Android

WebView, HTTP kommunikáció

Dr. Ekler Péter
peter.ekler@aut.bme.hu

BME AUT
Department of
Automation and
Applied Informatics

WebView használata

WebView

- Web tartalom megjelenítése *Activity*-ből
- WebKit/Chromium alapú render motor
- Fő funkciók:
  - Előre/hátra navigáció
  - History
  - Zoom
  - Szöveges keresés a tartalomban
  - Stb.
- *JavaScript* integráció
- *android.permission.INTERNET* szükséges a használatához

WebView megjelenítése 1/3

- Engedély beállítása a Manifest-be:

```
> <uses-permission android:name=
    "android.permission.INTERNET"/>
```

- XML erőforrás definiálása:

```
<?xml version="1.0" encoding="utf-8"?>
<WebView xmlns:android=
    "http://schemas.android.com/apk/res/android"
    android:id="@+id/webview"
    android:layout_width="match_parent"
    android:layout_height="match_parent" />
```

WebView megjelenítése 2/3

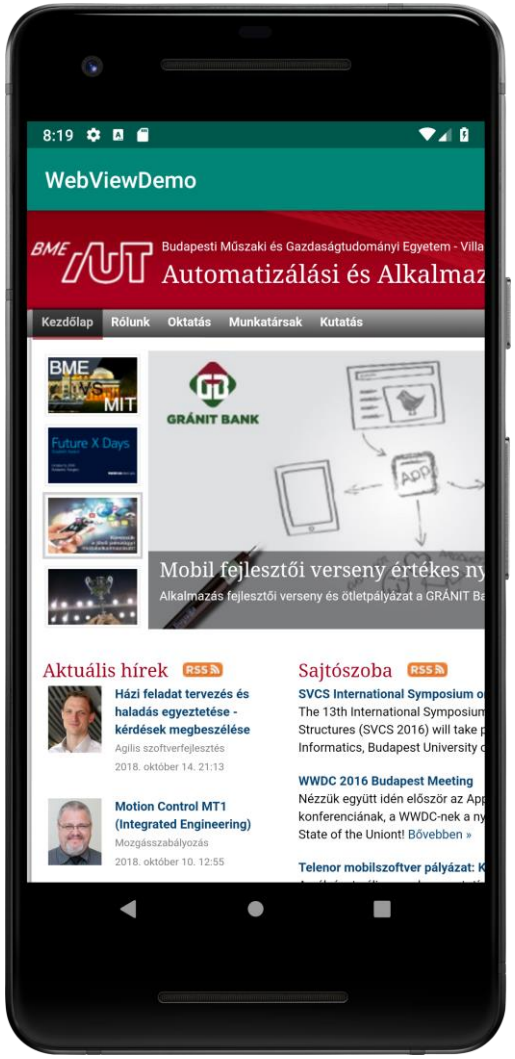
- Vezérlés Activity-ből:

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


    webView.settings.javaScriptEnabled = true
    webView.settings.builtInZoomControls = true

    webView.loadUrl("https://www.aut.bme.hu")
}
```

WebView megjelenítése 3/3



WebSettings

- *WebView* beállításainak kezelése
- Egy *WebView* példányosításakor az egyes tulajdonságok alapértelmezett értékeket vesznek fel
- Beállítások kezelése:
  - `mWebView.getSettings()`
  - getter és setter metódusok
- Példa beállítások:
  - *textSize*
  - *textZoom*
  - *zoomControl*
  - *userAgent*

Kitérő: billentyűzet kezelés

- Csak előtérben lévő Activity-n kezelhető a billentyűzet esemény
- Hangerő, vissza, egyéb gombok

```
override fun onKeyDown(keyCode: Int, event: KeyEvent): Boolean {
    if (keyCode == KeyEvent.KEYCODE_BACK) {
        // Billentyű esemény kezelése...
    }
    return super.onKeyDown(keyCode, event)
}
```

WebView navigáció

- A *WebView* komponensben előre/hátra navigálhatunk:

```
override fun onBackPressed() {
    if (webView.canGoBack()) {
        webView.goBack()
    } else {
        super.onBackPressed()
    }
}
```

WebViewClient

- Weboldalak megnyitásakor fontos események léphetnek fel, melyeket Android oldalról kezelhetünk
- Például ha a *WebView*-ban beállított weboldal átirányít valahova, akkor alapértelmezetten a beépített böngésző hívódik meg
- De definiálhatunk egy *WebViewClient*-et a különféle események kezelésére
  - `onFormResubmission(…)`
  - `onLoadResource(…)`
  - `onPageFinished(…)`
  - `onReceivedError(…)`
  - `onReceivedHttpAuthRequest (…)`
  - `onReceivedLoginRequest (…)`
  - `onReceivedSslError(…)`
  - `shouldOverrideKeyEvent(…)`
  - `shouldOverrideUrlLoading(…)`
  - Stb.

URL átirányítás felüldefiniálása

```
webView.webViewClient = object : WebViewClient() {
    override fun shouldOverrideUrlLoading(view: WebView,
      request: WebResourceRequest): Boolean {
        loadSite(request.url.toString())
        return true
    }

    override fun onPageFinished(view: WebView, url: String) {
        super.onPageFinished(view, url)
        progressBarWebLoad.progress = 100
        progressBarWebLoad.visibility = View.GONE
    }
}
```

WebChromeClient

- További böngésző események kezelése
- JavaScript események kezelése ☺
- Callback függvények felüldefiniálása
- Példa callback-ok:
  - `onJSTimeout()`
  - `onJSAlert()`
  - `onProgressChanged()`
  - `onJSConfirm()`
  - `onCreateWindow()`
  - Stb.

Weboldal letöltés állapota 1/2

```
webView.webChromeClient = object : WebChromeClient() {
    override fun onProgressChanged(view: WebView,
      newProgress: Int) {
        progressBarWebLoad.progress = newProgress
    }
}
```

Hálózati kapcsolatok

- Short-range
  - NFC
  - Bluetooth
  - Nearby API
  - Wifi-Direct
- Long-range/Internet
  - HTTP
  - TCP/IP-UDP

HTTP Kapcsolatok kezelése

HTTP kommunikáció Android platformon

- Egyik leggyakrabban használt kommunikációs technológia
- HTTP metódusok
  - GET, POST, PUT, DELETE
- Teljes körű HTTPS támogatás és certificate import lehetőség
- REST kommunikáció támogatása (Representational State Transfer)

HTTP kapcsolatok kezelése

- Szükséges engedély:
  <uses-permission android:name=
  "android.permission.INTERNET"/>
- Új szálban kell megvalósítani a hálózati kommunikáció hívást!
- Ellenőrizzük a HTTP válasz kódot:
  - http://www.restpatterns.org/HTTP_Status_Codes
  - http://www.w3.org/Protocols/HTTP/HTRESP.html
- HTTP REST
  - http://en.wikipedia.org/wiki/Representational_state_transfer
- Ügyeljünk az alapos hibakezelésre
- HTTP GET példa:
  - http://numbersapi.com/10/math

HTTP könyvtárak Android-on

- A rendszer két megvalósítás is tartalmaz:
  - Standard Java HTTP implementáció (***HttpURLConnection***)
  - **Apache** HTTP implementáció (*HttpClient*)
- Apache Depricated – Ne használjuk, ki is vették
- Igazán egyik sem jó
  - 3rd party megoldás – Square OkHttp
  - http://square.github.io/okhttp/

HTTP GET - HttpURLConnection

```
fun httpGet(urlAddr: String) {
    var reader: BufferedReader? = null
    try {
        val url = URL("http://mysrver.com/api/getitems")
        val conn = url.openConnection() as HttpURLConnection
        reader = BufferedReader(InputStreamReader(conn.inputStream))
        var line: String?
        do {
            line = reader.readLine()
            System.out.println(line)
        } while (line != null)
    } catch (e: IOException) {
        e.printStackTrace()
    } finally {
        if (reader != null) {
            try {
                reader.close()
            } catch (e: IOException) {
                e.printStackTrace()
            }
        }
    }
}
```

HTTP POST- HttpURLConnection

```
fun httpPost(urlAddr: String, content: ByteArray) {
    // ...
    var os: OutputStream? = null
    try {
        val url = URL("http://mysrver.com/api/refreshitems")
        val conn = url.openConnection() as HttpURLConnection
        conn.requestMethod = "POST"
        conn.doOutput = true
        conn.useCaches = false
        os = conn.outputStream
        os.write(content)
        os.flush()
        // …
    } catch (e: IOException) {
        e.printStackTrace()
    } finally {
        // …
        if (os != null) {
            try {
                os.close()
            } catch (e: IOException) {
                e.printStackTrace()
            }
        }
    }
}
```

Timeout értékek beállítása

- Fontos, hogy minden hálózati kommunikáció megfelelő módon kezelje a timeout-ot
- Timeout a kapcsolat megnyitásra
- Timeout az eredmény kiolvasására
- Példa:

```
…
val conn = url.openConnection() as HttpURLConnection
…
conn.setConnectTimeout(10000)
conn.setReadTimeout(10000)
…
```

Header paraméterek beállítása

- Egyszerű Header beállítása:

```
val conn = url.openConnection() as HttpURLConnection
conn.setRequestProperty(”[KEY]”,”[VALUE]”)
```

- Cookie beállítása:

```
val conn = url.openConnection() as HttpURLConnection
conn.setRequestProperty(”Cookie”,”sessionid=abc;age=15”)
```

- Összetett példa:

```
val conn = url.openConnection() as HttpURLConnection
conn.setRequestProperty(„Content-Type”,”application/json”)
conn.setRequestProperty(”Cookie”,”sessionid=abc;age=15”)
```

URL Encoding 1/2

- URL GET paraméterekben nem lehetnek „extra karakterek”
- Ilyen karakterek esetén URL encode-olásra van szükség
- Példa:
  - http://avalon.aut.bme.hu/~tyrael/phpget.php?name=John Doe
  - Vs.
  - http://avalon.aut.bme.hu/~tyrael/phpget.php?name=John%20Doe
- Szóköz esetén a kapott hiba:
  - 11-26 18:39:24.417: ERROR/AndroidRuntime(17232): java.lang.IllegalArgumentException: Illegal character in query at index 53: http://avalon.aut.bme.hu/~tyrael/phpget.php?name=John Doe
- Megoldás:
  - val name = URLEncoder.encode("John Doe")
  - connect("http://avalon.aut.bme.hu/~tyrael/phpget.php?name="+name)

URL Encoding 2/2

- *URLEncoder*:
  - Az ('a'..'z', 'A'..'Z') –n, számokon ('0'..'9') és '.', '-', '*', '_' karaktereken kívül minden karakter a hexadecimális megfelelőjévé kerül konvertálásra, például: '#' -> %23
  - `encode(String s)`
  - `encode(String s, String charsetName)`
- *URLDecoder*
  - *URLEncoder* fordítottja, az *application/x-www-form-urlencoded* MIME típusú szövegeket tudja visszalakítani
  - `decode(String s)`
  - `decode(String s, String encoding)`

Aszinkron kommunikáció

UI módosítása más szálból

- Az alkalmazás indításakor a rendszer létrehoz egy úgynevezett *main* szálat (UI szál)
- Sokáig tartó műveletek blokkolhatják a felhasználói felületet, ezért új szálba kell indítani őket
- Az ilyen műveletek a végén az eredményt a UI-on jelenítik meg, azonban az Android a UI-t csak a fő szálból engedi módosítani!
- Több megoldás is szóba jöhet:
  - *Activity.runOnUiThread(Runnable)*
  - *View.post(Runnable)*
  - *View.postDelayed(Runnable, long)*
  - *Handler*
  - *AsyncTask és LocalBroadcast (példa: laboron szerepelni fog)*
  - *Külső libek, pl. EventBus, Otto*
  - *REST külső lib: RetroFit*

Tipikus adatformátumok

Adatok küldése, válaszok feldolgozása

- Sokszor egy előre definiált formátumban/protokollon történik a kommunikáció kliens és szerver között
- Legtöbb esetben egy harmadik fél szerverétől kapott válasz is valamilyen jól strukturált formátumban érkezik
- Tipikus formátumok:
  - CSV (Comma Separated Value(s))
  - JSON (JavaScript Object Notation)
  - XML (Extensible Markup Language)
- Természetesen lehet saját protokoll is

JSON formátum

- Szintaktikai elemek: ’{’, ’]’ ,’[’, ’]’ ,’:’ ,’;’
- Példa:

```
{
   "keresztnev" : "Elek",
   "vezeteknev" : "Teszt",
   "kor" : 23,
   "cim" :
   {
      "utca" : "Baross tér",
      "varos" : "Budapest",
      "iranyitoszam" : "1087"
   },
   "telefon":
   [
      {
        "tipus"  : "otthoni",
        "szam": "123 322 1234"
      },
      {
        "tipus"  : "mobil",
        "szam": "626 515 1567"
      }
   ]
}
```

JSON feldolgozás

- *JSONObject*:
  - JSON objektumok parse-olása
  - Elemek elérhetősége a kulcs megadásával:
    - `getString(String name)`
    - `getJSONObject(String name)`
    - `getJSONArray(String name)`
  - JSON objektum létrehozása *String*-ből vagy *Map*-ból
- *JSONArray*:
  - *JSONObject*-hez hasonló működés JSON tömbökkel
  - Parse-olás, elemek lekérdezése index alapján, hossz
  - Létrehozás például *Collection*-ból

JSON API minták

- *Currency Exchange:*
  - *https://api.exchangeratesapi.io/latest?base=HUF*
- OpenWeather
  - http://api.openweathermap.org/data/2.5/weather?q=Budapest,hu&units=metric&appid=f3d694bc3e1d44c1ed5a97bd1120e8fe
- TV Show Data API:
  - http://api.tvmaze.com/search/shows?q=stargate

REST API gyűjtemények

- https://github.com/toddmotto/public-apis
- https://github.com/abhishekbanthia/Public-APIs
- https://github.com/Kikobeats/awesome-api

XML formátum

```
<?xml version="1.0"?>
<employees>
  <person>
    <name>Big Joe</name>
    <address>Beach Street 12.</address>
    <phone>111-222</phone>
  </person>
  <person>
    <name>Small Joe</name>
    <address>Hill Street 13.</address>
    <phone>222-333</phone>
  </person>
</employees>
```

XML feldolgozás

- Az Android gazdag eszközkészletet biztosít XML-ek feldolgozására
- SAX alapú feldolgozás
  - *javax.xml.parsers.SAXParser*
  - Különféle függvényekkel dolgozhatjuk fel az értelmező által generált eseményeket
  - Az eseményeket akkor generálja az értelmező, amikor a jelölő nyelv meghatározott részeihez ér
- DOM alapú feldolgozás
  - *javax.xml.parsers.DocumentBuilder*
  - *javax.xml.parsers.DocumentBuilderFactory*
  - Memóriába kerül beolvasásra az XML mint egy „fa”
  - Lekérdezhetők az elemek

Külső osztálykönyvtárak XML és JSON feldolgozásra

- XML:
  - SimpleXML
- JSON:
  - GSON
- REST API tesztelésére:
  - Postman Chrome Client

GSON POJO példa (Kotlin)

```
class PhoneInfo(
        @SerializedName("DeviceID")
        val deviceId: String,

        @SerializedName("OperatingSystem")
        val operatingSystem: String
)
```

GSON POJO példa (Java)

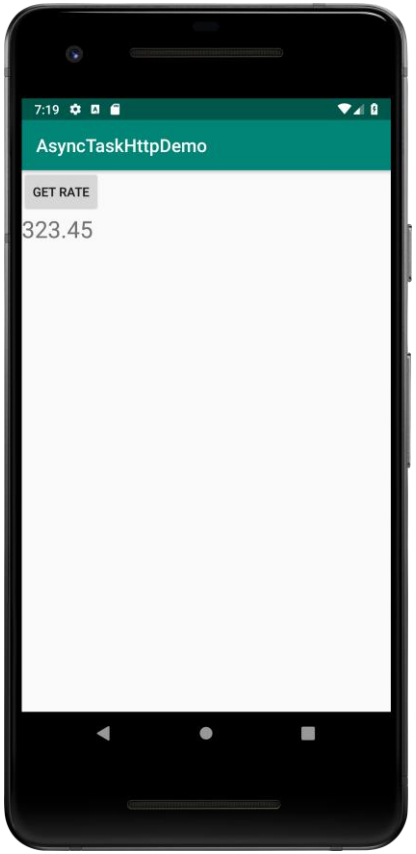
```
public class PhoneInfo {
  @SerializedName("DeviceID")
  private String deviceId;
  @SerializedName("OperatingSystem")
  private String operatingSystem;

  public PhoneInfo(String deviceId, String operatingSystem) {
    this.deviceId = deviceId;
    this.operatingSystem = operatingSystem;
  }
  public String getDeviceId() {
    return deviceId;
  }
  public String getOperatingSystem() {
    return operatingSystem;
  }
}
```

REST API-k kezelése

Példa

- https://api.exchangeratesapi.io/latest?base=EUR
- HttpURLConnection + AsyncTask



AsyncTask saját Callback-el

AsyncTask konstruktor paraméterében vár *Callback*-et:

```
class HttpGetTaskWithCalback(val callback: (String) -> Unit) :
  AsyncTask<String, Void, String>() {

    override fun doInBackground(vararg params: String): String {

        // … http lekérdezés és eredmény betöltése result String-be
        return result
    }

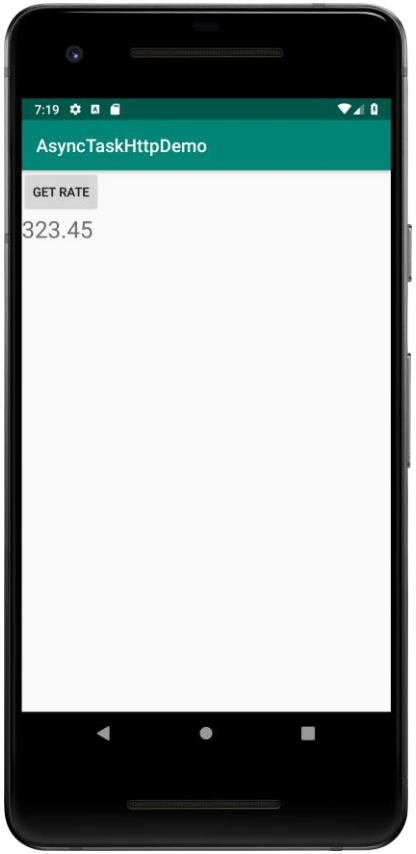
    override fun onPostExecute(result: String) {
        callback.invoke(result)
    }
}
```

Használat például Activity-ben:

```
btnGetRate.setOnClickListener{
    HttpGetTaskWithCalback { result ->
        Toast.makeText(this@ManActivity, result, Toast.LENGTH_LONG).show()
    }
}
```

Példa

- https://api.exchangeratesapi.io/latest?base=EUR
- OkHttp – Get példa



- https://api.exchangeratesapi.io/latest?base=EUR

- HOST URL:
  - https://api.exchangeratesapi.io/
- Path:
  - latest
- URL GET params
  - base = EUR

HTTP GET - OkHttp

```
object OkHttpHelper{
    fun getRates():String{
        val client=OkHttpClient.Builder()
            .connectTimeout(5000,TimeUnit.MILLISECONDS)
            .build()

        val request=Request.Builder()
            .url(
"https://api.exchangeratesapi.io/latest?base=EUR")
            .get()
            .build()

        val call=client.newCall(request)
        val response=call.execute()

        val responseStr=response.body()!!.string()
        return  responseStr
    }
}
```

Használat például Activity-ben:

```
Thread {
    var data = OkHttpHelper.getRates()
    runOnUiThread{
        Toast.makeText(this@MainActivity, data, Toast.LENGTH_LONG).show()
    }
}.start()
```

Retrofit

- HTTP API megjelenítése Java interface formában

```
interface ItemsService {
    @GET("/items/{item}/details")
    fun listItems(@Path("item") item: String): Call<List<Item>>
}
```

- Retrofit osztály a konkrét implementáció generálására

```
val retrofit = Retrofit.Builder()
        .baseUrl("https://api.myshop.com")
        .build()
val service = retrofit.create(ItemsService::class.java)
```

- Mindenhívás az *ItemsService* mehet szinkron és aszinkron módon:

```
val items: Call<List<Item>> = service.listItems("myItem")
```

Retrofit

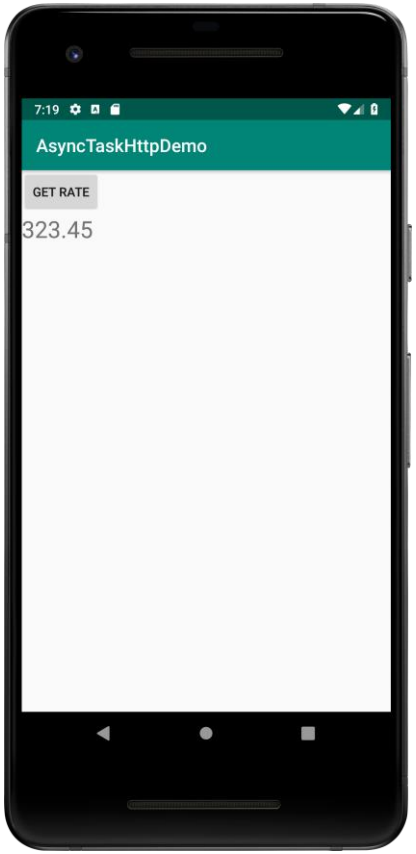
- HTTP kérések leírása annotációkkal:
  - URL és query paraméterek
  - Body – objektum konverzió (JSON, protocol buffers)
  - Multipart request és file feltöltés
- Gradle:
  - implementation 'com.squareup.retrofit2:retrofit:2.4.0'
- További információk:
  - http://square.github.io/retrofit/

Retrofit – GSON támogatás

- Automatikus konverzió a háttérben
  - Be kell állítani a Retrofitnak hogy mit használjon a konverzióhoz.
  - ```
    val retrofit=Retrofit.Builder()
            .baseUrl("http://api.myserver.com/")
            .addConverterFactory(GsonConverterFactory.create())
            .build()
    ```
- Gradle:
  - implementation 'com.google.code.gson:gson:2.8.5'
    implementation 'com.squareup.retrofit2:converter-gson:2.4.0'
- További információk:
  - http://square.github.io/retrofit/
    ```

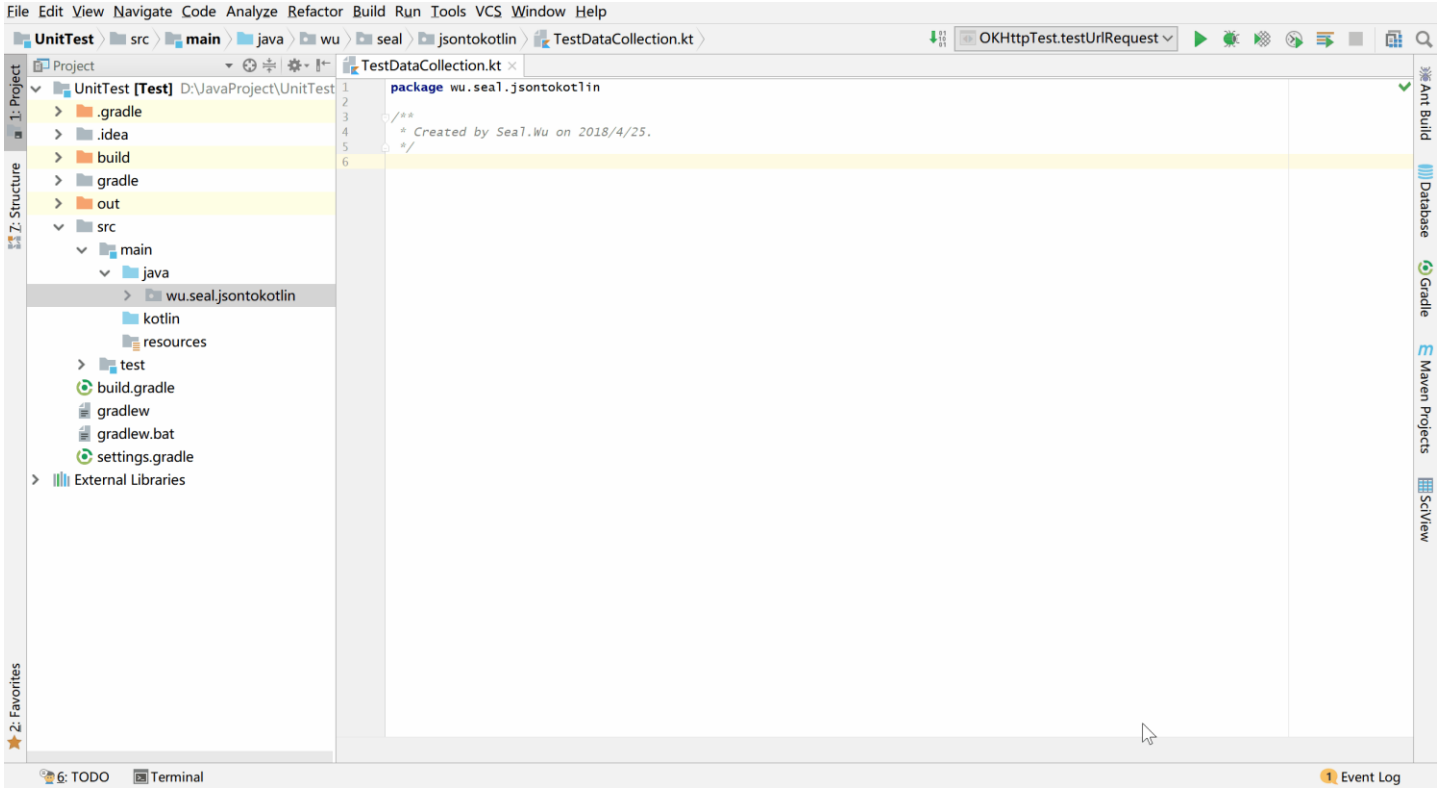
Példa - Retrofit

- https://api.exchangeratesapi.io/latest?base=EUR
- Retrofit 2 + GSON



Entitás vagy *data* class generálás JSON-ból

- *data* class, csak ha kellenek a *componentN* és egyéb függvények
- https://http4k-data-class-gen.herokuapp.com/
- https://github.com/wuseal/JsonToKotlinClass



Retrofit - Entitások / data class

```
data class MoneyResult(
    var date: String,
    var rates: Rates,
    var base: String
)

data class Rates(
    var BGN: Double,
    var CAD: Double,
…
}
```

Retrofit – API interface

```
import retrofit2.Call
import retrofit2.Callback
import retrofit2.http.GET
import retrofit2.http.Query


interface CurrencyExchangeAPI {
    @GET("/latest")
    fun getRates(@Query("base") base: String): Call<MoneyResult>
}
```

Retrofit - használat

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

    val retrofit = Retrofit.Builder()
        .baseUrl("https://api.exchangeratesapi.io/")
        .addConverterFactory(GsonConverterFactory.create())
        .build()

    val currencyAPI = retrofit.create(CurrencyExchangeAPI::class.java)

    btnGetRate.setOnClickListener {
        val ratesCall = currencyAPI.getRates("EUR")
        ratesCall.enqueue(object: Callback<MoneyResult> {
            override fun onFailure(call: Call<MoneyResult>, t: Throwable) {
                tvResult.text = t.message
            }

            override fun onResponse(call: Call<MoneyResult>,
              response: Response<MoneyResult>) {
                tvResult.text = response.body()?.rates?.HUF.toString()
            }
        })
    }
}
```

- http://api.openweathermap.org/data/2.5/weather?q=London&units=metric&appid=f3d694bc3e1d44c1ed5a97bd1120e8fe
- HOST URL:
  - https://api.openweathermap.org/
- Path:
  - data/2.5/weather
- URL params
  - q = London
  - units = metric
  - appid = f3d694bc3e1d44c1ed5a97bd1120e8fe

Összefoglalás

- Helyfüggő szolgáltatások
  - Geocoding/reverse geocoding
  - ProximityAlert
  - Geofences
  - Activity recoginition
- Google Maps képességek
  - Térkép beállítások
  - Markerek kezelése
  - Maps Utility Library
- Hálózati kommunikáció alapok
  - HTTP kapcsolatok kezelése

Kérdések