Android

Hálózati kommunikáció

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

BME AUT
Department of
Automation and
Applied Informatics

ZH

- Április 20. 12:15
- Moodle teszt

Adjon meg f-hez primitív függvényt, amelyre F(0)=0 $f(x)=3\sqrt[3]{x}\cdot x^2$, $f(x)=3e^{2x+2}$

$f(x)=\frac{5}{(2x+1)^6}$, $f(x)=3(x-1)^4$, $f(x)=2x^5+3x^4+2x^2-x+2$

Számítsa ki az alábbi határozatlan integrálokat!

$\int(1+x)^{10}$, $\int\frac{(x+2)^3}{x}$, $\int\frac{x^3-1}{x-1}$, $\int\frac{x^4-1}{x+1}$, $\int\sqrt{x^2+2x+1}$, $\int\left(\frac{x^3}{\sqrt{x}}+\frac{\sqrt{x}}{x^3}\right)$, $\int\frac{(2x+1)^2}{2x^2}$

$\int 2\sin 3x$, $\int 3\cos 2x$, $\int \sin 2x\cdot\cos 2x$, $\int(\cos^2 2x-\sin^2 2x)$, $\int 2\sin^2 x$, $\int\cos^2 2x$,

$\int\sin 3x\cdot\sin 2x$, $\int\cos 2x\cdot\cos 3x$, $\int\sin 2x\cdot\cos 3x$, $\int e^{2x-1}$, $\int e^{2-3x}$, $\int 2^{2x-1}$, $\int 3^{2-2x}$,

$\int\frac{1}{2x+1}$, $\int\sqrt{x\sqrt[3]{x}}$ $\int\frac{1}{x\ln x}$, $\int\frac{x-3}{x^2-6x+27}$, $\int\frac{e^{3x}}{e^{3x}+5}$, $\int ctg2x$

$\int x\sin x^2$, $\int\frac{e^x}{\sqrt{1-e^{2x}}}$, $\int\frac{e^{tgx}}{\cos^2 x}$, $\int\frac{\ln x}{\sqrt{x}}$, $\int x^4\cdot\ln x$, $\int(2x-1)\sin 2x$,

$\int(3x+2)\cos 3x$, $\int(x^2+2x)e^{-x}$, $\int(2x^2+x+1)\sin 4x$, $\int x^5e^{x^2}$, $\int(2-x^2)\cdot\cos 2x$

$\int\frac{2}{(x-1)(x+2)}$, $\int\frac{3x}{(x+1)(x-3)}$, $\int\frac{2x+1}{(x-1)(x-3)}$, $\int\frac{2}{x(x+2)}$, $\int\frac{3x-2}{x^2+x-6}$,

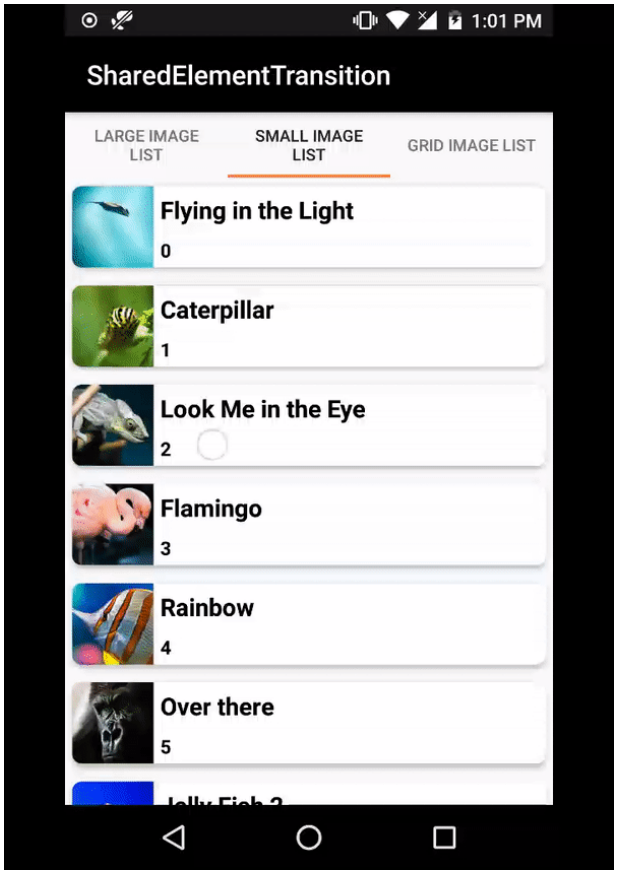
$\int\frac{x(2x+2)}{(x^2+x)(x^2-9)}$, $\int\frac{x^3-4x}{(x-2)(x^2+6x+8)}$, $\int\frac{3x+4}{(2x+3)(2-x)}$

Tartalom

- Hálózati kommunikáció – HTTP
  - Aszinkron kommunikáció
  - Retrofit
  - JSON parseolás
- WebView
- TCP/IP, UDP

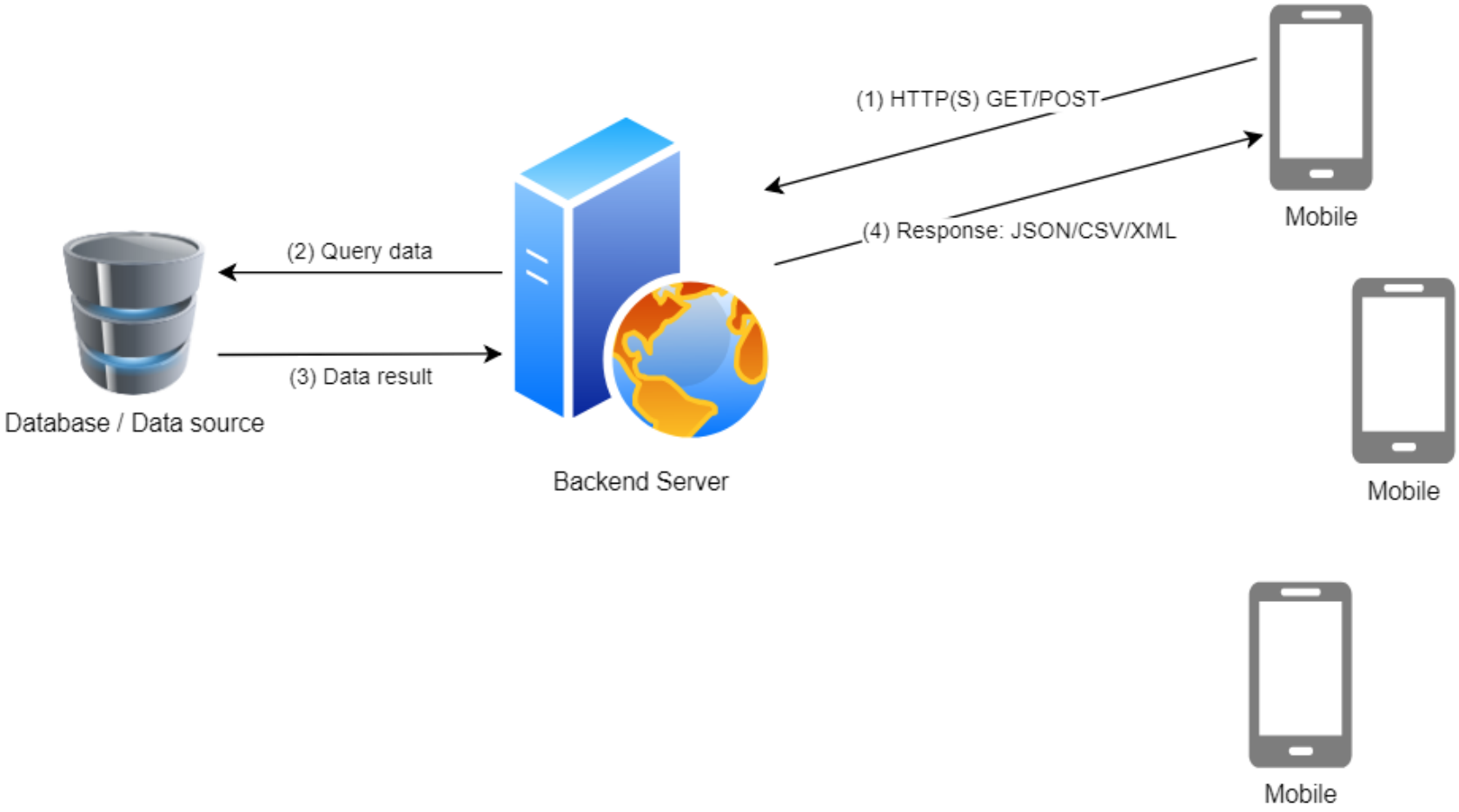
SharedElement (extra ☺)

- Transition animáció például Activiy váltáskor
  - https://developer.android.com/training/transitions/start-activity
  - https://www.uplabs.com/posts/shared-element-transition-kotlin



HTTP(s) Kapcsolatok kezelése

Tipikus architektúra



HTTP kommunikáció Android platformon

- Egyik leggyakrabban használt kommunikációs technológia
- HTTP metódusok
  - GET, POST, PUT, DELETE, stb.
- 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:
  - https://restfulapi.net/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*)
    - Depricated – Ne használjuk, ki is vették már
- Külső libraryk:
  - 3rd party megoldás – Square OkHttp
  - http://square.github.io/okhttp/
  - Retrofit: OkHttp alapú

Példa API

- http://api.openweathermap.org/
- data/2.5/weather
- ?q=Budapest&units=metric&appid=f3d694bc3e1d44c1ed5a97bd1120e8fe

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 (Deprecated)*
  - *Külső libek, pl. EventBus, Otto*
  - *REST külső lib: Retrofit (**preferált**)*

AsyncTask példa

```
class AsyncTaskUploadVote : AsyncTask<String, Void, String>() {

    override fun onPreExecute() {
        // ...
    }

    override fun doInBackground(vararg params: String): String? {
        val result = null
        // Hálózati kommunikáció, válasz mentése result-ba
        return result
    }

    override fun onPostExecute(result: String?) {
        // Eredmény használata UI szálon
        Log.d("RESULT", result)
    }

}

AsyncTaskUploadVote().execute("Yes")
```

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"
      }
   ]
}
```

```
class Person {

  val firstName: String
  val surName: String

  val address: Address
  val phone: List<PhoneNum>
}

class Address {
  val street: String
...
}

...
```

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:
  - Moshi vagy GSON
  - http://www.jsonschema2pojo.org/
- REST API tesztelésére:
  - PostMan (Chrome plugin)

GSON POJO példa (Kotlin)

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

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

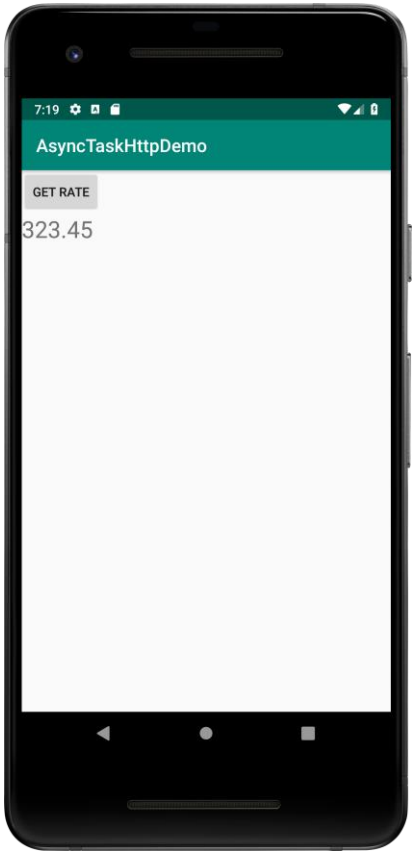
Moshi POJO példa

```
@JsonClass(generateAdapter = true)
class MoneyResult(val rates: Rates?, val base: String?, val date: String?)

@JsonClass(generateAdapter = true)
class Rates(val CAD: Double?, val HKD: Double?, val ISK: Double?, …
```

REST API-k kezelése

Példa

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



AsyncTask saját Callback-el

AsyncTask konstruktor paraméterében vár *Callback*-et:

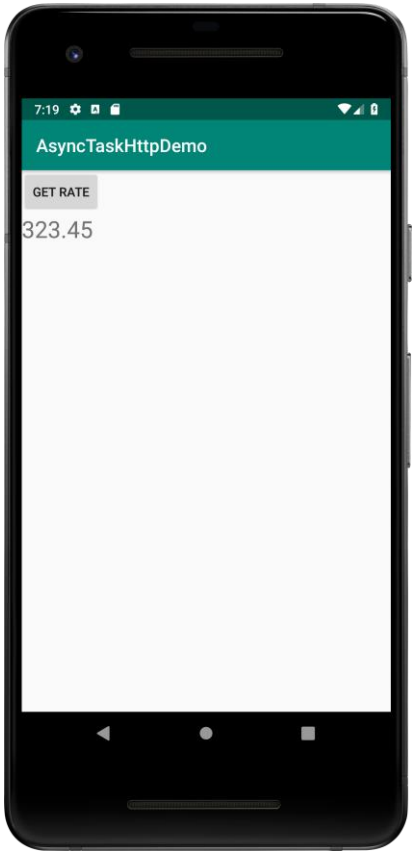
```
class HttpGetTaskWithCalback(val callback: (String) -> Unit) :
  AsyncTask<String, Void, String>() {

    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



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

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

- 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:2.9.0'
- További információk:
  - http://square.github.io/retrofit/

Retrofit – Moshi támogatás

- Automatikus konverzió a háttérben
  - Be kell állítani a Retrofitnak hogy mit használjon a konverzióhoz.

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

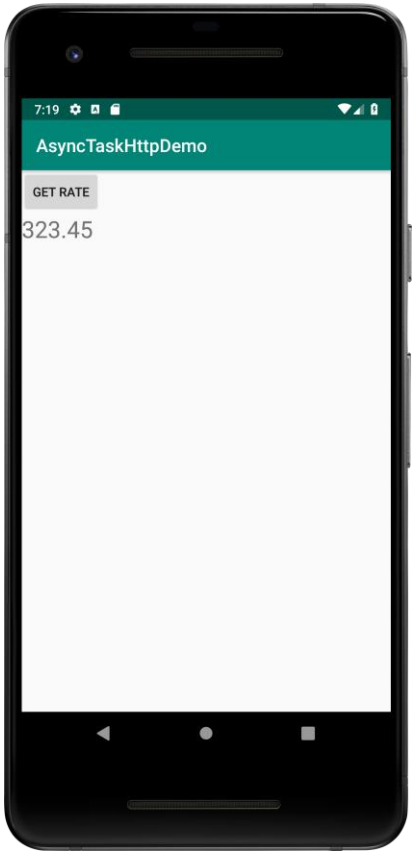
- Gradle:

```
def retrofit_version = '2.8.1'
implementation "com.squareup.retrofit2:retrofit:$retrofit_version"
implementation "com.squareup.retrofit2:converter-moshi:$retrofit_version"
def moshi_version = '1.9.2'
implementation "com.squareup.moshi:moshi:$moshi_version"
kapt "com.squareup.moshi:moshi-kotlin-codegen:$moshi_version"
```

- További információk:
  - http://square.github.io/retrofit/

Példa - Retrofit

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

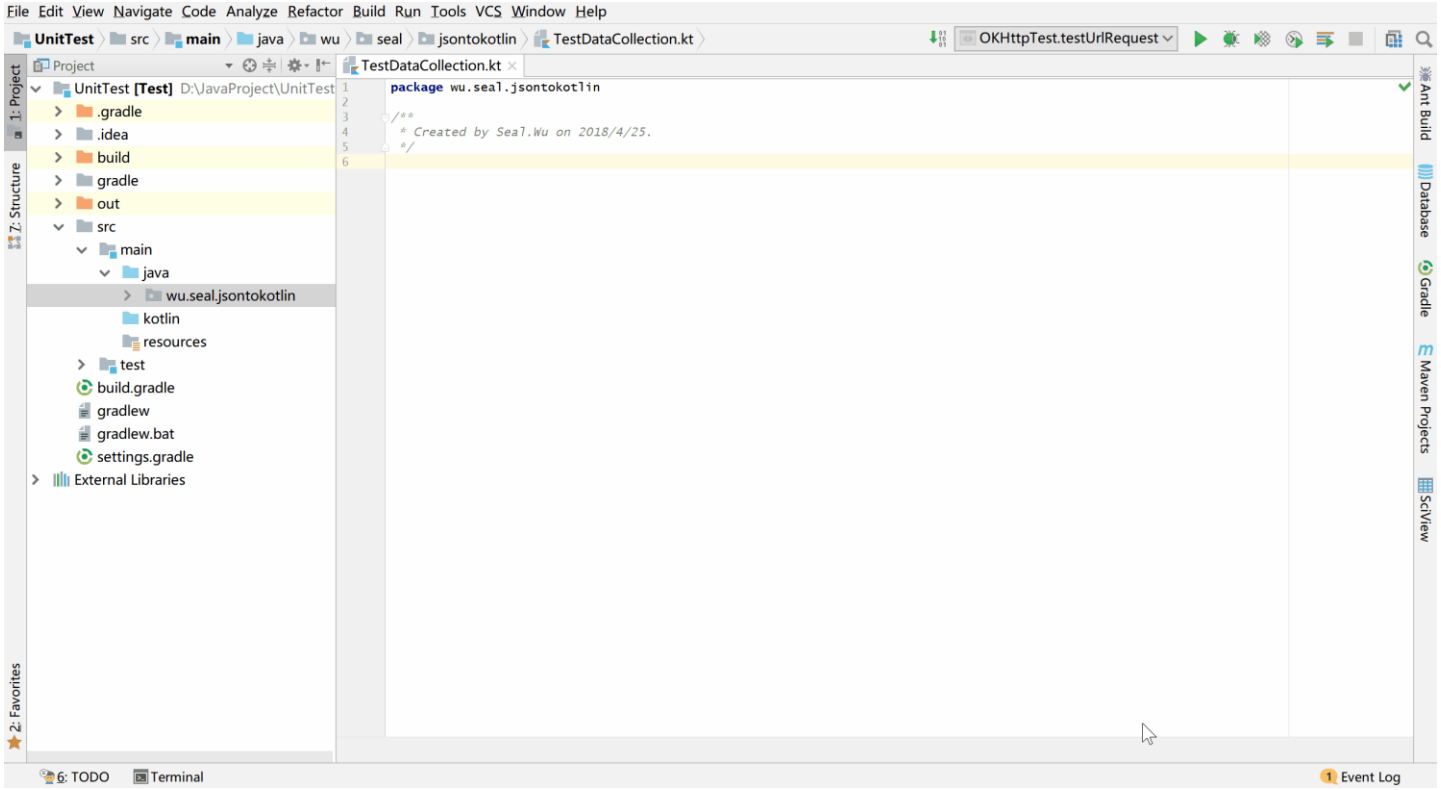


Gradle

```
apply plugin: 'com.android.application'
apply plugin: 'kotlin-android'
apply plugin: 'kotlin-android-extensions'
apply plugin: 'kotlin-kapt'

android {
    compileSdkVersion 29
    buildToolsVersion "29.0.3"

    defaultConfig {
        applicationId "hu.ait.httpmoneydemo"
        minSdkVersion 26
        targetSdkVersion 29
        versionCode 1
        versionName "1.0"

        testInstrumentationRunner "androidx.test.runner.AndroidJUnitRunner"
    }

    ...

    compileOptions {
        sourceCompatibility 1.8
        targetCompatibility 1.8
    }
    kotlinOptions {
        jvmTarget = "1.8"
    }
}
dependencies {
    …
    def retrofit_version = '2.9.0'
    implementation "com.squareup.retrofit2:retrofit:$retrofit_version"
    implementation "com.squareup.retrofit2:converter-moshi:$retrofit_version"
    def moshi_version = '1.9.2'
    implementation "com.squareup.moshi:moshi:$moshi_version"
    kapt "com.squareup.moshi:moshi-kotlin-codegen:$moshi_version"
}
```

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

```
@JsonClass(generateAdapter = true)
class MoneyResult(val rates: Rates?, val base: String?, val date: String?)

@JsonClass(generateAdapter = true)
class Rates(val CAD: Double?, val HKD: Double?, val ISK: Double?, val PHP: Double?,
            val DKK: Double?, val HUF: Double?, val CZK: Double?, val AUD: Double?, val RON: Double?,
            val SEK: Double?, val IDR: Double?, val INR: Double?, val BRL: Double?, val RUB: Double?,
            val HRK: Double?, val JPY: Double?, val THB: Double?, val CHF: Double?, val SGD: Double?,
            val PLN: Double?, val BGN: Double?, val TRY: Double?, val CNY: Double?, val NOK: Double?,
            val NZD: Double?, val ZAR: Double?, val USD: Double?, val MXN: Double?, val ILS: Double?,
            val GBP: Double?, val KRW: Double?, val MYR: 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(MoshiConverterFactory.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()
            }
        })
    }
}
```

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:

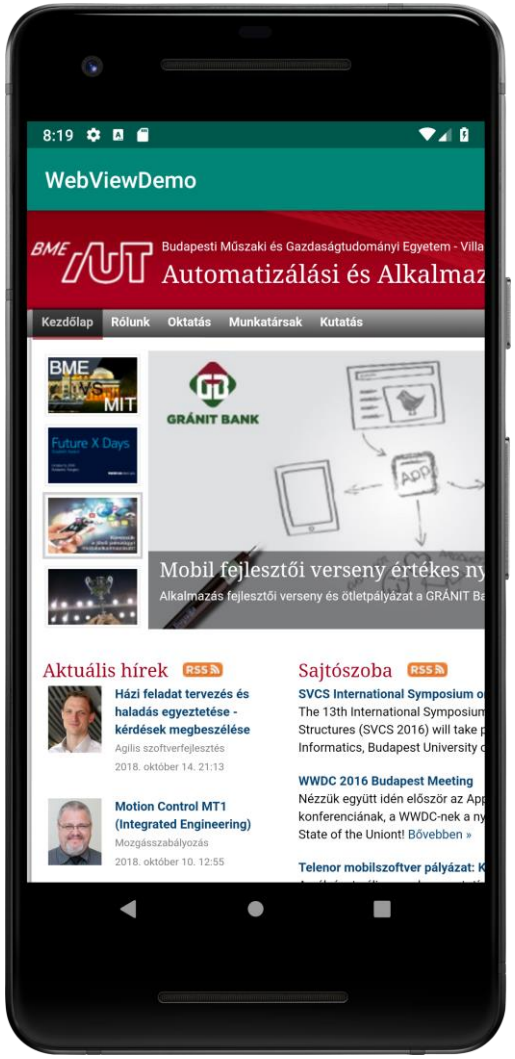
```
<?xml version="1.0" encoding="utf-8"?>
<WebView xmlns:android=
    "http://schemas.android.com/apk/res/android"
    android:id="@+id/webview"
    android:layout_width="match_parent"
    android:layout_height="match_parent" />
```

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

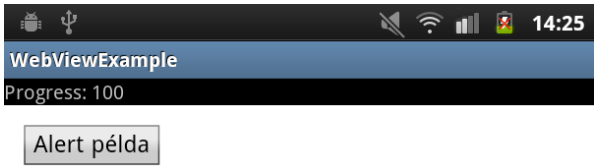
```
webView.webChromeClient = object : WebChromeClient() {
    override fun onProgressChanged(view: WebView,
      newProgress: Int) {
        progressBarWebLoad.progress = newProgress
    }
}
```

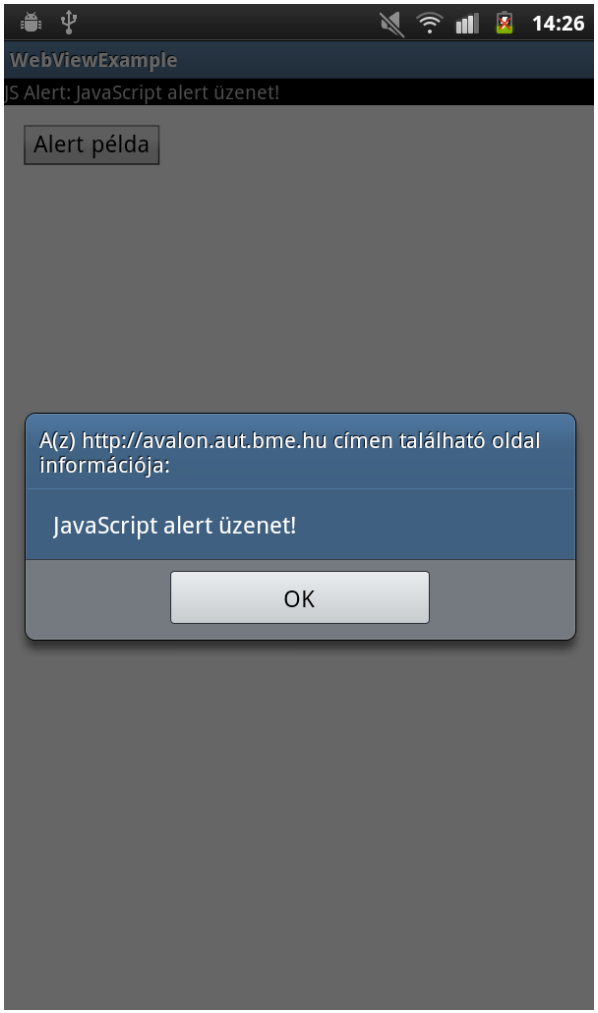
Weboldal letöltés állapota 2/2

JavaScript Alert példa 1/2

```
webView.getSettings().setJavaScriptEnabled(true)
webView.webChromeClient = object : WebChromeClient() {
    override fun onJsAlert(
        view: WebView, url: String,
        message: String, result: JsResult
    ): Boolean {
        tvStatus.setText("JS Alert: $message")
        // return true; esetén nem fut le a default
        // implementáció (pop-up ablak)
        return super.onJsAlert(view, url, message, result)
    }
}
```

JavaScript Alert példa 2/2





JavaScript kommunikáció WebViewn keresztül

- Fejlett JavaScript kezelő lib:
  - https://github.com/tcoulter/jockeyjs
- Szenzor értékek böngészőn keresztül
  - http://atleast.aut.bme.hu/mobilesensor/
  - Nyissuk meg egy mobil böngészőben és tekintsük át mely szenzor adatok érhetők el

Hálózati adatforgalom felügyelete - TrafficStats

- Gyakran szükség lehet az adatforgalom felügyeletére, fel/letöltés kijelzésére
- Megoldás: *android.net.TrafficStats* osztály
- Statikus függvények az adatforgalom lekérdezésére, például:
  - *static long getMobileRxBytes()*
- Adatforgalom tiltásakor törlődik az eddigi statisztika
- UDP és TCP forgalom külön lekérdezhető
- Adatforgalom lekérdezése az egyes processzekhez (UID megadásával):
  - *static long getUidRxBytes(int uid)*

TCP/IP és UDP kommunikáció

TCP/IP Socket

- Szabványos *Socket* implementáció
- Jól ismert *java.net.Socke*t osztály a kapcsolatok megnyitására
- *java.net.ServerSocket* osztály a bejövő kapcsolatok fogadására
  - Localhoston az alkalmazások egymás közti kommunikációja is megoldható
- InputStream és OutputStream támogatás az adatok olvasására és írására

Socket példa

```
val socket = Socket("192.168.2.112", 8787)
val inputStream = socket.getInputStream()
val isr = InputStreamReader(
    inputStream,
    "UTF-8"
)
val resultBuffer = StringBuilder()
var inChar: Int
while ((inChar = isr.read()) != -1) {
    resultBuffer.append(inChar.toChar())
}
val result = resultBuffer.toString()
// result kezelése
// ...
inputStream.close()
socket.close()
```

UDP kommunikáció

- User Datagram Protocol (UDP)
- Gyors kommunikáció
- Az UDP nem biztosítja a csomag megérkezését
- Példa használat:
  - Valós idejű multimédia átvitel
  - Játékok
  - Stb.

UDP Androidon

- Standard Java osztályok
- *java.net.DatagramSocket*: socket
- Broadcast támogatás:

```
val s: DatagramSocket = DatagramSocket()
s.setBroadcast(true)
```

- *java.net.DatagramPacket*: UDP csomag

UDP Androidon

- Standard Java osztályok
- *java.net.DatagramSocket*: socket
- Broadcast támogatás:

```
val s: DatagramSocket = DatagramSocket()
s.setBroadcast(true)
```

- *java.net.DatagramPacket*: UDP csomag

UDP üzenetek küldése

```
val msg = "UDP Test"
val socket = DatagramSocket()
// In case of broadcast
socket.setBroadcast(true)
//InetAddress localAddr =
//  InetAddress.getByName("192.168.0.110");
val message = msg.toByteArray()
val p = DatagramPacket(
    message,
    msg.length, localAddr, 10100
)
socket.send(p)
```

UDP üzenet fogadása

```
val message = ByteArray(1500)
val packet = DatagramPacket(message, message.size)
val socket = DatagramSocket(10100)
socket.receive(packet)
val msg = String(
    message, 0, packet.getLength()
)
Log.d("MYTAG", "received: $msg")
socket.close()
```

Népszerű osztálykönyvtárak

- KryoNet
  - https://github.com/EsotericSoftware/kryonet
- Near
  - https://github.com/adroitandroid/Near

KryoNet példa

- Adat osztály regisztráció (szerializációhoz)

```
class TextMessage {
    var text: String? = null
}

class ImageMessage {
    var image: ByteArray? = null
}


Log.set(Log.LEVEL_DEBUG) // sokat segít ☺

server = Server(131072, 131072) // write és object buffer méretek
val kryo = server.kryo
kryo.register(TextMessage::class.java)
kryo.register(ImageMessage::class.java)
kryo.register(ByteArray::class.java)
```

KryoNet – szerver inicializálás

```
server.addListener(object : Listener() {
    override fun connected(connection: Connection?) {
        super.connected(connection)
    }

    override fun received(connection: Connection, obj: Any) {
        …
    }

    override fun disconnected(connection: Connection?) {
        super.disconnected(connection)
        …
    }
})

server.start()
server.bind(54555, 54777) // tcp és udp portok
```

KryoNet – kliens felderítés és kapcsolódás

```
var client = Client()
client.start()
val kryo = client.kryo
kryo.register(TextMessage::class.java)
kryo.register(ImageMessage::class.java)
kryo.register(ByteArray::class.java)
val address = client.discoverHost(54777, 20000)

client.addListener(object : Listener() {
    override fun connected(connection: Connection?) {
        super.connected(connection)

    }

    override fun received(connection: Connection, obj: Any) {
        handleNetworkMessage(obj)
    }

    override fun disconnected(connection: Connection?) {
        super.disconnected(connection)
        runOnUiThread {
            Toast.makeText(this@MainActivity, "DISCONNECTED", Toast.LENGTH_LONG).show()
        }
    }
})
client.connect(20000, address, 54555, 54777)
```

Összefoglalás

- Hálózati kommunikáció – HTTP
  - Aszinkron kommunikáció
  - Retrofit
  - JSON parseolás
- WebView
- TCP/IP, UDP

Kérdések