

[RxJava \(http://developer.alexanderklimov.ru/android/rx/\)](http://developer.alexanderklimov.ru/android/rx/)

[Советы \(http://developer.alexanderklimov.ru/android/tips-android.php\)](http://developer.alexanderklimov.ru/android/tips-android.php)

[Статьи \(http://developer.alexanderklimov.ru/android/articles-android.php\)](http://developer.alexanderklimov.ru/android/articles-android.php)

[Книги \(http://developer.alexanderklimov.ru/android/books.php\)](http://developer.alexanderklimov.ru/android/books.php)

[Java \(http://developer.alexanderklimov.ru/android/java/java.php\)](http://developer.alexanderklimov.ru/android/java/java.php)

[Kotlin \(http://developer.alexanderklimov.ru/android/kotlin/\)](http://developer.alexanderklimov.ru/android/kotlin/)

[Дизайн \(http://developer.alexanderklimov.ru/android/design/\)](http://developer.alexanderklimov.ru/android/design/)

[Отладка \(http://developer.alexanderklimov.ru/android/debug/\)](http://developer.alexanderklimov.ru/android/debug/)

[Open Source \(http://developer.alexanderklimov.ru/android/opensource.php\)](http://developer.alexanderklimov.ru/android/opensource.php)

[Полезные ресурсы \(http://developer.alexanderklimov.ru/android/links.php\)](http://developer.alexanderklimov.ru/android/links.php)

Fragment (Фрагменты).

Часть пятая



Сохранение данных

Теперь рассмотрим важный вопрос, как сохранять данные при изменении конфигурации и других операциях.

Для примера возьмём старый урок по подсчёту ворон, но на этот раз будем считать котов. Вороны вечно порхают с ветки на ветку, их сложно считать. Коты - совсем другое дело, они лежат себе на одном месте и спят. Считать их одно удовольствие.



Кажется, я поторопился со своим утверждением.

Новый проект создавать не будем, а модифицируем старый. Заодно закрепим материал.

Создадим в проекте два новых фрагмента: **WithButtonFragment** и **WithTextViewFragment**. Из названий понятно, что в первом фрагменте будет кнопка, а во втором - **TextView**, в котором будет отражаться информация о количестве котов.

Повторяем прошлые шаги. Создаём новый класс, наследуемся от **Fragment**.

```
package ru.alexanderklimov.fragmentdemo;

import android.support.v4.app.Fragment;

public class WithButtonFragment extends Fragment {
}
```

Второй фрагмент.

```
package ru.alexanderklimov.fragmentdemo;

import android.support.v4.app.Fragment;

public class WithTextViewFragment extends Fragment {
}
```

Создадим две разметки для них. Для удобства будем использовать фон, чтобы различать фрагменты на экране.

fragment_with_button.xml

```
<?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:background="@android:color/holo_green_light"
    android:orientation="vertical">

    <Button
        android:id="@+id/button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center_horizontal"
        android:layout_margin="20dp"
        android:text="Считаем котов" />

</LinearLayout>
```

fragment_with_textview.xml

```
<?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:background="@android:color/holo_blue_light"
    android:orientation="vertical">

    <TextView
        android:id="@+id/textView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center_horizontal"
        android:layout_margin="20dp"
        android:text="Всего:"
        android:textAppearance="?android:attr/textAppearanceLarge" />

</LinearLayout>
```

Подключаем разметки к соответствующим фрагментам в методе **onCreateView()**.

```

@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) {
    return inflater.inflate(R.layout.fragment_with_button, container, false);
}

```

Для второго фрагмента напишите код самостоятельно.

Подключаем фрагменты в разметке основной активности **activity_main.xml**:

```

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/container"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:gravity="center_horizontal"
    android:orientation="vertical"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context=".MainActivity">

    <fragment
        android:id="@+id/fragment_withButton"
        android:name="ru.alexanderklimov.fragmentdemo.WithButtonFragment"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_gravity="center_horizontal"
        tools:layout="@layout/fragment_with_button" />

    <fragment
        android:id="@+id/fragment_withTextView"
        android:name="ru.alexanderklimov.fragmentdemo.WithTextViewFragment"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_gravity="center_horizontal"
        tools:layout="@layout/fragment_with_textview" />
</LinearLayout>

```

Не копируйте код с сайта, проделывайте операции самостоятельно через редактор в режиме **Design** и **Text**, как это объяснялось в предыдущих примерах.

Подключим кнопку в первом фрагменте.

```

package ru.alexanderklimov.fragmentdemo;

import android.os.Bundle;
import android.support.v4.app.Fragment;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;

public class WithButtonFragment extends Fragment implements View.OnClickListener {
    private int mCounter = 0; // счётчик котов

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) {
        View rootView =
            inflater.inflate(R.layout.fragment_with_button, container, false);
        Button button = (Button)rootView.findViewById(R.id.button);
        button.setOnClickListener(this);
        return rootView;
    }

    @Override
    public void onClick(View v) {
        mCounter++;
    }
}

```

Во втором фрагменте подключим компонент **TextView** и добавим метод для вывода текста.

```
package ru.alexanderklimov.fragmentdemo;

import android.os.Bundle;
import android.support.v4.app.Fragment;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.TextView;

public class WithTextViewFragment extends Fragment {
    private TextView mTextView;

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) {
        View rootView =
            inflater.inflate(R.layout.fragment_with_textview, container, false);
        mTextView = (TextView) rootView.findViewById(R.id.textview);
        return rootView;
    }

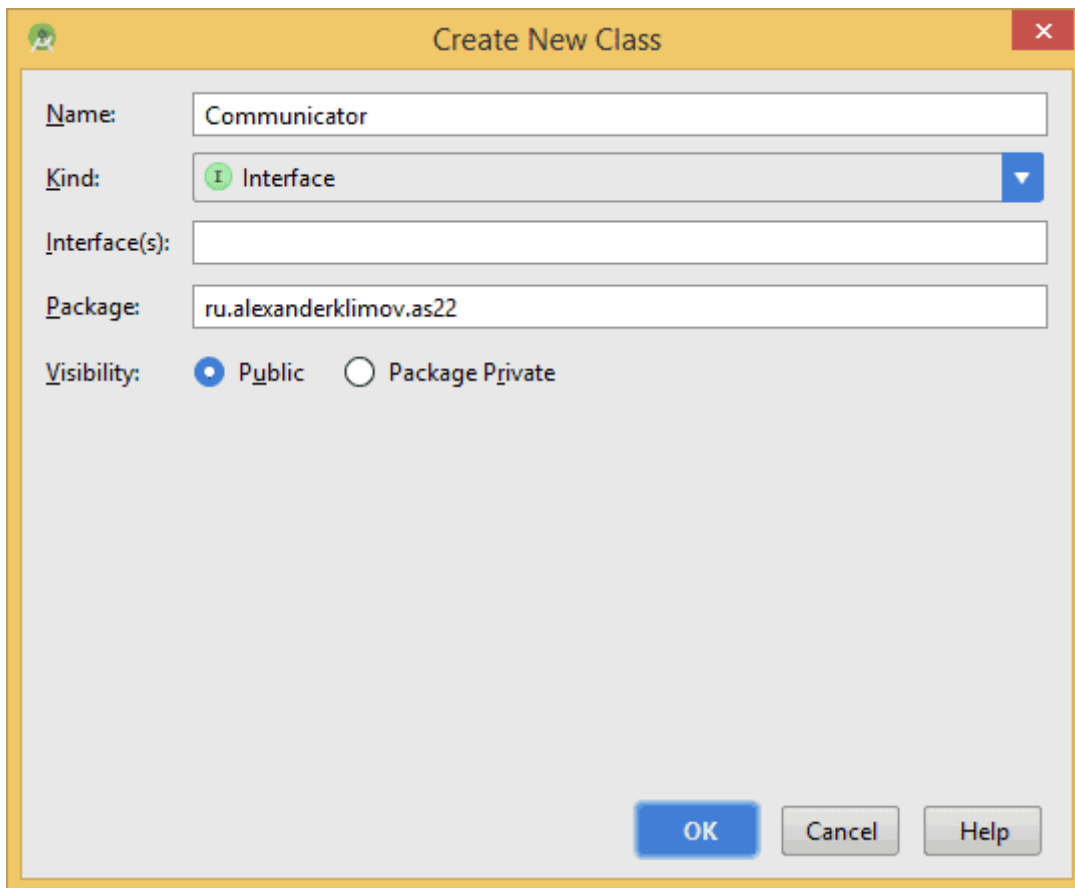
    public void changeText(String data) {
        mTextView.setText(data);
    }
}
```

Если мы вызовем метод **changeText()** с каким-нибудь текстом, то текст отобразится в **TextView**. Наша задача - научить фрагменты обмениваться данными через активность.

Если сейчас запустить пример, то фрагменты будут работать независимо друг от друга. Счётчик будет увеличиваться, но его значения пока не попадают во второй фрагмент и мы не можем увидеть число котов.

Напоминаю, мы не связываем два фрагмента между собой напрямую. Создаём интерфейс для этих целей.

В предыдущем примере мы создавали интерфейс внутри первого фрагмента. Для разнообразия изменим подход. Создадим новый класс через **New | Java Class** и в диалоговом окне для поля **Kind** выберем **Interface**. Также зададим ему имя **Communicator**.



Create New Class

Name: Communicator

Kind: Interface

Interface(s):

Package: ru.alexanderklimov.as22

Visibility: ☒ Public ☐ Package Private

OK Cancel Help

Интерфейс будет состоять из одного метода.

```
package ru.alexanderklimov.fragmentdemo;

public interface Communicator {
    public void count(String data);
}
```

Активность должна реализовать интерфейс.

```
public class MainActivity extends ActionBarActivity implements Communicator {
    @Override
    public void count(String data) {

    }
}
```

Фрагмент с кнопкой может использовать объект интерфейса для отправки данных. Инициализируем его в новом методе **onActivityCreated()**, который ранее не использовали. Метод сработает, когда активность будет создана и готова к работе.

```

package ru.alexanderklimov.fragmentdemo;

import android.os.Bundle;
import android.support.v4.app.Fragment;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;

public class WithButtonFragment extends Fragment implements View.OnClickListener {
    private int mCounter = 0; // счётчик котов
    private Communicator mCommunicator;

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        View rootView =
            inflater.inflate(R.layout.fragment_with_button, container, false);
        Button button = (Button)rootView.findViewById(R.id.button);
        button.setOnClickListener(this);
        return rootView;
    }

    @Override
    public void onActivityCreated(Bundle savedInstanceState) {
        super.onActivityCreated(savedInstanceState);
        mCommunicator = (Communicator)getActivity();
    }

    @Override
    public void onClick(View v) {
        mCounter++;
        mCommunicator.count("Я насчитал " + mCounter + " котов");
    }
}

```

При щелчках на кнопках мы вызываем метод **count()**, которому передаём нужную информацию.

А сам метод в активности будет выглядеть следующим образом:

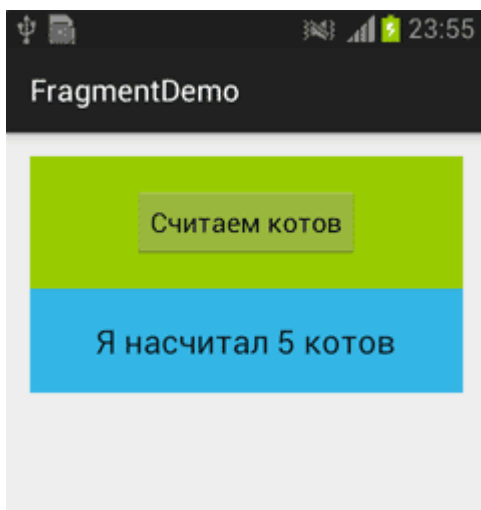
```

@Override
public void count(String data) {
    FragmentManager manager = getSupportFragmentManager();
    WithTextViewFragment withTextViewFragment =
        (WithTextViewFragment) manager.findFragmentById(R.id.fragment_withTextView);
    withTextViewFragment.changeText(data);
}

```

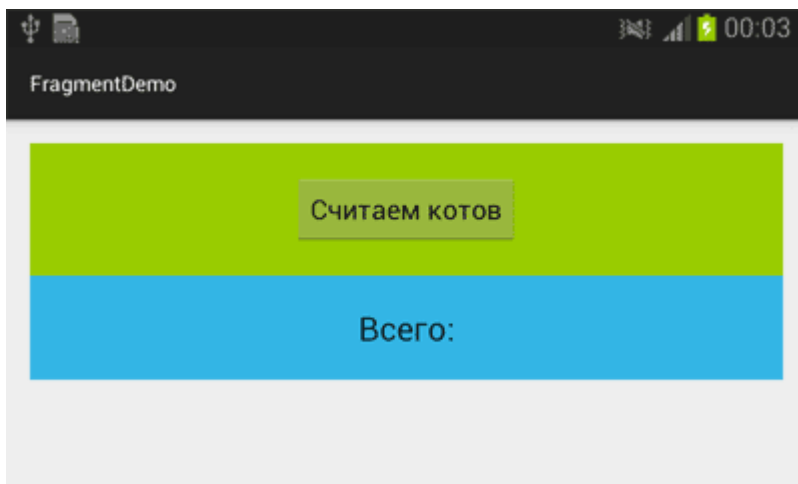
Фрагмент передаёт через метод **count()** данные **data**, а активность их принимает и передаёт их в метод второго фрагмента **changeText()**.

Подготовительные работы завершены и можно запустить пример для проверки.



Если вы создавали пример на основе предыдущих уроков, то не поворачивайте экран. Сначала удалите (или переименуйте) файлы для альбомной ориентации, чтобы не получить ошибку и крах приложения.

Теперь переходим непосредственно к теме нашего урока. Повернув экран, мы обнаружим, что данные из текстового блока пропали. Наши подсчёты - коту под хвост!



Мы можем в новой ориентации начать новый подсчёт, но повернув устройство обратно в портретный режим, снова потеряем данные.

Мы знаем, что при поворотах активность создаётся заново. Поэтому все данные сбрасываются. Чтобы сохранить данные, у фрагментов есть соответствующие методы, схожие с подобными методами у активностей. Задействуем их.

Метод **onSaveInstanceState()** поможет нам. Добавим метод в первый фрагмент с кнопкой.

```
@Override
public void onSaveInstanceState(Bundle outState) {
    super.onSaveInstanceState(outState);
    outState.putInt("counter", mCounter);
}
```

Параметр метода **outState** является объектом класса **Bundle** и позволяет хранить различные типы в формате "ключ-значение".

У фрагментов также есть метод **onCreate(Bundle savedInstanceState)**, где используется объект этого же класса **Bundle** только под другим именем **savedInstanceState**. Несмотря на разные имена, речь идёт об одном и том же объекте. И сохраняя данные в методе **onSaveInstanceState()**, мы можем их получить в методе **onCreate()**:

```
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    if (savedInstanceState == null) {
        mCounter = 0;
    } else {
        mCounter = savedInstanceState.getInt("counter", 0);
    }
}
```

При повороте фрагмент сохранит значение счётчика, перезапустится и восстановит значение счётчика. Если запустить пример, то увидим следующее. Щёлкнем несколько раз по кнопке и повернём экран. Данные сбросятся и мы снова увидим пустой текст. Но стоит нам нажать на кнопку, то увидим, что отсчёт пошёл не сначала, а продолжил со своего последнего значения. Мы видим, что первый фрагмент запоминает свои данные. А второй фрагмент пока тупит. Поможем ему.

```

package ru.alexanderklimov.fragmentdemo;

import android.os.Bundle;
import android.support.v4.app.Fragment;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.TextView;

public class WithTextViewFragment extends Fragment {
    private TextView mTextView;
    private String mData;

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) {
        View rootView =
            inflater.inflate(R.layout.fragment_with_textview, container, false);
        mTextView = (TextView) rootView.findViewById(R.id.textview);

        if(savedInstanceState == null){

        }else {
            mData = savedInstanceState.getString("text");
            mTextView.setText(mData);
        }

        return rootView;
    }

    @Override
    public void onSaveInstanceState(Bundle outState) {
        super.onSaveInstanceState(outState);
        outState.putString("text", mData);
    }

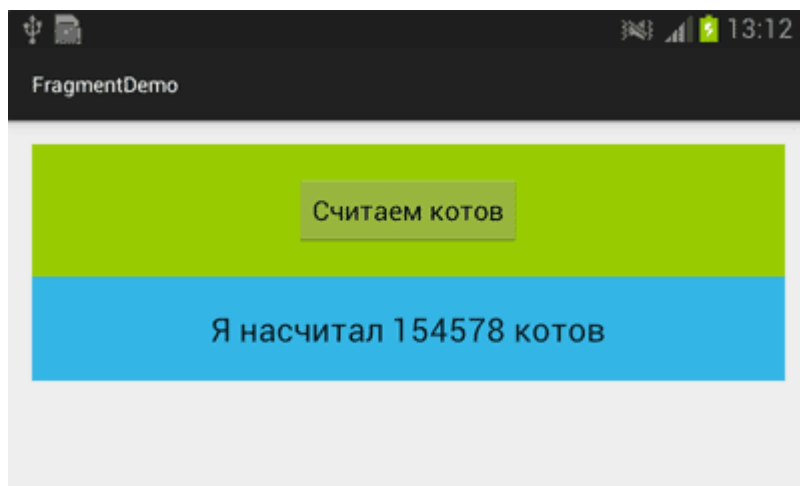
    public void changeText(String data) {
        mData = data;
        mTextView.setText(mData);
    }
}

```

Добавим новую переменную **mData**, которая будет хранить текст сообщения. У метода **onCreateView()** также есть параметр **savedInstanceState** класса **Bundle**, позволяющий извлечь сохранённые данные.

А текст мы сохраним в методе **onSaveInstanceState()**. В методе **changeText()** добавим строку кода, чтобы текст брался из новой переменной. Теперь при первом запуске всё работает как прежде. При повороте текст из **mData** сохраняется в методе **putString()** и восстанавливается через **getString()**.

После этих изменений программа больше не теряет своих данных и можно спокойно считать котов. Я стал смотреть на гифку и считать. Через 8 минут 16 секунд последний кот выпрыгнул в окно и в итоге получилось 154578 котов. Перепроверьте.



Часть шестая. Удержание состояния фрагмента (fragment6.php)

Дополнительное чтение

Обсуждение статьи (<http://forum.alexanderklimov.ru/viewtopic.php?id=31>) на форуме.

Реклама

Реклама