ANDROID
ET


Sommaire

1.1 - Préambule : les données JSON

1.1.1 - Généralités

Ce chapitre a pour objectif de vous introduire aux APIs Java pour JSON (JavaScript Object Notation).

JSON est un format d'échange de données à base de texte dérivé de JavaScript et est utilisé dans les Web Services et autres applications connectées.

JSON définit seulement deux structures de données : les objets et les tableaux. Un objet est un ensemble de paires clé-valeur, et un tableau est une liste de valeurs.
JSON définit six types de données : String, Number, object, array, true, false et null.

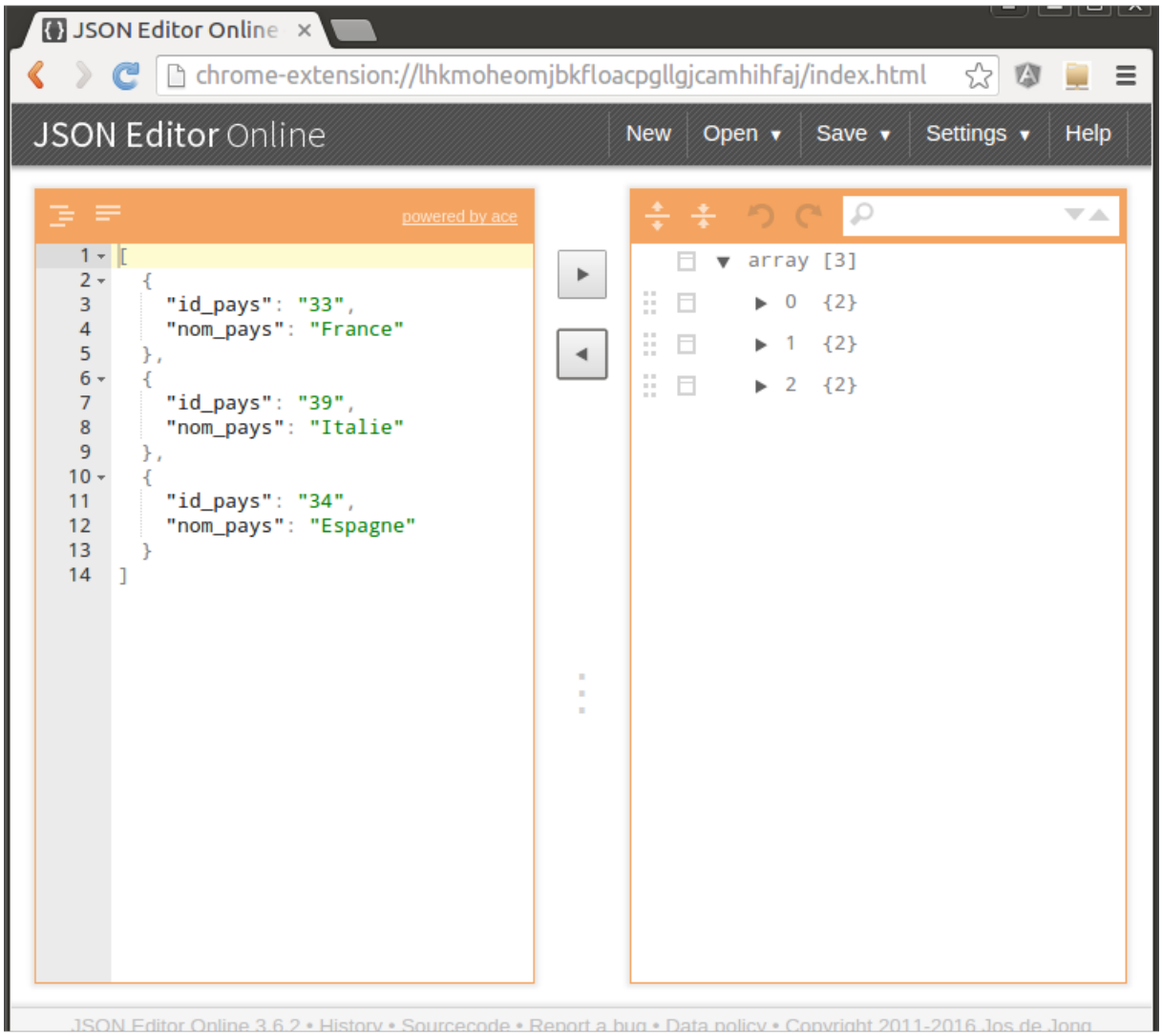
Les objets sont placées entre accolades ({}), leurs paires clé-valeur sont séparées par une virgule (,), et la clé et la valeur d'une paire sont séparées par deux points (:). Les clés dans un objet sont des chaînes, tandis que les valeurs peuvent être de n'importe quel type de données, y compris un autre objet ou un tableau.

Les tableaux sont placés entre crochets ([]), et leurs valeurs sont séparées par une virgule (,). Chaque valeur dans un tableau peut être d'un type différent, y compris un autre tableau ou un objet.

Lorsque les objets et les tableaux contiennent d'autres objets ou des tableaux, la structure des données est une arborescence.

1.1.2 - L'extension JSON Editor Online pour Chrome

Cette extension pour Chrome permet de valider un fichier JSON et/ou de le formater.



Il existe des validateurs online :

http://jsonlint.com/#
http://www.freeformatter.com/json-validator.html

Exemples :

Un objet

```
{
  "prenom": "Joe",
  "nom": "Dalton",
  "age": 33,
  "adresse": "6, rue Roubo",
  "codePostal": "75011",
  "ville": "Paris",
  "pays": "FR",
  "telephones": [
    { "mobile": "07-07-07-07-07" },
    { "maison": "01-01-01-01-01" }
  ]
}
```

Un tableau d'objets

```
[
{"France": "33", "Capitale": "Paris"},
{"Italie": "39", "Capitale": "Rome"},
{"Espagne": "34", "Capitale": "Madrid"}
]
```

JSON est souvent utilisé comme un format commun pour sérialiser et dé-sérialiser les données des applications, écrites dans des langages différents et s'exécutant sur des plates-formes différentes, qui communiquent entre elles via Internet. JSON est adapté à ce scenario parce que c'est un standard ouvert, facile à lire, à écrire et compact.
Les Services Web RESTful (REpresentational State Transfer) utilisent JSON comme format de données pour les requêtes et les réponses.
L'en-tête HTTP est le suivant : Content-Type: application/json
Les représentations JSON sont généralement plus compactes que les représentations XML parce JSON ne possède pas de balise de fermeture. Mais contrairement à XML il n'existe pas de schéma pour définir et valider la structure des données JSON.

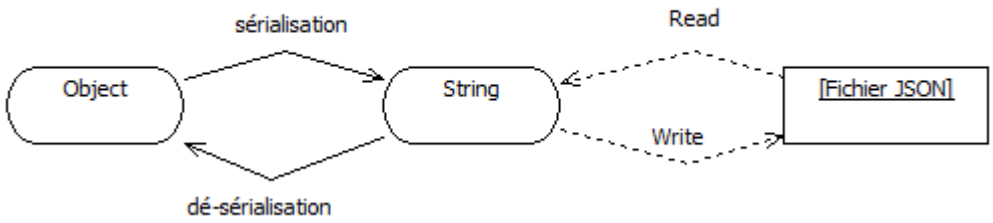
Pour la génération et l'analyse de données JSON, il existe deux modèles de programmation, qui sont similaires à ceux utilisés pour les documents XML :
Le modèle objet qui crée un arbre représentant les données JSON en mémoire. L'arbre peut ensuite être parcouru, analysé, ou modifié. Cette approche est la plus flexible mais lente et coûteuse en mémoire.
Le modèle de streaming utilise un analyseur basé sur les événements qui lit les données JSON élément par élément et s'arrête lorsqu'il trouve un objet ou un tableau ou une clé ou une valeur.

Le paquetage de javax.json (Oracle) fournit les classes et méthodes pour l'analyse globale.
Le paquet de javax.json.stream fournit les classes et méthodes pour l'analyse événementielle.

Il existe d'autres APIs : json-simple, org-json-java, gson (de Google), ...

Par la suite nous utiliserons l'API org.json.

Sérialisation et désérialisation



1.2 - UNE BIBLIOTHÈQUE À AJOUTER AU PROJET

Rien à ajouter. org.json est intégré à Android.

1.3 - SYNTAXES

Du pur Java !

Les imports :

```
import org.json.JSONObject;
import org.json.JSONArray;
import org.json.JSONException;
```

Traitement d'un tableau JSON

String 2 JSONArray

```
JSONArray tableauJSON = new JSONArray(String);
```

Tableau JSON vers tableau ordinal

```
String[] t = new String[tableauJSON.length()];
```

Dé-sérialisation

```
JSONObject objetJSON = (JSONObject) tableauJSON.get(i);
```

Traitement d'un objet JSON

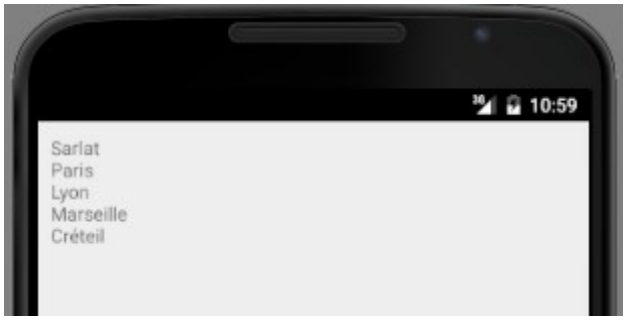
Récupération d'un objet JSON à partir d'une String

```
JSONObject objectJSON = new JSONObject(String);
```

Récupération de la valeur d'un champ

```
String valeur = objectJSON.get(asNomChamp).toString();
```

1.4 - LES FICHIERS UTILISÉS



villes.json

```
[
{"cp":"24200","nom_ville":"Sarlat"},
{"cp":"75000","nom_ville":"Paris"},
{"cp":"69000","nom_ville":"Lyon"},
{"cp":"13000","nom_ville":"Marseille"},
{"cp":"94000","nom_ville":"Créteil"}
]
```

ville.json

```
{"cp":"24200","nom_ville":"Sarlat"}
```

1.5 - Codes

1.5.1 - AnalyseurJSON.java

Cette classe contient deux méthodes getChampFromJSONArray() et getChampFromJSONObject() – une seule pourrait suffire avec un paramètre supplémentaire – qui renvoient un tableau de String à partir d'une String (le contenu du du fichier JSON) et d'un autre paramètre le champ demandé.

AnalyseurJSON
(+) getChampFromJSONArray() (+) getChampFromJSONObject()

```
import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;

import java.util.ArrayList;
import java.util.List;

public class AnalyseurJSON {

    /**
     * @param asNomChamp
     * @return
     */
    public static List<String> getChampFromJSONArray(String asContenu,
String asNomChamp) {
        List<String> liste = new ArrayList();

        try {
            // String 2 JSONArray
            JSONArray tableauJSON = new JSONArray(asContenu);
            // Dé-sérialisation
            JSONObject objetJSON;
            String[] t = new String[tableauJSON.length()];
            for (int i = 0; i < tableauJSON.length(); i++) {
                objetJSON = (JSONObject) tableauJSON.get(i);
                liste.add(objetJSON.get(asNomChamp).toString());
            }
        } catch (JSONException e) {
            liste.add(e.getMessage());
        } finally {
        }

        return liste;
    } /// getChampFromJSONArray


    /**
```

```
     * @param asContenu
     * @param asNomChamp
     * @return
     */
    public static List<String> getChampFromJSONObject(String asContenu,
String asNomChamp) {
        List<String> liste = new ArrayList();

        try {
            // String 2 JSONObject
            JSONObject objectJSON = new JSONObject(asContenu);
            liste.add(objectJSON.get(asNomChamp).toString());
        } catch (JSONException e) {
            liste.add(e.getMessage());
        } finally {
        }

        return liste;
    } /// getChampFromJSONObject

} /// classe
```

1.5.2 - JSONFromWeb.java

```
import android.app.Activity;
import android.os.AsyncTask;
import android.os.Bundle;
import android.util.Log;
import android.widget.TextView;

import java.io.*;
import java.net.HttpURLConnection;
import java.net.URL;
import java.util.List;

public class JSONFromWeb extends Activity {

    private TextView textViewJSON;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.json_from_web);

        textViewJSON = (TextView) findViewById(R.id.textViewJSON);

        String lsType;
        String lsURL;
        String lsResource;
        String lsChamp;

        // HOME
        lsURL = "http://192.168.122.1/PourSmartphones/";
        lsURL = "http://pascalbuguet.alwaysdata.net/PourSmartphones/";

        lsType = "object";
        lsResource = "ville.json";
        lsChamp = "nom_ville";

        lsType = "array";
        lsResource = "villes.json";
        lsChamp = "nom_ville";

        new TacheAsynchrone().execute(lsURL, lsResource, lsType, lsChamp);

    } /// onCreate

    /*
        * AsyncTask<Params, Progress, Result>
       */
    private class TacheAsynchrone extends AsyncTask<String, Integer,
List<String>> {
        @Override
        // ----------------------------------
        protected List<String> doInBackground(String... asParametres) {
            // String... parametre : nombre variable d'arguments
            // Se deplace dans un thread d'arriere-plan
            StringBuilder lsbResultat = new StringBuilder();
            List<String> liste = null;
```

```
            String lsURL = asParametres[0];
            String lsResource = asParametres[1];
            String lsType = asParametres[2];
            String lsChamp = asParametres[3];

            URL urlConnection = null;
            HttpURLConnection httpConnection = null;

            try {
                // Instanciation de HttpURLConnection avec l'objet url
                urlConnection = new URL(lsURL + lsResource);
                httpConnection = (HttpURLConnection)
urlConnection.openConnection();

                // Choix de la methode get ou post
                httpConnection.setRequestMethod("GET");

                // Autorise l'envoi de donnees
                // Sets the flag indicating whether this URLConnection
allows input.
                httpConnection.setDoInput(true);

                // Connexion
                httpConnection.connect();

                // EXECUTION DE LA REQUETE ET RESPONSE
                InputStream is = httpConnection.getInputStream();

//                // Comme l'on recoit un flux Text ASCII
                BufferedReader br = new BufferedReader(new
InputStreamReader(is));
                String lsLigne = "";
                while ((lsLigne = br.readLine()) != null) {
                    lsbResultat.append(lsLigne);
                    lsbResultat.append("\n");
                    Log.e("doInBackground", lsLigne);
                }
                //lsbResultat.deleteCharAt(lsbResultat.length()-1);
                br.close();
                is.close();

                if (lsType.equals("array")) {
                    /*
                    Un tableau JSON
                    */
                    liste =
AnalyseurJSON.getChampFromJSONArray(lsbResultat.toString(), lsChamp);
                } else if (lsType.equals("object")) {
                     /*
                     Un objet JSON
                     */
                    liste =
AnalyseurJSON.getChampFromJSONObject(lsbResultat.toString(), lsChamp);
                }
            } catch (IOException e) {
                Log.e("IOException", e.getMessage());
                lsbResultat.append(e.getMessage());
            } finally {
                // Deconnexion
                httpConnection.disconnect();
            }
```

```
            Log.e("doInBackground", lsbResultat.toString());

            // Renvoie la valeur a onPostExecute
            return liste;
        } /// doInBackground

        @Override
        // -------------------------
        protected void onPostExecute(List<String> liste) {
            // Synchronisation avec le thread de l'UI
            // Affiche le resultat final
            StringBuilder lsb = new StringBuilder();

            for (int i = 0; i < liste.size(); i++) {
                lsb.append(liste.get(i));
                lsb.append("\n");
            }
            //lsb.deleteCharAt(lsb.length() - 1);

            /*
            Dans ts les cas
             */
            textViewJSON.setText(lsb.toString());

        } /// onPostExecute

    } /// TacheAsynchrone

} /// classe
```

1.6 - Afficher du JSON produit par un script PHP