Services

Prof. Dr.-Ing. V.Brovkov

Service

- Service ist eine Anwendung, die im Hintergrund ohne UI läuft.
- Services können Hintergrundaufgaben übernehmen, die keine Interaktion mit dem Benutzer erfordern und auch weiterlaufen sollen, wenn andere Anwendungen im Vordergrund laufen.
  - Musik soll z.B. auch weiterlaufen, wenn wir gerade eine E-Mail verfassen oder auf Webseiten surfen.
- Hintergrunddienste können periodisch irgendetwas überwachen, z.B. die aktuelle Position, oder Statusmeldungen aus dem Netz empfangen und ggf, eine Benachrichtigung auslösen.
  - Das Herunterladen von Updates und auch die Synchronisierung von Diensten (Kontakte Bilder etc.) finden auch mittels Services statt.
- Grundsätzlich ist die Verwendung von Services mit Bedacht zu wählen, denn je nach Priorität und Ressourcenverbrauch erhöhen Hintergrunddienste den Stromverbrauch
  - Das ist ein häufiger Kritikpunkt am sehr mächtigen Multitasking-Konzept von Android

Beispiel

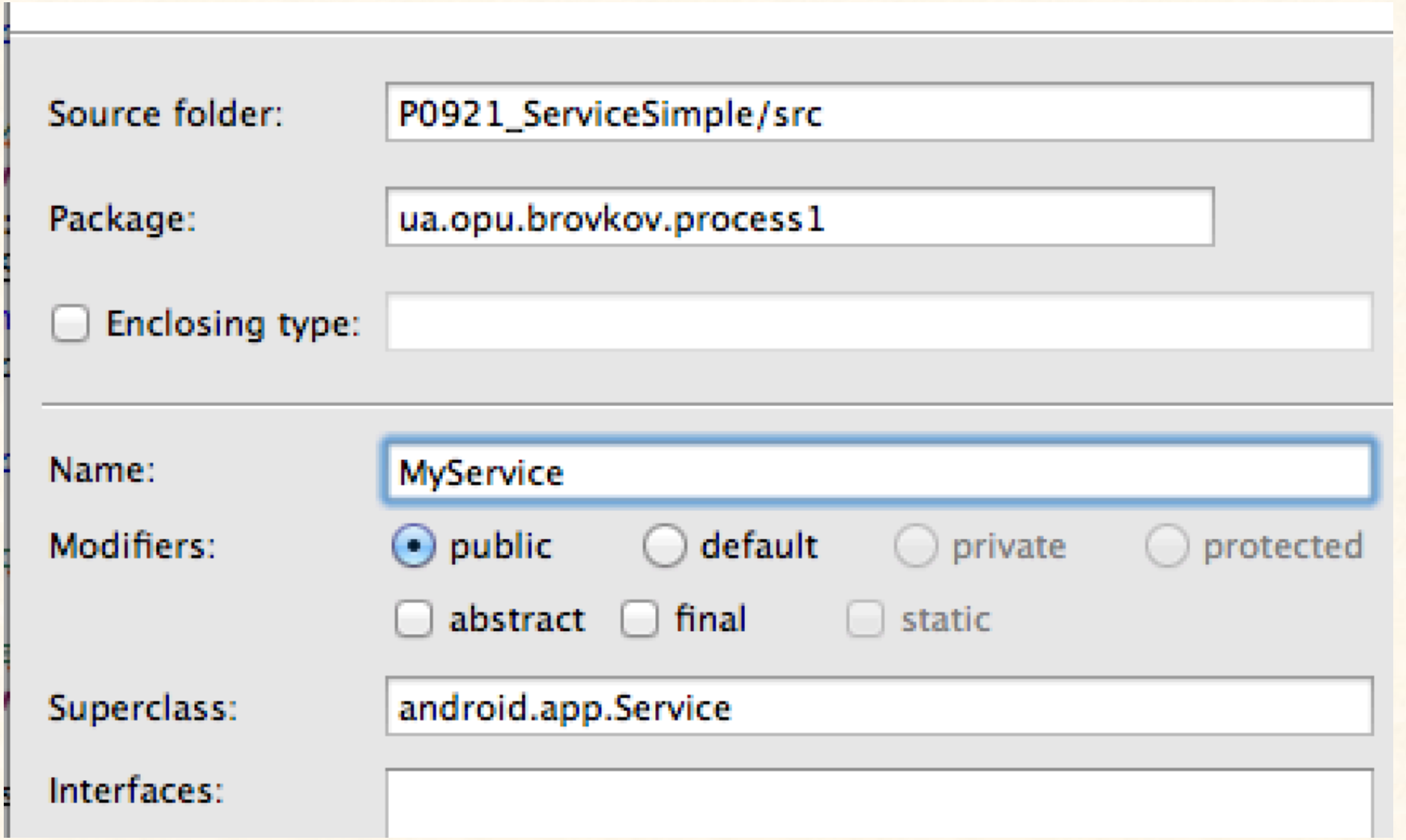
Projekt P0921_ServiceSimple

http://startandroid.ru/ru/uroki/vse-uroki-spiskom/157-urok-92-service-prostoj-primer.html

- Ein Service läuft in der gleicher Anwendung mit einer Activity
  - Was passiert wenn die Anwendung beendet und neu gestartet wird?
  - Wie kann man den Service beenden und neu starten lassen?

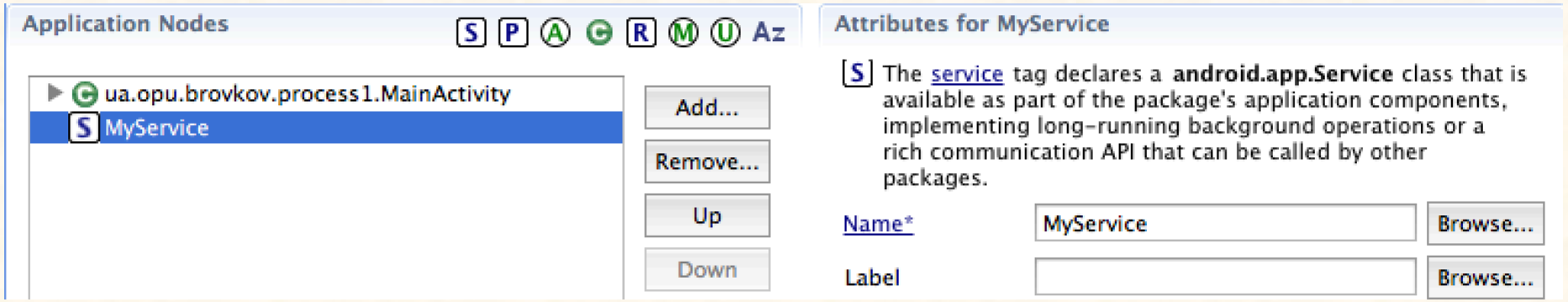
P0921_ServiceSimple Beispiel 1

P0921_ServiceSimple Beispiel 2



P0921_ServiceSimple Beispiel 3

- Projekterstellung:
  - Ein konventionelles Projekt erstellen
  - **public class MyService extends Service {}** erstellen
  - Das Service in Manifest registrieren
  - Die erforderliche Code zugeben
  - Done!



P0921_ServiceSimple Beispiel 4

```
3  import android.app.Activity;..
7
8  public class MainActivity extends Activity {
9    final String LOG_TAG = "myLogs";
10     public void onCreate(Bundle savedInstanceState) {
11       super.onCreate(savedInstanceState);
12       setContentView(R.layout.activity_main);
13     }
14     public void onClickStart(View v) {
15       startService(new Intent(this, MyService.class));
16     }
17     public void onClickStop(View v) {
18       stopService(new Intent(this, MyService.class));
19     }
20 }
```

Service Start

Service Stop

Einfach ein Intent verwenden!

P0921_ServiceSimple Beispiel 5

```
10  public class MyService extends Service {
11    final String LOG_TAG = "myLogs";
12    public void onCreate() {
13      super.onCreate();
14      Log.d(LOG_TAG, "onCreate");
15    }
16    public int onStartCommand(Intent intent, int flags, int startId) {
17      Log.d(LOG_TAG, "onStartCommand");
18      someTask();
19      return super.onStartCommand(intent, flags, startId);
20    }
21    public void onDestroy() {
22      super.onDestroy();
23      Log.d(LOG_TAG, "onDestroy");
24    }
```

Service creation

Service Start

Service Action

Service Destroy

P0921_ServiceSimple Beispiel 6

```
25    public IBinder onBind(Intent intent) {
26      Log.d(LOG_TAG, "onBind");
27      return null;
28    }
29    void someTask() {
30      for (int i = 1; i<=5; i++) {
31        Log.d(LOG_TAG, "i = " + i);
32        try {
33          TimeUnit.SECONDS.sleep(1);
34        } catch (InterruptedException e) {
35          e.printStackTrace();
36        }
37      }
38    }
39  }
```

Some Huge Task

Time Delay

P0921_ServiceSimple Beispiel 7

- Projekt Logs für den ersten Process start:

```
ua.opu.brovkov.process1        myLogs        onCreate
ua.opu.brovkov.process1        myLogs        onStartCommand
ua.opu.brovkov.process1        myLogs        i = 1
ua.opu.brovkov.process1        myLogs        i = 2
ua.opu.brovkov.process1        myLogs        i = 3
ua.opu.brovkov.process1        myLogs        i = 4
ua.opu.brovkov.process1        myLogs        i = 5
```

- Die Anwendung wurde beendet und neu gestartet. Die Logs:

```
ua.opu.brovkov.process1        myLogs        onStartCommand
ua.opu.brovkov.process1        myLogs        i = 1
ua.opu.brovkov.process1        myLogs        i = 2
ua.opu.brovkov.process1        myLogs        i = 3
ua.opu.brovkov.process1        myLogs        i = 4
ua.opu.brovkov.process1        myLogs        i = 5
```

P0921_ServiceSimple Beispiel 7

- Projekt Logs. Prozess wurde gestoppt und neu gestartet.

```
ua.opu.brovkov.process1     myLogs      onDestroy
ua.opu.brovkov.process1     myLogs      onCreate
ua.opu.brovkov.process1     myLogs      onStartCommand
ua.opu.brovkov.process1     myLogs      i = 1
ua.opu.brovkov.process1     myLogs      i = 2
ua.opu.brovkov.process1     myLogs      i = 3
ua.opu.brovkov.process1     myLogs      i = 4
ua.opu.brovkov.process1     myLogs      i = 5
```

- Die Anwendung wird gebremst, wenn das Process eine aufwändige Aufgabe erfüllt (siehe die Aktivitätsanzeige!). Ein separates Thread ist erforderlich!
- Die Lösung: sehe die nächste Folie

P0921_ServiceSimple Beispiel 8

```
18    public int onStartCommand(Intent intent, int flags, int startId) {
19        Log.d(LOG_TAG, "onStartCommand");
20        //someTask();
21        someTaskThreaded();
22        return super.onStartCommand(intent, flags, startId);
23    }

45    void someTaskThreaded() {
46        new Thread(new Runnable() {
47            public void run() {
48                for (int i = 1; i <= 5; i++) {
49                    Log.d(LOG_TAG, "i = " + i);
50                    try {
51                        TimeUnit.SECONDS.sleep(1);
52                    } catch (InterruptedException e) {
53                        e.printStackTrace();
54                    }
55                }
56                stopSelf();
57            }
58        }).start();
59    }
```

Service kann im separaten Thread ausgeführt werden. Keine Bremse mehr!

Service kann auch sich selbst stoppen

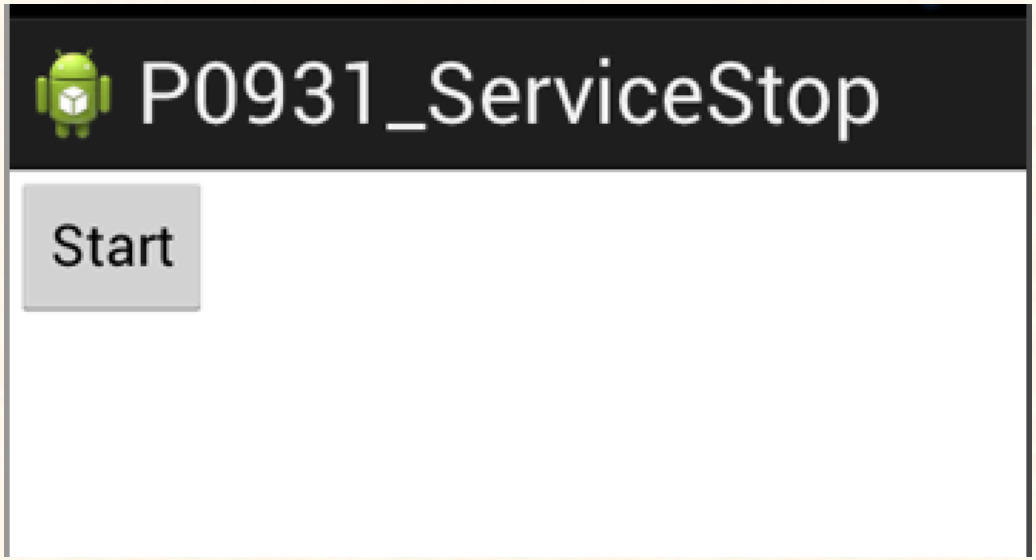
Beispiel

Projekt P0931_ServiceStop

http://startandroid.ru/ru/uroki/vse-uroki-spiskom/158-urok-93-service-peredacha-dannyh-v-servis-metody-ostanovki-servisa.html

- Ein Datenaustausch zwischen einer Activity und einem Service

Projekt P0931_ServiceStop 1



- Ein Projekt hat eine Activity und ein Service
- Mit dem Start Button erzeugen wir drei Intents, womit ein Service 3 Aufgaben erhält.
  - Jede Aufgabe hat eigene Parameters
  - Nur eine Aufgabe wird für paralleler Ausführung erlaubt
  - Nachdem alle Aufgaben beendet sind, stoppt der Service sich selbst

Projekt P0931_ServiceStop 2

```
 8  public class MainActivity extends Activity {
 9    /** Called when the activity is first created. */
10    @Override
11    public void onCreate(Bundle savedInstanceState) {
12      super.onCreate(savedInstanceState);
13      setContentView(R.layout.main);
14    }
15    public void onClickStart(View v) {
16      startService(new Intent(this, MyService.class).putExtra("time", 7));
17      startService(new Intent(this, MyService.class).putExtra("time", 2));
18      startService(new Intent(this, MyService.class).putExtra("time", 4));
19    }
20  }
```

Drei Service Aufgaben werden gestartet

```
12  public class MyService extends Service {
13      final String LOG_TAG = "myLogs";
14      ExecutorService es;
15      Object someRes;
16      public void onCreate() {..
22      public void onDestroy() {..
27      public int onStartCommand(Intent intent, int flags, int startId) {..
34      public IBinder onBind(Intent arg0) {..
37      class MyRun implements Runnable {..
66  }
```

Die getrennte Threads werden für jeder Aufgabe benutzt

Projekt P0931_ServiceStop 3

```
37    class MyRun implements Runnable {
38        int time;
39        int startId;
40        public MyRun(int time, int startId) {
41            this.time = time;
42            this.startId = startId;
43            Log.d(LOG_TAG, "MyRun#" + startId + " create");
44        }
45        public void run() {
46            Log.d(LOG_TAG, "MyRun#" + startId + " start, time = " + time);
47            try {
48                TimeUnit.SECONDS.sleep(time);
49            } catch (InterruptedException e) {
50                e.printStackTrace();
51            }
52            try {
53                Log.d(LOG_TAG,
54                        "MyRun#" + startId + " someRes = " + someRes.getClass());
55            } catch (NullPointerException e) {
56                Log.d(LOG_TAG, "MyRun#" + startId + " error, null pointer");
57            }
58            stop();
59        }
60        void stop() {
61            Log.d(LOG_TAG, "MyRun#" + startId + " end, stopSelf(" + startId
62                    + ")");
63            stopSelf(startId);
64        }
65    }
```

Ein Constructor

Logging & Sleeping

Das stopSelf(startID) Method stoppt den Service

Projekt P0931_ServiceStop 4

```
13      final String LOG_TAG = "myLogs";
14      ExecutorService es;
15      Object someRes;
16      public void onCreate() {
17          super.onCreate();
18          Log.d(LOG_TAG, "MyService onCreate");
19          es = Executors.newFixedThreadPool(1);
20          someRes = new Object();
21      }
22      public void onDestroy() {
23          super.onDestroy();
24          Log.d(LOG_TAG, "MyService onDestroy");
25          someRes = null;
26      }
27      public int onStartCommand(Intent intent, int flags, int startId) {
28          Log.d(LOG_TAG, "MyService onStartCommand");
29          int time = intent.getIntExtra("time", 1);
30          MyRun mr = new MyRun(time, startId);
31          es.execute(mr);
32          return super.onStartCommand(intent, flags, startId);
33      }
34      public IBinder onBind(Intent arg0) {
35          return null;
36      }
```

Nur eine Aufgabe auf Einmal!

Ein Thread mit gegebenen Parameter wird gestartet

Projekt P0931_ServiceStop 5

Tag	Text
myLogs	MyService onCreate
myLogs	MyService onStartCommand
myLogs	MyRun#1 create
myLogs	MyService onStartCommand
myLogs	MyRun#1 start, time = 7
myLogs	MyRun#2 create
myLogs	MyService onStartCommand
myLogs	MyRun#3 create
myLogs	MyRun#1 someRes = class java.lang.Object
myLogs	MyRun#1 end, stopSelf(1)
myLogs	MyRun#2 start, time = 2
myLogs	MyRun#2 someRes = class java.lang.Object
myLogs	MyRun#2 end, stopSelf(2)
myLogs	MyRun#3 start, time = 4
myLogs	MyRun#3 someRes = class java.lang.Object
myLogs	MyRun#3 end, stopSelf(3)
myLogs	MyService onDestroy

Projekt P0931_ServiceStop 6

Tag	Text
myLogs	MyService onCreate
myLogs	MyService onStartCommand
myLogs	MyRun#1 create
myLogs	MyService onStartCommand
myLogs	MyRun#2 create
myLogs	MyRun#1 start, time = 7
myLogs	MyService onStartCommand
myLogs	MyRun#3 create
myLogs	MyRun#2 start, time = 2
myLogs	MyRun#3 start, time = 4
myLogs	MyRun#2 someRes = class java.lang.Object
myLogs	MyRun#2 end, stopSelf(2)
myLogs	MyRun#3 someRes = class java.lang.Object
myLogs	MyRun#3 end, stopSelf(3)
myLogs	MyService onDestroy
myLogs	MyRun#1 error, null pointer
myLogs	MyRun#1 end, stopSelf(1)

```
16  public void onCreate() {
17      super.onCreate();
18      Log.d(LOG_TAG, "MyService onCreate");
19      es = Executors.newFixedThreadPool(3);
20      someRes = new Object();
21  }
```

- Eine Änderung: Alle drei Aufgaben laufen jetzt parallel und asynchron.
- Es gibt in Fehler mit null Pointer.
  - Der zuletzt gestartete MyRun#3 Thread ist beendet; das stopSelf(3)
  - Das Method onDestroy() hat den Objekt schon gelöscht! **Warum?**

Projekt P0931_ServiceStop 7

```
37    class MyRun implements Runnable {
38        int time;
39        int startId;
40        public MyRun(int time, int startId) {
41            this.time = time;
42            this.startId = startId;
43            Log.d(LOG_TAG, "MyRun#" + startId + " create");
44        }
45        public void run() {
46            Log.d(LOG_TAG, "MyRun#" + startId + " start, time = " + time);
47            try {
48                TimeUnit.SECONDS.sleep(time);
49            } catch (InterruptedException e) {
50                e.printStackTrace();
51            }
52            try {
53                Log.d(LOG_TAG,
54                        "MyRun#" + startId + " someRes = " + someRes.getClass());
55            } catch (NullPointerException e) {
56                Log.d(LOG_TAG, "MyRun#" + startId + " error, null pointer");
57            }
58            stop();
59        }
60        void stop() {
61            Log.d(LOG_TAG, "MyRun#" + startId + " end, stopSelf(" + startId
62                    + ")");
63            stopSelf(startId);
64        }
65    }
```

Ein Konstruktor

Logging & Sleeping

Das stopSelf(startId) Method stoppt den Service, wenn der **zuletzt gestartete** Service-Aufruf beendet ist. Es kann zur Datenverlust führen!

Beispiel

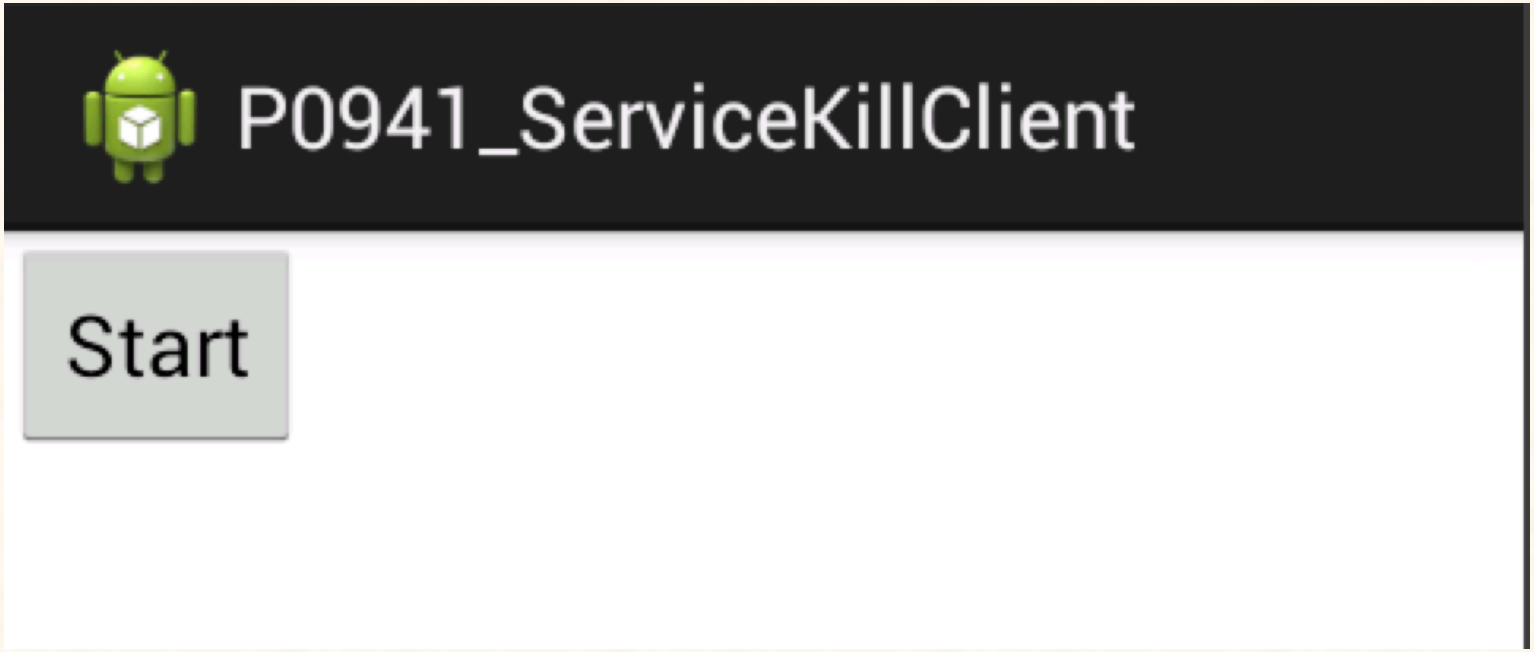
Projekte P0941_ServiceKillClient, P0942_ServiceKillServer

http://startandroid.ru/ru/uroki/vse-uroki-spiskom/159-urok-94-service-podrobno-pro-onstartcommand.html

- Ein Service läuft in der separater Anwendung
  - Wir erstellen zwei Projekte:
    - eine mit UI
    - eine nur mit einem Service
    - ein Zugriff zum Service: wie?
- Das System kann den Service beenden (Ressourcenmangel).
  - Was passiert wenn das Service neu gestartet wird?
  - Welche Optionen sind vorhanden?

P0941_ServiceKillClient 1

- Die Activity initiiert ein Service mit Hilfe eines Intentes.
- Alles Anderes passiert im Service ☺
- Wir werden den Service manuell stoppen und beobachten, was dabei passiert.



P0941_ServiceKillClient 2

```
 8  public class MainActivity extends Activity {
 9
10    /** Called when the activity is first created. */
11    @Override
12    public void onCreate(Bundle savedInstanceState) {
13      super.onCreate(savedInstanceState);
14      setContentView(R.layout.main);
15    }
16
17    public void onClickStart(View v) {
18      startService(new
19              Intent("ua.opu.brovkov.p0942_servicekillserver.MyService")
20                  .putExtra("name", "value"));
21    }
22  }
```

P0942_ServiceKillServer 1

```
10  public class MyService extends Service {
11    final String LOG_TAG = "myLogs";
12⊕   public void onCreate() {..
16⊕   public void onDestroy() {..
20⊕   public int onStartCommand(Intent intent, int flags, int startId) {..
29⊕   public IBinder onBind(Intent arg0) {..
32⊕   void readFlags(int flags) {..
38⊕   class MyRun implements Runnable {..
58  }
```

```
 9⊖    <application
10        android:icon="@drawable/ic_launcher"
11        android:label="@string/app_name" >
12⊖       <service android:name="MyService">
13⊖           <intent-filter>
14                <action android:name="ua.opu.brovkov.p0942_servicekillserver.MyService"/>
15            </intent-filter>
16        </service>
17    </application>
```

P0942_ServiceKillServer 2

Service: erstellt einen Thread mit 15 Sekunden Schlafzeit

```
38  class MyRun implements Runnable {
39    int startId;
40    public MyRun(int startId) {
41      this.startId = startId;
42      Log.d(LOG_TAG, "MyRun#" + startId + " create");
43    }
44    public void run() {
45      Log.d(LOG_TAG, "MyRun#" + startId + " start");
46      try {
47        TimeUnit.SECONDS.sleep(15);
48      } catch (InterruptedException e) {
49        e.printStackTrace();
50      }
51      stop();
52    }
53    void stop() {
54      Log.d(LOG_TAG, "MyRun#" + startId + " end, stopSelfResult("
55          + startId + ") = " + stopSelfResult(startId));
56    }
57  }
```

P0942_ServiceKillServer 3

```
10  public class MyService extends Service {
11    final String LOG_TAG = "myLogs";
12    public void onCreate() {
13      super.onCreate();
14      Log.d(LOG_TAG, "MyService onCreate");
15    }
16    public void onDestroy() {
17      super.onDestroy();
18      Log.d(LOG_TAG, "MyService onDestroy");
19    }
20    public int onStartCommand(Intent intent, int flags, int startId) {
21      Log.d(LOG_TAG, "MyService onStartCommand");
22      readFlags(flags);
23      MyRun mr = new MyRun(startId);
24      new Thread(mr).start();
25      return START_NOT_STICKY;
26      //return START_STICKY;
27      //return START_REDELIVER_INTENT;
28    }
29    public IBinder onBind(Intent arg0) {
30      return null;
31    }
32    void readFlags(int flags) {
33      if ((flags&START_FLAG_REDELIVERY) == START_FLAG_REDELIVERY)
34        Log.d(LOG_TAG, "START_FLAG_REDELIVERY");
35      if ((flags&START_FLAG_RETRY) == START_FLAG_RETRY)
36        Log.d(LOG_TAG, "START_FLAG_RETRY");
37    }
38    class MyRun implements Runnable {..
58  }
```

P0942_ServiceKillServer 4

- Ein normales Ablauf des Processes.
  - Das Process wurde gestartet und von selbst in 15 Sekunfen beendet.

Time	PID	TID	Ap	Tag	Text
03-03 08:07:36.338	2561	2561	u…	myLogs	MyService onCreate
03-03 08:07:36.338	2561	2561	u…	myLogs	MyService onStartCommand
03-03 08:07:36.338	2561	2561	u…	myLogs	MyRun#1 create
03-03 08:07:36.338	2561	2574	u…	myLogs	MyRun#1 start
03-03 08:07:51.348	2561	2574	u…	myLogs	MyRun#1 end, stopSelfResult(1) = true
03-03 08:07:51.348	2561	2561	u…	myLogs	MyService onDestroy

P0942_ServiceKillServer 5

```
20  public int onStartCommand(Intent intent, int flags, int startId) {
21    Log.d(LOG_TAG, "MyService onStartCommand");
22    readFlags(flags);
23    MyRun mr = new MyRun(startId);
24    new Thread(mr).start();
25    return START_NOT_STICKY;
26    //return START_STICKY;
27    //return START_REDELIVER_INTENT;
28  }
```

Time	PID	TID	Ap	Tag	Text
03-03 08:08:47.478	2561	2561	u…	myLogs	MyService onCreate
03-03 08:08:47.478	2561	2561	u…	myLogs	MyService onStartCommand
03-03 08:08:47.478	2561	2561	u…	myLogs	MyRun#1 create
03-03 08:08:47.488	2561	2575	u…	myLogs	MyRun#1 start

- Das Prozess wurde unterbrochenen. Er erstellt sich von selbst nicht mehr.

P0942_ServiceKillServer 6

```
20  public int onStartCommand(Intent intent, int flags, int startId) {
21    Log.d(LOG_TAG, "MyService onStartCommand");
22    readFlags(flags);
23    MyRun mr = new MyRun(startId);
24    new Thread(mr).start();
25    //return START_NOT_STICKY;
26    return START_STICKY;
27    //return START_REDELIVER_INTENT;
28  }
```

Time	PID	TID	Ap	Tag	Text
03-03 08:20:01.388	2597	2597	u…	myLogs	MyService onCreate
03-03 08:20:01.388	2597	2597	u…	myLogs	MyService onStartCommand
03-03 08:20:01.399	2597	2597	u…	myLogs	MyRun#1 create
03-03 08:20:01.399	2597	2610	u…	myLogs	MyRun#1 start
03-03 08:20:15.088	2611	2611	u…	myLogs	MyService onCreate
03-03 08:20:15.088	2611	2611	u…	myLogs	MyService onStartCommand
03-03 08:20:15.088	2611	2611	u…	myLogs	MyRun#2 create
03-03 08:20:15.099	2611	2624	u…	myLogs	MyRun#2 start
03-03 08:20:30.103	2611	2624	u…	myLogs	MyRun#2 end, stopSelfResult(2) = true
03-03 08:20:30.103	2611	2611	u…	myLogs	MyService onDestroy

- Das Prozess wurde unterbrochenen und startet sich von selbst neu. Der ursprüngliche Service-Request #1 wurde aber verloren; statt dessen wurde ein neues Service #2 erstellt.

P0942_ServiceKillServer 7

```
20  public int onStartCommand(Intent intent, int flags, int startId) {
21    Log.d(LOG_TAG, "MyService onStartCommand");
22    readFlags(flags);
23    MyRun mr = new MyRun(startId);
24    new Thread(mr).start();
25    //return START_NOT_STICKY;
26    //return START_STICKY;
27    return START_REDELIVER_INTENT;
28  }
```

Time	PID	TID	Ap	Tag	Text
03-03 08:28:11.328	2645	2645	u…	myLogs	MyService onCreate
03-03 08:28:11.328	2645	2645	u…	myLogs	MyService onStartCommand
03-03 08:28:11.328	2645	2645	u…	myLogs	MyRun#1 create
03-03 08:28:11.328	2645	2658	u…	myLogs	MyRun#1 start
03-03 08:28:30.558	2659	2659	u…	myLogs	MyService onCreate
03-03 08:28:30.558	2659	2659	u…	myLogs	MyService onStartCommand
03-03 08:28:30.558	2659	2659	u…	myLogs	START_FLAG_REDELIVERY
03-03 08:28:30.558	2659	2659	u…	myLogs	MyRun#1 create
03-03 08:28:30.568	2659	2672	u…	myLogs	MyRun#1 start
03-03 08:28:45.574	2659	2672	u…	myLogs	MyRun#1 end, stopSelfResult(1) = true
03-03 08:28:45.574	2659	2659	u…	myLogs	MyService onDestroy

- Das Prozess wurde unterbrochenen und startet sich von selbst neu. Der ursprüngliche Service-Request #1 wurde wiederhergestellt und richtig verarbeitet.

Services

Fragen?