Programación multimedia y dispositivos móviles

Android y JSON

Universidad Europea

LAUREATE INTERNATIONAL UNIVERSITIES

- **JSON** (JavaScript Object Notation) es un formato para el intercambios de datos, básicamente JSON describe los datos con una sintaxis dedicada que se usa para identificar y gestionar los datos. Nació como una alternativa a XML y, una de las mayores ventajas que tiene su uso es que puede ser leído por cualquier lenguaje de programación. Por lo tanto, puede ser usado para el intercambio de información entre distintas tecnologías. Además debido a su naturaleza y al ser más compacto suele ser mucho más rápido trabajar con JSON que con XML.

Ejemplo de código JSON

Si tuviéramos un listado de frutas, por ejemplo:
Frutas:

- Manzana - 6 unidades
- Pera - 8 unidades
- Naranja - 10 unidades

Esto se traduciría en JSON de la siguiente manera:

```
{
   "frutas": [
       { "nombre_fruta":"Manzana" , "cantidad":6 },
       { "nombre_fruta":"Pera" , "cantidad":8 },
       { "nombre_fruta":"Naranja" , "cantidad":10 }
    ]
}
```

JSONObject

- Para leer en java de un archivo JSON debemos tratar el contenido del archivo como un String.
- Partiendo de un String que contiene el código JSON utilizaremos la clase JSONObject
- La clase JsonObject representa un objeto JSON inmutable (una colección desordenada de cero o más pares de nombre / valor). También proporciona un mapa no modificable de las asignaciones de nombre / valor del objeto JSON.
- El valor de un JsonObject puede ser JsonObject, JsonArray, JsonString, JsonNumber, JsonValue.TRUE, JsonValue.FALSE, JsonValue.NULL.

JSONArray

- La clase **JSONArray** representa un array JSONObjects inmutable (una secuencia ordenada de cero o más valores). También proporciona una lista no modificable de los valores del array.
- El valor de un JSONArray puede ser JsonObject, JsonArray, JsonString, JsonNumber, JsonValue.TRUE, JsonValue.FALSE, JsonValue.NULL.
- Si sabemos que el valor de un nombre es una lista podemos obtener el array por medio del método **optJSONArray** de la clase JSONObject, pasándole como parámetro dicho nombre

Ejemplo

En el ejemplo de la fruta:

```
//Creamos un objeto JSON a partir de la cadena
JSONObject object = new JSONObject(cadenaJson);

//cogemos la lista dentro de la etiqueta "frutas"
JSONArray json_array = object.optJSONArray("frutas");

JSONObject objJSON = null;
for (int i = 0; i < json_array.length(); i++) {
    objJSON = json_array.getJSONObject(i);
    nombre = objJSON.getString("nombre_fruta");
    unidades = objJSON.getInt("cantidad");
}
```

Ejemplo

Para manejar los objetos de la lista deberíamos crear una clase Fruta:

```
public class Fruta {
    public String nombre;
    public int unidades;

    public Fruta(JSONObject objetoJSON) {
       nombre = objetoJSON.getString("nombre_fruta");
       unidades = objetoJSON.getInt("cantidad");
    }
}
```

De manera que al manejar el array:

```
Fruta f = null;
for (int i = 0; i < json_array.length(); i++) {
    f = new Fruta(json_array.getJSONObject(i));
}
```

Ejemplo

- En el proyecto llamado App Web Service creamos un paquete llamado **model** y en este paquete creamos la clase **Pokemon**:

```
public class Pokemon {
    public String url;
    public String nombre;

    public Pokemon(JSONObject objetoJSON) {
       url = objetoJSON.getString("url");
       nombre = objetoJSON.getInt("name");
    }

    public String toString() {
        return " url  : " + url + "\n name : " + name
            + "\n-----------------------------------\n";
    }
}
```

Ejemplo

- En el método **onPostExecute** del MainActivity.java:

```
protected void onPostExecute(Void v) {
    String salida = "";
    JSONObject jsonR = null;

    try {
        jsonR = new JSONObject(res);

        JSONArray jsonMainNode = jsonR.optJSONArray("results");

        Pokemon poke = null;
        for (int i=0; i < jsonMainNode.length(); i++) {
            poke = new Pokemos(jsonMainNode.getJSONObject(i));
            salida += poke.toString();
        }

        tvRes.setText(salida);

    } catch (JSONException e) {}
}
```

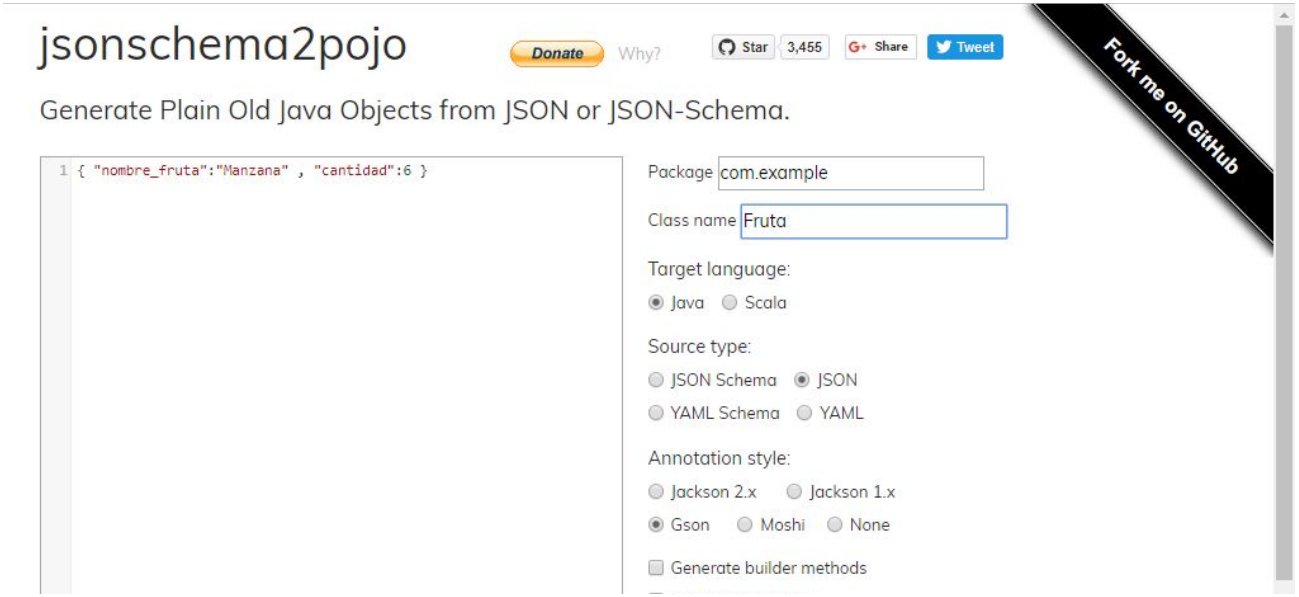
Retrofit también nos va a permitir parsear automáticamente la respuesta de un web service a un POJO (Plan Old Java Objects). En nuestro caso nos vamos a centrar en JSON

- Para poder convertir JSON a POJO con Retrofit tendremos que incluir en las dependencias las librerías de Gson de Google y la integración Gson de Retrofit.

```
compile 'com.google.code.gson:gson:2.8.2'
compile
'com.squareup.retrofit2:converter-gson:2.3.0'
```

jsonschema2pojo

- Para crear los POJOs necesarios podemos hacerlo utilizando la herramienta jsonschema2pojo.
- Pega la respuesta JSON en la caja de entrada. Selecciona el tipo de fuente de **JSON**, estilo de anotación de **Gson**, desmarca **Permitir propiedades adicionales** y cambia el nombre de la clase. Pulsa el botón **Preview** y copia el código.



```
import com.google.gson.annotations.Expose;
import com.google.gson.annotations.SerializedName;

public class Fruta {

    @SerializedName("nombre_fruta")
    @Expose
    private String nombreFruta;
    @SerializedName("cantidad")
    @Expose
    private Integer cantidad;
```

La anotación **@SerializedName** es necesaria para que Gson mapee las llaves JSON a campos de objeto Java.
La anotación **@Expose** indica que el miembro de la clase debería ser expuesto para serialización o deserialización JSON.

```
    public String getNombreFruta() {
        return nombreFruta;
    }

    public void setNombreFruta(String nombreFruta) {
        this.nombreFruta = nombreFruta;
    }

    public Integer getCantidad() {
        return cantidad;
    }

    public void setCantidad(Integer cantidad) {
        this.cantidad = cantidad;
    }
}
```

Ejemplo

- En el proyecto llamado App WS Retrofit.
- Incluimos las dependencias en el build.gradle(Module:app)

```
compile 'com.google.code.gson:gson:2.8.2'
compile
'com.squareup.retrofit2:converter-gson:2.3.0'
```

Ejemplo

- Creamos un paquete llamado **retrofitData** y creamos la clase Pokemon

```
public class Pokemon {

    @SerializedName("url")
    @Expose
    private String url;
    @SerializedName("name")
    @Expose
    private String name;

    // métodos get y set

    public String toString() {
        return " url    : " + url + "\n nombre : " + name
      + "\n-----------------------------------\n";
    }
}
```

Ejemplo

- Y la clase PokemonRes

```
public class PokemonRes {

    @SerializedName("count")
    @Expose
    private Integer count;
    @SerializedName("previous")
    @Expose
    private Object previous;
    @SerializedName("results")
    @Expose
    private List<Pokemon> results = null;
    @SerializedName("next")
    @Expose
    private String next;

    // métodos get y set
}
```

Ejemplo

- En el paquete retrofitUtils, en el interfaz APIRestService cambiamos el tipo de dato que maneja el servicio.

```
public interface APIRestService {
    public static final String BASE_URL =
"https://pokeapi.co/api/v2/";

    @GET("pokemon/")
    Call<PokemonRes> obtenerPokemon();

}
```

Ejemplo

- Y en la clase **RetrofitClient** cambiamos el conversor

```
public class RetrofitClient {
    private static Retrofit retrofit = null;

    public static Retrofit getClient(String baseUrl) {
        if (retrofit==null) {
            retrofit = new Retrofit.Builder()
                    .baseUrl(baseUrl)
        .addConverterFactory(GsonConverterFactory.create())
                    .build();
        }
        return retrofit;
    }

}
```

Ejemplo

- En el **MainActivity**.java, el método **consumirWS** quedaría de la siguiente manera:

```
public void consumirWS(View v) {

    Retrofit r =
RetrofitClient.getClient(APIRestService.BASE_URL);
    APIRestService ars = r.create(APIRestService.class);
    Call<PokemonRes> call = ars.obtenerPokemon();

    call.enqueue(new Callback<PokemonRes>() {

        String res;
        TextView tvRes = (TextView)findViewById(R.id.tvResultado);
```

Ejemplo

```
        @Override
        public void onResponse(Call<PokemonRes> call,
Response<PokemonRes> response) {
            if (!response.isSuccessful()) {
                Log.i(TAG, "Error" + response.code());
            } else {
                PokemonRes pokeRes = response.body();
                for (Pokemon poke: pokeRes.getResults()) {
                    res += poke.toString();
                }
                tvRes.setText(res);
            }
        }

        @Override
        public void onFailure(Call<PokemonRes> call, Throwable t){
            Log.e("error", t.toString());
        }
    });
}
```

Universidad Europea

LAUREATE INTERNATIONAL UNIVERSITIES

Madrid Valencia Canarias