### DialogFragment и создание диалоговых окон

Последнее обновление: 18.10.2021

Для создания своих диалоговых окон используется компонент AlertDialog в связке с классом фрагмента DialogFragment. Рассмотрим их применение.

Вначале добавим в проект новый класс фрагмента, который назовем CustomDialogFragment:

package com.example.dialogsapp;

import android.app.AlertDialog;

import android.app.Dialog;

import android.os.Bundle;

import androidx.fragment.app.DialogFragment;

import androidx.annotation.NonNull;

public class CustomDialogFragment extends DialogFragment {

@NonNull

public Dialog onCreateDialog(Bundle savedInstanceState) {

AlertDialog.Builder builder=new AlertDialog.Builder(getActivity());

return builder.setTitle("Диалоговое окно").setMessage("Для закрытия окна нажмите ОК").create();

}

}

Класс фрагмента содержит всю стандартную функциональность фрагмента с его жизненным циклом, но при этом наследуется от класса DialogFragment, который добавляет ряд дополнительных функций. И для его создания мы можем использовать два способа:

* Переопределение метода onCreateDialog(), который возвращает объект Dialog.
* Использование стандартного метода onCreateView().

Для создания диалогового окна в методе onCreateDialog() применяется класс AlertDialog.Builder. С помощью своих методов он позволяет настроить отображение диалогового окна:

* setTitle: устанавливает заголовок окна
* setView: устанавливает разметку интерфейса окна
* setIcon: устанавливает иконку окна
* setPositiveButton: устанавливает кнопку подтверждения действия
* setNeutralButton: устанавливает "нейтральную" кнопку, действие которой может отличаться от действий подтверждения или отмены
* setNegativeButton: устанавливает кнопку отмены
* setMessage: устанавливает текст диалогового окна, но при использовании setView данный метод необязателен или может рассматриваться в качестве альтернативы, если нам надо просто вывести сообщение.
* create: создает окно

В данном же случае диалоговое окно просто выводит некоторое сообщение.

Для вызова этого диалогового окна в файле activity\_main.xml определим кнопку:

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

<androidx.constraintlayout.widget.ConstraintLayout

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

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

android:layout\_width="match\_parent"

android:layout\_height="match\_parent">

<Button

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Dialog"

android:onClick="showDialog"

app:layout\_constraintTop\_toTopOf="parent"

app:layout\_constraintLeft\_toLeftOf="parent" />

</androidx.constraintlayout.widget.ConstraintLayout>

В коде MainActivity определим обработчик нажатия кнопки, который будет запускать диалоговое окно:

package com.example.dialogsapp;

import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;

import android.view.View;

public class MainActivity extends AppCompatActivity {

@Override

public void onCreate(Bundle savedInstance) {

super.onCreate(savedInstance);

setContentView(R.layout.activity\_main);

}

public void showDialog(View v) {

CustomDialogFragment dialog = new CustomDialogFragment();

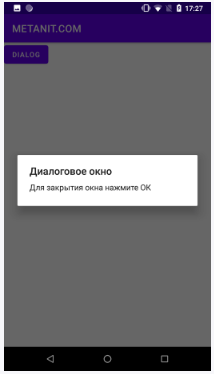
dialog.show(getSupportFragmentManager(), "custom");

}

}

Для вызова диалогового окна создается объект фрагмента CustomDialogFragment, затем у него вызывается метод show(). В этот метод передается менеджер фрагментов FragmentManager и строка - произвольный тег.

И поле нажатия на кнопку мы сможем ввести данные в диалоговое окно:



Теперь немного кастомизируем диалоговое окно:

public Dialog onCreateDialog(Bundle savedInstanceState) {

AlertDialog.Builder builder=new AlertDialog.Builder(getActivity());

return builder

.setTitle("Диалоговое окно")

.setIcon(android.R.drawable.ic\_dialog\_alert)

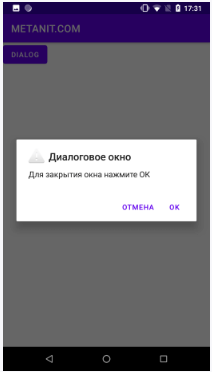
.setMessage("Для закрытия окна нажмите ОК")

.setPositiveButton("OK", null)

.setNegativeButton("Отмена", null)

.create();

}



Здесь добавляетcя иконка, которая в качестве изображения использует встроенный ресурс android.R.drawable.ic\_dialog\_alert и устанавливаются две кнопки. Для каждой кнопки можно установить текст и обработчик нажатия. В данном случае для обработчика нажатия передается null, то есть обработчик не установлен.

Теперь добавим в папку res/layout новый файл dialog.xml:

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

<androidx.constraintlayout.widget.ConstraintLayout

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

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

android:layout\_width="match\_parent"

android:layout\_height="match\_parent">

<TextView

android:gravity="center"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:text="Hello Android"

app:layout\_constraintTop\_toTopOf="parent"

app:layout\_constraintLeft\_toLeftOf="parent"

app:layout\_constraintRight\_toRightOf="parent"

app:layout\_constraintBottom\_toBottomOf="parent" />

</androidx.constraintlayout.widget.ConstraintLayout>

И изменим создание диалогового окна:

public Dialog onCreateDialog(Bundle savedInstanceState) {

AlertDialog.Builder builder=new AlertDialog.Builder(getActivity());

return builder

.setTitle("Диалоговое окно")

.setIcon(android.R.drawable.ic\_dialog\_alert)

.setView(R.layout.dialog)

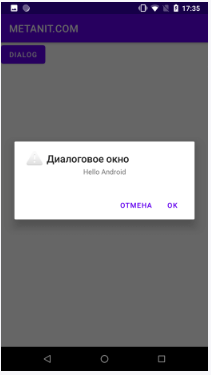
.setPositiveButton("OK", null)

.setNegativeButton("Отмена", null)

.create();

}

Метод setView() устанавливает в качестве интерфейса окна ранее добавленный ресурс layout dialog.xml.



При этом, как можно увидеть на скриншоте, кнопки и заголовок с иконкой не входят в разметку.

### Передача данных в диалоговое окно

Последнее обновление: 18.10.2021

Передача данных в диаговое окно, как и в любой фрагмент, осуществляется с помощью объекта Bundle.

Так, определим в файле activity\_main.xml список ListView:

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

<androidx.constraintlayout.widget.ConstraintLayout

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

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

android:layout\_width="match\_parent"

android:layout\_height="match\_parent">

<ListView

android:id="@+id/phonesList"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

app:layout\_constraintRight\_toRightOf="parent"

app:layout\_constraintTop\_toTopOf="parent"

app:layout\_constraintLeft\_toLeftOf="parent"

app:layout\_constraintBottom\_toBottomOf="parent"

/>

</androidx.constraintlayout.widget.ConstraintLayout>

В классе MainActivity определим для этого списка данные:

package com.example.dialogsapp;

import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;

import android.view.View;

import android.widget.AdapterView;

import android.widget.ArrayAdapter;

import android.widget.ListView;

import java.util.ArrayList;

public class MainActivity extends AppCompatActivity {

@Override

public void onCreate(Bundle savedInstance) {

super.onCreate(savedInstance);

setContentView(R.layout.activity\_main);

ListView phonesList = findViewById(R.id.phonesList);

ArrayList<String> phones = new ArrayList<>();

phones.add("Google Pixel");

phones.add("Huawei P9");

phones.add("LG G5");

phones.add("Samsung Galaxy S8");

ArrayAdapter<String> adapter = new ArrayAdapter<>(this, android.R.layout.simple\_list\_item\_1, phones);

phonesList.setAdapter(adapter);

phonesList.setOnItemClickListener(new AdapterView.OnItemClickListener() {

@Override

public void onItemClick(AdapterView<?> parent, View view, int position, long id) {

String selectedPhone = adapter.getItem(position);

CustomDialogFragment dialog = new CustomDialogFragment();

Bundle args = new Bundle();

args.putString("phone", selectedPhone);

dialog.setArguments(args);

dialog.show(getSupportFragmentManager(), "custom");

}

});

}

}

В обработчике нажатия на элемент в списке получаем выбранный элемент и добавляем его в объект Bundle. Далее через метод dialog.setArguments() передаем данные из Bundle во фрагмент.

Теперь определим следующий класс фрагмента CustomDialogFragment:

package com.example.dialogsapp;

import android.app.AlertDialog;

import android.app.Dialog;

import android.os.Bundle;

import androidx.annotation.NonNull;

import androidx.fragment.app.DialogFragment;

public class CustomDialogFragment extends DialogFragment {

@NonNull

public Dialog onCreateDialog(Bundle savedInstanceState) {

String phone = getArguments().getString("phone");

AlertDialog.Builder builder=new AlertDialog.Builder(getActivity());

return builder

.setTitle("Диалоговое окно")

.setIcon(android.R.drawable.ic\_dialog\_alert)

.setMessage("Вы хотите удалить " + phone + "?")

.setPositiveButton("OK", null)

.setNegativeButton("Отмена", null)

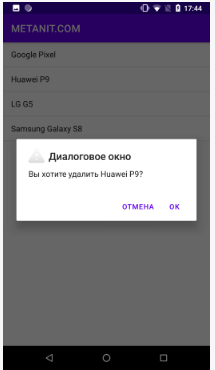
.create();

}

}

С помощью метода getArguments() получаем переданный в MainActivity объект Bundle. И так как была передана строка, то для ее извлечения применяется метод getString().

И при нажатии элемент списка будет передаваться в диалоговое окно:



В данном случае реального удаления не происходит, и в следующей статье рассмотрим, как добавить логику удаления и взаимодействия с Activity. Взаимодействие с Activity

Последнее обновление: 18.10.2021

Взаимодействие между Activity и фрагментом производится, как правило, через интерфейс. К примеру, в прошлой теме MainActivity выводила список объектов, и теперь определим удаление из этого списка через диалоговое окно.

Для этого добавим в проект интерфейс Removable:

package com.example.dialogsapp;

public interface Removable {

void remove(String name);

}

Единственный метод интерфейса remove получает удаляемый объект в виде параметра name.

Теперь реализуем этот интерфейс в коде MainActivity:

package com.example.dialogsapp;

import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;

import android.view.View;

import android.widget.AdapterView;

import android.widget.ArrayAdapter;

import android.widget.ListView;

import java.util.ArrayList;

public class MainActivity extends AppCompatActivity implements Removable{

private ArrayAdapter<String> adapter;

@Override

public void onCreate(Bundle savedInstance) {

super.onCreate(savedInstance);

setContentView(R.layout.activity\_main);

ListView phonesList = findViewById(R.id.phonesList);

ArrayList<String> phones = new ArrayList<>();

phones.add("Google Pixel");

phones.add("Huawei P9");

phones.add("LG G5");

phones.add("Samsung Galaxy S8");

adapter = new ArrayAdapter<>(this, android.R.layout.simple\_list\_item\_1, phones);

phonesList.setAdapter(adapter);

phonesList.setOnItemClickListener(new AdapterView.OnItemClickListener() {

@Override

public void onItemClick(AdapterView<?> parent, View view, int position, long id) {

String selectedPhone = adapter.getItem(position);

CustomDialogFragment dialog = new CustomDialogFragment();

Bundle args = new Bundle();

args.putString("phone", selectedPhone);

dialog.setArguments(args);

dialog.show(getSupportFragmentManager(), "custom");

}

});

}

@Override

public void remove(String name) {

adapter.remove(name);

}

}

Метод remove просто удаляет из адаптера переданный элемент.

Файл activity\_main.xml по прежнему определяет только элемент ListView:

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

<androidx.constraintlayout.widget.ConstraintLayout

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

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

android:layout\_width="match\_parent"

android:layout\_height="match\_parent">

<ListView

android:id="@+id/phonesList"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

app:layout\_constraintRight\_toRightOf="parent"

app:layout\_constraintTop\_toTopOf="parent"

app:layout\_constraintLeft\_toLeftOf="parent"

app:layout\_constraintBottom\_toBottomOf="parent"

/>

</androidx.constraintlayout.widget.ConstraintLayout>

И в конце определим фрагмент CustomDialogFragment:

package com.example.dialogsapp;

import android.app.AlertDialog;

import android.app.Dialog;

import android.content.Context;

import android.content.DialogInterface;

import android.os.Bundle;

import androidx.annotation.NonNull;

import androidx.fragment.app.DialogFragment;

public class CustomDialogFragment extends DialogFragment {

private Removable removable;

@Override

public void onAttach(Context context){

super.onAttach(context);

removable = (Removable) context;

}

@NonNull

public Dialog onCreateDialog(Bundle savedInstanceState) {

final String phone = getArguments().getString("phone");

AlertDialog.Builder builder=new AlertDialog.Builder(getActivity());

return builder

.setTitle("Диалоговое окно")

.setIcon(android.R.drawable.ic\_dialog\_alert)

.setMessage("Вы хотите удалить " + phone + "?")

.setPositiveButton("OK", new DialogInterface.OnClickListener() {

@Override

public void onClick(DialogInterface dialog, int which) {

removable.remove(phone);

}

})

.setNegativeButton("Отмена", null)

.create();

}

}

Метод onAttach() вызывается в начале жизненного цикла фрагмента, и именно здесь мы можем получить контекст фрагмента, в качестве которого выступает класс MainActivity. Так как MainActivity реализует интерфейс Removable, то мы можем преобразовать контекст к данному интерфейсу.

Затем в обработчике кнопки ОК вызывается метод remove объекта Removable, который удаляет переданный во фрагмент объект phone.

