



**SoftUni
Foundation**

android development

for beginners

Join at [Slido.com](https://www.slido.com) with [#AndroidSoftUni](#)

Background Operations

- Threads
- AsyncTask
- Broadcasts
- IntentService
- Service
- + Retrofit

threads

Threads and Processes

- Нишките служат за изпълнение на код
- Ако имаме една нишка, целият ни код ще се изпълнява последователно
- При повече нишки е възможно различни парчета код да се изпълняват едновременно
- В един процес може да има много нишки
- Нишките в един процес споделят рам памет и адреси
- Два процеса не могат да споделят ресурси, те са независими



Threading in Android

- Във всяко Андроид приложение има една основна нишка
- Тя се казва UI / Main thread
- На нея се изпълняват процесите свързани с живота на апп-а
- На нея се извършва рисуването на екрана, засичането на кликане, изпълняването на анимации
- Тя не трябва да бъде натоварвана с тежки операции, защото блокира UI-а и апп-а изглежда зависнал

ANRDemo isn't responding.

Do you want to close it?

WAIT

OK

asyncTask

Защо съществува?

- Използва се за бекграунд операции, но и когато трябва те да влияят на потребителския интерфейс
- UI на апп-а се променя само от основната /наричана още UI/ нишка, за това са нужни специални механизми за да може той да се ъпдейтва от бекграунд процеси
- Да се използва основно за локални, кратки бекграунд процеси



А защо не просто Thread?

```
new Thread(new Runnable() {  
    public void run() {  
        Bitmap b = loadImageFromNetwork("http://example.com/image.png");  
        mImageView.setImageBitmap(b);  
    }  
}).start();
```

- Не може да рисува по екрана
- Няма връзка с живота на активитито

Пример

```
new DownloadImageTask().execute("http://example.com/image.png");
```

- Трябва да се наследява `AsyncTask<String, Void, Bitmap>`
 - Първия тип показва типа на входните данни
 - Втория тип - на прогреса
 - Третия на резултата
- `Bitmap doInBackground(String... urls)`
 - Единствения метод от класа, който е на отделна нишка
- `void onPostExecute(Bitmap result)`

broadcast



BroadcastReceiver

- Клас, който получава broadcasts
- Broadcasts са съобщения, които, когато са пратени, достигат до всички приложения
- Broadcasts се пращат когато
 - Се вкл/изкл wi-fi, bluetooth, друго
 - Когато батерията стане критична, когато се включи зарядно
 - Когато телефона се е рестартирал
 - При всяко по-важно събитие свързано с телефона
 - Когато вие самите изпратите ваш собствен broadcast
- Публичните Broadcasts могат да се прихванат от всички приложения
- Има и локални, които се изпращат само до елементите на вашето приложение

Broadcasts

- За да **получавате** Broadcasts, трябва да си регистрирате receiver
 - Чрез Java кода
 - `registerReceiver(BroadcastReceiver, IntentFilter)`
 - В `intentFilter` е дефиниран типа интененти за който слушате
 - Чрез Манифеста

```
<receiver android:name=".NetworkChangeReceiver" >  
    <intent-filter>  
        <action android:name="android.net.conn.CONNECTIVITY_CHANGE"/>  
    </intent-filter>  
</receiver>
```

- За да **изпратите** Broadcast трябва да използвате
 - ```
Intent intent = new Intent();
intent.setAction("com.tutorialspoint.CUSTOM_INTENT");
sendBroadcast(intent);
```

**intent service**

# Прилики и разлики

- Wrapper на Service
- Живота ѝ зависи от живота на приложението, за разлика от Service
- По-самостоятелна от AsyncTask
- Умира сама след изпълнение на задачите си

```
public class RSSPullService extends IntentService {
 @Override
 protected void onHandleIntent(Intent workIntent) {
 // Gets data from the incoming Intent
 String dataString = workIntent.getDataString();
 ...
 // Do work here, based on the contents of dataString
 ...
 }
}
```

# Извикване

- Извиква се чрез експлицитен интент и метода `startService()`
- Може да ѝ се подаде информация при стартирането, като тя се запише в екстра
- Връща информация на другите компоненти чрез Broadcast:

```
Intent localIntent =
 new Intent(Constants.BROADCAST_ACTION)
 // Puts the status into the Intent
 .putExtra(Constants.EXTENDED_DATA_STATUS, status);
 // Broadcasts the Intent to receivers in this app.
LocalBroadcastManager.getInstance(this).sendBroadcast(localIntent);
```



**service**

# Как работи

- Ако сервиз е стартиран чрез метода `startService()` от активити или фрагмент, но живота му не зависи от живота на компонента, който го е стартирал
- Стартиран по този начин сервиз обикновено има определена цел и щом я изпълни трябва сам да се спре
- Сервиз може да се стартира и чрез `bindService()`. Много компоненти могат да се закачат за един и същ сервиз, чак когато всички са се разкачили от сервиза, той умира
- `onStartCommand()` - извиква се когато сервиза е стартиран
- `onBind()` - вика се когато друг компонент се закачи за сервиза

**bonus: retrofit**

# The smart way to do Internets

- Библиотека, която се използва при връзка на апп-а с някакъв сървър / REST API
- Много проста и лесна за използване
- Няма модерен андроид апп, който да не я ползва
- Опростява следенето на статуса на връзката/коловете
- <http://square.github.io/retrofit/>

**домашно**

# Задача #1

Създайте приложение, което стартира сервиз, който през 10 секунди записва местоположението на потребителя в SharedPreferences

В SharedPreferences се пише с:

- `PreferenceManager.getDefaultSharedPreferences(context).edit().putString(KEY, locations).commit();`

Чете се с:

- `String locations = PreferenceManager.getDefaultSharedPreferences(context).getString(KEY);`

При отваряне на приложението, всичката записана информация да се показва на екрана.

# Задача #2

Направете приложение което взима текущата локация на телефона и показва температурата и валежите в момента и за утрешния ден.

Нека има е подходяща картинка при облачно/слънчево/дъждовно време.

Използвайте Retrofit & <https://openweathermap.org/api>

# Ресурси

- <http://developer.android.com/guide/components/processes-and-threads.html>
- <http://developer.android.com/training/run-background-service/create-service.html>
- Показване на картинки в лист
  - <http://developer.android.com/training/displaying-bitmaps/index.html>
- <https://developer.android.com/training/background.html>
- <http://square.github.io/retrofit/>