Services and Broadcast

Services

Services

- Не требуют UI
- Выполняются в фоне
- Выполняются в главном потоке
- Могут работать постоянно, пока позволяют ресурсы
- Повышает приоритет приложения

Services

- Проигрывание музыки
- Синхронизация данных
- Скачивание файлов
- и т.д.

Иерархия процессов

- Foreground
- Visible
- Service with Foreground
- Service
- Background
- Empty

Manifest

```
<application
    <service
        android:name=".MyService”
        android:description="string resource”
        android:enabled=["true" | "false"]
        android:exported=["true" | "false"]
        android:icon="drawable resource"
        android:isolatedProcess=["true" | "false"]
        android:label="string resource"
        android:permission="string"
        android:process="string" >
        . . .
    </service>
</application>
```

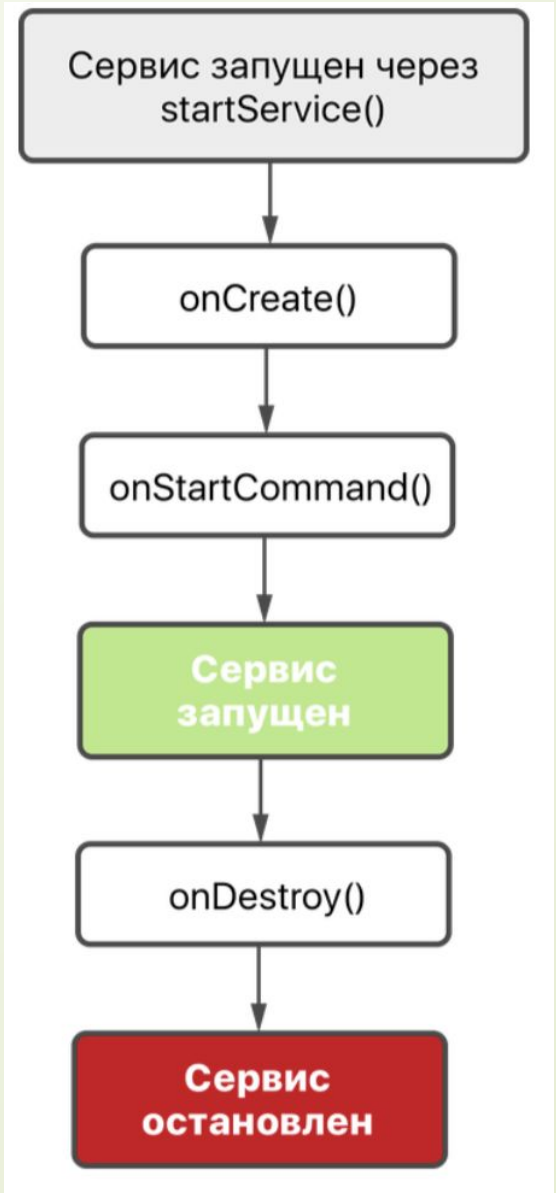
Service

```
public class MyService extends Service {
    public void onCreate() {
        super.onCreate();
    }

    public int onStartCommand(Intent intent, int flags, int startId) {
        return START_STICKY;
    }

    public void onDestroy() {
        super.onDestroy();
    }

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

Flags

- START_STICKY — перезапустить с Intent = null
- START_REDELIVER_INTENT — перезапустить с последним Intent
- START_NOT_STICKY — не перезапускать

Жизненный цикл



Уничтожение

- stopService(Intent)
- stopSelf()
- Ручное уничтожение приложения
- Сервис уничтожен системой

Service with Foreground

Service with Foreground

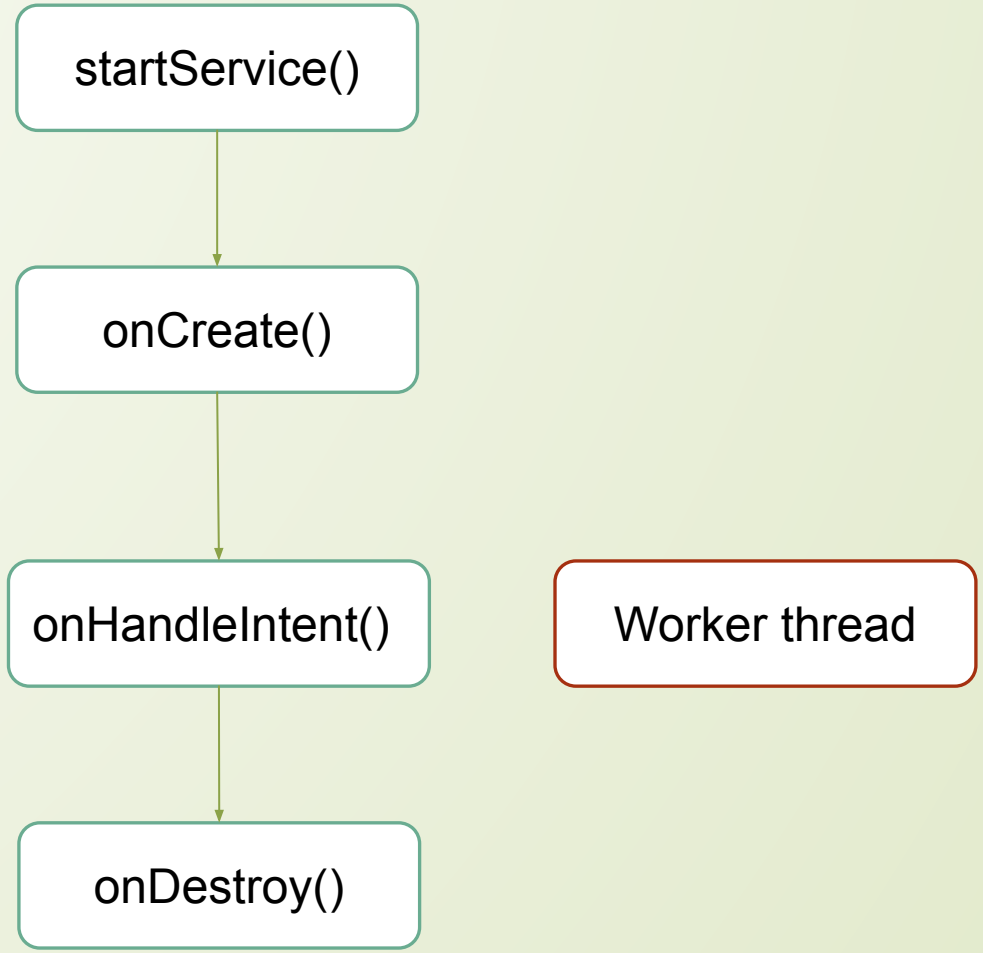
- Повышение приоритета
- Выводит уведомление в строку состояния

IntentService

IntentService

- Обеспечивает асинхронность выполнения
- Завершается автоматически после выполнения
- Обеспечивает очередь

IntentService



IntentService

```
public class MyIntentService extends IntentService {

    public MyIntentService() {
        super("name");
    }

    protected void onHandleIntent(Intent intent) {

    }
}
```

PendingIntent

PendingIntent

```
private final static int REQUEST_CODE = 1;
private final static String PENDING_KEY = "pending_key";

protected void onCreate(Bundle savedInstanceState) {

        Intent intent = new Intent(MainActivity.this, MyService.class);
        PendingIntent pi = createPendingResult(REQUEST_CODE, intent, flag);
        intent.putExtra(PENDING_KEY, pi);
        startService(intent);

}


protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    if (requestCode == REQUEST_CODE) {
        // code
    }
}
```

PendingIntent

```
public class MyService extends Service {
    public static final int RESULT_CODE = 12;
    public int onStartCommand(Intent intent, int flags, int startId) {
        PendingIntent pi = intent.getParcelableExtra(MainActivity.PENDING_KEY);
        Intent data = new Intent();
        pi.send(MyService.this, RESULT_CODE, data);
        return START_NOT_STICKY;
    }
}
```

PendingIntent.flags

- PendingIntent.*FLAG_UPDATE_CURRENT – заменит extra в существующем*
- PendingIntent.*FLAG_CANCEL_CURRENT – удалит существующий*
- PendingIntent.*FLAG_NO_CREATE – если нет похожего, то не будет создан*
- PendingIntent.*FLAG_ONE_SHOT – сработает только один раз*

Binding

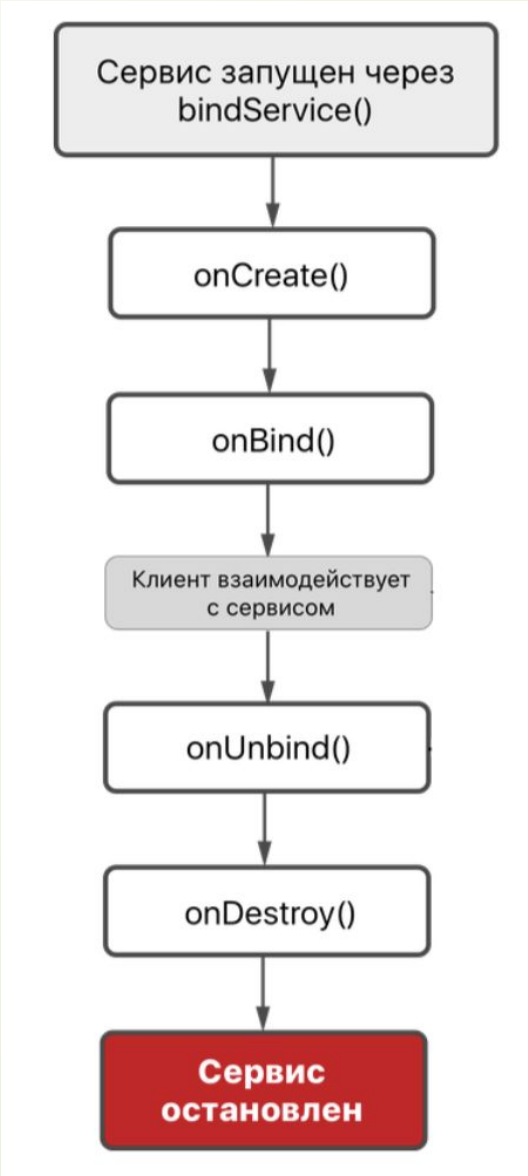
Binding

```
public class BindingService extends Service {
 public class MyBinder extends Binder {
        BindingService getService() {
            return BindingService.this;
        }
    }


    @Override
    public IBinder onBind(Intent intent) {
        return new MyBinder();
    }
}
```

Binding

```
ServiceConnection connection = new ServiceConnection() {
    @Override
    public void onServiceConnected(ComponentName name, IBinder service) {
        BindingService bindingService = ((BindingService.MyBinder) service).getService();
        ...
    }

    @Override
    public void onServiceDisconnected(ComponentName name) {
    }
};
```

Жизненный цикл



BroadcastReceiver

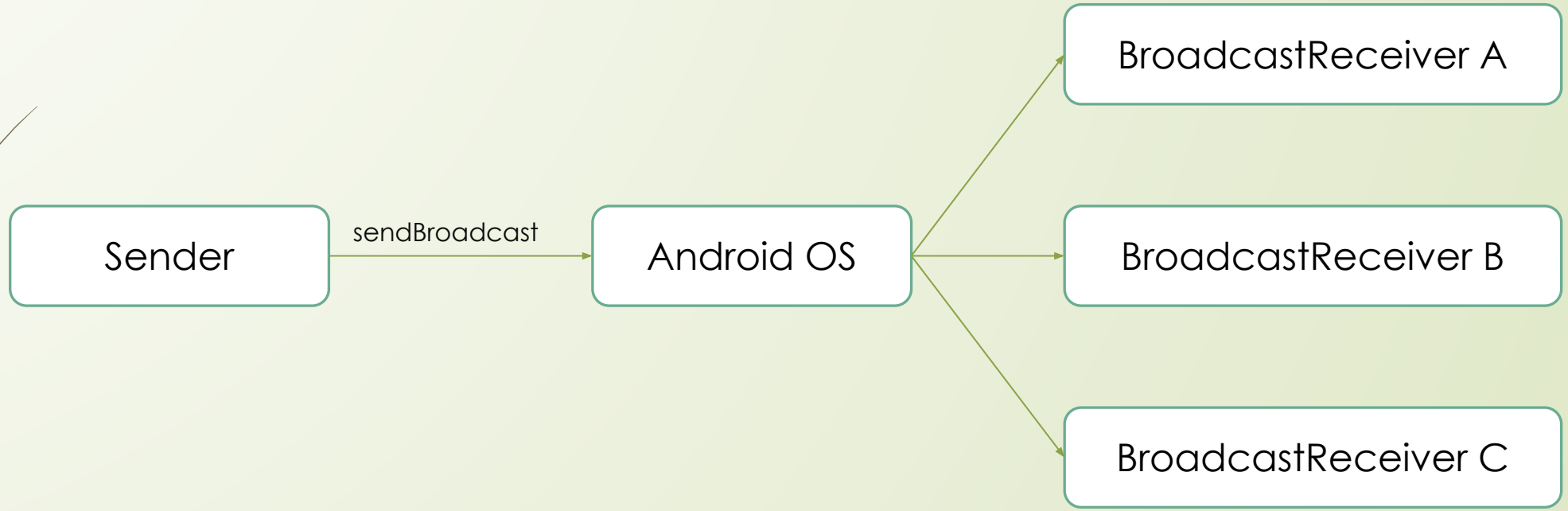
BroadcastReceiver

Приемник широковещательных сообщений – компонент для получения внешних событий и реакций на них.

Примеры

- Подключение устройства к источнику питания
- Нажатие на кнопку камеры
- Установка нового приложения
- Автозапуск приложения
- Входящие звонки
- Входящие смс

BroadcastReceiver



BroadcastReceiver

```
public class MessageReceiver extends BroadcastReceiver {


    @Override
    public void onReceive(Context context, Intent intent) {

        // code
    }
}
```

Регистрация Broadcast. Способ 1

```
<receiver
    android:name=".MessageReceiver"
    android:enabled="true"
    android:exported="true">
    <intent-filter>
        <action android:name="android.intent.action.ACTION_POWER_DISCONNECTED" />
        <action android:name="android.intent.action.ACTION_POWER_CONNECTED"/>
    </intent-filter>
</receiver>
```

Регистрация Broadcast. Способ 1

ПЛЮСЫ:

- Получаем уведомление всегда, даже если приложение не запущено.

МИНУСЫ:

- Не можем остановить получение уведомлений.
- Из метода onReceive() не имеем доступа к интерфейсу.
- Получаем уведомление всегда, даже если приложение не запущено.

КОГДА ИСПОЛЬЗОВАТЬ:

- Когда нужно получать уведомления всегда и всюду, даже если приложение не запущено.

Регистрация Broadcast. Способ 2

```
BroadcastReceiver br = new MessageReceiver();
IntentFilter filter = new IntentFilter();
filter.addAction(Intent.ACTION_POWER_DISCONNECTED);
filter.addAction(Intent.ACTION_POWER_CONNECTED);
registerReceiver(br, filter);


unregisterReceiver(br)
```

Регистрация Broadcast. Способ 2

ПЛЮСЫ:

- Получаем уведомления только тогда когда это нужно;
- Сами контролируем когда включить уведомления, а когда отключить.

МИНУСЫ:

- Из метода onReceive() по прежнему нет доступа к интерфейсу.

КОГДА ИСПОЛЬЗОВАТЬ:

- Когда нужно самим контролировать включение и отключение уведомлений, при условии, что не нужно изменять что-то в интерфейсе.

Регистрация Broadcast. Способ 3

```
BroadcastReceiver br = new BroadcastReceiver() {
    @Override
    public void onReceive(Context context, Intent intent) {
        // code
    }
};
IntentFilter filter = new IntentFilter();
filter.addAction(Intent.ACTION_POWER_DISCONNECTED);
filter.addAction(Intent.ACTION_POWER_CONNECTED);
registerReceiver(br, filter);


unregisterReceiver(br)
```

Регистрация Broadcast. Способ 3

ПЛЮСЫ:

- Получаем уведомления только тогда когда нужно;
- Сами контролируем когда включить уведомления, а когда отключить.
- Имеем доступ к интерфейсу

МИНУСЫ:

- Их нет.

КОГДА ИСПОЛЬЗОВАТЬ:

- Тогда, когда нужно самим контролировать включение и отключение уведомлений, при этом нужно изменять что-то в интерфейсе.

Примеры сообщений

- android.intent.action.ACTION_POWER_DISCONNECTED
- android.intent.action.ACTION_POWER_CONNECTED
- android.intent.action.ACTION_BATTERY_LOW
- android.intent.action.ACTION_BATTERY_OKAY
- android.intent.action.SCREEN_OFF
- android.intent.action.SCREEN_ON
- android.provider.Telephony.SMS_RECEIVED