[Хранение данных. Preferences.](https://startandroid.ru/ru/uroki/vse-uroki-spiskom/73-urok-33-hranenie-dannyh-preferences.html" \o "Урок 33. Хранение данных. Preferences.)

В этом уроке:

- хранение данных с помощью Preferences

Хватит об Intent и Activity. Поговорим о хранении данных. В Android есть несколько способов хранения данных:

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

**SQLite** - база данных, таблицы

**обычные файлы** - внутренние и внешние (на SD карте)

Мы начнем с самого простого – **Preferences**. Значения сохраняются в виде пары: **имя**, **значение**. Так же, как и например extras в Intent.

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:

**Project name**: P0331\_SharedPreferences  
**Build Target**: Android 2.3.3  
**Application name**: SharedPreferences  
**Package name**: ru.startandroid.develop.p0331sharedpreferences  
**Create Activity**: MainActivity

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

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32 | <?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**:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66 | 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();    }  } |

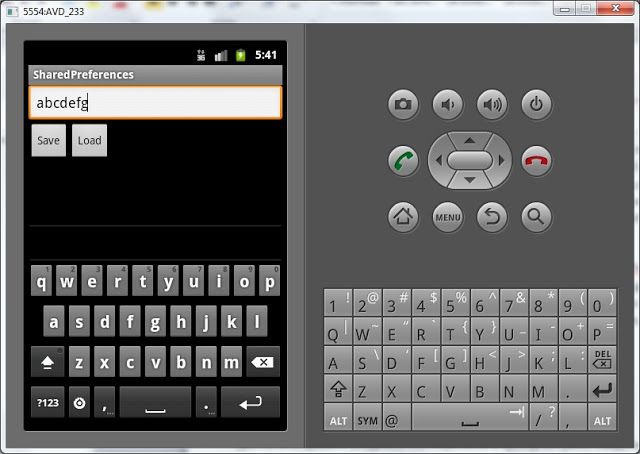
Определение элементов экрана, присвоение обработчиков и реализация onClick – тут все понятно и как обычно. Нам интересны **методы**, которые мы вызываем в onClick

**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,%20java.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.

**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,%20java.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.

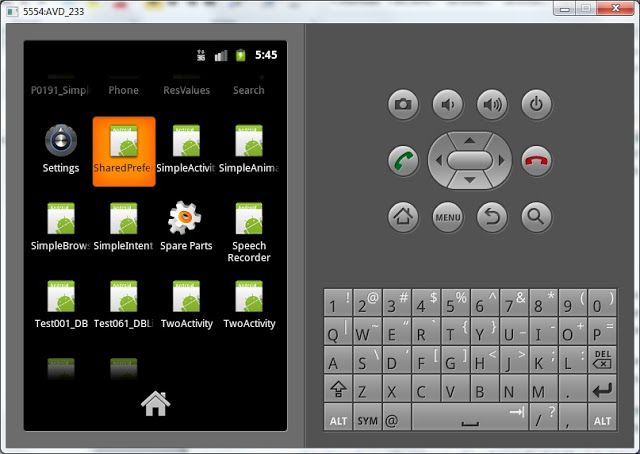
Все сохраняем, запускаем приложение.

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



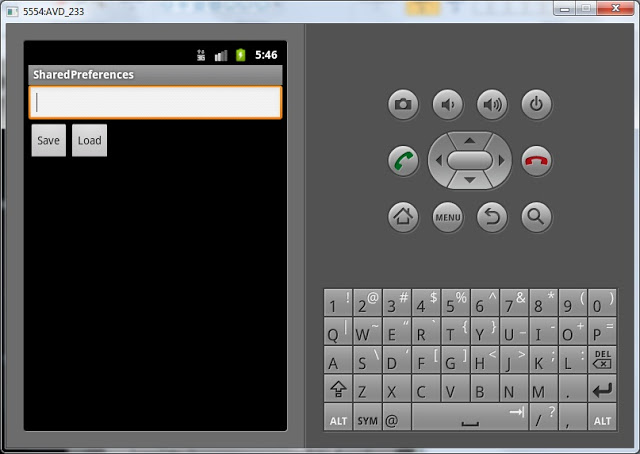
и не нажимая кнопку **Save**закройте приложение кнопкой **Назад**.

Теперь найдите приложение в общем списке приложений эмулятора

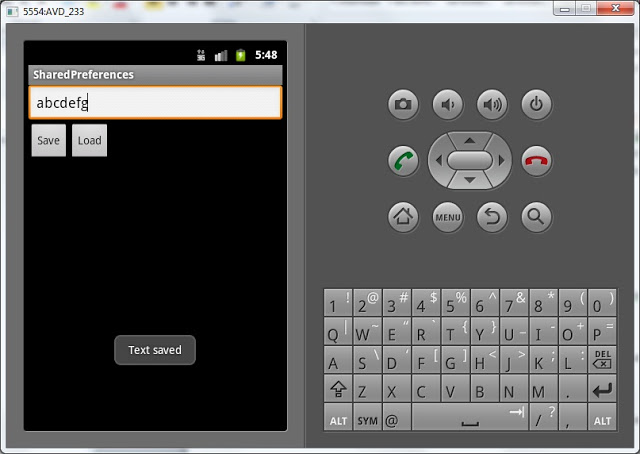


и запустите снова.

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

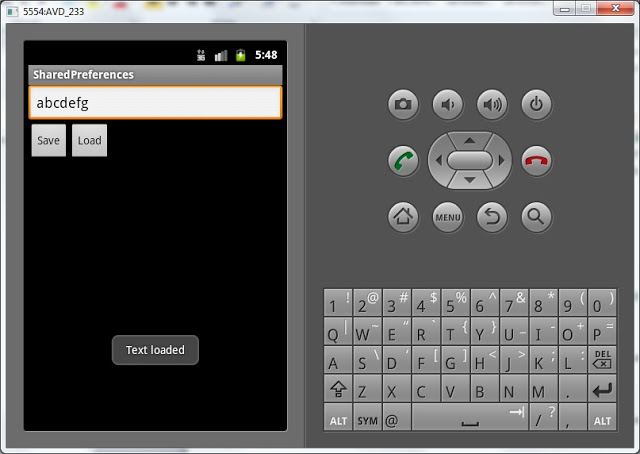


Давайте попробуем сохранять. Снова введите значение и нажмите **Save**.



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**.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | public void onCreate(Bundle savedInstanceState) {        ...        btnLoad = (Button) findViewById(R.id.btnLoad);      btnLoad.setOnClickListener(this);        loadText();  } |

(Добавляете только строку 8)

а метод saveText - в onDestroy

|  |  |
| --- | --- |
| 1  2  3  4  5 | @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.

Еще немного слов по этой теме.

Preferences-данные сохраняются в файлы и вы можете посмотреть их. Для этого в Eclipse откройте меню **Window**> **Show View**> **Other**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:

|  |  |
| --- | --- |
| 1  2  3  4 | <?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.

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" \l "getSharedPreferences(java.lang.String,%20int)" \t "_blank). 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:

|  |  |
| --- | --- |
| 1 | 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:

|  |  |
| --- | --- |
| 1  2  3 | 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. 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.

Полный код **MainActivity.java**:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75 | 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();    }    } |