**Введение в сервисы Android**

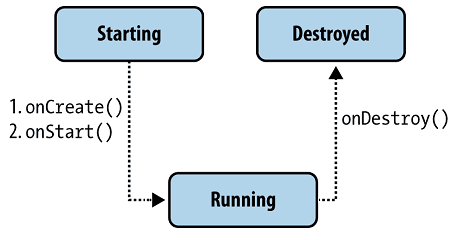
Сервисы представляют собой особую организацию приложения. В отличие от activity они не требуют наличия визуального интерфейса. Сервисы позволяют выполнять долговременные задачи без вмешательства пользователя.

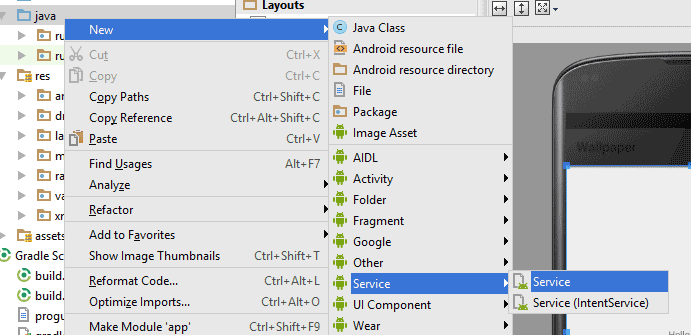
Все сервисы наследуются от класса **Service** и проходят следующие этапы жизненного цикла:

* Метод onCreate(): вызывается при создании сервиса
* Метод onStartCommand(): вызывается при получении сервисом команды, отправленной с помощью метода startService()
* Метод onBind(): вызывается при закреплении клиента за сервисом с помощью метода bindService()
* Метод onDestroy(): вызывается при завершении работы сервиса

## Жизненный цикл сервиса

Подобно активностям служба имеет свои методы жизненного цикла:





Создадим простейшее приложение с сервисом. Наш сервис будет воспроизводить музыкальный файл. И вначале добавим в проект в каталог *res* папку *raw* и в нее поместим какой-нибудь mp3-файл. Затем добавим новый класс сервиса. Назовем его MediaService:

|  |
| --- |
| package com.example.eugene.soundserviceapp;    import android.app.Service;  import android.content.Intent;  import android.os.IBinder;  import android.media.MediaPlayer;    public class MediaService extends Service {      MediaPlayer ambientMediaPlayer;      @Override      public IBinder onBind(Intent intent) {            throw new UnsupportedOperationException("Not yet implemented");      }      @Override      public void onCreate(){          ambientMediaPlayer=MediaPlayer.create(this, R.raw.zymotyx);          ambientMediaPlayer.setLooping(true);      }      @Override      public int onStartCommand(Intent intent, int flags, int startId){          ambientMediaPlayer.start();          return START\_STICKY;      }      @Override      public void onDestroy() {          ambientMediaPlayer.stop();      }  } |

Для воспроизведения музыкального файла сервис будет использовать компонент **MediaPlayer**.

В сервисе переопределяются все четыре метода жизненного цикла. Но по сути метод onBind() не имеет никакой реализации.

В методе onCreate() инициализируется медиа-проигрыватель с помощью музыкального ресурса, который добавлен в папку res/raw.

В методе onStartCommand() начинается воспроизведение.

Метод onStartCommand() может возвращать одно из значений, которое предполагает различное поведение в случае, если процесс сервиса был неожиданно завершен системой:

* **START\_STICKY**: в этом случае сервис снова возвращается в запущенное состояние, как будто если бы снова был бы вызван метод onStartCommand() без передачи в этот метод объекта Intent
* **START\_REDELIVER\_INTENT**: в этом случае сервис снова возвращается в запущенное состояние, как будто если бы снова был бы вызван метод onStartCommand() с передачей в этот метод объекта Intent
* **START\_NOT\_STICKY**: сервис остается в остановленном положении

Метод onDestroy() завершает воспроизведение.

Чтобы управлять сервисом, изменим activity. Сначала добавим в layout прау кнопок для управления сервисом:

<?xml version="1.0" encoding="utf-8"?>

<LinearLayout xmlns:android="<http://schemas.android.com/apk/res/android>"

    android:layout\_width="match\_parent"

    android:layout\_height="match\_parent"

    android:orientation="horizontal">

    <Button

        android:id="@+id/start"

        android:layout\_width="match\_parent"

        android:layout\_height="wrap\_content"

        android:layout\_weight="1"

        android:text="Старт"

        android:onClick="click"/>

    <Button

        android:id="@+id/stop"

        android:layout\_width="match\_parent"

        android:layout\_height="wrap\_content"

        android:layout\_weight="1"

        android:text="Стоп"

        android:onClick="click"/>

</LinearLayout>

И изменим код activity:

package com.example.eugene.soundserviceapp;

import android.support.v7.app.ActionBarActivity;

import android.os.Bundle;

import android.view.Menu;

import android.view.MenuItem;

import android.content.Intent;

import android.view.View;

public class MainActivity extends ActionBarActivity {

    @Override

    protected void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);

        setContentView(R.layout.activity\_main);

    }

    public void click(View v) {

        Intent i=new Intent(this, MediaService.class);

        if (v.getId()==R.id.start) {

            startService(i);

        }

        else {

            stopService(i);

        }

    }

}

Для запуска сервиса используется объект Intent:

|  |  |
| --- | --- |
|  | Intent i=new Intent(this, MediaService.class); |

Для запуска сервиса в классе Activity определен метод **startService()**, в который передается объект Intent. Этот метод будет посылать команду сервису и вызывать его метод onStartCommand(), а также указывать системе, что сервис должен продолжать работать до тех пор, пока не будет вызван метод stopService().

Метод **stopService()** также определен к классе Activity и принимает объект Intent. Он останавливает работу сервиса, вызывая его метод onDestroy()

И в конце нам надо зарегистрировать сервис в файле манифеста:

<?xml version="1.0" encoding="utf-8"?>

<manifest xmlns:android="<http://schemas.android.com/apk/res/android>"

    package="com.example.eugene.soundserviceapp" >

    <application

        android:allowBackup="true"

        android:icon="@mipmap/ic\_launcher"

        android:label="@string/app\_name"

        android:theme="@style/AppTheme" >

        <service

            android:name=".MediaService"

            android:enabled="true"

            android:exported="true" >

        </service>

        <activity

            android:name=".MainActivity"

            android:label="@string/app\_name" >

            <intent-filter>

                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />

            </intent-filter>

        </activity>

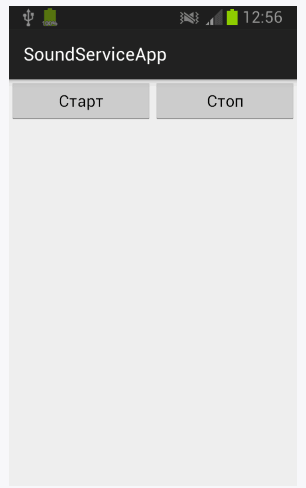
    </application>

</manifest>

Регистрация сервиса производится в узле application с помощью добавления элемента <service>. В нем определяется атрибут android:name, который хранит название класса сервиса. И кроме того может принимать еще ряд атрибутов:

* android:enabled: если имеет значение "true", то сервис может ли создаваться системой. Значение по умолчанию - "true".
* android:exported: указывает, могут ли другие компоненты приложения обращаться к сервису. Если имеет значение "true", то могут, если имеет значение "false", то нет.
* android:icon: значок сервиса, представляет собой ссылку на ресурс drawable
* android:isolatedProcess: если имеет значение true, то сервис может быть запущен как специальный процесс, изолированный от остальной системы.
* android:label: название сервиса, которое отображается пользователю
* android:permission: набор разрешений, которые должно применять приложение для запуска сервиса
* android:process: название процесса, в котором запущен сервис. Как правило, имеет то же название, что и пакет приложения.

Запустим приложение и нажмем на кнопку запуска сервиса:



После этого начнется воспроизведение добавленной нами в приложение мелодии.

# Телефония и коммуникация

## Телефонные звонки

Большинство гаджетов с ОС Android так или иначе являются мобильными телефонами или поддерживают функцию голосовых звонков, как некоторые планшеты и фаблеты. Android предоставляет необходимое API, используя которое, позволит выполнить некоторые функции телефона.

Осуществлять звонки довольно легко. Пусть у нас в файле интерфейса объявлено текстовое поле для ввода номера и кнопка для непосредственного звонка:

<LinearLayout

    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"

    android:orientation="vertical">

    <EditText android:id="@+id/number"

        android:layout\_width="match\_parent"

        android:layout\_height="wrap\_content"

        android:cursorVisible="true"

        android:hint="Введите номер"

        android:editable="true"

        android:singleLine="true" />

    <Button android:id="@+id/dial"

        android:layout\_width="match\_parent"

        android:layout\_height="wrap\_content"

        android:text="Позвонить"

        android:onClick="dial" />

</LinearLayout>

В коде activity пропишем обработчик кнопки:

package com.example.eugene.telephoneapp;

import android.support.v7.app.ActionBarActivity;

import android.os.Bundle;

import android.view.Menu;

import android.view.MenuItem;

import android.content.Intent;

import android.view.View;

import android.widget.EditText;

import android.net.Uri;

public class MainActivity extends ActionBarActivity {

    @Override

    protected void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);

        setContentView(R.layout.activity\_main);

    }

    public void dial(View v) {

        EditText number=(EditText)findViewById(R.id.number);

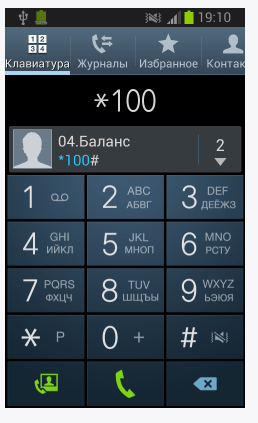
        String toDial="tel:"+number.getText().toString();

        startActivity(new Intent(Intent.ACTION\_DIAL, Uri.parse(toDial)));

    }

}

Для инициализации звонка надо использовать объект Intent с параметром **Intent.ACTION\_DIAL**, а также номером телефона. Однако в этом случае после ввоа номера и нажатии кнопки нас перекинет на стандартный экран ввода номера телефона, где уже будет введен номер и надо будет только нажать на кнопку звонка:



Но такое действие не всегда удобно, так как пользователю приходится еще на стандартном экране нажимать на кнопку звонка. Но мы можем испраивть данную ситуацию, используя вместо Intent.ACTION\_DIAL значение **Intent.ACTION\_CALL**:

|  |  |
| --- | --- |
|  | public void dial(View v) {        EditText number=(EditText)findViewById(R.id.number);      String toDial="tel:"+number.getText().toString();      startActivity(new Intent(Intent.ACTION\_CALL, Uri.parse(toDial)));  } |

Но при этом также надо добавить разрешение в файл манифеста:

|  |  |
| --- | --- |
|  | <?xml version="1.0" encoding="utf-8"?>  <manifest xmlns:android="<http://schemas.android.com/apk/res/android>"      package="com.example.eugene.telephoneapp" >      <uses-permission android:name="android.permission.CALL\_PHONE" /> |

Теперь звонки будут происходить сразу после нажатия кнопки.

## Отправка смс

Для отправки смс используется константа **Intent.ACTION\_SEND**. Создадим простейший интферфейс для отправки смс:

<LinearLayout

    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"

    android:orientation="vertical">

    <EditText android:id="@+id/number"

        android:layout\_width="match\_parent"

        android:layout\_height="wrap\_content"

        android:cursorVisible="true"

        android:hint="Введите номер"

        android:editable="true"

        android:singleLine="true" />

    <EditText android:id="@+id/message"

        android:layout\_width="match\_parent"

        android:layout\_height="wrap\_content"

        android:cursorVisible="true"

        android:hint="Введите сообщение"

        android:editable="true"

        android:singleLine="false" />

    <Button android:id="@+id/sms"

        android:layout\_width="match\_parent"

        android:layout\_height="wrap\_content"

        android:text="Отправить"

        android:onClick="smsSend" />

</LinearLayout>

И определим обработчик кнопки в коде activity:

package com.example.eugene.telephoneapp;

import android.support.v7.app.ActionBarActivity;

import android.os.Bundle;

import android.view.Menu;

import android.view.MenuItem;

import android.content.Intent;

import android.view.View;

import android.widget.EditText;

import android.net.Uri;

public class MainActivity extends ActionBarActivity {

    @Override

    protected void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);

        setContentView(R.layout.activity\_main);

    }

    public void smsSend(View v) {

        EditText number=(EditText)findViewById(R.id.number);

        EditText message=(EditText)findViewById(R.id.message);

        String toSms="smsto:"+number.getText().toString();

        String messageText= message.getText().toString();

        Intent sms=new Intent(Intent.ACTION\_SENDTO, Uri.parse(toSms));

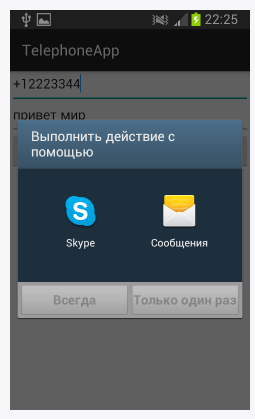
        sms.putExtra("sms\_body", messageText);

        startActivity(sms);

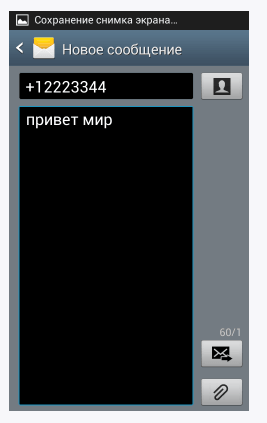
    }

}

После ввода номера телефона и сообщения нажмем на кнопку отправки, и система нам предоставит выбор между имеющимися программами для отправки сообщений. В моем случае это встроенная программа отправки смс и скайп.



После выбора программы можно будет подредактировать и уже потом окончательно отправить сообщение:



Но как и с телефоном мы также можем использовать прямую отправку смс без сторонних программ. Для этого, во-первых, добавим разрешение **SEND\_SMS** в файл манифеста:

|  |  |
| --- | --- |
|  | <?xml version="1.0" encoding="utf-8"?>  <manifest xmlns:android="<http://schemas.android.com/apk/res/android>"      package="com.example.eugene.telephoneapp" >      <uses-permission android:name="android.permission.SEND\_SMS" /> |

Исправим код обработчика кнопки:

|  |  |
| --- | --- |
|  | public void dial(View v) {        EditText number=(EditText)findViewById(R.id.number);      EditText message=(EditText)findViewById(R.id.message);        String numberText = number.getText().toString();      String messageText= message.getText().toString();        SmsManager.getDefault()             .sendTextMessage(numberText, null, messageText.toString(),  null, null);  } |

Для отправки используется класс **SmsManager** из пакета *android.telephony*. Его статический метод getDefault() возвращает объект данного класса. Для самой отправки применяется метод sendTextMessage(), который принимает пять параметров: номер телефона, адрес сервисного центра (в данном случае null), сообщение к отправке и два спецальных объкта PendingIntent, которые показывают статус отправки и доставки (в данном случае также null).

# Диалоговые окна

## DatePickerDialog и TimePickerDialog

По умолчанию в Android уже определены два диалоговых окна - **DatePickerDialog** и **TimePickerDialog**, которые позволяют выбрать дату и время.

Кроме установки даты DatePickerDialog позволяет обработать выбор даты с помощью слушателей **OnDateChangedListener** и **OnDateSetListener**. Что позволяет использовать выбранную дату далее в приложении.

Подобным образом TimePickerDialog позволяет обработать выбор времени с помощью слушателей **OnTimeChangedListener** и **OnTimeSetListener**

Пи работе с данными компонентами надо учитывать, что отсчет месяцев в DatePickerDialog начинается с нуля, то есть январь будет иметь номер 0, а декабрь - 11. И аналогично в TimePickerDialog отсчет секунд и минут будет идти с 0 до 59, а часов - с 0 до 23.

Используем DatePickerDialog и TimePickerDialog в приложении. Определим следующую разметку интерфейса в **activity\_main.xml**:

<LinearLayout xmlns:android="<http://schemas.android.com/apk/res/android>"

    android:layout\_width="match\_parent"

    android:layout\_height="match\_parent"

    android:orientation="vertical">

    <TextView android:id="@+id/currentDateTime"

        android:layout\_width="match\_parent"

        android:layout\_height="wrap\_content"

        android:textSize="18sp" />

    <Button android:id="@+id/timeButton"

        android:layout\_width="match\_parent"

        android:layout\_height="wrap\_content"

        android:text="Изменить время"

        android:onClick="setTime" />

    <Button android:id="@+id/dateButton"

        android:layout\_width="match\_parent"

        android:layout\_height="wrap\_content"

        android:text="Изменить дату"

        android:onClick="setDate"  />

</LinearLayout>

Здесь определены две кнопки для выбора даты и времени и текстовое поле, отображающее выбранные дату и время. И изменим код **MainActivity**:

package com.example.eugene.dialogsapp;

import android.app.DatePickerDialog;

import android.app.TimePickerDialog;

import android.support.v7.app.AppCompatActivity;

import android.os.Bundle;

import android.text.format.DateUtils;

import android.view.View;

import android.widget.DatePicker;

import android.widget.TextView;

import android.widget.TimePicker;

import java.util.Calendar;

public class MainActivity extends AppCompatActivity {

    TextView currentDateTime;

    Calendar dateAndTime=Calendar.getInstance();

    @Override

    public void onCreate(Bundle savedInstance) {

        super.onCreate(savedInstance);

        setContentView(R.layout.activity\_main);

        currentDateTime=(TextView)findViewById(R.id.currentDateTime);

        setInitialDateTime();

    }

    // отображаем диалоговое окно для выбора даты

    public void setDate(View v) {

        new DatePickerDialog(MainActivity.this, d,

                dateAndTime.get(Calendar.YEAR),

                dateAndTime.get(Calendar.MONTH),

                dateAndTime.get(Calendar.DAY\_OF\_MONTH))

                .show();

    }

    // отображаем диалоговое окно для выбора времени

    public void setTime(View v) {

        new TimePickerDialog(MainActivity.this, t,

                dateAndTime.get(Calendar.HOUR\_OF\_DAY),

                dateAndTime.get(Calendar.MINUTE), true)

                .show();

    }

    // установка начальных даты и времени

    private void setInitialDateTime() {

        currentDateTime.setText(DateUtils.formatDateTime(this,

                dateAndTime.getTimeInMillis(),

                DateUtils.FORMAT\_SHOW\_DATE | DateUtils.FORMAT\_SHOW\_YEAR

                        | DateUtils.FORMAT\_SHOW\_TIME));

    }

    // установка обработчика выбора времени

    TimePickerDialog.OnTimeSetListener t=new TimePickerDialog.OnTimeSetListener() {

        public void onTimeSet(TimePicker view, int hourOfDay, int minute) {

            dateAndTime.set(Calendar.HOUR\_OF\_DAY, hourOfDay);

            dateAndTime.set(Calendar.MINUTE, minute);

            setInitialDateTime();

        }

    };

    // установка обработчика выбора даты

    DatePickerDialog.OnDateSetListener d=new DatePickerDialog.OnDateSetListener() {

        public void onDateSet(DatePicker view, int year, int monthOfYear, int dayOfMonth) {

            dateAndTime.set(Calendar.YEAR, year);

            dateAndTime.set(Calendar.MONTH, monthOfYear);

            dateAndTime.set(Calendar.DAY\_OF\_MONTH, dayOfMonth);

            setInitialDateTime();

        }

    };

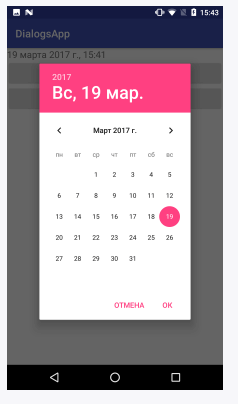
}

Ключевым классом здесь является **java.util.Calendar**, который хранится в стандартной библиоетке классов Java. В методе setInitialDateTime() мы получаем из экземпляра этого класса количество миллисекунд dateAndTime.getTimeInMillis() и с помощью форматирования выводим на текстовое поле.

Метод setDate(), вызываемый по нажатию на кнопку, отображает окно для выбора даты. При создании окна его объекту передается обработчик выбора даты DatePickerDialog.OnDateSetListener, который изменяет дату на текстовом поле.

Аналогично метод setTime() отображает окно для выбора времени. Объект окна использует обработчик выбора времени TimePickerDialog.OnTimeSetListener, который изменяет время на текстовом поле.

И поле запуска, нажав на кнопку изменения времени, мы сможем установить время:



Для установки времени на диалоговом окне определена кнопка "Установить". Аналогично работает окно установки даты:

