# MOBILNE APLIKACIJE

## Vežbe 7

## Mobilne komunikacije

2022/2023

# Sadržaj

# 1. JSON

JSON je format za lakšu razmenu podataka. Podaci se zapisuju kao parovi ključ/vrednost. Ključ se navodi kao tekst pod duplim navodnicima nakon čega sledi vrednost.

```
"firstName":"John"
```

JSON vrednosti mogu biti:

- A number (integer or floating point)
- A string (in double quotes)
- A Boolean (true or false)
- An array (in square brackets)
- An object (in curly braces)
- Null

JSON objekat se zapisuje u parovima ključ/vrednost koji se nalaze unutar vitičastih zagrada:

```
{"firstName":"John", "lastName":"Doe"}
```

JSON Array sadrži ključ, nakon čega sledi niz elemenata u uglasim zagradama:

```
"employees":[
    {"firstName":"John", "lastName":"Doe"},
    {"firstName":"Anna", "lastName":"Smith"},
    {"firstName":"Peter","lastName":"Jones"}
]
```

Na adresi http://www.jsonschema2pojo.org/ mogu se izgenerisati Java klase kopiranjem JSON sadržaja.

Potrebno je ispratiti par koraka:

- U *Package* delu napisati pun naziv paketa u projektu
- Za *Class* name dati naziv klasi koju generišemo
- Za *Source* type izabrati JSON
- Za *Annotation style* izabrati Gson
- [Opciono] *Preview* daje prikaz generisanih klasa
- Zip opcija preuzima zip fajl sa generisanim klasama i paketima koje možete importovati u projekat prostim *Copy-Paste* mehanizmom
- [Opciono] Ostala svojstva (*check boxovi*) mozete izmeniti po potrebi

## 2. REST web servis

Fokus RESTful servisa je na resursima i kako omogućiti pristup tim resursima. Resurs može biti predstavljen kao objekat, datoteka na disku, podaci iz aplikacije, baze podataka itd.

Prilikom dizajniranja sistema prvo je potrebno da identifikujemo resurse i da ustanovimo kako su međusobno povezani. Ovaj postupak je sličan modelovanju baze podataka.

Kada smo identifikovali resurse, sledeći korak je da pronađemo način kako da te resurse reprezentujemo u našem sistemu. Za te potrebe možemo koristiti bilo koji format za reprezentaciju resursa (npr. JSON).

Da bismo poslali zahtev nekom *web servisu* da dobijemo neki sadržaj, moramo napraviti jedan HTTP zahtev. HTTP zahtev se sastoji od nekoliko elemenata:

- <VERB> GET, PUT, POST, DELETE, OPTIONS itd. ili opis šta želimo da uradimo (HTTP metode)
- <URI> Putanja do resursa nad kojim ce operacija biti izvedena

Nakon svakog HTTP zahteva sledi i HTTP odgovor od servera klijentu tj. onome ko je zahtev poslao. Klijent kao odgovor dobija sadržaj i statusni kod. Ovaj kod nam govori da li je naša operacija izvršena uspešno ili ne. Kodovi su reprezentovani celim brojevima i to:

- *Success* 2xx - sve je prošlo ok.
- *Redirection* 3xx - desila se redirekcija na neki drugi servis.
- *Error* 4xx, 5xx - javila se greška.

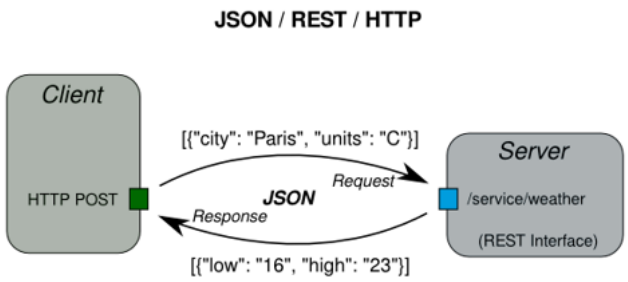
Svaki servis mora imati neku adresu na koju šaljemo HTTP zahtev. Primer:

http://MyService/Persons/1

Ako želimo da izvedemo nekakav upit nad našim servisom, to možemo da uradimo tako što dodamo **?** simbol na kraj putanje. Nakon specijalnog simbola, slede parovi ključ-vrednost spojeni **&** simbolom, ako tih parametara ima više od jednog. Primer:

http://MyService/Persons/1?format=json&encoding=UTF8

Servisi koje pozivamo preko interneta imaju već definisanu strukturu (tačnu putanju, metod kojim se pozivaju, podatke koje očekuju, kako se pretražuju). Jedino što je potrebno da uradimo je da nađemo konkretan servis i pogledamo kako on tačno očekuje da vršimo komunikaciju sa njim.




Slika 1. Primer komunikacije između klijenta i servera

## 2.1 Poziv REST web servisa koristeći *Retrofit*

Da bismo instalirali *Retrofit* biblioteku potrebno je dodati nekoliko biblioteka u *build.gradle* datoteci.

```
implementation 'com.google.code.gson:gson:2.8.6'
implementation 'com.squareup.retrofit2:retrofit:2.3.0'
implementation 'com.squareup.retrofit2:converter-gson:2.3.0'
```

Slika 2. Dodavanje biblioteka

Kada su biblioteke instalirane, potrebno je definisati *retrofit* instancu koja će vršiti HTTP zahteve ka određenom REST servisu (slika 3).

```
public static Retrofit retrofit = new Retrofit.Builder()
        .baseUrl(SERVICE_API_PATH)
        .addConverterFactory(GsonConverterFactory.create())
        .client(test())
        .build();
```

Slika 3. Kreiranje *retrofit* instance

Za svaki servis koji želimo da pozovemo moramo da definišemo metod sa kojim se poziva (GET, POST..), ali i dodatne parametre upita, dodatne putanje, a u slučaju POST zahteva i sadržaj koji je opisan JSON mapiranom Java klasom (u našem primeru klasa *TagToSend*).

Na slici 4 nalazi se primer *ReviewerService* interfejsa koji sadrži jednu metodu *add.* Ova metoda opisuje koju metodu koristimo i šta očekujemo kao rezultat.

```
public interface ReviewerService {

    @Headers({
            "User-Agent: Mobile-Android",
            "Content-Type:application/json"
    })
    @POST(ServiceUtils.ADD)
    Call<ResponseBody> add(@Body TagToSend tag);
}
```

Slika 4. Interfejs *ReviewerService*

Spisak dostupnih anotacija, koje *retrofit* podržava, se nalazi na slici 5.

| Annotation | Description |
|---|---|
| `@Path` | variable substitution for the API endpoint (i.e. username will be swapped for `{username}` in the URL endpoint). |
| `@Query` | specifies the query key name with the value of the annotated parameter. |
| `@Body` | payload for the POST call (serialized from a Java object to a JSON string) |
| `@Header` | specifies the header with the value of the annotated parameter |

Slika 5. Anotacije i njihova značenja

Ukoliko neki servis, koji želimo da pozovemo, zahteva dodatne elemente, koje treba poslati unutar zaglavlja HTTP zahteva, to možemo definisati koristeći *Header* anotaciju. Svi elementi koji se navode su parovi ključ-vrednost (slika 6).

```
@Headers({"Cache-Control: max-age=640000", "User-Agent: My-App-Name"})
@GET("/some/endpoint")
```

Slika 6. Primer zaglavlja

U metodi *onStartCommand* klase *SyncService* nalazi se kod koji obrađuje odgovor REST servisa (slika 3). Servis koji pozivamo izgleda ovako:

`http://<service_ip_adress>:<service_port>/rs.ftn.reviewer.rest/rest/proizvodi`

Sam poziv REST servisa se odvija u pozadini i mi ne moramo da vodimo računa o tome, samo je potrebno da registrujemo šta da se desi kada odgovor stigne do nas. Taj deo se implementira dodavanjem *Callback<List<Event>>* unutar *enqueue* metode (slika 7). U *if*-u proveravamo da li je statusni kod HTTP odgovora jednak 200 i u tom slučaju u logujemo prvu poruku.

```
        Call<ResponseBody> call = ServiceUtils.reviewerService.add(tts);
        call.enqueue(new Callback<ResponseBody>() {
            @Override
            public void onResponse(Call<ResponseBody> call, Response<ResponseBody> response) {
                if (response.code() == 200){
                    Log.d( tag: "REZ", msg: "Meesage recieved");
                }else{
                    Log.d( tag: "REZ", msg: "Meesage recieved: "+response.code());
                }
            }

            @Override
            public void onFailure(Call<ResponseBody> call, Throwable t) {
                Log.d( tag: "REZ", t.getMessage() != null?t.getMessage():"error");
            }
        });
    }

    sendBroadcast(ints);

    stopSelf();

    return START_NOT_STICKY;
}
```

Slika 7. Logovanje poruka u zavisnosti od HTTP odgovora

## 3. Učitavanje slika sa interneta

Kada želimo da učitavamo slike sa interneta potrebno je da posao odvijamo u pozadini – inače blokiramo glavnu nit aplikacije. Ovaj posao možemo da radimo sami ili da koristimo pomoć. ***Picasso*** je zgodna biblioteka za rešavanje ovog problema. Da bi koristili biblioteku potrebno je dodati:

```
implementation 'com.squareup.picasso:picasso:2.71828'
```

Slika 8: Dodavanje biblioteke

unutar *build.gradle* datoteke na nivou modula.

Nakon toga slike možemo učitati na sledeći način:

```
Picasso.get().load( path: "https://i.imgur.com/My0BRjX.jpeg")
        .placeholder(R.drawable.ic_action_error).into((ImageView) requireView()
                .findViewById(R.id.image_con));
```

Slika 9: Učitavanje slike

## 4. Primer

Domaći se nalazi na *Canvas*-u (*canvas.ftn.uns.ac.rs*) na putanji *Вежбе/07 Вежбе/07 Задатак.pdf*.

Primer možete preuzeti na sledećem linku: *https://gitlab.com/antesevicceca/mobilne-aplikacije*.

Za dodatna pitanja možete se obratiti asistentima:

- Svetlana Antešević (*svetlanaantesevic@uns.ac.rs*)
- Jelena Matković (*matkovic.jelena@uns.ac.rs*)