**Сохранение данных при повороте экрана**

В одной из предыдущих тем был рассмотрен жизненный цикл Activity в приложении на Android, где после создания Activity вызывался метод **onRestoreInstanceState**, который восстанавливал ее состояние, а перед завершением работы вызывался метод **onSaveInstanceState**, который сохранял состояние Actiity. Оба этих метода в качестве параметра принимают объект **Bundle**, который как раз и хранит состояние activity:

|  |  |
| --- | --- |
| 1  2 | protected void onRestoreInstanceState(Bundle saveInstanceState);  protected void onSaveInstanceState(Bundle saveInstanceState); |

В какой ситуации могут быть уместно использование подобных методов? Банальная ситуация - переворот экрана и переход от портретной ориентации к альбомной и наоборот. Если, к примеру, графический интерфейс содержит текстовое поле для вывода TextView, и мы программно изменяем его текст, то после изменения ориентации экрана его текст может исчезнуть. Или если у нас глобальные переменные, то при изменении ориентации экрана их значения могут быть сброшены до значений по умолчанию.

Чтобы точнее понять проблему, с которой мы можем столкнуться, рассмотрим пример. Изменим файл **activity\_main** следующим образом:

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

    android:layout\_width="match\_parent"

    android:layout\_height="match\_parent"

    android:orientation="vertical">

    <EditText

        android:id="@+id/nameBox"

        android:layout\_width="match\_parent"

        android:layout\_height="wrap\_content"

        android:hint="Введите имя"/>

    <Button

        android:layout\_width="wrap\_content"

        android:layout\_height="wrap\_content"

        android:text="Сохранить"

        android:onClick="saveName"/>

    <TextView

        android:id="@+id/nameView"

        android:layout\_width="wrap\_content"

        android:layout\_height="wrap\_content"

        android:textSize="18dip"/>

    <Button

        android:layout\_width="wrap\_content"

        android:layout\_height="wrap\_content"

        android:text="Получить имя"

        android:onClick="getName"/>

</LinearLayout>

Здесь определено поле EditText, в которое вводим имя. И также определена кнопка для его сохранения.

Далее для вывода сохраненного имени предназначено поле TextView, а для получения сохраненного имени - вторая кнопка.

Теперь изменим класс **MainActivity**:

package com.example.eugene.viewsapplication;

import android.support.v7.app.AppCompatActivity;

import android.os.Bundle;

import android.view.View;

import android.widget.EditText;

import android.widget.TextView;

public class MainActivity extends AppCompatActivity {

    String name ="undefined";

    protected void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);

        setContentView(R.layout.activity\_main);

    }

    public void saveName(View view) {

        // получаем введенное имя

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

        name = nameBox.getText().toString();

    }

    public void getName(View view) {

        // получаем сохраненное имя

        TextView nameView = (TextView) findViewById(R.id.nameView);

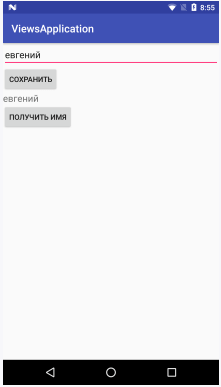
        nameView.setText(name);

    }

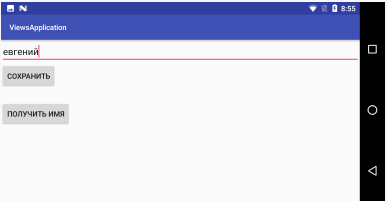
}

Для хранения имени в программе определена переменная name. При нажатии на первую кнопку сохраняем текст из EditText в переменную name, а при нажатии на вторую кнопку - обратно получаем текст из переменной name в поле TextView.

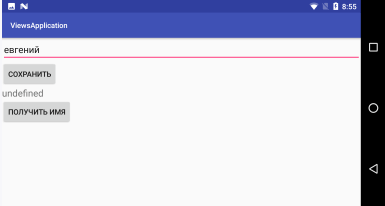
Запустим приложение введем какое-нибудь имя, сохраним и получим его в TextView:



Но если мы перейдем к альбомному режиму, то TextView окажется пустым, несмотря на то, что в него вроде бы уже получили нужное значение:



И даже если мы попробуем заново получить значение из переменной name, то мы увидим, что она обнулилась:



Чтобы избежать подобных ситуаций как раз и следует сохранять и восстанавливать состояние activity. Для этого изменим код MainActivity:

package com.example.eugene.viewsapplication;

import android.support.v7.app.AppCompatActivity;

import android.os.Bundle;

import android.view.View;

import android.widget.EditText;

import android.widget.TextView;

public class MainActivity extends AppCompatActivity {

    String name ="undefined";

    final static String nameVariableKey = "NAME\_VARIABLE";

    final static String textViewTexKey = "TEXTVIEW\_TEXT";

    protected void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);

        setContentView(R.layout.activity\_main);

    }

    // сохранение состояния

    @Override

    protected void onSaveInstanceState(Bundle outState) {

        outState.putString(nameVariableKey, name);

        TextView nameView = (TextView) findViewById(R.id.nameView);

        outState.putString(textViewTexKey, nameView.getText().toString());

        super.onSaveInstanceState(outState);

    }

    // получение ранее сохраненного состояния

    @Override

    protected void onRestoreInstanceState(Bundle savedInstanceState) {

        super.onRestoreInstanceState(savedInstanceState);

        name = savedInstanceState.getString(nameVariableKey);

        String textViewText= savedInstanceState.getString(textViewTexKey);

        TextView nameView = (TextView) findViewById(R.id.nameView);

        nameView.setText(textViewText);

    }

    public void saveName(View view) {

        // получаем введенное имя

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

        name = nameBox.getText().toString();

    }

    public void getName(View view) {

        // получаем сохраненное имя

        TextView nameView = (TextView) findViewById(R.id.nameView);

        nameView.setText(name);

    }

}

В методе onSaveInstanceState() сохраняем состояние. Для этого вызываем у параметра Bundle метод **putString(key, value)**, первый параметр которого - ключ, а второй - значение сохраняемых данных. В данном случае мы сохраняем две строки, поэтому вызываем метод putString(). Для сохранения объектов других типов данных мы можем вызвать соответствующий метод:

* **putInt()**
* **putDouble()**
* **putFloat()**
* **putChar()**
* **putByte()**

Каждый такой метод также в качестве первого параметра принимает ключа, а в качестве второго - значение.

В методе **onRestoreInstanceState** происходит обратный процесс - с помощью метода **getString(key)** по ключу получаем из сохраненного состояния строку по ключу. Соответственно для получения данных других типов мы можем использовать аналогичные методы:

* **getInt()**
* **getDouble()**
* **getFloat()**
* **getChar()**
* **getByte()**