[Урок 80. Handler. Немного теории. Наглядный пример использования](https://startandroid.ru/ru/uroki/vse-uroki-spiskom/143-urok-80-handler-nemnogo-teorii-nagljadnyj-primer-ispolzovanija.html" \o "Урок 80. Handler. Немного теории.  Наглядный пример использования)

Handler дает нам две интересные и полезные возможности:

1) реализовать отложенное по времени выполнение кода

2) выполнение кода не в своем потоке

public void onclick(View v) {

  switch (v.getId()) {

  case R.id.btnStart:

    Thread t = new Thread(new Runnable() {

      public void run() {

        for (int i = 1; i <= 10; i++) {

          // долгий процесс

          downloadFile();

          // обновляем TextView

          tvInfo.setText("Закачано файлов: " + i);

          // пишем лог

          Log.d(LOG\_TAG, "i = " + i);

        }

      }

    });

    t.start();

    break;

  case R.id.btnTest:

    Log.d(LOG\_TAG, "test");

    break;

  default:

    break;

  }

}

<https://startandroid.ru/ru/uroki/vse-uroki-spiskom/143-urok-80-handler-nemnogo-teorii-nagljadnyj-primer-ispolzovanija.html>

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

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)

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();

  }

<https://startandroid.ru/ru/uroki/vse-uroki-spiskom/73-urok-33-hranenie-dannyh-preferences.html>

[Урок 46. События ExpandableListView](https://startandroid.ru/ru/uroki/vse-uroki-spiskom/88-urok-46-sobytija-expandablelistview.html)

Дерево-список строить мы умеем, теперь посмотрим, как с ним можно взаимодействовать. Нам предоставлена возможность обрабатывать следующие события: нажатие на группу, нажатие на элемент, сворачивание группы, разворачивание группы.

<ExpandableListView

            android:id="@+id/elvMain"

            android:layout\_width="match\_parent"

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

        </ExpandableListView>

<https://startandroid.ru/ru/uroki/vse-uroki-spiskom/88-urok-46-sobytija-expandablelistview.html>

dependencies **{** implementation fileTree(dir: "libs", include: ["\*.jar"])  
 implementation "com.vk:android-sdk-core:3.1.0"   
 implementation "com.vk:android-sdk-api:3.1.0"  
 implementation "org.jetbrains.kotlin:kotlin-stdlib:$kotlin\_version"  
 implementation 'androidx.core:core-ktx:1.5.0'  
 implementation 'androidx.appcompat:appcompat:1.3.0'  
 implementation 'androidx.activity:activity-ktx:1.2.3'  
 implementation 'androidx.constraintlayout:constraintlayout:2.0.4'  
 implementation 'androidx.recyclerview:recyclerview:1.2.0'  
 testImplementation 'junit:junit:4.12'  
 androidTestImplementation 'androidx.test.ext:junit:1.1.2'  
 androidTestImplementation 'androidx.test.espresso:espresso-core:3.3.0'  
  
**}**

<uses-permission android:name="android.permission.INTERNET"/>

<integer name="com\_vk\_sdk\_AppId">7864502</integer>

<?xml version="1.0" encoding="utf-8"?>  
<LinearLayout 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="wrap\_content"  
 android:orientation="vertical"  
 android:paddingLeft="12sp"  
 android:paddingRight="12sp"  
 android:layout\_margin="10sp"  
 >  
  
 <LinearLayout  
 android:layout\_width="fill\_parent"  
 android:layout\_height="wrap\_content"  
 android:orientation="vertical"  
 android:padding="15sp"  
 android:background="@drawable/list\_row\_bg"  
 >  
  
 <LinearLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:gravity="center\_vertical"  
 android:orientation="horizontal">  
  
 <ImageView  
 android:id="@+id/list\_row\_icon"  
 android:layout\_width="35sp"  
 android:layout\_height="35sp"  
 android:layout\_marginLeft="4dp"  
 android:layout\_marginTop="4dp"  
 android:layout\_marginRight="15dp"  
 app:srcCompat="@drawable/ic\_baseline\_person\_24" />  
  
 <LinearLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:orientation="vertical">  
  
 <TextView  
 android:id="@+id/list\_row\_text"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="Some"  
 android:textSize="18sp"  
 android:textColor="@android:color/black"  
 />  
  
 <TextView  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:id="@+id/list\_row\_description"  
 />  
 </LinearLayout>  
 </LinearLayout>  
 </LinearLayout>  
</LinearLayout>

<?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"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 tools:context=".MainActivity">  
  
 <LinearLayout  
 android:layout\_width="fill\_parent"  
 android:layout\_height="fill\_parent"  
 android:orientation="vertical"  
 app:layout\_constraintTop\_toTopOf="parent"  
 tools:layout\_editor\_absoluteX="-6dp">  
  
 <Button  
 android:id="@+id/buttonUsers"  
 android:layout\_width="162dp"  
 android:layout\_height="wrap\_content"  
 android:text="Загрузить" />  
  
 <Button  
 android:id="@+id/buttonSave"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="Сохранить" />  
 <Button  
 android:id="@+id/buttonLoad"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="Из бд" />  
  
 <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 tools:context=".MainActivity"  
 android:background="#ffffff"  
 android:orientation="vertical">  
  
 <ListView  
 android:id="@+id/listView"  
 android:dividerHeight="15sp"  
 android:divider="@android:color/transparent"  
 android:layout\_width="fill\_parent"  
 android:layout\_height="wrap\_content" />  
 </RelativeLayout>  
 </LinearLayout>  
  
</androidx.constraintlayout.widget.ConstraintLayout>

package com.example.mp  
  
import android.content.Intent  
import androidx.appcompat.app.AppCompatActivity  
import android.os.Bundle  
import android.util.Log  
import com.example.mp.dao.UserDao  
import com.example.mp.friends.ListAdapter  
import com.example.mp.models.User  
import com.example.mp.services.DbService  
import com.vk.api.sdk.VK  
import com.vk.api.sdk.auth.VKAccessToken  
import com.vk.api.sdk.auth.VKAuthCallback  
import com.vk.api.sdk.auth.VKScope  
import kotlinx.android.synthetic.main.activity\_main.\*  
  
class MainActivity : AppCompatActivity() {  
 override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 setContentView(R.layout.*activity\_main*)  
  
 VK.login(this, *arrayListOf*(VKScope.*WALL*, VKScope.*PHOTOS*))  
 var friendsList = *listOf*<User>()  
  
 buttonUsers.setOnClickListener **{** val userDao = UserDao()  
 userDao.getFriends(**{** friends: List<User> **->** *run* **{** Log.i("vk", friends.toString())  
 listView.*adapter* = ListAdapter(this, friends)  
 friendsList = friends  
 **}  
 }**)  
 **}** buttonSave.setOnClickListener **{** val userDao = UserDao()  
 userDao.saveFriends(friendsList, this)  
 **}** buttonLoad.setOnClickListener **{** val userDao = UserDao()  
 val friends = userDao.getFriendsFromDB(this)  
 Log.i("vk", friends.toString())  
 **}** }  
  
 override fun onActivityResult(requestCode: Int, resultCode: Int, data: Intent?) {  
 val callback = object: VKAuthCallback {  
 override fun onLogin(token: VKAccessToken) {  
 // User passed authorization  
 }  
  
 override fun onLoginFailed(errorCode: Int) {  
 // User didn't pass authorization  
 }  
 }  
 if (data == null || !VK.onActivityResult(requestCode, resultCode, data, callback)) {  
 super.onActivityResult(requestCode, resultCode, data)  
 }  
 }  
}

package com.example.mp.services  
  
import android.util.Log  
import com.vk.api.sdk.VK  
import com.vk.api.sdk.VKApiCallback  
import com.vk.api.sdk.requests.VKRequest  
  
class VkApiService {  
 val tag: String = "vk"  
  
 fun <T>execute(request: VKRequest<T>, onSuccess: (result: T) -> Unit) {  
 VK.execute(request, object: VKApiCallback<T> {  
 override fun fail(error: Exception) {  
 Log.e(tag, error.toString())  
 }  
  
 override fun success(result: T) {  
 onSuccess(result)  
 }  
 })  
 }  
  
 companion object {  
 private var instance: VkApiService? = null  
  
 fun getInstance(): VkApiService {  
 if (instance == null) {  
 instance = VkApiService()  
 }  
  
 return instance!!  
 }  
 }  
}

package com.example.mp.services  
  
import android.content.Context  
import android.database.sqlite.SQLiteDatabase  
import android.database.sqlite.SQLiteOpenHelper  
import android.util.Log  
import com.example.mp.models.User  
  
class DbService(context: Context) : SQLiteOpenHelper(context, "app.db", null, 1) {  
 override fun onCreate(db: SQLiteDatabase?) {  
 Log.i("vk", db.*toString*())  
 db?.execSQL(User.CREATE\_TABLE)  
 Log.i("vk", "db created")  
 }  
  
 override fun onUpgrade(db: SQLiteDatabase?, oldVersion: Int, newVersion: Int) {  
 onCreate(db)  
 }  
  
 companion object {  
 private var instance: DbService? = null  
  
 fun getInstance(context: Context? = null): DbService {  
 if (instance == null) {  
 instance = DbService(context!!)  
 }  
  
 return instance!!  
 }  
 }  
}

package com.example.mp.requests  
  
import com.example.mp.models.User  
import com.vk.api.sdk.requests.VKRequest  
import org.json.JSONObject  
  
class VKUsersRequest(method: String) : VKRequest<List<User>>(method) {  
 override fun parse(r: JSONObject): List<User> {  
 val response = r.getJSONObject("response")  
 val users = response.getJSONArray("items")  
 val result = ArrayList<User>()  
 for (i in 0 *until* users.length()) {  
 result.add(User.parse(users.getJSONObject(i)))  
 }  
 return result  
 }  
}

package com.example.mp.models  
  
import android.os.Parcel  
import android.os.Parcelable  
import org.json.JSONObject  
  
data class User(  
 val id: Int = 0,  
 val firstName: String = "",  
 val lastName: String = "",  
 val photo: String = "") {  
 companion object {  
 fun parse(json: JSONObject)  
 = User(id = json.optInt("id", 0),  
 firstName = json.optString("first\_name", ""),  
 lastName = json.optString("last\_name", ""),  
 photo = json.optString("photo\_200", ""))  
  
 private const val TABLE\_NAME = "Users"  
  
 const val CREATE\_TABLE = "CREATE TABLE IF NOT EXISTS $TABLE\_NAME(" +  
 "id integer primary key autoincrement, " +  
 "firstName text, " +  
 "lastName text, " +  
 "photo text);"  
  
 }  
}

package com.example.mp.friends  
  
import android.content.Context  
import android.graphics.BitmapFactory  
import android.view.LayoutInflater  
import android.view.View  
import android.view.ViewGroup  
import android.widget.BaseAdapter  
import android.widget.ImageView  
import android.widget.TextView  
import com.example.mp.R  
import com.example.mp.models.User  
import kotlinx.coroutines.GlobalScope  
import kotlinx.coroutines.launch  
import kotlinx.coroutines.runBlocking  
import java.net.URL  
import androidx.fragment.app.Fragment  
  
class ListAdapter(context: Context, private val data: List<User>): BaseAdapter() {  
  
 private var inflater: LayoutInflater = LayoutInflater.from(context)  
  
 override fun getView(position: Int, convertView: View?, parent: ViewGroup?): View {  
 var view = convertView  
 if (view == null) view = inflater.inflate(R.layout.*list\_row*, null)!!  
  
 val currentItem = data[position]  
  
 val textView = view.findViewById<View>(R.id.*list\_row\_text*) as TextView  
 textView.*text* = "${currentItem.firstName} ${currentItem.lastName}"  
  
 val description = view.findViewById<View>(R.id.*list\_row\_description*) as TextView  
  
 val imageView: ImageView = view.findViewById(R.id.*list\_row\_icon*) as ImageView  
  
 *runBlocking* **{** val photo = GlobalScope.*launch* **{** val url = URL(currentItem.photo)  
 val image = BitmapFactory.decodeStream(url.openConnection().getInputStream());  
 imageView.setImageBitmap(image)  
 **}** photo.join()  
 **}** return view  
 }  
  
 override fun getItem(position: Int): Any {  
 return data[position]  
 }  
  
 override fun getItemId(position: Int): Long {  
 return position.toLong()  
 }  
  
 override fun getCount(): Int {  
 return data.size  
 }  
}

package com.example.mp.dao  
  
import android.content.ContentValues  
import android.content.Context  
import android.util.Log  
import com.example.mp.models.User  
import com.example.mp.requests.VKUsersRequest  
import com.example.mp.services.DbService  
import com.example.mp.services.VkApiService  
  
class UserDao {  
 private val api: VkApiService = VkApiService.getInstance()  
  
 fun getFriends(onSuccess: (res: List<User>) -> Unit, userId: Int? = null) {  
 val req = VKUsersRequest("friends.get")  
 if (userId != null) req.addParam("user\_id", userId)  
 req.addParam("fields", "photo\_200")  
 req.addParam("count", "5")  
 api.execute(req, onSuccess)  
 }  
  
 fun saveFriends(friends: List<User>, context: Context) {  
 val dbService = DbService.getInstance(context)  
 val db = dbService.*writableDatabase* friends.*forEach* **{** val values = ContentValues().*apply* **{** put("firstName", **it**.firstName)  
 put("lastName", **it**.lastName)  
 put("photo", **it**.photo)  
 **}** val newRowId = db?.insert("Users", null, values)  
 **}** }  
  
 fun getFriendsFromDB(context: Context): List<User> {  
 val dbService = DbService.getInstance(context)  
 val db = dbService.*writableDatabase* val friends = *mutableListOf*<User>()  
 val cursor = db.query("Users", null, null, null, null, null, null)  
  
 while (cursor?.moveToNext()!!) {  
 val id = cursor.getInt(0)  
 val firstName = cursor.getString(1)  
 val lastName = cursor.getString(2)  
 val photo = cursor.getString(3)  
  
 val user = User(id, firstName, lastName, photo)  
 friends.add(user)  
 }  
  
 return friends  
 }  
}