreferences getPreferences(int mode) {

return getSharedPreferences(getLocalClassName(), mode);

}

Используется метод **getSharedPreferences**, а в качестве имени файла берется имя класса текущего **Activity**. Отсюда и появилось имя файла **MainActivity.xml**.

В итоге:

- используете **getPreferences**, если работаете с данными для текущего Activity и не хотите выдумывать имя файла.

- используете **getSharedPreferences**, если сохраняете, например, данные - общие для нескольких Activity и сами выбираете имя файла для сохранения.

Кстати, в File Explorer вы можете видеть [юниксовые rwx-права доступа](http://younglinux.info/rwximg" \t "_blank) к файлу (приложение 2). Попробуйте при сохранении данных использовать не **MODE\_PRIVATE**, а [MODE\_WORLD\_READABLE](http://developer.android.com/reference/android/content/Context.html#MODE_WORLD_READABLE) или [MODE\_WORLD\_WRITEABLE](http://developer.android.com/reference/android/content/Context.html#MODE_WORLD_WRITEABLE) и посмотрите, как будут меняться права.

ПРИЛОЖЕНИЕ 1

Полный код **MainActivity.java**:

package ru.startandroid.develop.p0331sharedpreferences;

import android.app.Activity;

import android.content.SharedPreferences;

import android.content.SharedPreferences.Editor;

import android.os.Bundle;

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 {

  EditText etText;

  Button btnSave, btnLoad;

  SharedPreferences sPref;

  final String SAVED\_TEXT = "saved\_text";

    /\*\* Called when the activity is first created. \*/

    @Override

    public void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);

        setContentView(R.layout.main);

        etText = (EditText) findViewById(R.id.etText);

        btnSave = (Button) findViewById(R.id.btnSave);

        btnSave.setOnClickListener(this);

        btnLoad = (Button) findViewById(R.id.btnLoad);

        btnLoad.setOnClickListener(this);

        loadText();

    }

  @Override

  public void onClick(View v) {

    switch (v.getId()) {

    case R.id.btnSave:

      saveText();

      break;

    case R.id.btnLoad:

      loadText();

      break;

    default:

      break;

    }

  }

  void saveText() {

    sPref = getPreferences(MODE\_PRIVATE);

    Editor ed = sPref.edit();

    ed.putString(SAVED\_TEXT, etText.getText().toString());

    ed.commit();

    Toast.makeText(this, "Text saved", Toast.LENGTH\_SHORT).show();

  }

  void loadText() {

    sPref = getPreferences(MODE\_PRIVATE);

    String savedText = sPref.getString(SAVED\_TEXT, "");

    etText.setText(savedText);

    Toast.makeText(this, "Text loaded", Toast.LENGTH\_SHORT).show();

  }

  @Override

  protected void onDestroy() {

    super.onDestroy();

    saveText();

  }

}

ПРИЛОЖЕНИЕ 2

Права записываются одной строкой сразу для трёх типов пользователей:

* владельца файла (u);
* других пользователей, входящих в группу владельца (g);
* всех прочих пользователей (o);

В числовом виде файлу или каталогу устанавливаются абсолютные права , в то же время в символьном виде можно установить отдельные права для разных типов пользователей.

Пример: в числовом виде, установить права rwx-rx-rx:

chmod 755 filename

|  |  |  |  |
| --- | --- | --- | --- |
| **Пример — значение права «755»** | | | |
|  | **владелец** | **группа** | **остальные** |
| восьмеричное значение | 7 | 5 | 5 |
| символьная запись | rwx | r-x | r-x |
| обозначение типа пользователя | u | g | o |

Таким образом, права «755» записываются в символьном виде как «rwxr-xr-x». При этом для понимания сути задания прав в Unix-системах полезно знать представление чисел в [двоичной](https://ru.wikipedia.org/wiki/%D0%94%D0%B2%D0%BE%D0%B8%D1%87%D0%BD%D0%B0%D1%8F_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0_%D1%81%D1%87%D0%B8%D1%81%D0%BB%D0%B5%D0%BD%D0%B8%D1%8F) [системе счисления](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0_%D1%81%D1%87%D0%B8%D1%81%D0%BB%D0%B5%D0%BD%D0%B8%D1%8F).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Три варианта записи прав пользователя** | | | | |
| **двоичная** | **восьмеричная** | **символьная** | **права на файл** | **права на директорию** |
| 000 | 0 | --- | нет | нет |
| 001 | 1 | --x | выполнение | чтение файлов и их свойств |
| 010 | 2 | -w- | запись | нет |
| 011 | 3 | -wx | запись и выполнение | всё, кроме чтения списка файлов |
| 100 | 4 | r-- | чтение | чтение имён файлов |
| 101 | 5 | r-x | чтение и выполнение | доступ на чтение |
| 110 | 6 | rw- | чтение и запись | чтение имён файлов |
| 111 | 7 | rwx | все права | все права |

Часть разрешений имеет смысл только в сочетании с другими. Из первых четырёх пунктов (не дающих права на чтение файла) для файлов обычно используется только «---», то есть полный запрет доступа к файлу данному типу пользователей. Для директорий из всего списка обычно применяются только 0, 5 и 7 — запрет, чтение и выполнение, и полный доступ.

Суммировав эти коды для трёх типов пользователей, можно получить числовую или символьную запись. Например, **chmod 444 {имяфайла}**: 400+40+4=444 — все имеют право только на чтение (идентично «r--r--r--»).

Помимо стандартных разрешений 'rwx', команда chmod осуществляет также управление битами SGID, [SUID](https://ru.wikipedia.org/wiki/Suid) и T. Установленные атрибуты SUID или SGID позволяют запускать файл на выполнение с правами владельца файла или группы соответственно.

Для SUID вес — 4000, а для SGID — 2000. Данные атрибуты имеют смысл при установленном соответствующем бите исполнения и обозначаются при символьной записи буквой «s»: «rw**s**rwxrwx» и «rwxrw**s**rwx» соответственно.

Пример: **chmod 4555 {имяфайла}** — все имеют право на чтение и выполнение, но запускаться файл на исполнение будет с правами владельца.

Установка SGID для директории приведёт к установке принадлежности каждого нового создаваемого файла к той же группе, к которой принадлежит сама директория, а не к основной группе владельца, как это происходит по умолчанию. SUID для директории не имеет смысла[[1]](https://ru.wikipedia.org/wiki/Chmod#cite_note-pocket-1).

**sticky bit** или **restricted deletion flag** (t-бит) используется только с директориями. Когда t-бит для директории не установлен, файл в данной директории может удалить (переименовать) любой пользователь, имеющий доступ на запись к данному файлу. Устанавливая t-бит на директорию, мы меняем это правило таким образом, что удалить (переименовать) файл может только владелец этого файла. Следуя приведённой выше кодировке, t-бит имеет вес 1000.

Примечание: Право на запись (w) даёт пользователю возможность записывать или изменять файл, а право на запись для каталога — возможность создавать новые файлы или удалять файлы из этого каталога. Если на каталоге стоит возможность записи (w), то файл внутри этого каталога **можно будет удалить, даже если право на запись для него не установлено**. (В соответствии с концепцией файловой системы POSIX).

### Популярные значения

**400 (-r--------)**

Владелец имеет право чтения; никто другой не имеет права выполнять никакие действия

**644 (-rw-r--r--)**

Все пользователи имеют право чтения; владелец может редактировать

**660 (-rw-rw----)**

Владелец и группа могут читать и редактировать; остальные не имеют права выполнять никаких действий

**664 (-rw-rw-r--)**

Все пользователи имеют право чтения; владелец и группа могут редактировать

**666 (-rw-rw-rw-)**

Все пользователи могут читать и редактировать

**700 (-rwx------)**

Владелец может читать, записывать и запускать на выполнение; никто другой не имеет права выполнять никакие действия

**744 (-rwxr--r--)**

Каждый пользователь может читать, владелец имеет право редактировать и запускать на выполнение

**755 (-rwxr-xr-x)**

Каждый пользователь имеет право читать и запускать на выполнение; владелец может редактировать

**777 (-rwxrwxrwx)**

Каждый пользователь может читать, редактировать и запускать на выполнение

**1555 (-r-xr-xr-t)**

Каждый пользователь имеет право читать и запускать на выполнение; удалить файл может только владелец этого файла

**2555 (-r-xr-sr-x)**

Каждый пользователь имеет право читать и запускать на выполнение с правами группы(user group) владельца файла

**0440 (-r--r-----)**

Владелец и группа имеет право чтения никто другой не имеет права выполнять никакие действия

**4555 (-r-sr-xr-x)**

Каждый пользователь имеет право читать и запускать на выполнение с правами владельца файла