JSON, REST, Networking

JSON

DEFINICIJA, OPIS, VIZUELIZACIJA, GENERISANJE JAVA KLASA

JSON: JavaScript Object Notation

- JSON je format za laksu razmenu podataka.
- Podaci se zapisuju kao parovi kljuc/vrednost
- Kljuc se navodi kao tekst pod duplim navodnicima nakon cega 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 object i JSON array

- JSON object se zapisuje u parovima kljuc/vrednost koji se nalaze unutar viticastih zagrada
  - **{"firstName":"John", "lastName":"Doe"}**

- JSON Array sadrzi kljuc, nakon cega sledi niz elemenata u uglasim zagradama
  - **"employees":[**

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

Vizualizacija JSON sadrzaja

- Na adresi http://jsonviewer.stack.hu/ moze se izvrsiti vizuelni prikaz JSON sadrzaja
- JSON sadrzaj je potrebno kopiratu u **Text** tab
- **Viewer** tab nam prikazuje sadrzaj JSON fajla, podeljen na objekte nizove i daje jasan prikaz.
- Ukoliko sadrzaj nije dobar, alat izbacuje gresku sto je jasan pokazatelj da JSON sadrzaj nije leo formatiran.
- Voditi racuna da se sav JSON sadrzaj na kraju nadje izmeju **{}** zagrada.

Generisanje Java klasa iz JSON sadrzaja

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

- Potrebno je ispratiti par koraka:
  - U **Package** delu napisati pun naziv paketa u projektu
  - Za **Class name** dati naziv klasi koju generisemo
  - 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 mozete importovati u projekat prostim **Copy-Paste** mehanizmom
  - **[Opciono]** Ostala svojstva (check boxovi) mozete izmeniti po potrebi

Rest web service

DEFINICIJA, OPIS, UPOTREBA

Rest servce

- Fokus RESTful service je na resursima i kako omoguciti pristup tim resursima.
- Resurs moze biti predstavljen kao objekat, fajl na disku, podaci iz aplikacije, baze podataka, ...
- Prilikom dizajniranja sistema prvo je potrebno da identifikujemo resurse i da ustanovimo kako su medjusobno povezani.
- Ovaj postupaka je slican modelovanju baze podataka
- Kada smo identifikovali resurse, sledeci korak je da pronadjemo nacin kako da te resurse reprezentujemo u nasem sistemu.
- Za te potrebe mozemo koristiti bilo koji format za reprezentaciju resursa (JSON npr.).

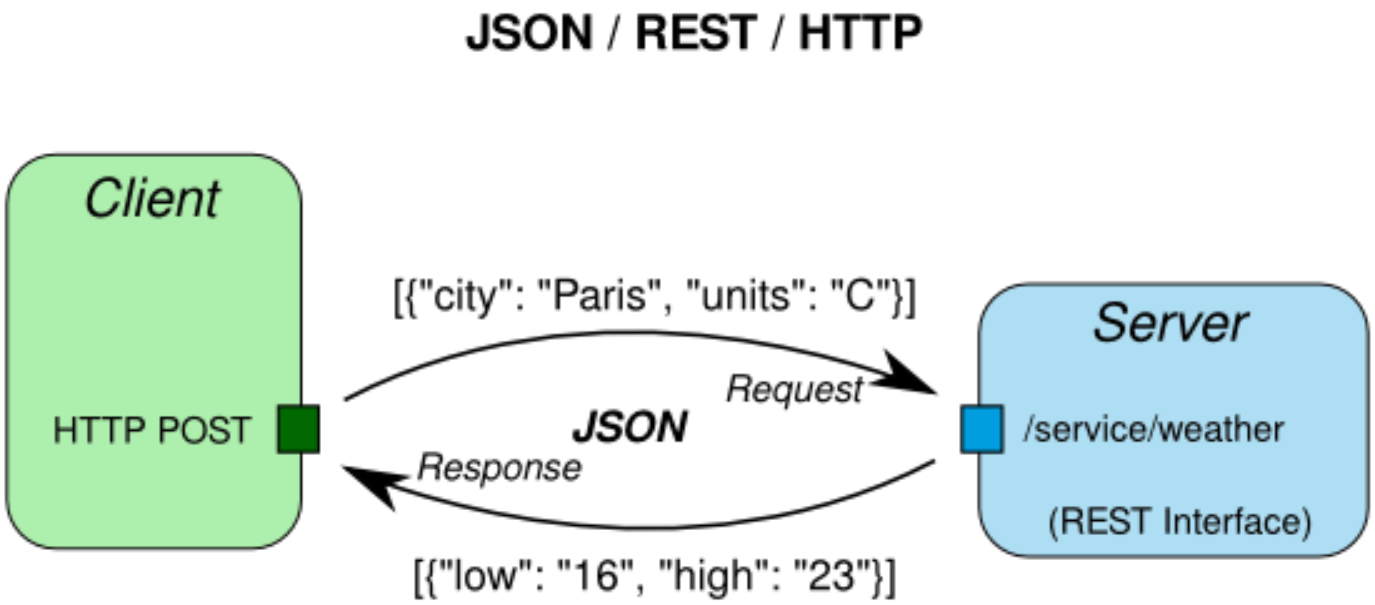
Rest servce

- Da bi poslali zahtev nekom web servisu da dobijemo neki sadrzaj, moramo napraviti jedna HTTP zahteh.
- Nakon svakog HTTP zahteva sledi i HTTP odgovor od servera klijentu tj onome ko je zahtev poslao.
- HTTP zahtev se sastoji od nekoliko elemenata:
  - **<VERB> GET, PUT, POST, DELETE, OPTIONS, etc ili opis sta zelmo da uradimo**
  - **<URI> Putanja do resursa nad kojim ce operacija biti izvedena**
- Kada klijent dobije odgovor dobice sadrzaj ali i **status cod.**
- Ovaj kod nam govori da li je nasa operacija izvrsena uspesno ili ne.
- Ovi kodovi su reprezentovani celim brojevima I to:
  - **Success 2xx sve je proslo ok**
  - **Redirection 3xx desio se redirect na neki drugi servis**
  - **Error 4xx, 5xx javila se greska**

Rest servce

- Svaki servis mora imati neku adresu na koju saljemo HTTP zahtev i onda se oznacava kao
  - **http://MyService/Persons/1**
- Ako zelimo da izvedemo nekakav upit nad nasim servisom, to mozemo da uradimo dodavajuci **?** simbol na kraj putanje.
- Nakon specijanog simbola slede parovi **kljuc-vrednost** spojeni **&** simbolom ako tih parametara ima vise od jednog
  - **http://MyService/Persons/1?format=json&encoding=UTF8**
- Servisi koje pozivamo preko internet imaju vec definisanu strukturu (tacnu putanju metod kojim se pozivaju, podatke koje ocekuju, kako se pretrazuju). Jedino sto je potrebno da uradimo je da nadjemo konkretan servis I pogledamo kako on tacno ocekuje da vrsimo komunkiaciju sa njim.

Mobile Rest komunikacija



Retrofit

POZIV REST WEB SERVICE-A KORISTECI RETROFIT

Instalacija

- Da bi instalirali retrofit biblioteku ta poziv Rest servisa poterbno je dodati nekoliko biblioteka u *build.gradle* fajl:

```
dependencies {
    compile 'com.google.code.gson:gson:2.6.2'
    compile 'com.squareup.retrofit2:retrofit:2.1.0'
    compile 'com.squareup.retrofit2:converter-gson:2.1.0'

}
```

Podešavanje servisa

- Kada su biblioteke instalirane potrebno je definisati *retrofit* instancu koja će vršiti HTTP zahteve ka određenom REST servisu. Takođe treba dodati osnovu servisa na koji se nadograđuju ostali pozivi:

```
// Trailing slash is needed
public static final String BASE_URL = "http://api.myservice.com/";
Retrofit retrofit = new Retrofit.Builder()
    .baseUrl(BASE_URL)
    .addConverterFactory(GsonConverterFactory.create())
    .build();
```

```
Gson gson = new GsonBuilder()
        .setDateFormat("yyyy-MM-dd'T'HH:mm:ssZ")
        .create();

Retrofit retrofit = new Retrofit.Builder()
    .baseUrl(BASE_URL)
    .addConverterFactory(GsonConverterFactory.create(gson))
    .build();
```

Definicija mapiranih servisa

- Za svaki servis koji žeimo 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:

```
public interface MyApiEndpointInterface {
    // Request method and URL specified in the annotation
    // Callback for the parsed response is the last parameter

    @GET("users/{username}")
    void getUser(@Path("username") String username, Callback<User> cb);

    @GET("group/{id}/users")
    void groupList(@Path("id") int groupId, @Query("sort") String sort, Callback<List<User>>

    @POST("users/new")
    void createUser(@Body User user, Callback<User> cb);
}
```

Maperi za parametre poziva servisa

- Spisak dostupnih anotacija koje retrofit podržava
- Nova verzija podržava i *QueryMap* što olakšava pisanje više od jednog filter/selekcionog parametra prilikom poziva service-a)

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

Dodavanje dodatnih header parametara

- Ukoliko neki servis koji žeimo da pozovemo zahteva dodatne elemente koje treba poslati unutar zaglavlja HTTP zahteva to možemo definisati za servis koristeći *Header* anotaciju. Svi elementi koji se navode su parovi ključ-vrednost:

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

Definicija instance servisa

Pre nego što počnemo da vršimo pozive servisa kroz android aplikaciju, potrebno je da spojimo retrofit instancu sa interface-om koji sadrži opise svih mapiranih servisa. Nakon toga slobodno možemo pozivati servise:

```
MyApiEndpointInterface apiService =
    retrofit.create(MyApiEndpointInterface.class);
```

Asinhroni poziv servisa

- Pozive možemo izvoditi koristeći *AsyncTask,* međutim, retrofit nam pruža način da pozive izvodimo asinhrono ali da ne moramo da pišemo dodatan kod
- Potrebno je da pozovemo naš servis i da mu prosledimo metodu, koja opisuje šta da se desi kada dobijmo odgovor sa servisa. U ovoj metodi voditi rašuna o kodu statusa koji stiše sa odgovorom.

```
String username = "sarahjean";
Call<User> call = apiService.getUser(username);
call.enqueue(new Callback<User>() {
    @Override
    public void onResponse(Call<User> call, Response<User> response) {
        int statusCode = response.code();
        User user = response.body();
    }

    @Override
    public void onFailure(Call<User> call, Throwable t) {
        // Log error here since request failed
    }
});
```

Ucitavanje slika sa interneta

- Kada zelimo da ucitavamo slike sa internet potrebno je da posao odvijamo u pozadini – inace blokiramo glavnu nit aplikacije
- Ovaj posao mozemo da radimo sami ili da koristimo pomoc ☺
- **Picasso** je zgodna bibliteka za resavanje ovog problema.

```
compile 'com.squareup.picasso:picasso:2.5.2'
```

```
Picasso.with(context).load("http://i.imgur.com/DvpvklR.png").into(imageView);
```

Dodatni sadržaj

- Retrofit wiki
  - https://github.com/codepath/android_guides/wiki/Consuming-APIs-with-Retrofit
- Retrofit main page
  - http://square.github.io/retrofit/
- Retrofit tutorial step by step
  - https://futurestud.io/tutorials/retrofit-getting-started-and-android-client
- JSON sintaksa:
  - http://www.w3schools.com/js/js_json_syntax.asp
- Restfull web service
  - http://www.drdobbs.com/web-development/restful-web-services-a-tutorial/240169069

Pitanja?

HVALA NA PAŽNJI