第十九章

ANDROID網路程式

課程簡述

- 如何使用App呼叫Server API
- 如何處理JSON
- 如何建立自己的API Server
- 如何使用MongoDB
- 如何讓自己的API Server連接MongoDB並回傳結果
- 如何讓App呼叫自己的API Server

概念介紹

與SERVER溝通流程

API

API

- Application Programming Interface
- 程式撰寫時不同軟體模組間的接口
- Web API
  - http://<主機位置>/<分類>/<分類>/...
    e.g.
    https://api.twitter.com/1.1/statuses/user_timeline.json
- 最常使用GET或POST呼叫

API

- GET將參數帶在HTTP的Request網址中
  - 網址 http://xxx.toright.com/api/?id=010101
  - 封包

```
GET /?id=010101 HTTP/1.1
Host: xxx.toright.com
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; zh-TW;
rv:1.9.2.13) Gecko/20101203 Firefox/3.6.13 GTB7.1 ( .NET CLR 3.5.30729)
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: zh-tw,en-us;q=0.7,en;q=0.3
Accept-Encoding: gzip,deflate
Accept-Charset: UTF-8,*
Keep-Alive: 115
Connection: keep-alive
```

API

- POST將參數帶在HTTP Request的封包中
  - 網址 http://xxx.toright.com/api/insert
  - 封包

```
POST / HTTP/1.1
Host: xxx.toright.com
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; zh-TW;
rv:1.9.2.13) Gecko/20101203 Firefox/3.6.13 GTB7.1 ( .NET CLR
3.5.30729)
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: zh-tw,en-us;q=0.7,en;q=0.3
Accept-Encoding: gzip,deflate
Accept-Charset: UTF-8,*
Keep-Alive: 115
Connection: keep-alive

Content-Type: application/x-www-form-urlencoded
</code><code>Content-Length: 9
id=010101
```

SERVER及資料庫

SERVER及資料庫

- 可以用PHP, ASP, ASP.NET, VB.NET, JSP來製作
- 資料庫常使用MySQL, SQL Server, Postgres SQL, Oracle等等

- 本次教學，用較新的**Node.js**為Server與**MongoDB**為資料庫

與SERVER溝通流程

JSON

- JavaScript Object Notation
- 輕量化資料交換格式
- Example: https://graph.facebook.com/prettyklicks/posts?access_token=123975213079|nqKWO89vVW4QH_bNmKH-Wiy3W0w&limit=1

JSON OBJECT

- 以 { 開始，以 } 結束
- 內容 Key(索引值)與Value(內容)
- 內容可以是
  - 整數或浮點數、字串(要用”” 框起)、布林(true, false)、JSON Array、JSON Object
- 中間以冒號:分隔 { Key : Value }
- e.g.

```
{”name” : ”android”}
```

JSON ARRAY

- 以 [開頭，以] 結果
- 中間內容有序列表，每個陣列物件以 , 分隔
- e.g.

```
{
        “values”:[
                {“value”, 1.0}, {“value”, 2.0}, {“value”, 3.0}
        ]
}
```

JSON範例

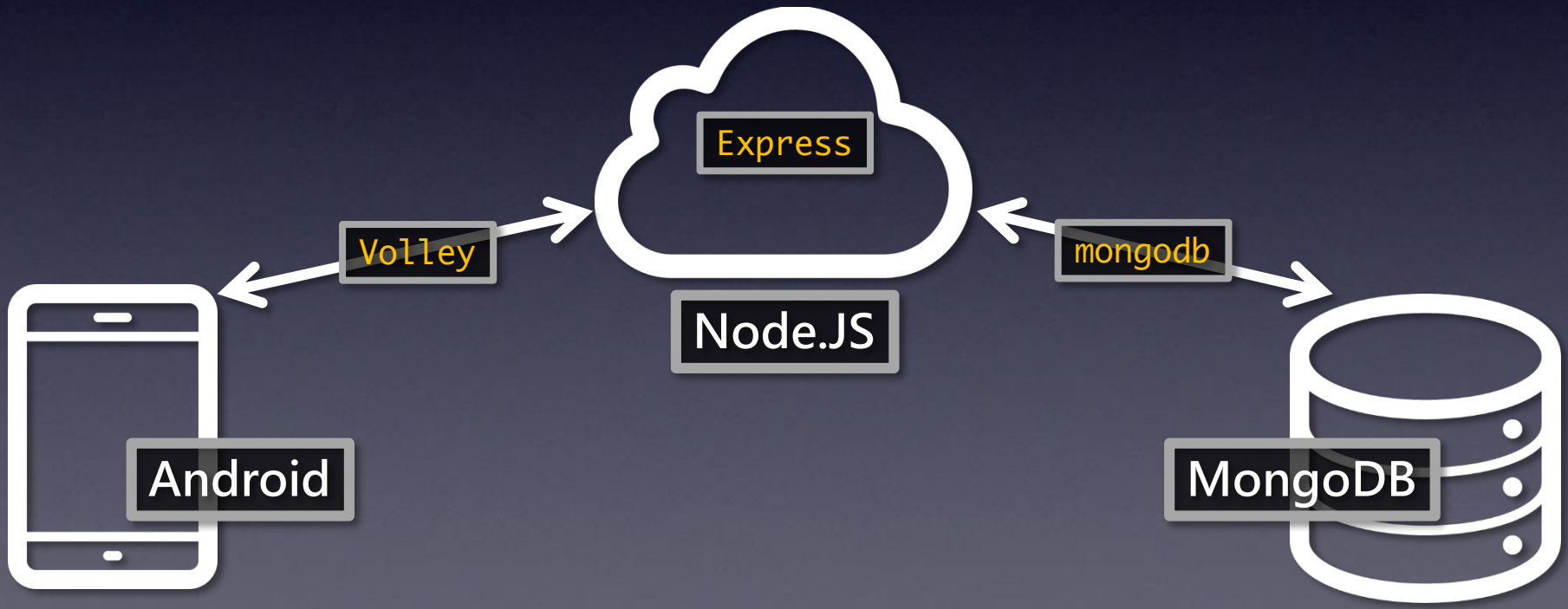
```
{
        "id": "187393124625179",
        "name": "Web Design"
}
```

```
{
        "id": "187393124625179",
       "name": {"first" : "Justin", "last" : "Liu"}
}
```

```
{
        "id": "187393124625179",
        "names": [
                {"nickname", "John"},
                {"nickname", "David"}
        ]
}
```

總結

- 即將使用的技術
  - **白字**表示將會用到的平台
  - **橘字**表示將會用到的函式庫或模組



ANDROID呼叫WEB API

APP呼叫WEB API

範例WEB API

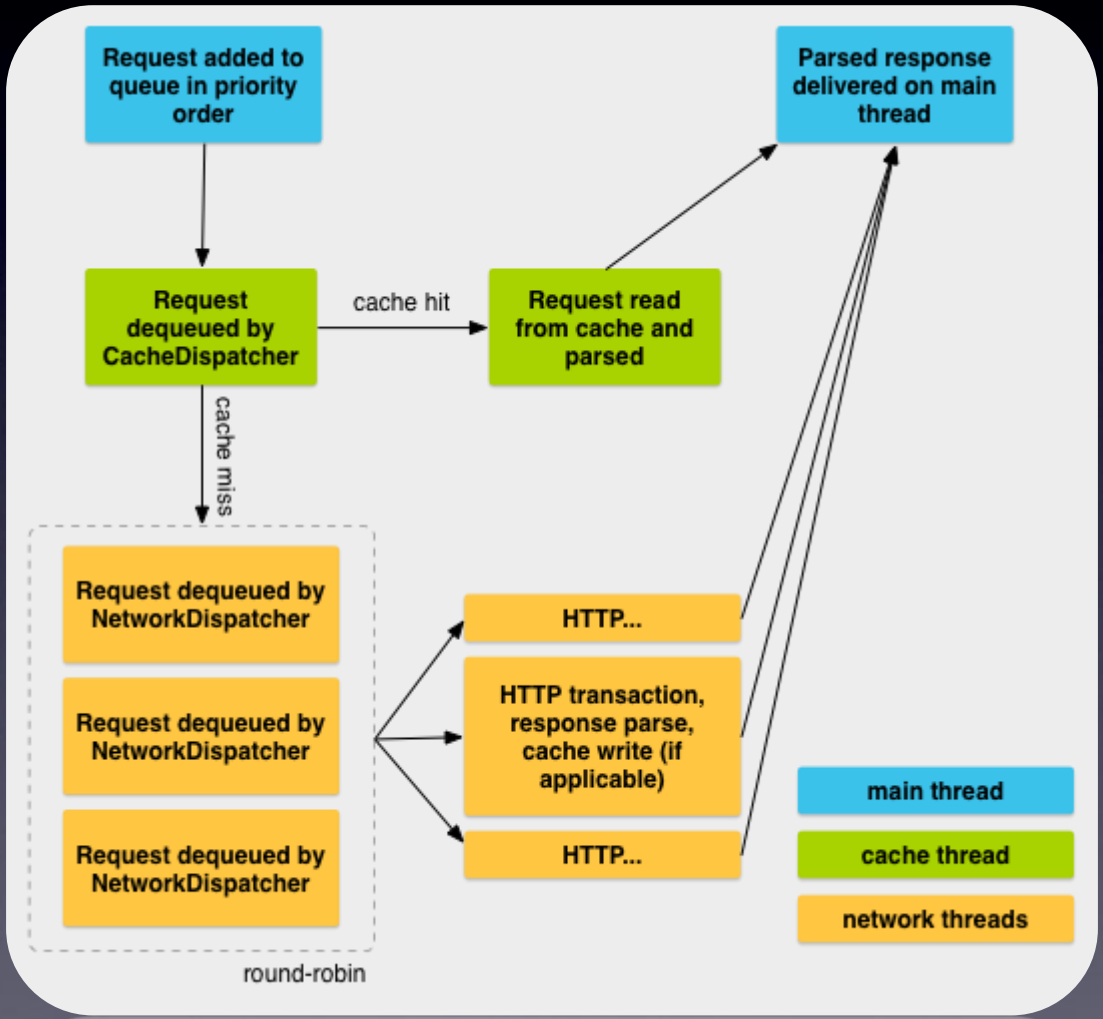
- 牛刀小試
- GET http://jsonplaceholder.typicode.com/posts/1
  - 用瀏覽器打開試看看

ANDROID網路連線

對應專案
NetworkCommunication_step1

- Android網路程式的困難
  - 不能在主執行緒進行網路連線
  - 多執行緒溝通的困難
  - 很多人不能了解AsyncTask來完成
  - 錯誤情況很難處理
- 使用第三方函式庫(Library)，VOLLEY

VOLLEY流程圖


Request added to queue in priority order
Parsed response delivered on main thread
Request dequeued by CacheDispatcher
cache hit
Request read from cache and parsed
cache miss
Request dequeued by NetworkDispatcher
Request dequeued by NetworkDispatcher
Request dequeued by NetworkDispatcher
HTTP...
HTTP transaction, response parse, cache write (if applicable)
HTTP...
main thread
cache thread
network threads
round-robin

VOLLEY的使用步驟

1. 設定使用網路的權限
2. 放入volley的函式庫
3. 建立Request
4. 使用NetworkManager
5. 接收回應與處理錯誤
6. 停止volley

1.設定使用網路的權限

- 在AndroidManifest.xml中要設定Internet的permission

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
   package="com.example.networkcommunication"
   android:versionCode="1"
   android:versionName="1.0" >
         <uses-permission android:name="android.permission.INTERNET"/>
         ……
</manifest>
```

2.放入VOLLEY的函式庫

- 將volley.jar放置到以下資料夾 <專案>/lib
- volley是開放原始碼專案，可由以下網址取得
  - https://android.googlesource.com/platform/frameworks/volley
- 更多volley的資訊
  - http://developer.android.com/training/volley/index.html

3.建立REQUEST

- 在需要使用的地方，加上Request

```
public class MainActivity extends Activity {

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

        StringRequest request = new StringRequest(Request.Method.GET,
"http://jsonplaceholder.typicode.com/posts/1", mResponseListener,
mErrorListener);
        NetworkManager.getInstance(this).request(null, request);
    }
```

3.建立REQUEST

```
StringRequest request = new StringRequest(
Request.Method.GET,
"http://jsonplaceholder.typicode.com/posts/1",
mResponseListener,
mErrorListener);
```

- 參數1：GET或是POST(或是其他HTTP的方法)
- 參數2：API網址
- 參數3：Server回應後的Listener
- 參數4：錯誤出現時的Listener

4.使用NETWORKMANAGER

- NetworkManager可以將Request藉由Volley發送給API Server

```
public class MainActivity extends Activity {

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

        StringRequest request = new StringRequest(Request.Method.GET,
"http://jsonplaceholder.typicode.com/posts/1", mResponseListener,
mErrorListener);
        NetworkManager.getInstance(this).request(null, request);
    }
```

5.接收回應與處理錯誤

- 若連線沒有出錯且得到Server正常的回應，就會從Request的第三個參數得到結果
  - 由onResponse()收到Server回應的字串

```
private Listener<String> mResponseListener =
new Listener<String>() {

    @Override
    public void onResponse(String string) {
    }
};
```

5.接收回應與處理錯誤

- 若連線過程出錯，或Server回應錯誤，就會從Request的第四個參數回應
  - onErrorResponse()會得到錯誤的通知

```
private ErrorListener mErrorListener = new ErrorListener() {

    @Override
    public void onErrorResponse(VolleyError error) {
    }
};
```

6.停止VOLLEY

- 最後在第一個使用NetworkManager的Activity，其中的onDestroy()呼叫stop()以便釋放資源

```
@Override
protected void onDestroy() {
        super.onDestroy();
        NetworkManager.getInstance(this).stop();
}
```

剖析JSON與JSON ARRAY

剖析JSON

對應專案
NetworkCommunication_step2

- 由外至內剖析，先取得JSONObject
再藉由索引值取得對應的內容

JSONObject json

```
{
  "userId": 1,
  "id": 1,
  "title": "sunt ….",
  "body": "quia et ….."
}
```

剖析JSON

- 使用getInt(索引值)取得數字的值

```
{
  "userId": 1,
  "id": 1,
  "title": "sunt ….",
  "body": "quia et ….."
}
```

```
json.getInt(“userId”)
```

剖析JSON

- 使用getInt(索引值)取得數字的值

```
{
  "userId": 1,
  "id": 1,
  "title": "sunt ….",
  "body": "quia et ….."
}
```

```
json.getInt(“id”)
```

剖析JSON

- 使用getString(索引值)取得字串的值

```
{
  "userId": 1,
  "id": 1,
  "title": "sunt ….",
  "body": "quia et ….."
}
```

```
json.getString(“title”)
```

剖析JSON

```
public void onResponse(String string) {
   try {
      JSONObject json = new JSONObject(string);
      String title = json.getString("title");
      String body = json.getString("body");

      TextView text1 = (TextView) findViewById(R.id.textView1);
      text1.setText(title);
      TextView text2 = (TextView) findViewById(R.id.textView2);
      text2.setText(body);
   } catch (JSONException e) {
       e.printStackTrace();
   }
}
```

剖析JSON ARRAY

對應專案
NetworkCommunication_step3

- 試著將網址改為
http://jsonplaceholder.typicode.com/todos

```
[
    {
      "userId": 1,
      "id": 1,
      "title": "delectus aut autem",
      "completed": false
    },
    {
      "userId": 1,
      "id": 2,
      "title": "quis ut nam facilis et
officia qui",
      "completed": false
    },
    ...
]
```

JSONArray ary

剖析JSON ARRAY

```
ary.getJSONObject(0)
```

```
[
    {
      "userId": 1,
      "id": 1,
      "title": "delectus aut autem",
      "completed": false
    },
    {
      "userId": 1,
      "id": 2,
      "title": "quis ut nam facilis et
officia qui",
      "completed": false
    },
    ...
]
```

剖析JSON ARRAY

```
[
    {
      "userId": 1,
      "id": 1,
      "title": "delectus aut autem",
      "completed": false
    },
    {
      "userId": 1,
      "id": 2,
      "title": "quis ut nam facilis et
officia qui",
      "completed": false
    },
    ...
]
```

```
ary.getJSONObject(1)
```

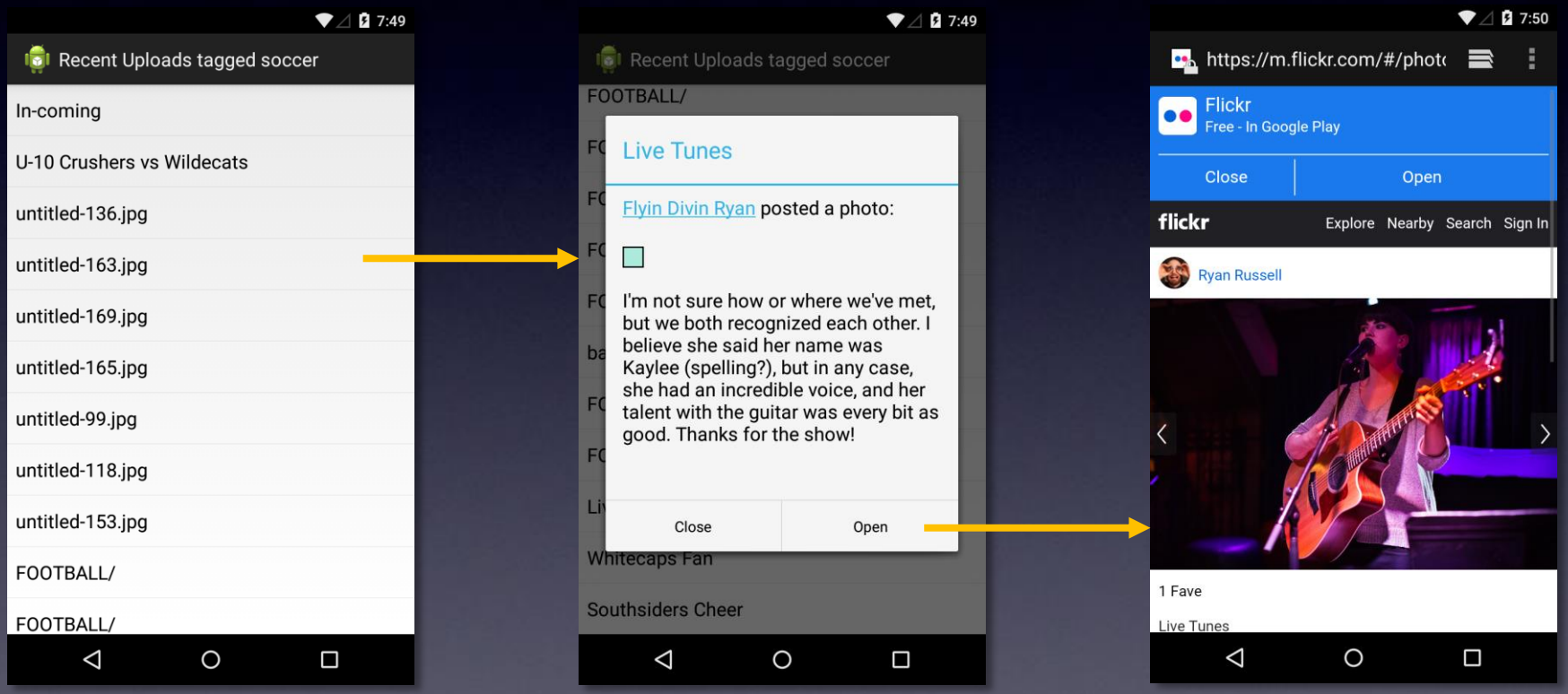
剖析JSON ARRAY

```
public void onResponse(String string) {
    try {
        JSONArray ary = new JSONArray(string);
        StringBuilder userIds = new StringBuilder();
        StringBuilder titles = new StringBuilder();
        for (int i = 0; i < ary.length(); i++) {
            JSONObject json = ary.getJSONObject(i);
            int userId = json.getInt("userId");
            userIds.append(userId);
            userIds.append(",");
            String title = json.getString("title");
            titles.append(title);
            titles.append(",");
        }
        TextView text1 = (TextView) findViewById(R.id.textView1);
        text1.setText(userIds.toString());
        TextView text2 = (TextView) findViewById(R.id.textView2);
        text2.setText(titles.toString());
    } catch (JSONException e) {
         e.printStackTrace();
    }
}
```

作業1

- 使用GET方式取得Flickr的圖片資訊
  - URL: http://www.flickr.com/services/feeds/photos_public.gne?tags=soccer&format=json&jsoncallback=?
- 將抓回圖片資訊，標題呈現在ListView上
- 點下ListView，會顯示AlertDialog
  - AlertDialog標題呈現圖片標題
  - AlertDialog的Message呈現關於圖片說明
  - AlertDialog的按鈕按下後要用瀏覽器開啟圖片資訊的網頁
  - AlertDialog的另一個按鈕按下後關閉AlertDialog

作業1

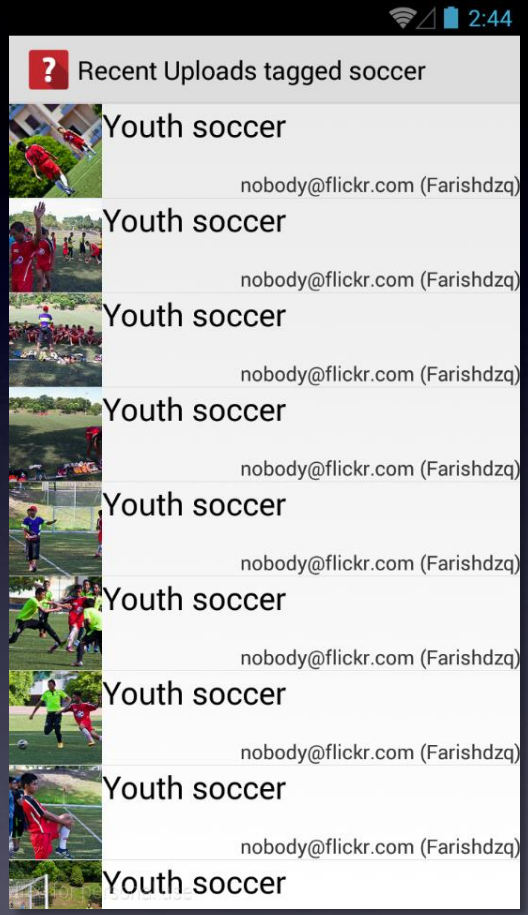

7:49
Recent Uploads tagged soccer
In-coming
U-10 Crushers vs Wildcats
untitled-136.jpg
untitled-163.jpg
untitled-169.jpg
untitled-165.jpg
untitled-99.jpg
untitled-118.jpg
untitled-153.jpg
FOOTBALL/
FOOTBALL/
7:49
Recent Uploads tagged soccer
FOOTBALL/
Live Tunes
Flyin Divin Ryan posted a photo:
I'm not sure how or where we've met, but we both recognized each other. I believe she said her name was Kaylee (spelling?), but in any case, she had an incredible voice, and her talent with the guitar was every bit as good. Thanks for the show!
Close
Open
Whitecaps Fan
Southsiders Cheer
7:50
https://m.flickr.com/#/photo
Flickr
Free - In Google Play
Close
Open
flickr
Explore
Nearby
Search
Sign In
Ryan Russell
1 Fave
Live Tunes

客製化LISTVIEW

客製化LISTVIEW

對應專案
NetworkCommunication_step4

- 大部分的App都需要在ListView上呈現各樣的資訊
- Android的ListView，簡單的用法只能夠呈現文字
- 若依照作業1，希望最後產生的結果如右，該怎麼做？



客製化LISTVIEW

- ListView都是一列一列呈現的
- 在寫ListView時都會看到以下的程式碼

```
ListView listView = (ListView) findViewById(…);
ArrayAdapter<Photo> adapter = new ArrayAdapter…;
listView.setAdapter(adapter);
```

- ListView的每一列該怎麼呈現，就是由Adapter負責
- 客製化ListView就表示要自己寫Adapter
- 以下共有五個步驟幫助你寫出自己的Adapter

客製化LISTVIEW步驟

- 第一步、以物件方式準備資料
- 第二步、建立客製化的Layout
- 第三步、建立BaseAdapter的子類別
- 第四步、填入必備資訊
- 第五步、在Adapter組出每一列

客製化ListView

第一步、以物件方式準備資料

1.以物件方式準備資料

- 依照作業指示，會連到API取得JSON，將JSON內容轉為物件

```
{
        "title": "Recent Uploads tagged soccer",
        ....
        "items": [
            {
                "title": ".",
                "link": "...",
                "media": {"m":"..."},
                "description": " ...",
            },
            {
                "title": ".",
                "link": "...",
                "media": {"m":"..."},
                "description": " ...",
            },
        ]
}
```

針對JSONObject轉為自訂**Object**

```
public class Photo {
    public String title;
    public String link;
    public String media;
    public String desc;
}
```

1.以物件方式準備資料

- 對JSONArray內的每一個JSONObject，都進行轉換，請不要用陣列去儲存這些值

```
{
        "title": "Recent Uploads tagged soccer",
        ....
        "items": [
            {
                "title": ".",
                "link": "...",
                "media": {"m":"..."},
                "description": " ...",
            },
            {
                "title": ".",
                "link": "...",
                "media": {"m":"..."},
                "description": " ...",
            },
        ]
}
```

```
public class Photo {
    public String title;
    public String link;
    public String media;
    public String desc;
}
```

針對JSONObject轉為自訂Object

1.以物件方式準備資料

- 最後每一個JSONObject都存在ArrayList中，下頁來看看實際程式碼怎麼做

```
{
        "title": "Recent Uploads tagged soccer",
        ....
        "items": [
           {
                 "title": ".",
                 "link": "...",
                 "media": {"m":"..."},
                 "description": " ...",
           },
           {
                 "title": ".",
                 "link": "...",
                 "media": {"m":"..."},
                 "description": " ...",
           },
        ]
}
```

存成ArrayList
e.g. ArrayList<Photo> list;

1.以物件方式準備資料

- 建立物件Photo

```
public class Photo {
        public String title;
        public String link;
        public String media;
        public String desc;
        public String author;
}
```

1.以物件方式準備資料

- 處理API回傳字串

response為API回傳的字串

```
JSONObject json = new JSONObject(response);
ArrayList<Photo> datas = new ArrayList<Photo>();
JSONArray array = json.getJSONArray("items");
for (int i = 0; i < array.length(); i++) {
    JSONObject jsonPhoto = array.getJSONObject(i);
    Photo photo = new Photo();
    photo.title = jsonPhoto.getString("title");
    photo.media = jsonPhoto.getString(“media");
    photo.desc = jsonPhoto.getString("description");
    photo.author = jsonPhoto.getString("author");
    datas.add(photo);
}
```

1.以物件方式準備資料

預備好ArrayList
接下來用來儲存物件

```
JSONObject json = new JSONObject(response);
ArrayList<Photo> datas = new ArrayList<Photo>();
JSONArray array = json.getJSONArray("items");
for (int i = 0; i < array.length(); i++) {
    JSONObject jsonPhoto = array.getJSONObject(i);
    Photo photo = new Photo();
    photo.title = jsonPhoto.getString("title");
    photo.media = jsonPhoto.getString("media");
    photo.desc = jsonPhoto.getString("description");
    photo.author = jsonPhoto.getString("author");
    datas.add(photo);
}
```

1.以物件方式準備資料

從API回傳的內容中
取JSONArray

```
JSONObject json = new JSONObject(response);
ArrayList<Photo> datas = new ArrayList<Photo>();
JSONArray array = json.getJSONArray("items");
for (int i = 0; i < array.length(); i++) {
    JSONObject jsonPhoto = array.getJSONObject(i);
    Photo photo = new Photo();
    photo.title = jsonPhoto.getString("title");
    photo.media = jsonPhoto.getString("media");
    photo.desc = jsonPhoto.getString("description");
    photo.author = jsonPhoto.getString("author");
    datas.add(photo);
}
```

1.以物件方式準備資料

```
JSONObject json = new JSONObject(response);
ArrayList<Photo> datas = new ArrayList<Photo>();
JSONArray array = json.getJSONArray("items");
for (int i = 0; i < array.length(); i++) {
    JSONObject jsonPhoto = array.getJSONObject(i);
    Photo photo = new Photo();
    photo.title = jsonPhoto.getString("title");
    photo.media = jsonPhoto.getString("media");
    photo.desc = jsonPhoto.getString("description");
    photo.author = jsonPhoto.getString("author");
    datas.add(photo);
}
```

開始以迴圈的方式
處理JSONArray

1.以物件方式準備資料

```
JSONObject json = new JSONObject(response);
ArrayList<Photo> datas = new ArrayList<Photo>();
JSONArray array = json.getJSONArray("items");
for (int i = 0; i < array.length(); i++) {
    JSONObject jsonPhoto = array.getJSONObject(i);
    Photo photo = new Photo();
    photo.title = jsonPhoto.getString("title");
    photo.media = jsonPhoto.getString("media");
    photo.desc = jsonPhoto.getString("description");
    photo.author = jsonPhoto.getString("author");
    datas.add(photo);
}
```

依照迴圈的索引
取出對應的JSONObject

1.以物件方式準備資料

```
JSONObject json = new JSONObject(response);
ArrayList<Photo> datas = new ArrayList<Photo>();
JSONArray array = json.getJSONArray("items");
for (int i = 0; i < array.length(); i++) {
    JSONObject jsonPhoto = array.getJSONObject(i);
    Photo photo = new Photo();
    photo.title = jsonPhoto.getString("title");
    photo.media = jsonPhoto.getString("media");
    photo.desc = jsonPhoto.getString("description");
    photo.author = jsonPhoto.getString("author");
    datas.add(photo);
}
```

建立屬於目前JSONObject
對應的物件Photo

1.以物件方式準備資料

```
JSONObject json = new JSONObject(response);
ArrayList<Photo> datas = new ArrayList<Photo>();
JSONArray array = json.getJSONArray("items");
for (int i = 0; i < array.length(); i++) {
    JSONObject jsonPhoto = array.getJSONObject(i);
    Photo photo = new Photo();
    photo.title = jsonPhoto.getString("title");
    photo.media = jsonPhoto.getString("media");
    photo.desc = jsonPhoto.getString("description");
    photo.author = jsonPhoto.getString("author");
    datas.add(photo);
}
```

將JSONObject的值取出
設定給物件Photo

1.以物件方式準備資料

```
JSONObject json = new JSONObject(response);
ArrayList<Photo> datas = new ArrayList<Photo>();
JSONArray array = json.getJSONArray("items");
for (int i = 0; i < array.length(); i++) {
    JSONObject jsonPhoto = array.getJSONObject(i);
    Photo photo = new Photo();
    photo.title = jsonPhoto.getString("title");
    photo.media = jsonPhoto.getString("media");
    photo.desc = jsonPhoto.getString("description");
    photo.author = jsonPhoto.getString("author");
    datas.add(photo);
}
```

將物件Photo擺入ArrayList中

客製化ListView

第二步、建立客製化的LAYOUT

2.建立客製化的LAYOUT

- 本範例會做成如下的Layout，名稱為list_item.xml儲存在專案資料夾的res/layout下

2.建立客製化的LAYOUT

- 本範例會做成如下的Layout

2.建立客製化的LAYOUT

- 本範例會做成如下的Layout，名稱為list_item.xml儲存在專案資料夾的res/layout下

客製化ListView

第三步、建立BASEADAPTER的子類別

3.建立BASEADAPTER的子類別

- 建立一個Class，去繼承BaseAdapter
- 當你繼承了BaseAdapter後，編輯器會請你新增4個基本的Function
  - getCount()
  - getItem()
  - getItemId()
  - getView()

3.建立BASEADAPTER的子類別

```
public class PhotoAdapter extends BaseAdapter {
    public int getCount() {
        return 0;
    }
    public Object getItem(int position) {
        return null;
    }
    public long getItemId(int position) {
        return position;
    }
    public View getView(int position, View convertView, ViewGroup parent) {
        return null;
    }
}
```

3.建立BASEADAPTER的子類別

```
public class PhotoAdapter extends BaseAdapter {
    public int getCount() {
        return 0;
    }
    public Object getItem(int position) {
        return null;
    }
    public long getItemId(int position) {
        return position;
    }
    public View getView(int position, View convertView, ViewGroup parent) {
        return null;
    }
}
```

繼承BaseAdapter

3.建立BASEADAPTER的子類別

```
public class PhotoAdapter extends BaseAdapter {
    public int getCount() {
        return 0;
    }
    public Object getItem(int position) {
        return null;
    }
    public long getItemId(int position) {
        return position;
    }
    public View getView(int position, View convertView, ViewGroup parent) {
        return null;
    }
}
```

負責回傳ListView有幾列要呈現

3.建立BASEADAPTER的子類別

```
public class PhotoAdapter extends BaseAdapter {
    public int getCount() {
        return 0;
    }
    public Object getItem(int position) {
        return null;
    }
    public long getItemId(int position) {
        return position;
    }
    public View getView(int position, View convertView, ViewGroup parent) {
        return null;
    }
}
```

負責回傳某一列對應的物件
position就是
ListView的哪一列

3.建立BASEADAPTER的子類別

```
public class PhotoAdapter extends BaseAdapter {
    public int getCount() {
        return 0;
    }
    public Object getItem(int position) {
        return null;
    }
    public long getItemId(int position) {
        return position;
    }
    public View getView(int position, View convertView, ViewGroup parent) {
        return null;
    }
}
```

一般直接回傳參數
position

3.建立BASEADAPTER的子類別

```
public class PhotoAdapter extends BaseAdapter {
    public int getCount() {
        return 0;
    }
    public Object getItem(int position) {
        return null;
    }
    public long getItemId(int position) {
        return position;
    }
    public View getView(int position, View convertView, ViewGroup parent) {
        return null;
    }
}
```

ListView每一列的組成處

position表示目前ListView
要組成的列位置

3.建立BASEADAPTER的子類別

```
public class PhotoAdapter extends BaseAdapter {
    public int getCount() {
        return 0;
    }
    public Object getItem(int position) {
        return null;
    }
    public long getItemId(int position) {
        return position;
    }
    public View getView(int position, View convertView, ViewGroup parent) {
        return null;
    }
}
```

回傳的View就會被擺在ListView的列上

客製化ListView

第四步、填入必備資訊

4.填入必備資訊

- 來擴充PhotoAdapter，讓第一步準備的資料可以成為Adapter需要的資訊

```
public class PhotoAdapter extends BaseAdapter {

    private ArrayList<Photo> mData;

    public PhotoAdapter(ArrayList<Photo> data) {
        mData = data;
    }
    public int getCount() {
        return (mData != null) ? mData.size() : 0;
    }
    public Object getItem(int position) {
        return mData.get(position);
    }
    public long getItemId(int position) {
        return position;
    }
    public View getView(int position, View convertView, ViewGroup parent) {
        return null;
    }
}
```

4.填入必備資訊

```
public class PhotoAdapter extends BaseAdapter {
    private ArrayList<Photo> mData;
    public PhotoAdapter(ArrayList<Photo> data) {
        mData = data;
    }
    public int getCount() {
        return (mData != null) ? mData.size() : 0;
    }
    public Object getItem(int position) {
        return mData.get(position);
    }
    public long getItemId(int position) {
        return position;
    }
    public View getView(int position, View convertView, ViewGroup parent) {
        return null;
    }
}
```

建立一個成員變數
負責管理整個Adapter的資料

4.填入必備資訊

```
public class PhotoAdapter extends BaseAdapter {
    private ArrayList<Photo> mData;
    public PhotoAdapter(ArrayList<Photo> data) {
        mData = data;
    }
    public int getCount() {
        return (mData != null) ? mData.size() : 0;
    }
    public Object getItem(int position) {
        return mData.get(position);
    }
    public long getItemId(int position) {
        return position;
    }
    public View getView(int position, View convertView, ViewGroup parent) {
        return null;
    }
}
```

建立一個建構子
參數為資料的ArrayList

將建構子的參數帶入儲存在
成員變數中

4.填入必備資訊

```
public class PhotoAdapter extends BaseAdapter {
    private ArrayList<Photo> mData;
    public PhotoAdapter(ArrayList<Photo> data) {
        mData = data;
    }
    public int getCount() {
        return (mData != null) ? mData.size() : 0;
    }
    public Object getItem(int position) {
        return mData.get(position);
    }
    public long getItemId(int position) {
        return position;
    }
    public View getView(int position, View convertView, ViewGroup parent) {
        return null;
    }
}
```

看有幾筆資料
ListView就要呈現幾列

4.填入必備資訊

```
public class PhotoAdapter extends BaseAdapter {
    private ArrayList<Photo> mData;
    public PhotoAdapter(ArrayList<Photo> data) {
        mData = data;
    }
    public int getCount() {
        return (mData != null) ? mData.size() : 0;
    }
    public Object getItem(int position) {
        return mData.get(position);
    }
    public long getItemId(int position) {
        return position;
    }
    public View getView(int position, View convertView, ViewGroup parent) {
        return null;
    }
}
```

依照參數position
回傳對應的資料

客製化ListView

第五步、在ADAPTER組出每一列

5.在ADAPTER組出每一列

```
public View getView(int position, View convertView, ViewGroup parent) {
    if (convertView == null) {
        convertView =
            LayoutInflater.from(parent.getContext())
            .inflate(R.layout.list_item, null);
    }
    ImageView image = (ImageView) convertView.findViewById(R.id.icon);
    TextView title = (TextView) convertView.findViewById(R.id.title);
    TextView author = (TextView) convertView.findViewById(R.id.author);
    Photo photo = (Photo) getItem(position);
    title.setText(photo.title);
    author.setText(photo.author);
    return convertView;
}
```

5.在ADAPTER組出每一列

```
public View getView(int position, View convertView, ViewGroup parent) {
    if (convertView == null) {
        convertView =
            LayoutInflater.from(parent.getContext())
            .inflate(R.layout.list_item, null);
    }
    ImageView image = (ImageView) convertView.findViewById(R.id.icon);
    TextView title = (TextView) convertView.findViewById(R.id.title);
    TextView author = (TextView) convertView.findViewById(R.id.author);
    Photo photo = (Photo) getItem(position);
    title.setText(photo.title);
    author.setText(photo.author);
    return convertView;
}
```

position表示listview
目前要組成哪一列

5.在ADAPTER組出每一列

```
public View getView(int position, View convertView, ViewGroup parent) {
    if (convertView == null) {
        convertView =
            LayoutInflater.from(parent.getContext())
            .inflate(R.layout.list_item, null);
    }
    ImageView image = (ImageView) convertView.findViewById(R.id.icon);
    TextView title = (TextView) convertView.findViewById(R.id.title);
    TextView author = (TextView) convertView.findViewById(R.id.author);
    Photo photo = (Photo) getItem(position);
    title.setText(photo.title);
    author.setText(photo.author);
    return convertView;
}
```

ListView有重複使用View的機制
ListView檢查系統發現有可重複使用的View，就會從這參數傳入

5.在ADAPTER組出每一列

```
public View getView(int position, View convertView, ViewGroup parent) {
    if (convertView == null) {
        convertView =
            LayoutInflater.from(parent.getContext())
            .inflate(R.layout.list_item, null);
    }
    ImageView image = (ImageView) convertView.findViewById(R.id.icon);
    TextView title = (TextView) convertView.findViewById(R.id.title);
    TextView author = (TextView) convertView.findViewById(R.id.author);
    Photo photo = (Photo) getItem(position);
    title.setText(photo.title);
    author.setText(photo.author);
    return convertView;
}
```

這個就是ListView

5.在ADAPTER組出每一列

如果發現參數沒有傳入
可重複使用的View
就自己**新建立一個**

```
public View getView(int position, View convertView, ViewGroup parent) {
    if (convertView == null) {
        convertView =
            LayoutInflater.from(parent.getContext())
            .inflate(R.layout.list_item, null);
    }
    ImageView image = (ImageView) convertView.findViewById(R.id.icon);
    TextView title = (TextView) convertView.findViewById(R.id.title);
    TextView author = (TextView) convertView.findViewById(R.id.author);
    Photo photo = (Photo) getItem(position);
    title.setText(photo.title);
    author.setText(photo.author);
    return convertView;
}
```

5.在ADAPTER組出每一列

新建立View的方式：
使用LayoutInflater
將步驟二的layou轉為View

```
public View getView(int position, View convertView, ViewGroup parent) {
    if (convertView == null) {
        convertView =
            LayoutInflater.from(parent.getContext())
            .inflate(R.layout.list_item, null);
    }
    ImageView image = (ImageView) convertView.findViewById(R.id.icon);
    TextView title = (TextView) convertView.findViewById(R.id.title);
    TextView author = (TextView) convertView.findViewById(R.id.author);
    Photo photo = (Photo) getItem(position);
    title.setText(photo.title);
    author.setText(photo.author);
    return convertView;
}
```

5.在ADAPTER組出每一列

```
public View getView(int position, View convertView, ViewGroup parent) {
    if (convertView == null) {
        convertView =
            LayoutInflater.from(parent.getContext())
            .inflate(R.layout.list_item, null);
    }
    ImageView image = (ImageView) convertView.findViewById(R.id.icon);
    TextView title = (TextView) convertView.findViewById(R.id.title);
    TextView author = (TextView) convertView.findViewById(R.id.author);
    Photo photo = (Photo) getItem(position);
    title.setText(photo.title);
    author.setText(photo.author);
    return convertView;
}
```

使用convertView的
findViewById()來找出
步驟二layout內部的View

5.在ADAPTER組出每一列

```
public View getView(int position, View convertView, ViewGroup parent) {
    if (convertView == null) {
        convertView =
            LayoutInflater.from(parent.getContext())
            .inflate(R.layout.list_item, null);
    }
    ImageView image = (ImageView) convertView.findViewById(R.id.icon);
    TextView title = (TextView) convertView.findViewById(R.id.title);
    TextView author = (TextView) convertView.findViewById(R.id.author);
    Photo photo = (Photo) getItem(position);
    title.setText(photo.title);
    author.setText(photo.author);
    return convertView;
}
```

使用getItem()找出
對應這一列的Photo資料

5.在ADAPTER組出每一列

```
public View getView(int position, View convertView, ViewGroup parent) {
    if (convertView == null) {
        convertView =
            LayoutInflater.from(parent.getContext())
            .inflate(R.layout.list_item, null);
    }
    ImageView image = (ImageView) convertView.findViewById(R.id.icon);
    TextView title = (TextView) convertView.findViewById(R.id.title);
    TextView author = (TextView) convertView.findViewById(R.id.author);
    Photo photo = (Photo) getItem(position);
    title.setText(photo.title);
    author.setText(photo.author);
    return convertView;
}
```

將資料內的title和author
設定給對應的View

5.在ADAPTER組出每一列

```
public View getView(int position, View convertView, ViewGroup parent) {
    if (convertView == null) {
        convertView =
            LayoutInflater.from(parent.getContext())
            .inflate(R.layout.list_item, null);
    }
    ImageView image = (ImageView) convertView.findViewById(R.id.icon);
    TextView title = (TextView) convertView.findViewById(R.id.title);
    TextView author = (TextView) convertView.findViewById(R.id.author);
    Photo photo = (Photo) getItem(position);
    title.setText(photo.title);
    author.setText(photo.author);
    return convertView;
}
```

最後記得，一定要回傳
convertView

5.在ADAPTER組出每一列

- 執行結果，似乎少了圖片
- 圖片下載並顯示在ListView上是一件滿有難度的事情
- 可以採用第三方的Library協助完成這項困難的工作

Recent Uploads tagged soccer

Ensaio-Campo do Ourinhos...
nobody@flickr.com (nariobarbosa)

KFCO Beerschot Wilrijk - Ly...
nobody@flickr.com (Kristof Vermeulen)

KFCO Beerschot Wilrijk - Ly...
nobody@flickr.com (Kristof Vermeulen)

KFCO Beerschot Wilrijk - Ly...
nobody@flickr.com (Kristof Vermeulen)

KFCO Beerschot Wilrijk - Ly...
nobody@flickr.com (Kristof Vermeulen)

KFCO Beerschot Wilrijk - Ly...
nobody@flickr.com (Kristof Vermeulen)

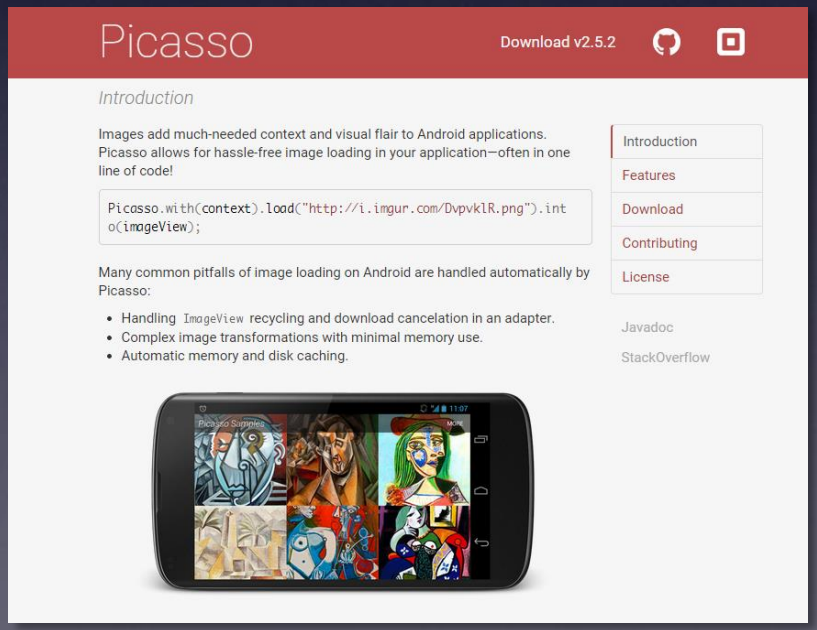
KFCO Beerschot Wilrijk - Ly...
nobody@flickr.com (Kristof Vermeulen)

KFCO Beerschot Wilrijk - Ly...
nobody@flickr.com (Kristof Vermeulen)

KFCO Beerschot Wilrijk - Ly...

5.在ADAPTER組出每一列

- Picasso
  - http://square.github.io/picasso/
- Library下載位置
  - http://repo1.maven.org/maven2/com/squareup/picasso/picasso/2.5.2/picasso-2.5.2.jar



5.在ADAPTER組出每一列

- 將下載的Library(.jar)擺在專案的lib資料夾下
- 在官網首頁就有教學，如何使用Picasso去下載圖並呈現在ImageView上
  - Picasso.with(context).load(‘圖片網址’).into(某個ImageView)
- 現在來在Adapter的getView()補上程式碼吧

5.在ADAPTER組出每一列

```
public View getView(int position, View convertView, ViewGroup parent) {
    if (convertView == null) {
        convertView =
            LayoutInflater.from(parent.getContext())
            .inflate(R.layout.list_item, null);
    }
    ImageView image = (ImageView) convertView.findViewById(R.id.icon);
    TextView title = (TextView) convertView.findViewById(R.id.title);
    TextView author = (TextView) convertView.findViewById(R.id.author);
    Photo photo = (Photo) getItem(position);
    title.setText(photo.title);
    author.setText(photo.author);
    Picasso.with(parent.getContext()).load(photo.media).into(image);
    return convertView;
}
```

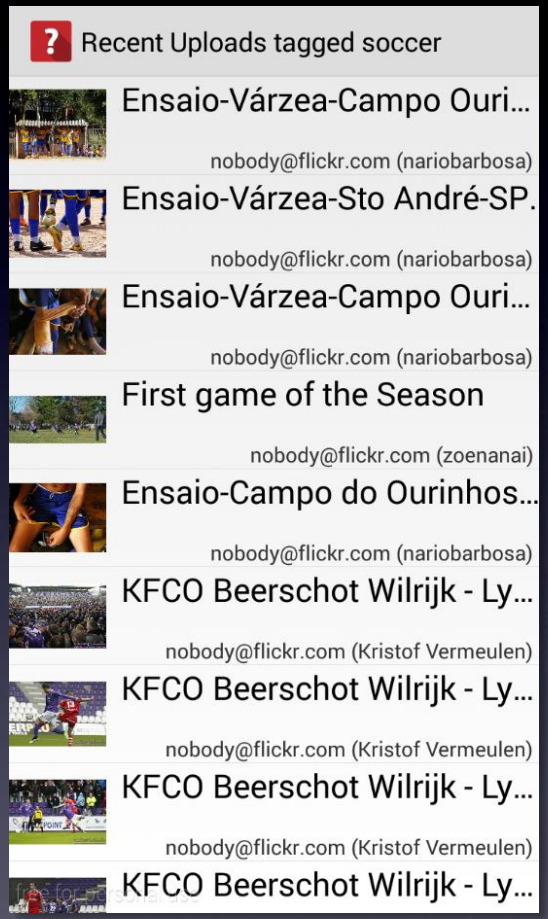
要求下載完的圖片呈現在imageview中

將Photo資料的link傳給Picasso

5.在ADAPTER組出每一列

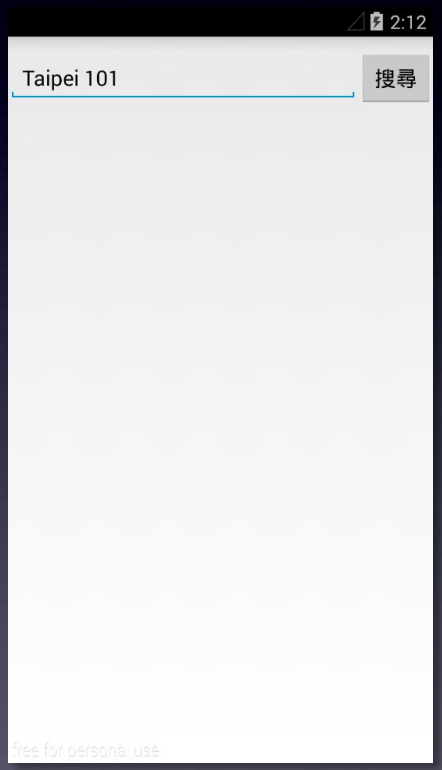
- 如此圖片就可以自動呈現了
- 若要圖片改為正方形呈現，可以使用以下程式碼

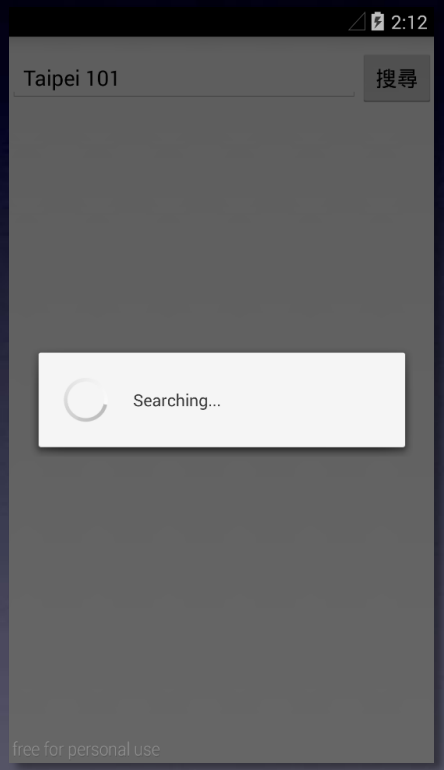
```
Picasso.with(parent.getContext())
.load(photo.media)
.fit().centerCrop()
.into(image);
```



作業2

- 做一個圖片搜尋的App





作業2

- 使用的API是GET https://ajax.googleapis.com/ajax/services/search/images?v=1.0&q=<關鍵字>&start=<頁面數>
  - 頁面數一定是4的倍數，起始為0，下一次的分頁就是4，依序為8, 12, 16…
- 自己用瀏覽器試試看

作業2

- 讀取更多可以使用ListView的Footer來製作，請參考以下網址
  - http://www.cnblogs.com/loulijun/archive/2012/10/25/2738952.html
- 讀取後的內容要「連接」在原本的內容之後，可以參考以下連結
  - http://givemepass.blogspot.tw/2011/10/listview.html

Q & A