

Programmation mobile


Cours 5 : Réseau et `RecyclerView`


Julien Grange <julien.grange@lacl.fr>


Mardi 5 novembre 2024


- 1 Récupérer une discothèque en ligne, format JSON : Volley
- 2 L'afficher sous forme de longue liste : `RecyclerView`


Cours5

Mass in B Minor par Frans Bruggen
 Compositeur : Johann Sebastian Bach
Interprète : Frans Bruggen
Titre : Mass in B Minor
Année : 2009

The Art of Fugue par Gustav Leonhardt
 Compositeur : Johann Sebastian Bach
Interprète : Gustav Leonhardt
Titre : The Art of Fugue
Année : 1953

The Goldberg Variations par Jean Rondeau
 Compositeur : Johann Sebastian Bach
Interprète : Jean Rondeau
Titre : The Goldberg Variations
Année : 2017

Pièces de Clavecin par Scott Ross
 Compositeur : Johann Sebastian Bach
Interprète : Scott Ross
Titre : Pièces de Clavecin
Année : 1975

The Messiah par John Eliot Gardiner
 Compositeur : George Friedric Haendel
Interprète : John Eliot Gardiner
Titre : The Messiah
Année : 1982

The Art of Fugue par Emerson String Quartet

- Asynchrones (on ne veut pas bloquer)
- Sujet complexe, si on fait tout nous-mêmes : gérer les threads
- Sujet aisé, si on se repose sur les autres : Volley

Demander la permission dans le Manifest :

```
<uses-permission android:name="android.permission.INTERNET"/>
```

```
[
  {
    "composer": "Johann Sebastian Bach",
    "work": "Mass in B Minor",
    "interpret": "Frans Bruggen",
    "image": "bach.bmp",
    "year": 2009
  },
  {
    "composer": "Johann Sebastian Bach",
    "work": "The Art of Fugue",
    "interpret": "Gustav Leonhardt",
    "image": "bach.bmp",
    "year": 1953
  },
  {
    "composer": "Johann Sebastian Bach",
    "work": "The Goldberg Variations",
    "interpret": "Jean Rondeau",
    "image": "bach.bmp",
    "year": 2017
  },
  {
    "composer": "Johann Sebastian Bach",
    "work": "Pièces de Clavecin",
    "interpret": "Scott Ross",
    "image": "rameau.bmp",
    "year": 1975
  },
  {
    "composer": "George Friedric Haendel",
    "work": "The Messiah",
    "interpret": "John Eliot Gardiner",
    "image": "haendel.bmp",
    "year": 1982
  },
  {
    "composer": "Johann Sebastian Bach",
    "work": "The Art of Fugue",
    "interpret": "Emerson String Quartet",
    "image": "bach.bmp",
    "year": 2003
  },
  {
    "composer": "Johann Sebastian Bach",
    "work": "Mass in B Minor",
    "interpret": "John Eliot Gardiner",
    "image": "bach.bmp",
    "year": 1985
  },
  {
    "composer": "George Friedric Haendel",
    "work": "Dixit Dominus",
    "interpret": "John Eliot Gardiner",

```

- array - [, , ,]
- object - { , , , }
- int
- boolean
- string
- clef-valeur - "clef" : valeur

```
[
  {
    "composer": "Johann Sebastian Bach",
    "work": "Mass in B Minor",
    "interpret": "Frans Bruggen",
    "image": "bach.bmp",
    "year": 2009
  },
  {
    "composer": "Johann Sebastian Bach",
    "work": "The Art of Fugue",
    "interpret": "Gustav Leonhardt",
    "image": "bach.bmp",
    "year": 1953
  },
  {
    "composer": "Johann Sebastian Bach",
    "work": "The Goldberg Variations",
    "interpret": "Jean Rondeau",
    "image": "bach.bmp",
    "year": 2017
  },
  {
    "composer": "Johann Sebastian Bach",
    "work": "Pièces de Clavecin",
    "interpret": "Scott Ross",
    "image": "rameau.bmp",
    "year": 1975
  },
  {
    "composer": "George Friedric Haendel",
    "work": "The Messiah",
    "interpret": "John Eliot Gardiner",
    "image": "haendel.bmp",
    "year": 1982
  },
  {
    "composer": "Johann Sebastian Bach",
    "work": "The Art of Fugue",
    "interpret": "Emerson String Quartet",
    "image": "bach.bmp",
    "year": 2003
  },
  {
    "composer": "Johann Sebastian Bach",
    "work": "Mass in B Minor",
    "interpret": "John Eliot Gardiner",
    "image": "bach.bmp",
    "year": 1985
  },
  {
    "composer": "George Friedric Haendel",
    "work": "Dixit Dominus",
    "interpret": "John Eliot Gardiner",

```

- array - [, , ,]
- object - { , , , }
- int
- boolean
- string
- clef-valeur - "clef" : valeur

On a donc

- un array...
- ...contenant des objects...
- ...contenant chacun cinq clefs-valeurs
- ...dont quatre strings...
- ...et un int.

- Très utilisé sur le web pour représenter des données pas trop larges
- Pas du tout efficace du point de vue *théorie de l'information*...
- ...mais facilement compréhensible et facile à analyser.

- Très utilisé sur le web pour représenter des données pas trop larges
- Pas du tout efficace du point de vue *théorie de l'information*...
- ...mais facilement compréhensible et facile à analyser.
- Format arborescent, sous format clef-valeur.
- On peut représenter des types composés : object et array

Volley : RequestQueue et JSONArrayRequest

```
public class MainActivity extends AppCompatActivity {  
  
    private RecyclerView recyclerView;  
    private Discotheque discotheque;  
    private DiscothequeAdapter discothequeAdapter;  
    private LinearLayoutManager linearLayoutManager;  
    private RequestQueue queue;  
    public final String baseAddr = "https://lacl.fr/julien_grange/Enseignements/Programmation_mobile/23_24/Cours5/";  
    private final String discoAddr = baseAddr+"discotheque.json";  
  
    // le callback appelé dès que le JSONArray a été récupéré sur le web  
    Response.Listener<JSONArray> arrayListener = new Response.Listener<JSONArray>() {...};  
  
    // le callback appelé en cas d'échec de la connexion HTTP  
    Response.ErrorListener arrayErrorListener = new Response.ErrorListener() {...};  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
  
        recyclerView = findViewById(R.id.recyclerView);  
        discotheque = new Discotheque();  
        discothequeAdapter = new DiscothequeAdapter(discotheque);  
        linearLayoutManager = new LinearLayoutManager(getApplicationContext());  
        recyclerView.setLayoutManager(linearLayoutManager);  
        recyclerView.setAdapter(discothequeAdapter);  
  
        // on crée un queue par défaut, via laquelle on va pouvoir faire des requêtes web  
        queue = Volley.newRequestQueue(context, this);  
  
        // on crée notre requête, en en précisant l'adresse et les callbacks  
        JSONArrayRequest arrayRequest = new JSONArrayRequest(discoAddr, arrayListener, arrayErrorListener);  
  
        // on demande à ce que cette requête soit traitée dès que possible - le résultat est asynchrone !  
        queue.add(arrayRequest);  
    }  
}
```


Volley : Response.Listener<JSONArray>

```
// le callback appelé dès que le JSONArray a été récupéré sur le web
Response.Listener<JSONArray> arrayListener = new Response.Listener<JSONArray>() {
    @Override
    public void onResponse(JSONArray response) {

        // on traite tous les éléments du tableau
        for (int i = 0; i < response.length(); i++) {
            try {
                // chaque élément est parsé en un Disque...
                Disque disque = Disque.jsonToDisque(response.getJSONObject(i));
                // ...puis est ajouté à la discothèque
                discotheque.addDisque(disque);

                // pour récupérer l'image, il faut faire une autre requête réseau
                ImageRequest imageRequest = new ImageRequest(
                    // premier argument de l'ImageRequest : l'adresse
                    url + baseAddr + disque.getImagePath(),
                    // deuxième argument : le callback en cas de réponse positive
                    new Response.Listener<Bitmap>() {...},
                    // arguments 3,4,5,6 : paramètres de l'image
                    Disque.dimension,
                    Disque.dimension,
                    ImageView.ScaleType.CENTER,
                    Bitmap.Config.ARGB_8888,
                    // dernier argument : le callback d'erreur
                    new Response.ErrorListener() {...}
                );

                // ...et l'ajouter à la queue
                queue.add(imageRequest);
            } catch (JSONException e) {
                Toast.makeText(getApplicationContext(), text: "Error in parsing", Toast.LENGTH_SHORT).show();
                Log.e("tag", "parsing", msg: "In json parsing: " + e.getMessage());
            }

            // puis on prévient l'Adapter qu'il y a eu du changement dans les données
            discothequeAdapter.notifyDataSetChanged();
        }
    }
};
```

JSONObject::getString, JSONObject::getInt

```
vity_main.xml x MainActivity.java x DisqueViewHolder.java x Disque.java x DiscothequeAdapter.v x Discotheque.java x
public class Disque {
    private String compositeur, interprete, titre, imagePath;
    private int annee;
    public final static int dimension = 100;
    private Bitmap image;

    // les identifiants du JSON en question
    public static final String strCompositeur="composer", strInterprete="interpret",
        strTitre="work",strAnnee="year", strImage="image";

    public Disque(String compositeur,String interprete,String titre, int annee, String imagePath){
        this.compositeur = compositeur;
        this.interprete = interprete;
        this.annee = annee;
        this.titre = titre;
        this.imagePath = imagePath;
        // avant d'avoir récupéré l'image, on en crée une vide
        this.image = Bitmap.createBitmap(dimension,dimension,Bitmap.Config.ARGB_8888);
    }

    public String getInterprete() { return interprete; }
    public String getTitre() { return titre; }
    public String getImagePath() { return imagePath; }
    public Bitmap getImage() { return image; }
    public void setImage(Bitmap bitmap){image = bitmap;}

    // méthode parsant un JSONObject en Disque
    public static Disque jsonToDisque(JSONObject jsonObject) throws JSONException {
        return new Disque(jsonObject.getString(strCompositeur),
            jsonObject.getString(strInterprete),
            jsonObject.getString(strTitre),
            jsonObject.getInt(strAnnee),
            jsonObject.getString(strImage)
        );
    }

    @Override
    public String toString() {
        return "Compositeur : " + compositeur + "\n" +
            "Interprète : " + interprete + "\n" +
            "Titre : " + titre + "\n" +
            "Année : " + annee;
    }
}
```

```
1 package com.example.cours5;
2
3 import ...
4
5
6 public class Discotheque {
7
8     private ArrayList<Disque> disques;
9
10    public Discotheque() { disques = new ArrayList<>(); }
11
12    public void addDisque(Disque disque) { disques.add(disque); }
13
14
15    public int nbDisques() { return disques.size(); }
16
17    public Disque getDisque(int i) { return disques.get(i); }
18
19 }
20
21
22
23
24
25
26
27
```

Deux classes :

- `JSONObject`

```
int getInt(String clef)
String getString(String clef)
JSONObject getJSONObject(String clef)
JSONArray getJSONArray(String clef)
```

- `JSONArray`

```
int getInt(int index)
String getString(int index)
JSONObject getJSONObject(int index)
JSONArray getJSONArray(int index)
```

Ces méthodes lancent une `JSONException` si la clef n'existe pas.

Pourquoi les `RecyclerView` ?

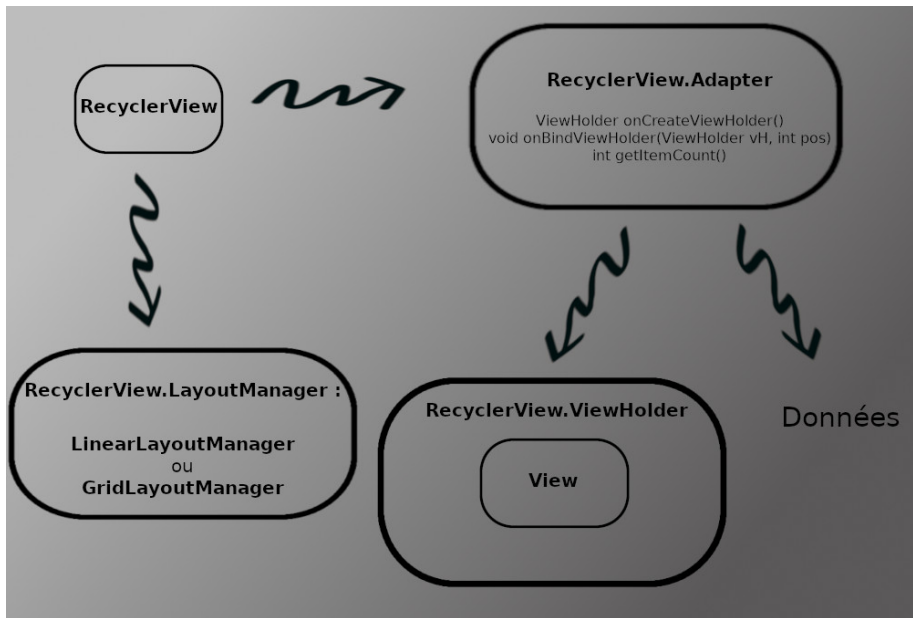
On veut afficher une longue liste (contacts, albums, tweets, etc.)

Il est préférable de créer les `View` associées à la volée

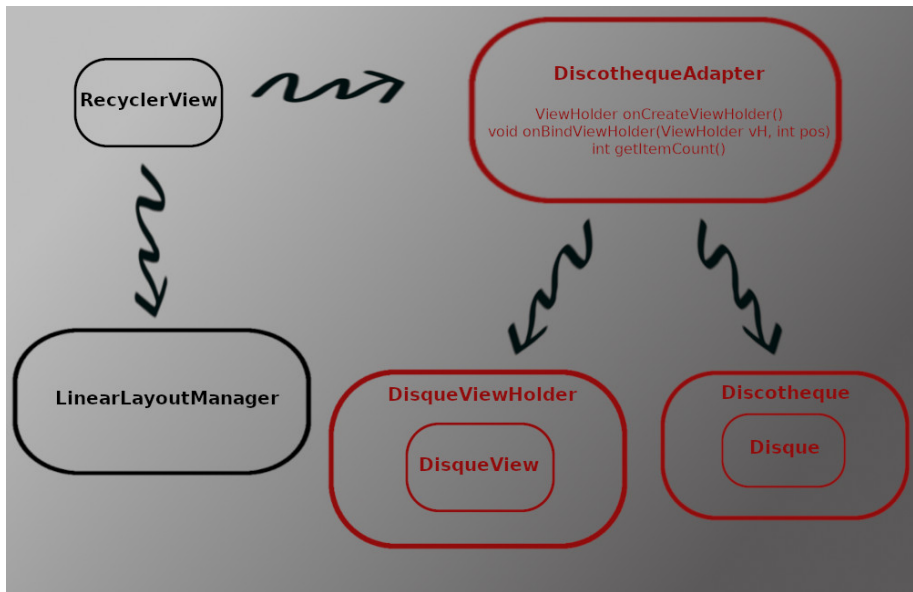
→ économie de ressources.

`RecyclerView` gère ça pour nous, au prix d'une petite lourdeur.

Taxonomie des classes utilisées



Taxonomie des classes utilisées



```
// DisqueView est notre View "maison", adaptée à nos besoins propres
public class DisqueView extends ConstraintLayout {

    private TextView header, body;
    private ImageView imageView;
    private final static int margin = 30;

    public DisqueView(Context context) {
        super(context);
        // on crée les views, et on les ajoute au ConstraintLayout
        header = new TextView(context); body = new TextView(context); imageView = new ImageView(context);
        this.addView(header); this.addView(body); this.addView(imageView);

        // on génère des ID pour tout le monde
        header.setId(View.generateViewId());
        body.setId(View.generateViewId());
        imageView.setId(View.generateViewId());
        this.setId(View.generateViewId());

        // on s'occupe des contraintes
        ConstraintSet constraintSet = new ConstraintSet();
        constraintSet.clone(constraintLayout, this);

        constraintSet.connect(header.getId(), ConstraintSet.LEFT, this.getId(), ConstraintSet.LEFT, margin);
        constraintSet.connect(header.getId(), ConstraintSet.TOP, this.getId(), ConstraintSet.TOP, margin);

        constraintSet.connect(imageView.getId(), ConstraintSet.LEFT, this.getId(), ConstraintSet.LEFT, margin);
        constraintSet.connect(imageView.getId(), ConstraintSet.TOP, header.getId(), ConstraintSet.BOTTOM, margin);

        constraintSet.connect(body.getId(), ConstraintSet.LEFT, imageView.getId(), ConstraintSet.RIGHT, margin);
        constraintSet.connect(body.getId(), ConstraintSet.TOP, imageView.getId(), ConstraintSet.TOP, margin);
        constraintSet.connect(body.getId(), ConstraintSet.BOTTOM, imageView.getId(), ConstraintSet.BOTTOM, margin);

        constraintSet.applyTo(constraintLayout, this);

        body.setPadding(margin, margin, margin, margin);
        header.setPadding(margin, margin, margin, margin);

        TextViewCompat.setTextAppearance(header, android.R.style.TextAppearance_Holo);
    }
}
```

- header

- image
- body

Mass in B Minor par Frans Bruggen

Compositeur : Johann Sebastian Bach

Interprète : Frans Bruggen

Titre : Mass in B Minor

Année : 2009



```
// la méthode qui remplit un DisqueView avec les valeurs d'un Disque
public void setDisque(Disque disque){
    header.setText(disque.getTitre() + " par " + disque.getInterprete());
    body.setText(disque.toString());
    imageView.setImageBitmap(disque.getImage());
}
```

DisqueViewHolder et DiscothequeAdapter

```
ity_main.xml x MainActivity.java x DisqueViewHolder.java x Disque.java x DiscothequeAdapter x DiscothequeAdapter.java x
package com.example.cours5;
import ...
public class DisqueViewHolder extends RecyclerView.ViewHolder {
    DisqueView disqueView;
    public DisqueViewHolder(View itemView) {
        super(itemView);
        disqueView = (DisqueView) itemView;
    }
    public void setDisque(Disque disque) { disqueView.setDisque(disque); }
}

package com.example.cours5;
import ...
public class DiscothequeAdapter extends RecyclerView.Adapter<DisqueViewHolder> {
    private Discotheque discotheque;
    public DiscothequeAdapter(Discotheque discotheque) { this.discotheque = discotheque; }
    @NonNull
    @Override
    public DisqueViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
        DisqueView disqueView = new DisqueView(parent.getContext());
        return new DisqueViewHolder(disqueView);
    }
    @Override
    public void onBindViewHolder(@NonNull DisqueViewHolder holder, int position) {
        Disque disque = discotheque.getDisque(position);
        holder.setDisque(disque);
    }
    @Override
    public int getItemCount() { return discotheque.nbDisques(); }
}
```


Récupérer les images : une étape en deux temps

- Un JSON ne peut pas contenir d'image.
- La valeur du champ "image" indique le nom du fichier image stocké sur le serveur web.

Récupérer les images : une étape en deux temps

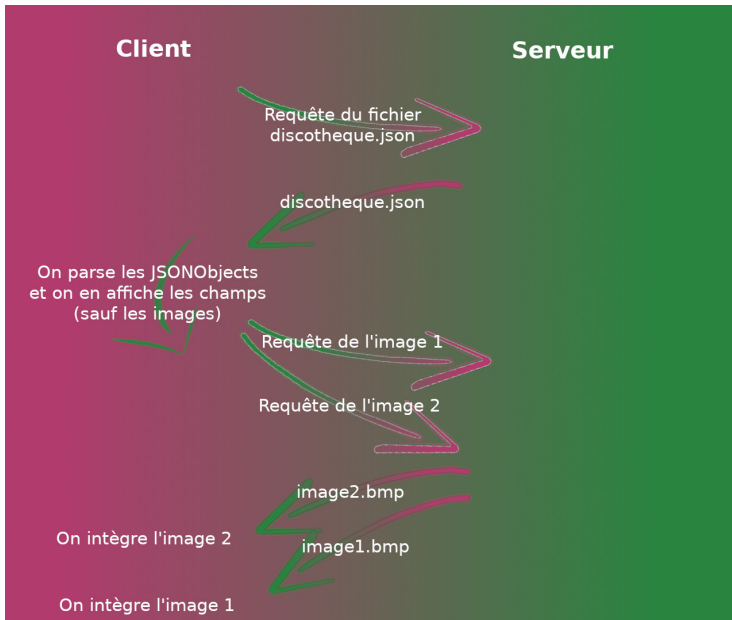
- Un JSON ne peut pas contenir d'image.
- La valeur du champ "image" indique le nom du fichier image stocké sur le serveur web.
- Il s'agit donc d'une manœuvre en deux requêtes :
 - ① on demande au serveur le JSONArray, qu'on parse
 - ② on demande au serveur chaque image mentionnée

Récupérer les images : une étape en deux temps

- Un JSON ne peut pas contenir d'image.
- La valeur du champ "image" indique le nom du fichier image stocké sur le serveur web.
- Il s'agit donc d'une manœuvre en deux requêtes :
 - ① on demande au serveur le JSONArray, qu'on parse
 - ② on demande au serveur chaque image mentionnée

NB : Volley gère un cache de manière transparente. Si on demande une image qui a déjà été téléchargée, alors Volley nous en fournira la copie locale, sans redoubler la requête.

Récupérer les images : une étape en deux temps



Première étape : récupérer et parser discotheque.json

Pour prévenir que les données ont changé :

`RecyclerView.Adapter::notifyDataSetChanged()`.

```
main.xml | MainActivity.java | DisqueViewHolder.java | Disque.java | DiscothequeAdapter.java | MainActivity.java
private RecyclerView.Adapter<DisqueViewHolder>
private Discotheque discotheque;
private DiscothequeAdapter discothequeAdapter;
private LinearLayoutManager layoutManager;
private RequestQueue queue;
public final String baseAddr = "https://laci.fr/jeuLien-grange/Enseignements/Programmation_mobile/23-24/Cours9/";
private final String discaAddr = baseAddr + "discotheque.json";

// Le callback appelé dès que le JSONArray a été récupéré sur le web
Response.Listener<JSONArray> arrayListener = new Response.Listener<JSONArray>() {

// Le callback appelé en cas d'échec de la connexion HTTP
Response.ErrorListener arrayErrorListener = new Response.ErrorListener() {

@Override
public void onErrorResponse(VolleyError error) {
    Toast.makeText(getApplicationContext(), text, "Error in getting the array", Toast.LENGTH_SHORT).show();
    Log.e("tag", "connection", msg, "In Volley: " + error.getMessage());
};

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

    recyclerView = findViewById(R.id.recyclerView);
    discotheque = new Discotheque();
    discothequeAdapter = new DiscothequeAdapter(discotheque);
    layoutManager = new LinearLayoutManager(getApplicationContext());
    recyclerView.setLayoutManager(layoutManager);
    recyclerView.setAdapter(discothequeAdapter);

// on crée un queue par défaut, via laquelle on va pouvoir faire des requêtes web
queue = Volley.newRequestQueue(this);

// on crée notre requête, en précisant l'adresse et les callbacks
JSONArrayRequest arrayRequest = new JSONArrayRequest(discaAddr, arrayListener, arrayErrorListener);

// on demande à ce que cette requête soit traitée dès que possible - le résultat est asynchrone /
queue.add(arrayRequest);
};

// Le callback appelé dès que le JSONArray a été récupéré sur le web
Response.Listener<JSONArray> arrayListener = new Response.Listener<JSONArray>() {
@Override
public void onResponse(JSONArray response) {
// on traite tous les éléments du tableau
for (int i = 0; i < response.length(); i++) {
try {
// chaque élément est parsé en un Disque...
Disque disque = Disque.fromJson(response.getJSONObject(i));
// ...puis est ajouté à la discotheque
discotheque.addDisque(disque);

// pour récupérer l'image, il faut faire une autre requête réseau
ImageRequest imageRequest = new ImageRequest(
// premier argument de l'ImageRequest : l'adresse
baseAddr + disque.getImagePath(),
// deuxième argument : le callback en cas de réponse positive
new Response.Listener<Bitmap>() { ... },
// arguments 3,4,5,6 : paramètres de l'image
Disque.dimension,
Disque.dimension,
ImageView.ScaleType.CENTER,
Bitmap.Config.ARGB_8888,
// dernier argument : le callback d'erreur
new Response.ErrorListener() { ... }
);

// ...et l'ajouter à la queue
queue.add(imageRequest);
} catch (JSONException e) {
    Toast.makeText(getApplicationContext(), text, "Error in parsing", Toast.LENGTH_SHORT).show();
    Log.e("tag", "parsing", msg, "In json parsing: " + e.getMessage());
}
};

// puis on prévient l'adapter qu'il y a eu du changement dans les données
discothequeAdapter.notifyDataSetChanged();
};
};
```

Deuxième étape : récupérer chacune des images

```
// on traite tous les éléments du tableau
for (int i = 0; i < response.Length(); i++) {
    try {
        // chaque élément est parsé en un Disque...
        Disque disque = Disque.jsonToDisque(response.getJSONObject(i));
        // ...puis est ajouté à la discothèque
        discotheque.addDisque(disque);

        // pour récupérer l'image, il faut faire une autre requête réseau
        ImageRequest imageRequest = new ImageRequest(
            // premier argument de l'ImageRequest : l'adresse
            url: baseAddr+disque.getImagePath(),
            // deuxième argument : le callback en cas de réponse positive
            new Response.Listener<Bitmap>() {
                @Override
                public void onResponse(Bitmap response) {
                    // on change l'image, et on prévient d'Adapteur qu'il y a du nouveau
                    disque.setImage(response);
                    discothequeAdapter.notifyDataSetChanged();
                }
            },
            // arguments 3,4,5,6 : paramètres de l'image
            Disque.dimension,
            Disque.dimension,
            ImageView.ScaleType.CENTER,
            Bitmap.Config.ARGB_8888,
            // dernier argument : le callback d'erreur
            new Response.ErrorListener() {
                @Override
                public void onErrorResponse(VolleyError error) {
                    Log.v( tag: "cours5", msg: "In getting image: " + error.toString());
                }
            }
        );
    }
};
```

Cours5

Mass in B Minor par Frans Bruggen



Compositeur : Johann Sebastian Bach
Interprète : Frans Bruggen
Titre : Mass in B Minor
Année : 2009

The Art of Fugue par Gustav Leonhardt



Compositeur : Johann Sebastian Bach
Interprète : Gustav Leonhardt
Titre : The Art of Fugue
Année : 1953

The Goldberg Variations par Jean Rondeau



Compositeur : Johann Sebastian Bach
Interprète : Jean Rondeau
Titre : The Goldberg Variations
Année : 2017

Pièces de Clavecin par Scott Ross



Compositeur : Johann Sebastian Bach
Interprète : Scott Ross
Titre : Pièces de Clavecin
Année : 1975

The Messiah par John Eliot Gardiner



Compositeur : George Frideric Haendel
Interprète : John Eliot Gardiner
Titre : The Messiah
Année : 1982

The Art of Fugue par Emerson String Quartet