Aplikacje mobilne. Projektowanie aplikacji Android. Przetwarzanie asynchroniczne AsyncTask, Notification. BroadcastReceiver. Services

AsyncTask

Android AsyncTask to klasa, która służy do wykonywania długich operacji w tle i synchronizowania wątku podrzędnego z wątkiem głównym. AsyncTask używa trzech parametrów:

- Params - jest to typ parametru, który zostanie przesłany do naszego zadania do wykonania.
- Progress - drugi parametr służy do określania typu postępu publikowanego metodą doInBackground.
- Result, jest to zwracany typ parametru, który zostanie zwrócony do głównego wątku po zakończeniu przetwarzania w tle.

AsyncTask ma cztery metody, które są wyzwalane w różnych momentach cyklu życia wykonywania zadania asynchronicznego.

- PreExecute: Jest to wywoływane w wątku interfejsu użytkownika przed wykonaniem AsyncTask. W tej metodzie możemy wyświetlić ProgressBar lub wykonać dowolne zadania związane z interfejsem użytkownika.
- doInBackground: Kod wykonywania w tle jest umieszczany w tej metodzie. Nie możemy wywołać wystąpień interfejsu użytkownika działania w tej metodzie.
- onProgressUpdate: Ta metoda jest wyzwalana, gdy postęp publikowania jest wywoływany w metodzie doInBackground. Ta metoda jest przydatna, jeśli chcesz poinformować wątek interfejsu użytkownika o bieżącym postępie.
- onPostExecute: zostaje wywołany po zakończeniu doInBackground. Wartości zwrócone z doInBackground są odbierane tutaj. Możemy tutaj wprowadzić zmiany w interfejsie użytkownika, takie jak aktualizowanie widoków za pomocą zwracanych wartości.

AsyncTask tworzymy za pomocą poniższego kodu:

```
val task = MyAsyncTask(this)
task.execute(10)
```

Zadanie:

1. Utwórzmy aplikację AsyncTaskApp z pustą aktywnością, w której zdefiniujemy takie komponenty jak TextView, Button oraz ProgressBar w RelativeLayout.
2. Wzorcowy kod activity_main.xml poniżej:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">


    <TextView
        android:id="@+id/textView"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_above="@+id/btnDoAsync"
        android:gravity="center"
        android:text="Value if returned from the AsyncTask, will be displayed here" />

    <Button
        android:id="@+id/btnDoAsync"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerInParent="true"
        android:layout_margin="16dp"
        android:text="START ASYNC TASK" />


    <ProgressBar
        android:id="@+id/progressBar"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/btnDoAsync"
        android:visibility="gone"
        android:layout_centerHorizontal="true" />

</RelativeLayout>
```

3. Plik MainActivity.kt jest nie co bardziej skomplikowany.

```
import android.support.v7.app.AppCompatActivity
import android.os.Bundle
import kotlinx.android.synthetic.main.activity_main.*
import android.os.AsyncTask
import android.view.View
import android.widget.Toast
import java.lang.ref.WeakReference


class MainActivity : AppCompatActivity() {
```

```
  var myVariable = 10

  override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_main)

    btnDoAsync.setOnClickListener {
      val task = MyAsyncTask(this)
      task.execute(10)
    }
  }

  companion object {
    class MyAsyncTask internal constructor(context: MainActivity) : AsyncTask<Int, String,
String?>() {

      private var resp: String? = null
      private val activityReference: WeakReference<MainActivity> =
WeakReference(context)

      override fun onPreExecute() {
        val activity = activityReference.get()
        if (activity == null || activity.isFinishing) return
        activity.progressBar.visibility = View.VISIBLE
      }

      override fun doInBackground(vararg params: Int?): String? {
        publishProgress("Sleeping Started") // Calls onProgressUpdate()
        try {
          val time = params[0]?.times(1000)
          time?.toLong()?.let { Thread.sleep(it / 2) }
          publishProgress("Half Time") // Calls onProgressUpdate()
          time?.toLong()?.let { Thread.sleep(it / 2) }
          publishProgress("Sleeping Over") // Calls onProgressUpdate()
          resp = "Android was sleeping for " + params[0] + " seconds"
        } catch (e: InterruptedException) {
          e.printStackTrace()
          resp = e.message
        } catch (e: Exception) {
          e.printStackTrace()
          resp = e.message
        }

        return resp
      }


      override fun onPostExecute(result: String?) {

        val activity = activityReference.get()
```

```
            if (activity == null || activity.isFinishing) return
            activity.progressBar.visibility = View.GONE
            activity.textView.text = result.let { it }
            activity.myVariable = 100
        }

        override fun onProgressUpdate(vararg text: String?) {

            val activity = activityReference.get()
            if (activity == null || activity.isFinishing) return

            Toast.makeText(activity, text.firstOrNull(), Toast.LENGTH_SHORT).show()

        }
    }
  }
}
```

4. W skrócie
    - Po kliknięciu przycisku AsyncTask zostaje uruchomiony w tle.
    - W AsyncTask przekazujemy wartość Int, która reprezentuje liczbę sekund.
    - W doInBackgroundmetodzie usypiamy wątek na podaną liczbę sekund.
    - Uruchamiamy progressUpdateciąg, który jest wyświetlany w Toast .
    - W onPostExecutemetodzie ukrywamy ProgressBar i ustawiamy TextView na ciąg zwracany przez rozpakowanie typu Kotlin Nullable .
5. Zastanów się do czego można w praktyce wykorzystać AsyncTask. Zmodyfikuj kod by program wyświetlał coś innego.

Notification – Powiadomienia

Powiadomienia w systemie Android są jednym z mechanizmów komunikacji między aplikacjami zainstalowanymi na urządzeniu a użytkownikiem. Powiadomienia wykorzystywane są w przypadku wystąpienia szczególnych zdarzeń, które mogą być ważne i interesujące dla użytkownika. Projektując własny system powiadomień, należy zwrócić uwagę, aby nie były one uciążliwe dla użytkownika. Przykładowymi powiadomieniami stosowanymi w codziennej pracy są:
• aplikacja obsługująca pocztę powiadamia o przychodzącej nowej poczcie;
• system Android powiadamia o nowej wersji programu.

Powiadomienia są wyświetlane w górnej części ekranu, jeśli urządzenie jest odblokowane lub, w zależności od ustawień zabezpieczeń, na ekranie blokady, gdy urządzenie jest zablokowane.

Zadanie:
Stwórzmy aplikację, która pozwoli nam przesłać powiadomienie z aplikacji i wyświetlić te powiadomienie na ekranie startowym naszego smartfonu.

1. Stwórz aplikację o nazwie NotificationApp.
2. W pliku activity_main.xml dodaj RelativeLayout z komponentem button.

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout 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"
  tools:context=".MainActivity">
  <TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_centerHorizontal="true"
    android:layout_marginTop="50dp"
    android:text="Tutorials Point"
    android:textAlignment="center"
    android:textColor="@android:color/holo_green_dark"
    android:textSize="32sp"
    android:textStyle="bold" />
  <Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_centerInParent="true"
    android:onClick="createNotification"
    android:text="create notification" />
</RelativeLayout>
```

3. W pliku MainActivity.kt dodaj

```
import android.app.NotificationChannel
import android.app.NotificationManager
import android.app.PendingIntent
import android.content.Context
import android.content.Intent
import android.os.Build
import android.os.Bundle
import android.view.View
import androidx.appcompat.app.AppCompatActivity
import androidx.core.app.NotificationCompat
class MainActivity : AppCompatActivity() {
  var count = 0
  private val channelId = "10001"
  private val defaultChannelId = "default"
  override fun onResume() {
    super.onResume()
    count = 0
  }
  override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_main)
    title = "KotlinApp"
  }
  fun createNotification(view: View) {
    count++
    val notificationIntent = Intent(applicationContext, MainActivity::class.java)
    notificationIntent.putExtra("fromNotification", true)
    notificationIntent.flags = Intent.FLAG_ACTIVITY_CLEAR_TOP or
    Intent.FLAG_ACTIVITY_SINGLE_TOP
    val pendingIntent = PendingIntent.getActivity(this, 0, notificationIntent, 0)
```

```
        val notificationManager = getSystemService(Context.NOTIFICATION_SERVICE) as
        NotificationManager
        val builder = NotificationCompat.Builder(applicationContext, defaultChannelId)
        builder.setContentTitle("My Notification")
        builder.setContentIntent(pendingIntent)
        builder.setContentText("Hello World, to jest napis")
        builder.setSmallIcon(R.drawable.ic_launcher_foreground)
        builder.setAutoCancel(true)
        builder.setBadgeIconType(NotificationCompat.BADGE_ICON_SMALL)
        builder.setNumber(count)
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
          val importance = NotificationManager.IMPORTANCE_HIGH
          val notificationChannel = NotificationChannel(channelId,
          "NOTIFICATION_CHANNEL_NAME", importance)
          builder.setChannelId(channelId)
          notificationManager.createNotificationChannel(notificationChannel)
        }
        notificationManager.notify(System.currentTimeMillis().toInt(), builder.build())
      }
    }
```

4. W manifeście dodaj zezwolenia:
   `<uses-permission android:name="android.permission.VIBRATE" />`
   `<uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED" />`
5. **Zmodyfikuj program by notyfikacje były wyświetlane w innej kolorystyce. Zastanów się jak dodać ikonę/obrazek do powiadomienia oraz wywołać intencję po kliknięciu w powiadomienie.**

Więcej o notyfikacjach:
https://developer.android.com/training/notify-user/build-notification

BroadcastReceiver.

BroadcastReceiver to komponent który umożliwia rejestrację zdarzeń systemowych lub aplikacji. Wszyscy odbiorcy zdarzenia są powiadamiani przez środowisko wykonawcze systemu Android o wystąpieniu tego zdarzenia np. możemy otrzymać informację po włączeniu trybu samolotowego, że ten tryb jest uruchomiony.

1. Stwórzmy aplikację, która pozwoli nam zmienić status sieci WiFi czy też nie. Stwórz aplikację o nazwie BroadCastReceiver.
2. W pliku activity_main.xml dodaj RelativeLayout z komponentem Switch.

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout 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"
  tools:context=".MainActivity">
  <Switch
    android:id="@+id/wifiSwitch"
    android:layout_width="wrap_content"
```

```
    android:layout_height="wrap_content"
    android:layout_centerInParent="true" />
</RelativeLayout>
```

3. Stwórzmy teraz BroadcastReceiver w pliku MainActivity.kt

```
import android.content.BroadcastReceiver
import android.content.Context
import android.content.Intent
import android.content.IntentFilter
import android.net.wifi.WifiManager
import android.os.Bundle
import android.widget.Switch
import android.widget.Toast
import androidx.appcompat.app.AppCompatActivity
class MainActivity : AppCompatActivity() {
  lateinit var wifiSwitch: Switch
  lateinit var wifiManager: WifiManager
  override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_main)

    wifiSwitch = findViewById(R.id.wifiSwitch)
    wifiManager = applicationContext.getSystemService(Context.WIFI_SERVICE) as
WifiManager
    wifiSwitch.setOnCheckedChangeListener { _, isChecked ->
      if (isChecked) {
        wifiManager.isWifiEnabled = true
        wifiSwitch.text = "WiFi is ON"
      } else {
        wifiManager.isWifiEnabled = false
        wifiSwitch.text = "WiFi is OFF"
      }
    }
  }
  override fun onStart() {
    super.onStart()
    val intentFilter = IntentFilter(WifiManager.WIFI_STATE_CHANGED_ACTION)
    registerReceiver(wifiStateReceiver, intentFilter)
  }
  override fun onStop() {
    super.onStop()
    unregisterReceiver(wifiStateReceiver)
  }
  private val wifiStateReceiver: BroadcastReceiver = object : BroadcastReceiver() {
    override fun onReceive(context: Context, intent: Intent) {
      when (intent.getIntExtra(WifiManager.EXTRA_WIFI_STATE,
      WifiManager.WIFI_STATE_UNKNOWN)) {
        WifiManager.WIFI_STATE_ENABLED -> {
          wifiSwitch.isChecked = true
```

```
            wifiSwitch.text = "WiFi is ON"
            Toast.makeText(this@MainActivity, "Wifi is On", Toast.LENGTH_SHORT).show()
          }
          WifiManager.WIFI_STATE_DISABLED -> {
            wifiSwitch.isChecked = false
            wifiSwitch.text = "WiFi is OFF"
            Toast.makeText(this@MainActivity, "Wifi is Off", Toast.LENGTH_SHORT).show()
          }
        }
      }
    }
  }
```

4. W Manifeście dołącz zezwolenie na zmianę statusu WiFi.
   <uses-permission android:name="android.permission.CHANGE_WIFI_STATE" />
5. Dodaj do swojej aplikaci ImageView z logo WiFi. Dostosuj aplikację do pracy w trybie poziomym.
6. **Spróbuj zmodyfikować tą aplikację by sprawdzała czy uruchomiony jest tryb samolotowy czy nie. Spróbuj również wykonać aplikację uruchamiającą i wyłączającą latarkę. Czy jest to możliwe?**

Services

Services czyli usługi systemu Android to składniki aplikacji, które mogą wykonywać długotrwałe operacje w tle i nie zapewniają interfejsu użytkownika. Sterowanie usługami realizowane jest zazwyczaj z Aktywności i do niej przekazywane są wyniki działania usług. Dla systemu operacyjnego Android Usługi mają wyższy priorytet niż Aktywności, dlatego w przypadku braku zasobów ich działanie jest wstrzymywane na samym końcu, gdy brak jest możliwości uruchomienia nowej aplikacji.

1. Utwórz aplikację o nazwie AlarmManagerApp z dwoma przyciskami buton.
2. Plik activity_main.xml powinien wyglądać mniej więcej tak:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
  android:id="@+id/activity_main"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  android:gravity="center_horizontal"
  android:orientation="vertical"
  android:padding="16sp">

  <Button
    android:id="@+id/btnStartService"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="30dp"
    android:text="Start Service Alarm" />
  <Button
    android:id="@+id/btnStopService"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="10dp"
    android:text="Cancel Service" />
```

```
</LinearLayout>
```

3. Utwórz plik MainActivity.kt i wklej poniższe fragmenty kodu w odpowiednie miejsca.

```
import android.app.AlarmManager
import android.app.PendingIntent
import android.content.Context
import android.content.Intent
import android.os.Bundle
import android.widget.Button
import android.widget.Toast
import androidx.appcompat.app.AppCompatActivity
import java.util.*
class MainActivity : AppCompatActivity() {
  private lateinit var btnStart: Button
  private lateinit var btnStop: Button
  lateinit var pendingIntent: PendingIntent
  override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_main)


    btnStart = findViewById(R.id.btnStartService)
    btnStop = findViewById(R.id.btnStopService)

    btnStart.setOnClickListener {
      val myIntent = Intent(this@MainActivity, MyAlarmService::class.java)
      pendingIntent = PendingIntent.getService(this@MainActivity, 0, myIntent, 0)

      val alarmManager: AlarmManager = getSystemService(Context.ALARM_SERVICE) as
AlarmManager
      val calendar: Calendar = Calendar.getInstance()
      calendar.timeInMillis = System.currentTimeMillis()
      calendar.add(Calendar.SECOND, 3)
      alarmManager.set(AlarmManager.RTC_WAKEUP,                calendar.timeInMillis,
pendingIntent)
      Toast.makeText(baseContext, "Starting Service Alarm", Toast.LENGTH_LONG).show()
    }


btnStop.setOnClickListener {
      val alarmManager: AlarmManager = getSystemService(Context.ALARM_SERVICE) as
AlarmManager
      alarmManager.cancel(pendingIntent)
      Toast.makeText(baseContext, "Service Cancelled", Toast.LENGTH_LONG).show()
    }
  }
}
```

4. Dodajmy teraz plik MyAlarmService.kt a w nim:

```
import android.app.Service
import android.content.Intent
import android.os.IBinder
import android.widget.Toast
@Suppress("DEPRECATION")
class MyAlarmService : Service() {
  override fun onCreate() {
    Toast.makeText(this, "MyAlarmService.onCreate()", Toast.LENGTH_LONG).show();
    super.onCreate()
  }
  override fun onBind(intent: Intent?): IBinder? {
    Toast.makeText(this, "MyAlarmService.onBind()", Toast.LENGTH_LONG).show();
    return null
  }
  override fun onDestroy() {
    super.onDestroy()
    Toast.makeText(this, "MyAlarmService.onDestroy()", Toast.LENGTH_LONG).show()
  }
  override fun onStart(intent: Intent?, startId: Int) {
    super.onStart(intent, startId)
    Toast.makeText(this, "MyAlarmService.onStart()", Toast.LENGTH_LONG).show()
  }
  override fun onUnbind(intent: Intent?): Boolean {
    Toast.makeText(this, "MyAlarmService.onUnbind()", Toast.LENGTH_LONG).show()
    return super.onUnbind(intent)
  }
}
```

5. Przetestuj aplikację. Co się stanie? Zastanów się gdzie można wykorzystać usługi?

Dodatkowe zadania do wykonania:
Battery Monitor - https://www.youtube.com/watch?v=IekM_vjIiL0
Bluetooth Manager - https://www.youtube.com/watch?v=PtN6UTIu7yw
StepCounter - https://www.youtube.com/watch?v=WSx2a99kPY4