

# ANDROID TP 8 {JSON}

Khaled BELKADI

# PLAN

---

- INTRODUCTION
  - JSON ? C'EST QUOI ?
  - JSON ? POURQUOI ?
  - EXEMPLE
- ANDROID & JSON
- TP
- Documentation
  - JSONObject, JSONArray
  - Ou JsonReader, JsonWriter

# INTRODUCTION : JSON ? C'EST QUOI ?

---

- JavaScript Object Notation ou JSON, est une notation dérivée des objets du langage JavaScript ;
- Permet de sérialiser/structurer de l'information (objets, tableaux, nombres, chaînes de caractères, booléen et de null) sous forme **textuelle** comme le XML ;
- Basé sur le principe d'ensemble de paires clé/valeur (tel une HashMap en Java) ;
- Démystification :
  - “It is based upon JavaScript syntax but is distinct from it: some JavaScript is not JSON, and some JSON is not JavaScript.” - [https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global\\_Objects/JSON](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/JSON)
  - Plus information : <http://timelessrepo.com/json-isnt-a-javascript-subset>

# INTRODUCTION : JSON ? POURQUOI ?

---

- Avantages :
  - Peu verbeux ;
  - Facile à apprendre ;
  - Simple à parser ;
  - Format light-weight permettant d'échanger de l'informations entre applications en sérialisant les données au format JSON;
- Inconvénients :
  - Les données ne sont pas typées.
  - Représentation non-extensible.

# EXEMPLE JSON Contact

---

```
{  
  "prenom": "Khaled",  
  "nom": "Belkadi",  
  "adresse": {  
    "rue": "Paul Langevin",  
    "ville": "Villeneuve d'Ascq",  
    "codePostal": 59260  
  },  
  "numTelephone": [  
    "06 00 01 11 01",  
    "06 11 10 01 11"  
  ]  
}
```

Vous n'êtes pas sur de vous ?  
Alors : <http://jsonlint.com/>

# EXEMPLES : List

```
[  
  {"id": 115, "titre": "Linux VS Windows", "Description" : "Salut", "date": 15508489},  
  {"id": 134, "titre": "JSON VS XML", "Description" : "Coucou", "date": 15508489}  
]
```

Le code ci-dessus convient que dans certains cas ;

Que faire si nous avons besoin d'un accès direct à l'article id = 134 ?

# EXEMPLES : List avec accès direct

```
{  
  "115" : {"titre": "Linux VS Windows", "Description" : "Salut", "date": 15508489},  
  "134" : {"titre": "JSON VS XML", "Description" : "Coucou", "date": 15508489}  
}
```

Nous allons aider le parser en structurant les données plus intelligemment selon nos besoins.

# ANDROID & JSON

- android.util
  - [JsonReader](#), [JsonWriter](#)
  - Implémentation bas niveau
  - Très verbeux
  - A partir d'Android 3.0.x | API : 11
- org.json
  - [JSONObject](#)
  - [JSONArray](#)
  - Implémentation haut niveau proposant du databinding simple.
  - Pas efficace lorsque le JSON est volumineux car celui-ci est parser en entièreté dès l'instance de la classe.

Un plus : Compromis des deux solutions : [GSON](#)



# TP 8 JSON : CRÉER DU JSON

1. Transformer les informations du formulaire en JSON , le format à respecter est le suivant :

```
{
  "prenom": "Khaled",
  "nom": "Belkadi",
  "adresse": {
    "numero": 10,
    "rue": "Hum",
    "codePostal": 59260,
    "ville": "Villeneuve D'Ascq"
  },
  "telephones": ["06 10 10 15 23", "06 78 17 94 44"]
}
```

A implémenter : `String Contact.contactToJson(Contact contact);`

Le but est de sérialiser une instance de Contact en JSON

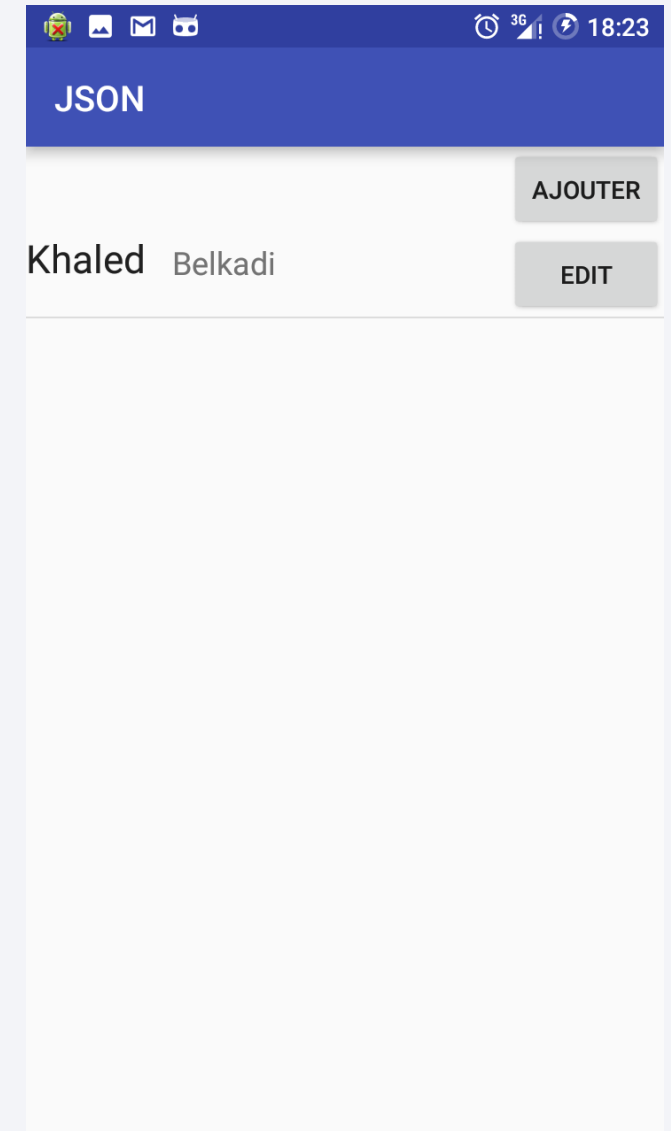
The screenshot shows a mobile application interface with a blue header labeled "JSON". The form contains the following fields and controls:

- First Name:** A text input field containing "Khaled".
- Last Name:** A text input field containing "Belkadi".
- Adresse:** A section header for the address fields.
- Address Number:** A text input field containing "10".
- Street:** A text input field containing "Hum".
- Postal Code:** A text input field containing "59260".
- City:** A text input field containing "Villeneuve D'Ascq".
- Telephone:** A section header for the telephone fields.
- Phone Number 1:** A text input field containing "06 10 10 15 23".
- Phone Number 2:** A text input field containing "06 78 17 94 44".
- Ajouter un Numéro:** A button to add a new phone number.
- VALIDER:** A button at the bottom right to validate the form.

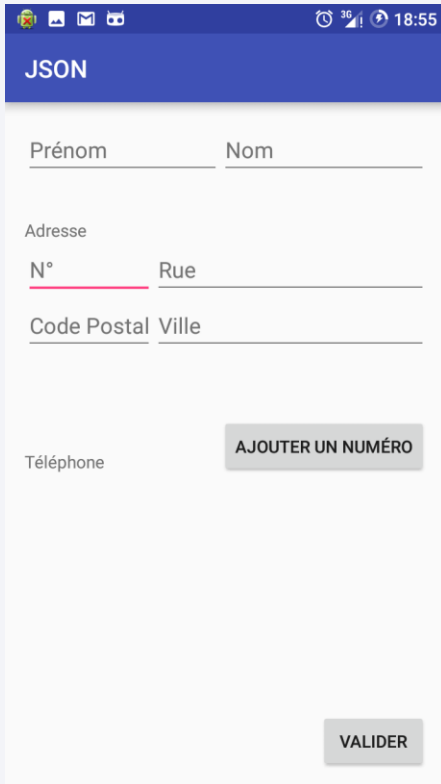
# TP 8 JSON : LIRE DU JSON

- L'ajout renvoi le contact Json au format d'un String ;
- Il faut donc le parser afin de créer une instance de contact pour alimenter ma ListView.

**A implémenter :** `Contact Contact.fromJson(String json) ;`



# LIENS POUR LE TP



JSON

Prénom \_\_\_\_\_ Nom \_\_\_\_\_

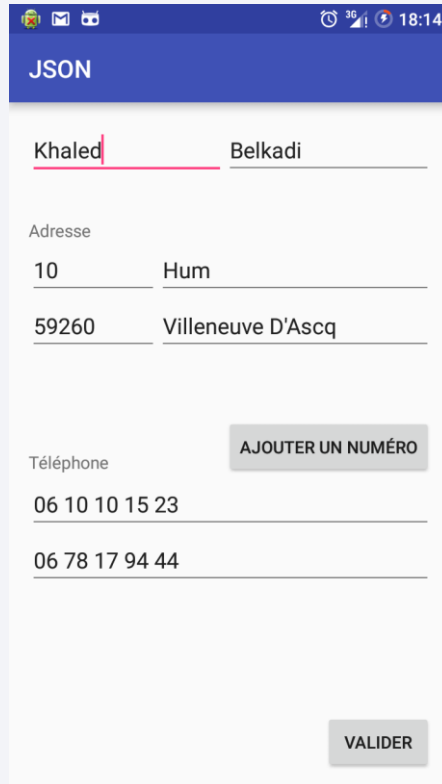
Adresse

N° \_\_\_\_\_ Rue \_\_\_\_\_

Code Postal \_\_\_\_\_ Ville \_\_\_\_\_

Téléphone \_\_\_\_\_ AJOUTER UN NUMÉRO

VALIDER



JSON

Khaled \_\_\_\_\_ Belkadi \_\_\_\_\_

Adresse

10 \_\_\_\_\_ Hum \_\_\_\_\_

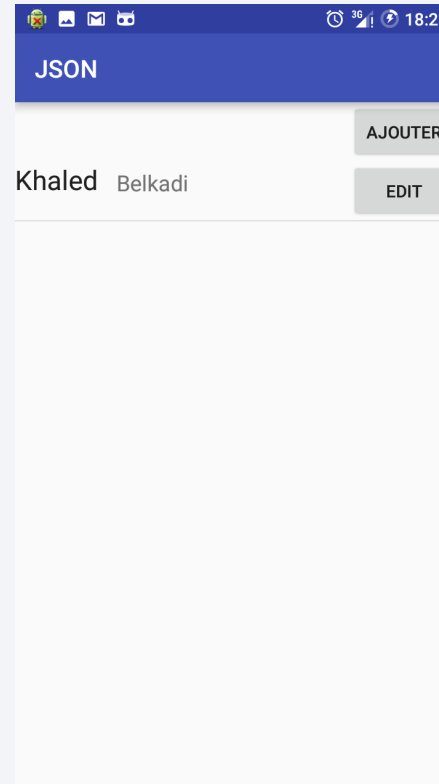
59260 \_\_\_\_\_ Villeneuve D'Ascq \_\_\_\_\_

Téléphone 06 10 10 15 23

06 78 17 94 44

AJOUTER UN NUMÉRO

VALIDER



JSON

Khaled Belkadi

AJOUTER

EDIT

TP 8 JSON :

[https://drive.google.com/open?id=0B\\_DyleomNOifOGJGZWlKcTR3Wkk](https://drive.google.com/open?id=0B_DyleomNOifOGJGZWlKcTR3Wkk)

Solution version org.json :

[https://drive.google.com/open?id=0B\\_DyleomNOifVVVjMHZNU2N1akE](https://drive.google.com/open?id=0B_DyleomNOifVVVjMHZNU2N1akE)

Solution version android.util :

[https://drive.google.com/open?id=0B\\_DyleomNOifWEQwWjlxT3N3UVE](https://drive.google.com/open?id=0B_DyleomNOifWEQwWjlxT3N3UVE)



Documentation

# Exemple de code org.json.JSONObject

- Cet classe permet de créer un objet JSON implémentant l'interface `Map<String,JsonValue>`
- Rappels :
  - La clé doit être impérativement une chaine de caractère.
  - JsonValue est soit :
    - Un autre objet JSON
    - Un tableau (JSONArray)
    - Un nombre
    - Une chaine de caractères
    - Un booléen
    - Ou un null

# Exemple de code org.json.JSONObject

- `JSONObject fiche = JSONObject();`
  - `{ }`
- `fiche.put("prenom", "Khaled");`
  - `fiche => {"prenom": "Khaled"}`
- `JSONObject adresse = JSONObject(); // adresse vaut : { }`
- `adresse.put("codePostal", 59260);`
  - `adresse => {"codePostal": 59260}`
- `adresse.put("ville", "Villeneuve d'Ascq");`
  - `adresse => {"codePostal": 59260, "ville": "Villeneuve d'Ascq"}`
- `fiche.put("adresse", adresse);`
  - `fiche => {"prenom": "Khaled", "adresse": {"codePostal": 59260, "ville": "Villeneuve d'Ascq"}}`

# Exemple de code org.json.JSONObject read

- `JSONObject fiche = new JSONObject("{\"prenom\": \"Khaled\", \"adresse\": {\"codePostal\": 59260, \"ville\": \"Villeneuve d'Ascq\"}, \"telephone\": [\"06121464\", \"06124778\"]});`
- `fiche.getString(\"prenom\");`
  - Retourne une string comprenant: Khaled
- `JSONObject adresse = fiche.getJSONObject(\"adresse\");`
  - Retourne une JSONObject ayant comme instance `{\"codePostal\": 59260, \"ville\": \"Villeneuve d'Ascq\"}`
  - `adresse.getInt(\"codePostal\");`
    - Retourne un Int valant 23
- `JSONArray telephones = fiche.getJSONArray(\"telephone\");`
  - `telephones.getString(1);`
    - Retourne la string : 06124778

# Exemple de code org.json.JSONArray

- Le fonctionnement est basé sur une `List<JsonValue>`
- `JSONArray telephones = new JSONArray();`
  - `telephones => [ ]`
- `telephones.put("06121464");`
  - `telephones => ["06121464"]`
- `telephones.put("06124778");`
  - `telephones => ["06121464", "06124778"]`
- `telephones.getString(1)`
  - Retourne le string : `06124778`

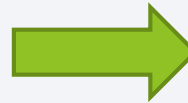




Documentation parseur ANDROID NATIF

# JsonWriter : créer un JSON

```
{  
  "prenom": "Khaled",  
  "adresse": {  
    "rue": "Paul Langevin",  
    "ville": "Villeneuve d'Ascq",  
  },  
  "ips": [  
    "195.142.54.14",  
    "197.142.51.17",  
  ]  
}
```



```
StringWriter resultToStringWriter = new StringWriter(); /* output stream */  
  
JsonWriter jsonWriter = new JsonWriter(resultToStringWriter);  
jsonWriter.beginObject(); /* création de l'objet parent */  
  
jsonWriter.name("prenom").value("Khaled");  
  
jsonWriter.value("adresse");  
jsonWriter.beginObject(); /* création de l'objet adresse */  
jsonWriter.name("rue").value("Paul Langevin");  
jsonWriter.name("ville").value("Villeneuve d'Ascq");  
jsonWriter.endObject(); /* fermeture de l'objet adresse */  
  
jsonWriter.value("ips");  
jsonWriter.beginArray(); /* création de l'array filmsPref */  
jsonWriter.value("195.142.54.14");  
jsonWriter.value("197.142.51.17");  
jsonWriter.endArray(); /* fermeture l'array filmsPref */  
  
jsonWriter.endObject(); /* fermeture de l'objet parent */  
  
/* close des streams */  
jsonWriter.close(); resultToStringWriter.close();  
  
/* output text => resultToStringWriter.toString()
```

# Exemple de code JsonReader

```
{  
  "prenom": "Khaled",  
  "adresse": {  
    "rue": "Paul Langevin",  
    "ville": "Villeneuve d'Ascq",  
  },  
  "ips": [  
    "195.142.54.14",  
    "197.142.51.17",  
  ]  
}
```



```
StringReader sr = new StringReader(<string du json à lire>);  
JsonReader jr = new JsonReader(sr);  
  
jr.beginObject(); /* lecture de l'objet parent */  
while (jr.hasNext()) /* lire tant qu'il y a une clé non-lue */  
{  
    String key = jr.nextName(); /* récupération de la clé suivante */  
    switch (key)  
    {  
        case "prenom": /* quand la clé vaut prenom */  
            prenom = jr.nextString(); /* récupération de la valeur suivante */  
            break;  
  
        case "adresse":  
            jr.beginObject(); /* lecture de l'objet adresse */  
            /* lire tant qu'il y a une clé non-lue dans l'objet adresse */  
            while (jr.hasNext()) /* tant qu'il y a une clé non-lue dans l'objet  
courant (dans notre cas : adresse) */  
            {  
                key = jr.nextName(); /* lire la next key de l'objet adresse */  
                switch (key)  
                {  
                    case "rue":  
                        rue = jr.nextString();  
                        break;  
                    case "ville":  
                        ville = jr.nextString();  
                        break;  
                }  
            }  
            jr.endObject(); /* fermeture de l'objet adresse car il a été lu */  
            break;  
  
        case "filmsPref":  
            jr.beginArray();  
            int i = 0;  
            while (jr.hasNext()) /* tant qu'il y a une valeur dans l'array filmsPref */  
                ips[i++] = jr.nextString();  
            jr.endArray();  
            break;  
    }  
}  
jr.endObject(); /* fermeture de l'objet parent */  
  
jr.close(); sr.close(); /* close des streams */
```