КАЛУЖСКИЙ ФИЛИАЛ
ФЕДЕРАЛЬНОГО ГОСУДАРСТВЕННОГО БЮДЖЕТНОГО
ОБРАЗОВАТЕЛЬНОГО УЧРЕЖДЕНИЯ ВЫСШЕГО ОБРАЗОВАНИЯ
«МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
ИМЕНИ Н.Э. БАУМАНА (национальный исследовательский университет)»

Факультет «Информатика и управление"

Кафедра "Программное обеспечение ЭВМ, информационные технологии"

Сервисы

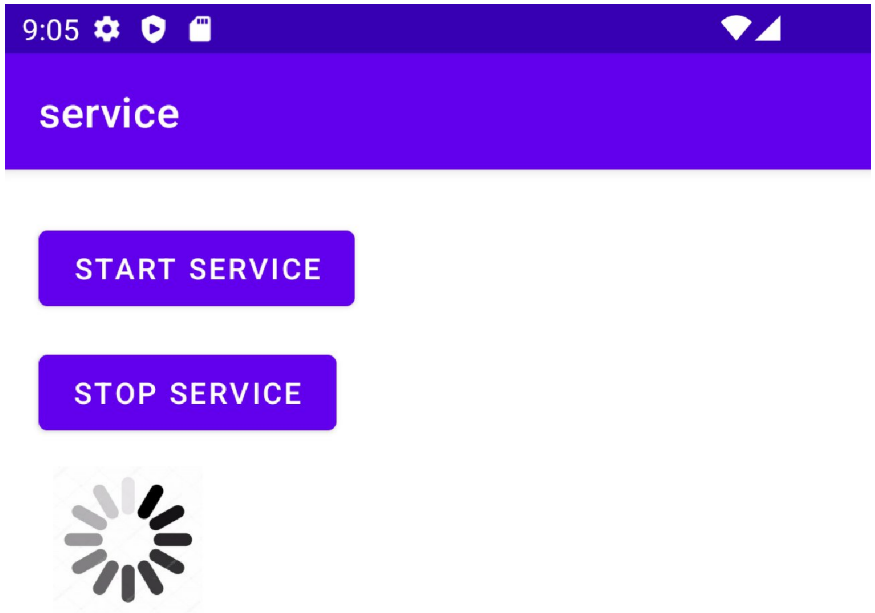
Калуга

Сервис – это некая задача, которая работает в фоне и не использует UI. Запускать и останавливать сервис можно из приложений и других сервисов. Также можно подключиться к уже работающему сервису и взаимодействовать с ним.
В качестве примера можно рассмотреть алгоритм почтовой программы. Она состоит из приложения и сервиса. Сервис работает в фоне и периодически проверяет наличие новой почты, скачивает ее и выводит уведомления. А когда вы запускаете приложение, оно отображает вам эти загруженные сервисом письма. Также приложение может подключиться к сервису и поменять в нем, например, период проверки почты или совсем закрыть сервис, если постоянная проверка почты больше не нужна.
Т.е. сервис нужен, чтобы ваша задача продолжала работать, даже когда приложение закрыто

```
public class MyService extends Service {

	final String LOG_TAG = "myLogs";

	public void onCreate() {
		super.onCreate();
		Log.d(LOG_TAG, "onCreate");
	}

	public int onStartCommand(Intent intent, int flags, int
startId) {
		Log.d(LOG_TAG, "onStartCommand");
		someTask();
		return super.onStartCommand(intent, flags, startId);
	}

	public void onDestroy() {
		super.onDestroy();
		Log.d(LOG_TAG, "onDestroy");
	}

	public IBinder onBind(Intent intent) {
		Log.d(LOG_TAG, "onBind");
		return null;
	}

	void someTask() {}
}
```

```
<service android:name=".MyService"
    android:singleUser="true">
    <intent-filter>
        <action
android:name="android.service.example" />
    </intent-filter>
</service>
```

```
public class MainActivity extends Activity {

    final String LOG_TAG = "myLogs";

    public void onCreate(Bundle
savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

    }

    public void onClickStart(View v) {
        startService(new Intent(this,
MyService.class));
    }

    public void onClickStop(View v) {
        stopService(new Intent(this,
MyService.class));
    }
}
```



```
void someTask() {
    for (int i = 1; i<=5; i++) {
        Log.d(LOG_TAG, "i = " + i);
        try {

TimeUnit.SECONDS.sleep(1);
        } catch (InterruptedException e)
{
            e.printStackTrace();
        }
    }
}
```

VS

```
void someTask() {
    new Thread(new Runnable() {
        public void run() {
            for (int i = 1; i<=5; i++) {
                Log.d(LOG_TAG, "i = " + i);
                try {

TimeUnit.SECONDS.sleep(1);
                } catch (InterruptedException
e) {
                    e.printStackTrace();
                }
            }
            stopSelf();
        }
    }).start();
}
```

Обратная связь через PendingIntent

•в Activity создать PendingIntent с помощью метода createPendingResult
•поместить PendingIntent в обычный Intent, который используется для старта сервиса и вызывов startService
•в сервисе извлекается PendingIntent из полученного в методе onStartCommand объекта Intent
•когда необходимо передать результаты работы из сервиса в Activity, вызывается метод send для объекта PendingIntent
•эти результаты из сервиса можно отловить в Activity в методе onActivityResult

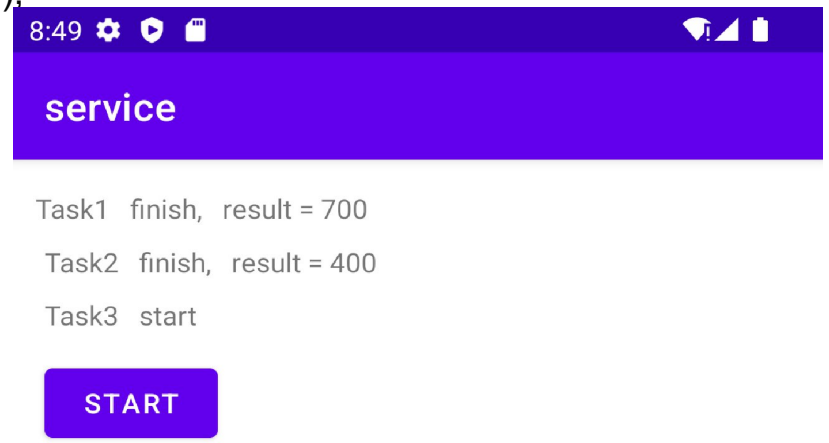
```
public class MainActivity extends Activity {
    final String LOG_TAG = "myLogs";
    final int TASK1_CODE = 1;
    final int TASK2_CODE = 2;
    final int TASK3_CODE = 3;
    public final static int STATUS_START = 100;
    public final static int STATUS_FINISH = 200;
    public final static String PARAM_TIME = "time";
    public final static String PARAM_PINTENT =
"pendingIntent";
    public final static String PARAM_RESULT = "result";
    TextView tvTask1, tvTask2, tvTask3;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        tvTask1 = (TextView) findViewById(R.id.tvTask1);
        tvTask1.setText("Task1");
        tvTask2 = (TextView) findViewById(R.id.tvTask2);
        tvTask2.setText("Task2");
        tvTask3 = (TextView) findViewById(R.id.tvTask3);
        tvTask3.setText("Task3");
    }
```

```
public void onClickStart(View v) {
    PendingIntent pi;
    Intent intent;
    pi = createPendingResult(TASK1_CODE, null, 0);
    intent = new Intent(this, MyService.class)
.putExtra(PARAM_TIME,
7).putExtra(PARAM_PINTENT, pi);
    startService(intent);
    pi = createPendingResult(TASK2_CODE, null, 0);
    intent = new Intent(this, MyService.class)
.putExtra(PARAM_TIME,
4).putExtra(PARAM_PINTENT, pi);
    startService(intent);
    pi = createPendingResult(TASK3_CODE, null, 0);
    intent = new Intent(this, MyService.class)
.putExtra(PARAM_TIME,
6).putExtra(PARAM_PINTENT, pi);
    startService(intent);
}
```

```
@Override
    protected void onActivityResult(int requestCode, int resultCode,
                                    Intent data) {
        super.onActivityResult(requestCode, resultCode, data);
        Log.d(LOG_TAG, "requestCode = " + requestCode + ",
                resultCode = " + resultCode);
        if (resultCode == STATUS_START) {
            switch (requestCode) {
                case TASK1_CODE:
                    tvTask1.setText("Task1 start");
                    break;
                case TASK2_CODE:
                    tvTask2.setText("Task2 start");
                    break;
                case TASK3_CODE:
                    tvTask3.setText("Task3 start");
                    break;
            }
        }
        if (resultCode == STATUS_FINISH) {
            int result = data.getIntExtra(PARAM_RESULT, 0);
            switch (requestCode) {
                case TASK1_CODE:
                    tvTask1.setText("Task1 finish, result = " + result); break;
                case TASK2_CODE:
                    tvTask2.setText("Task2 finish, result = " + result); break;
                case TASK3_CODE:
                    tvTask3.setText("Task3 finish, result = " + result); break;
            }
        }
    }
```

```
class MyRun implements Runnable {
    int time;
    int startId;
    PendingIntent pi;
    public MyRun(int time, int startId, PendingIntent pi) {
        this.time = time;
        this.startId = startId;
        this.pi = pi;
        Log.d(LOG_TAG, "MyRun#" + startId + " create");
    }
    public void run() {
        Log.d(LOG_TAG, "MyRun#" + startId + " start, time = " + time);
        try {
            pi.send(MainActivity.STATUS_START);
            TimeUnit.SECONDS.sleep(time);
            Intent intent = new Intent().putExtra(MainActivity.PARAM_RESULT, time * 100);
            pi.send(MyService.this, MainActivity.STATUS_FINISH, intent);

        } catch (InterruptedException e) {
            e.printStackTrace();
        } catch (CanceledException e) {
            e.printStackTrace();
        }
        stop();
    }
    void stop() {
        Log.d(LOG_TAG, "MyRun#" + startId + " end, stopSelfResult("
                + startId + ") = " + stopSelfResult(startId));
    }
}
```

```
public class MyService extends Service {
    final String LOG_TAG = "myLogs";
    ExecutorService es;

    public void onCreate() {
        super.onCreate();
        Log.d(LOG_TAG, "MyService onCreate");
        es = Executors.newFixedThreadPool(2);
    }

    public void onDestroy() {
        super.onDestroy();
        Log.d(LOG_TAG, "MyService onDestroy");
    }

    public int onStartCommand(Intent intent, int flags, int startId) {
        Log.d(LOG_TAG, "MyService onStartCommand");

        int time = intent.getIntExtra(MainActivity.PARAM_TIME, 1);
        PendingIntent pi =
intent.getParcelableExtra(MainActivity.PARAM_PINTENT);

        MyRun mr = new MyRun(time, startId, pi);
        es.execute(mr);

        return super.onStartCommand(intent, flags, startId);
    }

    public IBinder onBind(Intent arg0) {
        return null;
    }
```



Обратная связь через BroadcastReciever

•в Activity создаем BroadcastReceiver, а также создаем IntentFilter, настроенный на определенный Action, и регистрируем (включаем) эту пару. Теперь BroadcastReceiver будет получать Intent-ы подходящие под условия IntentFilter

•в сервисе, когда нам понадобится передать данные в Activity, мы создаем Intent (с Action из предыдущего пункта), кладем в него данные, которые хотим передать, и посылаем его на поиски BroadcastReceiver

•BroadcastReceiver в Activity ловит этот Intent и извлекает из него данные

```
    setContentView(R.layout.main);
    tvTask1 = (TextView) findViewById(R.id.tvTask1);
    tvTask1.setText("Task1");
    tvTask2 = (TextView) findViewById(R.id.tvTask2);
    tvTask2.setText("Task2");
    tvTask3 = (TextView) findViewById(R.id.tvTask3);
    tvTask3.setText("Task3");
    br = new BroadcastReceiver() {
        public void onReceive(Context context, Intent
intent) {
            int task = intent.getIntExtra(PARAM_TASK,
0);
            int status =
intent.getIntExtra(PARAM_STATUS, 0);
            Log.d(LOG_TAG, "onReceive: task = " +
task + ", status = " +   status);
            if (status  == STATUS_START) {
                switch (task) {
                    case TASK1_CODE:
                        tvTask1.setText("Task1 start");
                        break;
…
                }
            }
            if (status == STATUS_FINISH) {
                int result =
intent.getIntExtra(PARAM_RESULT, 0);
                switch (task) {
                  case TASK1_CODE:
                  tvTask1.setText("Task1 finish, result = "
+ result);
                  break;
```

```
                }
            }
        };
        IntentFilter intFilt = new
IntentFilter(BROADCAST_ACTION);
        registerReceiver(br, intFilt);
    }
    @Override
    protected void onDestroy() {
        super.onDestroy();
        unregisterReceiver(br);
    }

    public void onClickStart(View v) {
        Intent intent;
        intent = new Intent(this,
MyService.class).putExtra(PARAM_TIME, 7)
                .putExtra(PARAM_TASK,
TASK1_CODE);
        startService(intent);

        intent = new Intent(this,
MyService.class).putExtra(PARAM_TIME, 4)
                .putExtra(PARAM_TASK,
TASK2_CODE);
        startService(intent);

        intent = new Intent(this,
MyService.class).putExtra(PARAM_TIME, 6)
                .putExtra(PARAM_TASK,
TASK3_CODE);
        startService(intent);
    }
```

```
class MyRun implements Runnable {
	int time;
	int startId;
	int task;
	public MyRun(int startId, int time, int task) {
		this.time = time;
		this.startId = startId;
		this.task = task;
		Log.d(LOG_TAG, "MyRun#" + startId + " create");
	}
	public void run() {
		Intent intent = new Intent(MainActivity.BROADCAST_ACTION);
		Log.d(LOG_TAG, "MyRun#" + startId + " start, time = " + time);
		try {
			intent.putExtra(MainActivity.PARAM_TASK, task);
			intent.putExtra(MainActivity.PARAM_STATUS, ainActivity.STATUS_START);
			sendBroadcast(intent);
			TimeUnit.SECONDS.sleep(time);
			intent.putExtra(MainActivity.PARAM_STATUS, ainActivity.STATUS_FINISH);
			intent.putExtra(MainActivity.PARAM_RESULT, time * 100);
			sendBroadcast(intent);
		} catch (InterruptedException e) {
			e.printStackTrace();
		}
		stop();
	}
	void stop() {
		Log.d(LOG_TAG, "MyRun#" + startId + " end, stopSelfResult(" + startId + ") = " +
stopSelfResult(startId));
	}
}
```

```
public class MyService extends Service {

    final String LOG_TAG = "myLogs";
    ExecutorService es;

    public void onCreate() {
        super.onCreate();
        Log.d(LOG_TAG, "MyService onCreate");
        es = Executors.newFixedThreadPool(2);
    }

    public void onDestroy() {
        super.onDestroy();
        Log.d(LOG_TAG, "MyService onDestroy");
    }

    public int onStartCommand(Intent intent, int flags, int startId) {
        Log.d(LOG_TAG, "MyService onStartCommand");
        int time = intent.getIntExtra(MainActivity.PARAM_TIME, 1);
        int task = intent.getIntExtra(MainActivity.PARAM_TASK, 0);
        MyRun mr = new MyRun(startId, time, task);
        es.execute(mr);
        return super.onStartCommand(intent, flags, startId);
    }

    public IBinder onBind(Intent arg0) {
        return null;
    }
}
```

Биндинг

```
public class MainActivity extends Activity {
    final String LOG_TAG = "myLogs";
    boolean bound = false;
    ServiceConnection sConn;
    Intent intent;

    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        intent = new Intent("android.MyService");
            sConn = new ServiceConnection() {
                public void
onServiceConnected(ComponentName name,
                                        IBinder binder) {
                    Log.d(LOG_TAG, "MainActivity
onServiceConnected");
                    bound = true;
                }
                public void
onServiceDisconnected(ComponentName name) {
                    Log.d(LOG_TAG, "MainActivity
onServiceDisconnected");
                    bound = false;
                }
            };
    }

    public void onClickStart(View v) {
        startService(intent);
    }
    public void onClickStop(View v) {
        stopService(intent);
    }

    public void onClickBind(View v) {
        bindService(intent, sConn,
BIND_AUTO_CREATE);
    }

    public void onClickUnBind(View v) {
        if (!bound) return;
        unbindService(sConn);
        bound = false;
    }

    protected void onDestroy() {
        super.onDestroy();
        onClickUnBind(null);
    }
}
```

```
public class MyService extends Service {

    final String LOG_TAG = "myLogs";

    public void onCreate() {
        super.onCreate();
        Log.d(LOG_TAG, "MyService onCreate");
    }

    public IBinder onBind(Intent intent) {
        Log.d(LOG_TAG, "MyService onBind");
        return new Binder();
    }

    public void onRebind(Intent intent) {
        super.onRebind(intent);
        Log.d(LOG_TAG, "MyService onRebind");
    }

    public boolean onUnbind(Intent intent) {
        Log.d(LOG_TAG, "MyService onUnbind");
        return super.onUnbind(intent);
    }

    public void onDestroy() {
        super.onDestroy();
        Log.d(LOG_TAG, "MyService onDestroy");
    }
}
```

Отправка уведомлений

```
public class MyService extends Service {
    NotificationManager nm;
    @Override
    public void onCreate() {
        super.onCreate();
        nm = (NotificationManager) getSystemService(NOTIFICATION_SERVICE);
    }
    public int onStartCommand(Intent intent, int flags, int startId) {
        try {
            TimeUnit.SECONDS.sleep(5);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
        sendNotif();
        return super.onStartCommand(intent, flags, startId);
    }
    void sendNotif() {
        Notification notif = new Notification(R.drawable.ic_launcher, "Text in status bar”
System.currentTimeMillis());
        Intent intent = new Intent(this, MainActivity.class);
        intent.putExtra(MainActivity.FILE_NAME, "somefile");
        PendingIntent pIntent = PendingIntent.getActivity(this, 0, intent, 0);
        notif.setLatestEventInfo(this, "Notification's title", "Notification's text", pIntent);
        notif.flags |= Notification.FLAG_AUTO_CANCEL;
        nm.notify(1, notif);
    }
    public IBinder onBind(Intent arg0) {
        return null;
    }
}
```

Foreground

Вы можете сообщить системе, что ваш сервис очень важен для пользователя и его нельзя удалять при нехватке памяти (или удалении активности). Это актуально, например, для музыкального плеера. В статус-бар при этом будет помещено уведомление.

Делается это методом startForeground (int id, Notification notification).

На вход он принимает те же параметры, что и NotificationManager.notify – ID и Notification.

Т.е. вы создаете уведомление, назначаете ему ID и передаете это в startForeground

Автозагрузка

Сервисы для получения погоды или почты имеет смысл помещать в автозагрузку. Для этого нам надо создать BroadcastReceiver, настроить его IntentFilter на Action = android.intent.action.BOOT_COMPLETED, и добавить права android.permission.RECEIVE_BOOT_COMPLETED. Этот BroadcastReceiver будет вызван системой при старте системы и в нем мы кодим запуск сервиса.

Допустим, есть проект с сервисом MyService.

Создаем в проекте класс MyBroadReceiv

```
public class MyBroadReceiv extends BroadcastReceiver {
  final String LOG_TAG = "myLogs";


  public void onReceive(Context context, Intent intent) {
    Log.d(LOG_TAG, "onReceive " + intent.getAction());
    context.startService(new Intent(context, MyService.class));
  }
}
```