Санкт-Петербургский политехнический университет Петра Великого

Кафедра компьютерных систем и программных технологий

**Отчёт по лабораторной работе №7**

**Дисциплина**: Проектирование мобильных приложений

**Тема**: Сервисы и Broadcast Receivers

Выполнил студент гр. 3530901/70203 И.Д. Иванов

(подпись)

Преподаватель И.С. Егорова

(подпись)

“\_\_\_”\_\_\_\_\_\_\_\_\_\_\_\_\_ 2019 г.

Санкт-Петербург

2019

**1. Цели работы**

Получить практические навыки разработки сервисов (started и bound) и Broadcast Receivers.

**2. Программа работы**

**2.1. Задача 1.** Started сервис для скачивания изображения

В лабораторной работе №6 было разработан код, скачивающий картинку из интернета. На основе этого кода разработайте started service, скачивающий файл из интернета. URL изображения для скачивания должен передаваться в Intent. Убедитесь, что код для скачивания исполняется не в UI потоке

Добавьте в разработанный сервис функцию отправки broadcast сообщения по завершении скачивания. Сообщение (Intent) должен содержать путь к скачанному файлу.

Service является компонентом приложения, который может выполнять длительные операции в фоновом режиме и не содержит пользовательского интерфейса. Другой компонент приложения может запустить службу, которая продолжит работу в фоновом режиме даже в том случае, когда пользователь перейдет в другое приложение.

Started service - Служба является «запущенной», когда компонент приложения (например, операция) запускает ее вызовом startService(). После запуска служба может работать в фоновом режиме в течение неограниченного времени, даже если уничтожен компонент, который ее запустил.

Любой компонент приложения может использовать службу (даже из отдельного приложения) подобно тому, как любой компонент может использовать операцию — запустив ее с помощью Intent.

IntentService - это подкласс класса Service, который использует рабочий поток для обработки всех запросов запуска поочередно. Достаточно реализовать метод onHandleIntent(), который получает намерение для каждого запроса запуска, позволяя выполнять фоновую работу.

Исходный код службы:

class DownloadService : IntentService("DownloadService") {  
  
 val URL\_EXTRA= "url"  
 val IMG\_TITLE = "I\_love\_android"  
 val BROADCAST\_TAG = "load\_url\_message"  
 val MESSAGE\_EXTRA = "path\_extra"  
  
 override fun onHandleIntent(intent: Intent?) {  
 val url = intent?.getStringExtra(URL\_EXTRA)  
 if (url == null)  
 sendBroadcast("null url")  
 else {  
 download(url)  
 }  
  
 }  
  
 private fun download(url: String): String {  
 val bitmap = URL(url).openStream().*use* **{** return@use BitmapFactory.decodeStream(**it**)  
 **}** val path = saveImage(bitmap, IMG\_TITLE)  
 sendBroadcast(path)  
 return path  
 }  
  
 private fun saveImage(b: Bitmap, imageName: String): String {  
 val outStream: FileOutputStream  
 try {  
 outStream = this.*applicationContext*.openFileOutput(imageName, Context.*MODE\_PRIVATE*)  
 b.compress(Bitmap.CompressFormat.PNG, 100, outStream)  
 outStream.close()  
 } catch (e: Exception) {  
 Log.d("saveImage", " Error with saving!")  
 e.printStackTrace()  
 }  
 return *applicationContext*.getFileStreamPath(imageName).*absolutePath* }  
  
 private fun sendBroadcast(msg: String) {  
 sendBroadcast(Intent(BROADCAST\_TAG).putExtra(MESSAGE\_EXTRA, msg))  
 }  
}

Скачивание гарантированно произойдет не в UI потоке, поскольку DownloadService наследуется от IntentService, который использует рабочий поток для обработки всех запросов запуска поочередно.

По завершению скачивания сервис отправляет broadcast сообщение, содержащее путь к скачанному файлу.

IntentService останавливается после обработки всех запросов запуска, поэтому нам не требуется вызывать stopSelf().

Исходный код MainActivity:

class MainActivity : AppCompatActivity() {  
  
 override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 setContentView(R.layout.*activity\_main*)  
  
 button.setOnClickListener **{** startService(  
 Intent(this, DownloadService::class.*java*).putExtra(  
 "url",  
 "https://namobilu.com/u/img/ib/003/195/195003-1.jpg"  
 )  
 )  
 **}** }  
}

При нажатии на кнопку запускается служба. URL изображения для скачивания передаётся в Intent.

**2.2. Задача 2.** Broadcast Receiver

Разработайте два приложения: первое приложение содержит 1 activity с 1 кнопкой, при нажатии на которую запускается сервис по скачиванию файла. Второе приложение содержит 1 broadcast receiver и 1 activity. Broadcast receiver по получении сообщения из сервиса инициирует отображение пути к изображению в TextView в Activity.

Android приложения могут посылать и получать broadcast сообщения от ОС и других приложений. Для обработки broadcast сообщений используется BroadcastReceiver.

В качестве первого приложения используем приложение, созданное в задаче 1.

Исходный код MainActivity второго приложения:

class MainActivity : AppCompatActivity() {  
  
 private lateinit var broadcastReceiver: BroadcastReceiver  
 val MESSAGE\_EXTRA = "path\_extra"  
 val BROADCAST\_TAG = "load\_url\_message"  
   
 override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 setContentView(R.layout.*activity\_main*)  
   
 broadcastReceiver = object : BroadcastReceiver() {  
 override fun onReceive(context: Context?, intent: Intent?) {  
 val msg = intent?.getStringExtra(MESSAGE\_EXTRA)  
 path\_text.*text* = msg  
 }  
 }  
  
 registerReceiver(broadcastReceiver, IntentFilter(BROADCAST\_TAG))  
  
 }  
  
 override fun onDestroy() {  
 super.onDestroy()  
 unregisterReceiver(broadcastReceiver)  
 }  
}

В методе onCreate() создаётся broadcastReceiver, обрабатывающий broadcast сообщения с тегом BROADCAST\_TAG. При получении сообщения в текстовом поле отображается путь к скачанной картинке, полученный из первого приложения.

Результат работы:



Рис.1. Внешний вид второго приложения после получения сообщения

**2.3. Задача 3.** Bound Service для скачивания изображения

Сделайте разработанный сервис одновременно bound И started: переопределите метод onBind. Из тела метода возвращайте IBinder, полученный из класса Messenger. Убедитесь, что код скачивания файла исполняется не в UI потоке.

Измените способ запуска сервиса в первом приложении: вместо startService используйте bindService. При нажатии на кнопку отправляйте сообщение Message, используя класс Messenger, полученный из интерфейса IBinder в методе onServiceConnected.

Добавьте в первое приложение TextView, а в сервис отправку обратного сообщения с местоположением скачанного файла. При получении сообщения от сервиса приложение должно отобразить путь к файлу на экране.

Bound Service - это служба, которая допускает привязку к ней компонентов приложения посредством вызова bindService() для создания долговременного соединения. К службе могут быть привязаны несколько компонентов одновременно, но, когда все они отменяют привязку, служба уничтожается, поэтому привязанную службу не требуется останавливать вручную.

Интерфейс между службой и клиентом должен быть реализацией объекта IBinder, которую служба должна возвращать из метода обратного вызова onBind(). После того, как клиент получает объект IBinder, он может начать взаимодействие со службой посредством этого интерфейса.

Если необходимо, чтобы служба взаимодействовала с удаленными процессами, для предоставления интерфейса службы можно воспользоваться объектом Messenger. При его использовании для клиента отсутствуют «методы» для отправки вызова службе. Вместо этого клиент отправляет «сообщения» (объекты Message), которые служба получает в своем объекте Handler.

Исходный код службы:

class DownloadService : IntentService("DownloadService") {  
  
 val URL\_EXTRA = "url"  
 val IMG\_TITLE = "I\_love\_android"  
 val BROADCAST\_TAG = "load\_url\_message"  
 val MESSAGE\_EXTRA = "path\_extra"  
 val MESSAGE\_RESPONSE = "response"  
 val MESSAGE\_TO\_CLIENT = 1  
 val MESSAGE\_TO\_SERVICE = 2  
  
 private lateinit var messenger: Messenger  
  
 override fun onBind(intent: Intent): IBinder? {  
 messenger = Messenger(@SuppressLint("HandlerLeak") object : Handler() {  
 override fun handleMessage(msg: Message) {  
 when (msg.what) {  
 MESSAGE\_TO\_SERVICE -> {  
 DownloadAsyncTask(msg.replyTo).execute(msg.*data*.getString(URL\_EXTRA))  
 }  
 }  
 }  
 })  
 return messenger.*binder* }  
  
 override fun onHandleIntent(intent: Intent?) {  
 val url = intent?.getStringExtra(URL\_EXTRA)  
 if (url == null)  
 sendBroadcast("null url")  
 else {  
 download(url)  
 }  
 }  
  
 private fun download(url: String): String {  
 val bitmap = URL(url).openStream().*use* **{** return@use BitmapFactory.decodeStream(**it**)  
 **}** val path = saveImage(bitmap, IMG\_TITLE)  
 sendBroadcast(path)  
 return path  
 }  
  
 private fun saveImage(b: Bitmap, imageName: String): String {  
 val outStream: FileOutputStream  
 try {  
 outStream = this.*applicationContext*.openFileOutput(imageName, Context.*MODE\_PRIVATE*)  
 b.compress(Bitmap.CompressFormat.PNG, 100, outStream)  
 outStream.close()  
 } catch (e: Exception) {  
 Log.d("saveImage", " Error with saving!")  
 e.printStackTrace()  
 }  
 return *applicationContext*.getFileStreamPath(imageName).*absolutePath* }  
 private fun sendBroadcast(msg: String) {  
 sendBroadcast(Intent(BROADCAST\_TAG).putExtra(MESSAGE\_EXTRA, msg))  
 }  
  
 inner class DownloadAsyncTask(private val receiver: Messenger? = null) :  
 AsyncTask<String, Void, String>() {  
  
 override fun doInBackground(vararg urls: String): String? {  
 val url = urls[0]  
 var str: String? = null  
 try {  
 str = download(url)  
 } catch (e: Exception) {  
 Log.e("Error loading", e.message)  
 }  
 return str  
 }  
  
 override fun onPostExecute(result: String) {  
 super.onPostExecute(result)  
 val message = Message.obtain(null, MESSAGE\_TO\_CLIENT, 0, 0).*apply* **{** *data* = Bundle().*apply* **{** putString(MESSAGE\_RESPONSE, result) **}  
 }** receiver?.send(message)  
 }  
 }  
}

Сервис одновременно является и bound, и started. При запуске сервиса с помощью startService() он будет работать аналогично заданию 1, однако, при запуске сервиса с помощью bindService() будет выполняться привязка сервиса и создание AsyncTask для асинхронного скачивания картинки.

Исходный код MainActivity:

package org.bigawesometurtle.lab7\_3  
  
import android.annotation.SuppressLint  
import android.content.ComponentName  
import android.content.Intent  
import android.content.ServiceConnection  
import android.os.\*  
import androidx.appcompat.app.AppCompatActivity  
import kotlinx.android.synthetic.main.activity\_main.\*  
  
class MainActivity : AppCompatActivity() {  
  
 val URL\_EXTRA = "url"  
 val MESSAGE\_TO\_CLIENT = 1  
 val MESSAGE\_TO\_SERVICE = 2  
 val MESSAGE\_RESPONSE = "response"  
 val URL = "https://namobilu.com/u/img/ib/003/195/195003-1.jpg"  
  
 private var serviceMessenger: Messenger? = null  
 private var serviceConnected = false  
 private val messenger = Messenger(ClientHandler())  
  
 private val serviceConnection = object : ServiceConnection {  
 override fun onServiceDisconnected(name: ComponentName) {  
 serviceMessenger = null  
 serviceConnected = false  
 }  
  
 override fun onServiceConnected(name: ComponentName, service: IBinder) {  
 serviceMessenger = Messenger(service)  
 serviceConnected = true  
 }  
 }  
  
 override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 setContentView(R.layout.*activity\_main*)  
  
 button.setOnClickListener **{** val message = Message.obtain(null, MESSAGE\_TO\_SERVICE, 0, 0).*apply* **{** replyTo = messenger  
 *data* = Bundle().*apply* **{** putString(URL\_EXTRA, URL) **}  
 }** serviceMessenger?.send(message)  
 **}** }  
  
 override fun onStart() {  
 super.onStart()  
 val intent = Intent(this, DownloadService::class.*java*)  
 bindService(intent, serviceConnection, *BIND\_AUTO\_CREATE*)  
 }  
  
 override fun onStop() {  
 super.onStop()  
 if (serviceConnected) {  
 unbindService(serviceConnection)  
 serviceConnected = false  
 }  
 }  
  
 inner class ClientHandler : Handler() {  
 override fun handleMessage(msg: Message) {  
 when (msg.what) {  
 MESSAGE\_TO\_CLIENT -> {  
 path\_text.*text* = msg.*data*.getString(MESSAGE\_RESPONSE)  
 }  
 }  
 }  
 }  
}

Сервис запускается при помощи bindService(). Сообщения отправляются с помощью класса Messenger. Сервису отправляется сообщение, содержащее URL изображения для скачивания, он скачивает картинку и отправляет обратное сообщение, содержащее путь к скачанной картинке. Полученный путь отображается в текстовом поле.

Результат работы:

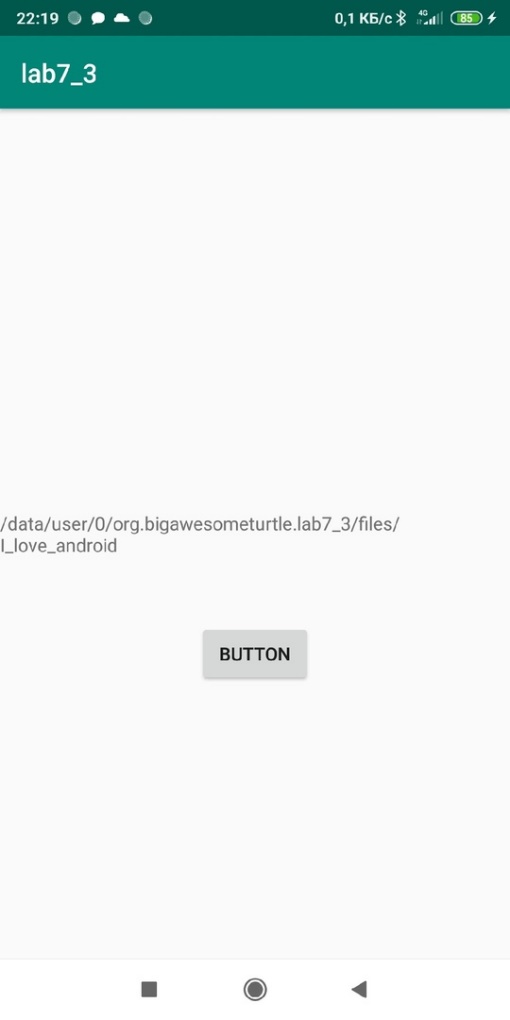


Рис.2. Результат работы приложения

**3. Вывод**

В ходе выполнения данной лабораторной работы были получены практические навыки разработки сервисов (started и bound) и Broadcast Receivers.