Android App 開發技術分享

(非同步處理、資料處理、顯示與交換)

jameschen0125@gmail.com

資料處理與、顯示與交換

- 非同步處理技巧與時機 (*另一份講義)
- 檔案處理、SQLite 資料庫
- 大量資料顯示的議題
  - ListView/Spinner/Gallery
  - RecyclerView(RecyclerViewAdapter) / CardView / …
- 畫面設計: from Activity to Fragments
  - Fragment 畫面切割與處理
- 網路資料交換：Socket, JSON, Bluetooth…

資料處理(交換) - 1

SharedPreferences / PreferenceActivity

File I/O、SQLite

http://developer.android.com/guide/components/fragments.html

SharedPreferences 基本操作

- 使用 **SharedPreferences** 儲存所需要保存的資料，系統會將其存放在特定的位置 **(存為XML** 格式**)**
  - /data/data/_____/_____
- **基本語法**

```
// Write
Editor sped = getSharedPreferences( filename, mode ).edit();
sped.putXXX( key, value).commit();
// Read
SharedPreferemces sp = getSharedPreferences( filename, mode);
sp.getXXX(key);
```

練習時間： SharedPreferences

- 請匯入 LoginApp_New.zip
  - LoginApp600SP 專案
- 設計重點
  - 登入成功後的動作
    - 記錄使用者帳號 ➔ 存入 [喜好設定檔]
  - 畫面開啟時(登入前) 可先載入、顯示前次成功登入的帳號
    - 讀取 [喜好設定檔] 內的資料

File I/O – 處理資料

- 內部空間(專屬目錄): /data/data/_____/_____/abc.txt)

  FileInputStream *fin =* ***openFileInput****(檔名);*

  FileOutputStream *fout =* ***openFileOutput****(檔名, mode);*

- 檔案系統(一般是指定到SD的路徑)

  FileInputStream *fin =* ***new FileInputStream****(/路徑/檔名);*

  FileOutputStream *fout =* ***new FileOutputStream****(/路徑/檔名);*

必要時轉為
Reader/Writer

絕對路徑、內部資訊存取

- **取得 App(Package) 私有路徑**

```
String appFilesPath = "/data/data/"
+ context.getApplicationContext().getPackageName() + "/files";
```

- **取得 SD 掛載路徑**

```
String filesPath =
   Environment.getExternalStorageDirectory().getAbsolutePath()
   + "/mpos";
```

- **取得 Assets 內部資料**

```
FileInputStream fin = Context.getAssets().open(filename);
```

- **取得 raw 原生資料夾內部資料**

```
FileInputStream fin = getResources().openRawResource(R.raw.id);
```

練習時間： File I/O

- 請匯入 LoginApp_New.zip
  - LoginApp601File 專案
- 設計重點
  - 登入成功時的動作
    - 寫入 [設定檔]
  - 畫面開啟時(登入前) 載入並顯示先前登入過的帳密。
    - 讀取 [設定檔]

SQLite Database 的基本使用方法

- **取得 DB 控制物件 (*SQLiteOpenHelper)**

```
SQLiteDatabase db =
        SQLiteDatabase.openOrCreateDatabase( dbFilePath, null);
        // openOrCreateDatabase( dbFilename, mode, null);
```

- **執行指令** (亦可使用 CRUD 對應的四種方法)

```
db.execSQL("SQL Statement");
Cursor cursor = db.rawQuery("SQL Statement");
```

- *釋放資源*

```
cursor.close();
db.close();
```

Cursor 相關指令

- 利用 **Cursor** 物件 **(myCursor)** 來讀取資料集內容 **(RecordSet/ResultSet)**

1. *int num = myCursor.**getCount()**;*
2. ***XXX** data = myCursor.**getXXX(fieldIdx)**; // 0 ~ …*
3. *myCursor.**moveToFirst()**;*
4. *myCursor.**moveToNext()**;*
5. *myCursor.**moveToPosition(idx)**;*
6. *myCursor.**close()**;*

利用 SQLiteOpenHelper 處理資料庫

1. **定義 SQLite Helper Class**：繼承**SQLiteOpenHelper**、並覆寫以下方法:
   - *<Constructor>:* // 呼叫父類別建構子, 必要時會建立DB File、升級(onUpgrade).
     super(context, DB_NAME, CursorFactory, DATABASE_VERSION);
   - *onCreate(SQLiteDatabase):* 初始化動作：ex:利用SQL 建立程式中需要的表格.
   - *onUpgrade(SQLiteDatabase, int, int):* 必要時會被叫用: 更新資料庫(新版本).
   - *onOpen(SQLiteDatabase):* 每次開啟資料庫後的初始化指令。

2. **利用 Helper 物件 (*myDbHelper*)取得 SQLiteDatabase 管理物件**:
   SQLiteDatabase** db = **myDbHelper.getWritableDatabase( ) ;
   或
   SQLiteDatabase** db = **myDbHelper.getReadableDatabase( );
3. **取得 DB 控制物件之後，處理資料的步驟則與前面相同。**

練習時間： SQLiteLab01

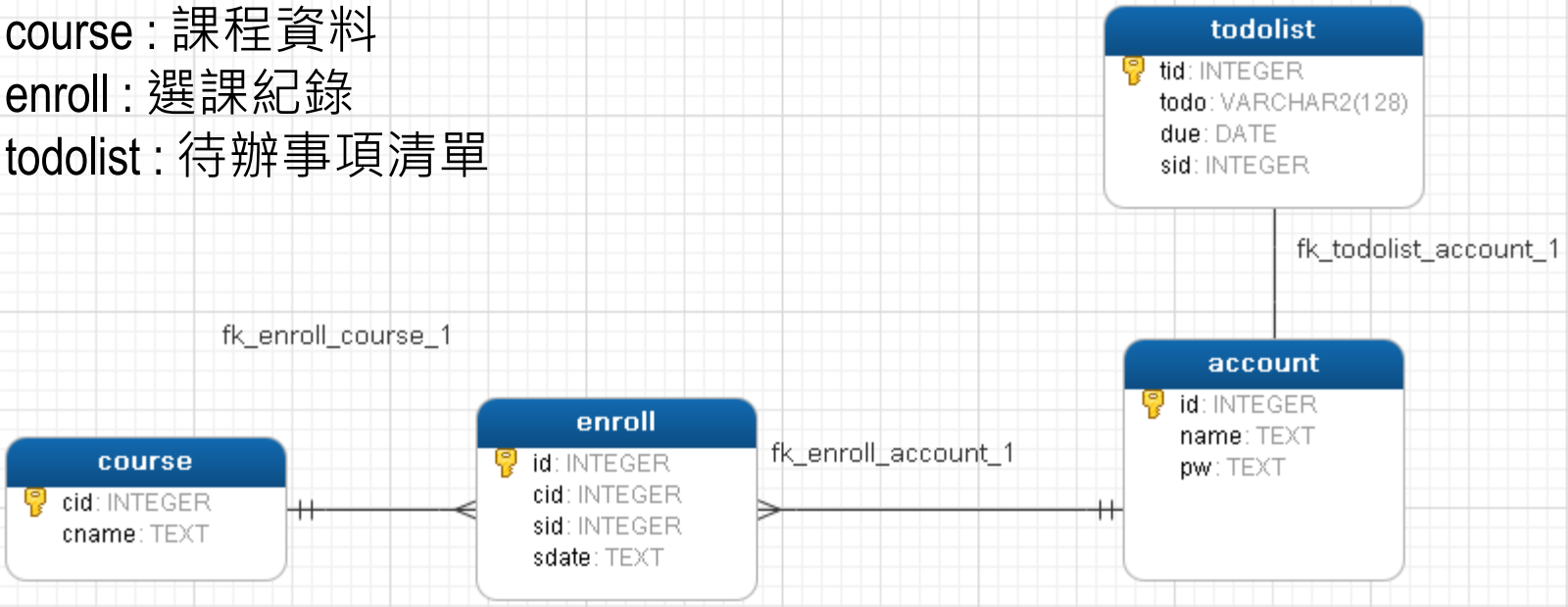
- 請匯入 LoginApp_New.zip
  - LoginApp700SQLiteLab01 專案
- 設計重點
  - 登入時檢查帳密是否正確：開啟 SQLite DB

```
public boolean isLegalAccount(String acc, String pw) {
    1. 開啟資料庫連線 (Schema, username, password, encoding...)
    2. 查詢帳號、密碼
    3. 確認結果
    4. 關閉必要連線、釋放資源.
    5. 回傳
}
```

註： 專案已內建一個認證用的 SQLite DB (loginapp.sqlite) ；可用 Navicat 開啟

loginapp.sqlite

account : 帳號資料
course : 課程資料
enroll : 選課紀錄
todolist : 待辦事項清單



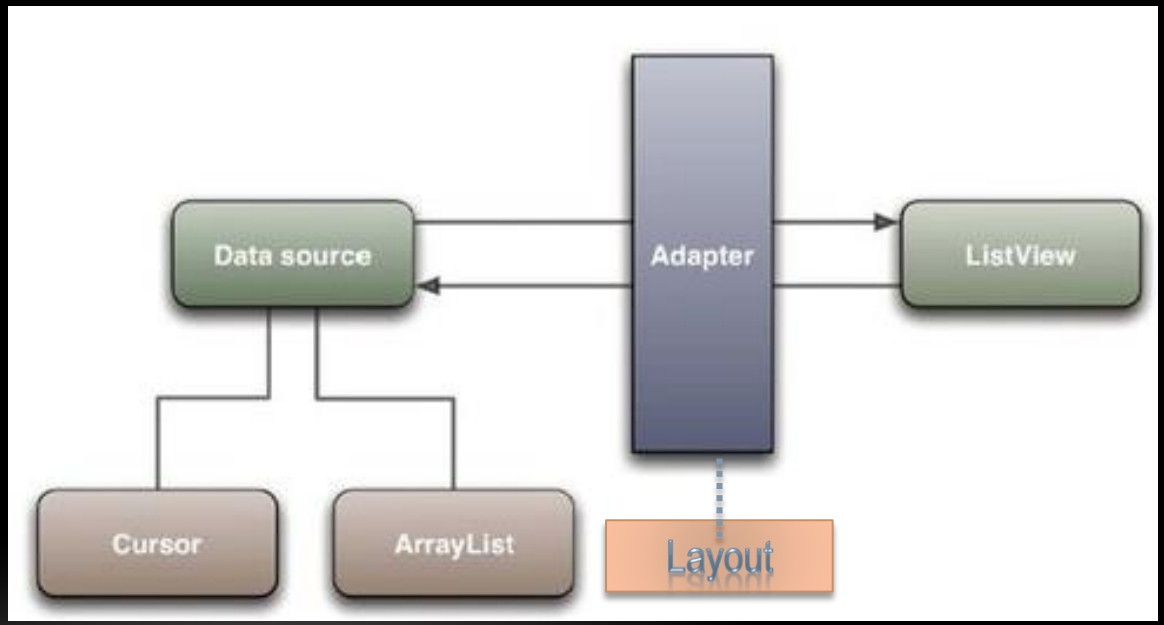
Adapter 與 AdapterView (Adapter pattern)

動態、大量資料的顯示

ListView & Adapter

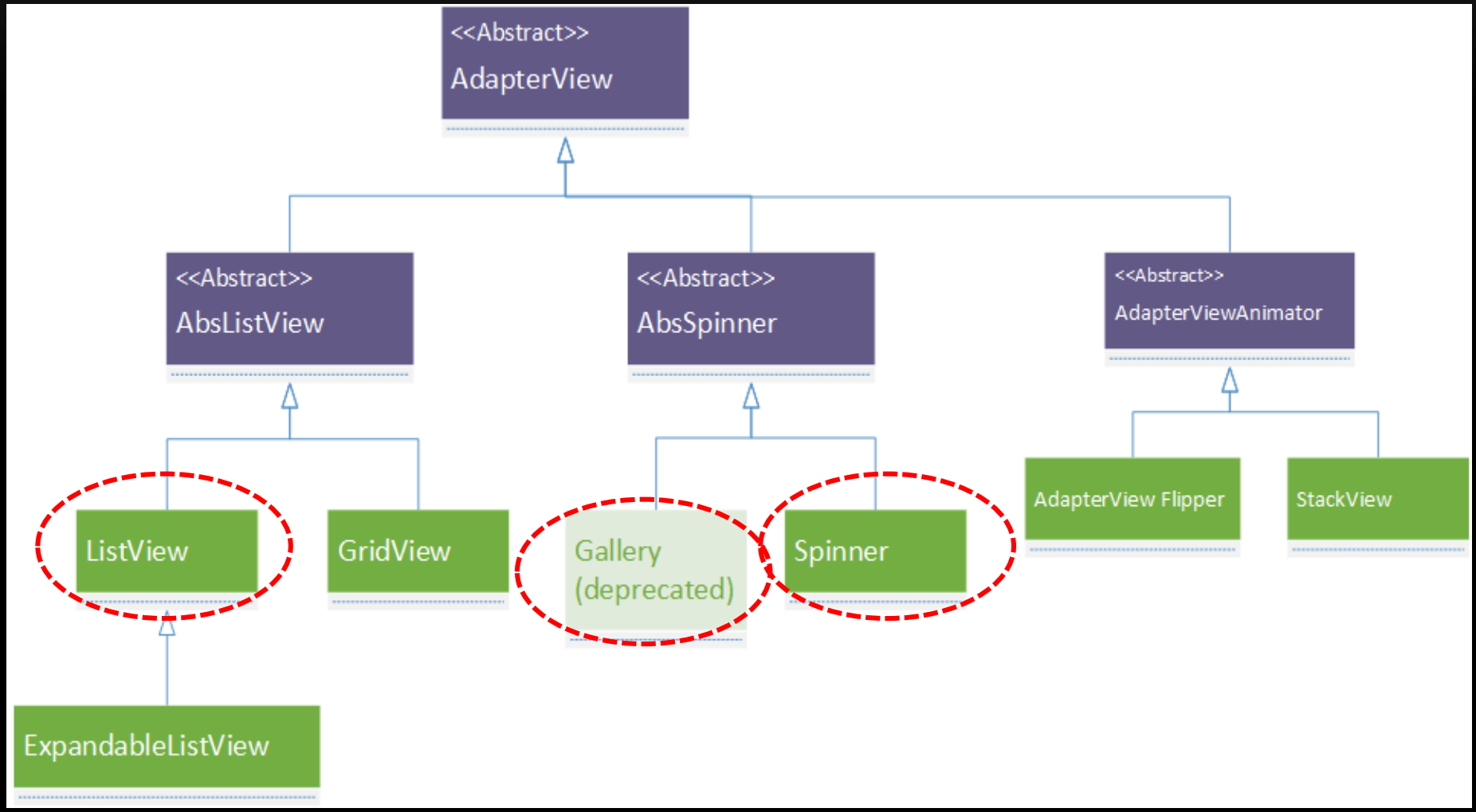
- ListView 是 Android 常見的一種視圖 (例：電話簿、大量資料列表)
  - ListView 一般會搭配 Adapter 來顯示資料；
- Adapter負責提供資料(以**特定樣式**來包裝資料)給 AdapterView 顯示

Adapter
BaseAdapter
ArrayAdapter<T>
CursorAdapter
SimpleAdapter
ResourceCursorAdapter
SimpleCursorAdapter



- 亦可繼承 ListActivity (其內部已經包含一個全螢幕的 ListView)

AdapterView 繼承架構圖

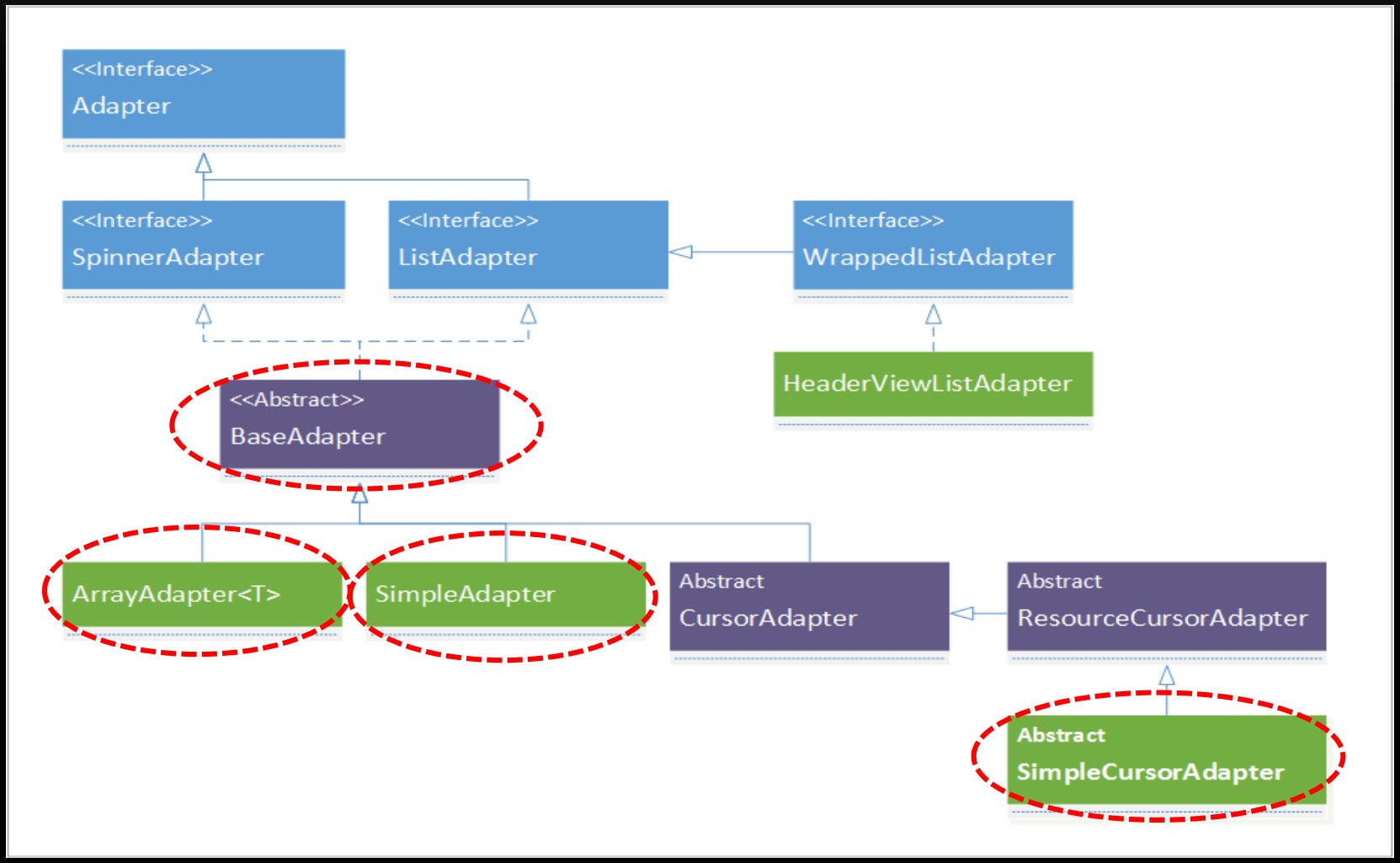


http://www.intertech.com/Blog/android-adapters-adapterviews/

常用的Adapters

- **BaseAdapter**：抽象類別
  - 適用於需要自訂 ListView 外觀等應用的場合
  - 需要實作多個方法
- **ArrayAdapter**：適合處理單一內容的陣列(或List)資料
- **SimpleAdapter**：適用於較複雜的資料顯示 (List+HashMap)
- **SimpleCursorAdapter**：處理資料庫的查詢結果(Cursor)

Adapter 繼承架構圖



http://www.intertech.com/Blog/android-adapters-adapterviews/

以 ArrayAdapter 載入字串陣列

(A) 從 Resources 內讀取:

```
ArrayAdapter<CharSequence> adapter
    = ArrayAdapter.createFromResource(
      MainActivity.this,
      R.array.myArray,
      android.R.layout.simple_list_item_1);
```

(B) 在程式內宣告並讀取: (list : CharSequence[ ])

```
ArrayAdapter<CharSequence> adapter
    = new ArrayAdapter<CharSequence>(
      MainActivity.this,
      android.R.layout.simple_list_item_1,
      list  );
```

練習時間： ListViewDemo01

- 請匯入 ListViewDemo.zip
  - ListViewDemo01 專案
- 設計重點
  - 利用 ListView 顯示陣列資料、完成點擊事件處理

1. *產生資料(Array, List, ...)*
2. *生成 ArrayAdapter*
3. *找到 ListView 物件*
4. *將 Adapter 指定給 ListView 物件.*
5. *註冊事件處理，完成事件處理程式*

以 SimpleAdapter 顯示**較複雜**資料?

```
SimpleAdapter adapter
  = new SimpleAdapter(
          this,
           R.layout.myItem,
          dataList,
           new String[ ] {  "master", "detail" },
           new int[ ] {   R.id.master, R.id.detail }
    );
```

dataList: List of 物件
內部元素: HashMap<Key,Value>

練習時間： ListViewDemo02

- 請匯入 ListViewDemo.zip
  - ListViewDemo02 專案
- 設計重點
  - 利用 ListView 顯示複雜資料(泛型)、搭配 SimpleAdapter

1. *產生資料 : ArrayList, HashMap*
2. *生成 SimpleAdapter*
3. *找到 ListView 物件*
4. *將 Adapter 指定給 ListView 物件.*
5. *註冊事件處理，完成事件處理程式*

客製化 Adapter

```
1.  public class MyAdapter extends BaseAdapter {
2.      public MyAdapter(Activity context, List list) {
3.              // 初始化程序
4.      }
5.      public int getCount() {
6.         // 回傳資料總數
7.      }
8.      public Object getItem(int position) {       }
9.      public long getItemId(int position) {       }
10.     public View getView(int position, View convertView, ViewGroup parent) {
11.        // 將資料來源填充成 View, 再回傳以利 AdapterView 顯示
12.           return convertView;
13.     }
14. }
```

練習時間： ListViewDemo03

- 請匯入 ListViewDemo.zip
  - ListViewDemo03 專案
- 設計重點
  - 利用 ListView 顯示複雜資料(泛型)、搭配客製化的Adapter

1. *產生資料 : ArrayList, HashMap*
2. ***利用客製化 Adapter***
   - *定義客製化 MyAdapter: 繼承 BaseAdapter*
3. *找到 ListView 物件*
4. *將 Adapter 指定給 ListView 物件.*
5. *註冊事件處理，完成事件處理程式*

以 SimpleCursorAdapter 讀取資料表

```
SimpleCursorAdapter adapter
  = new SimpleCursorAdapter(
          this,
          R.layout.myItem,
          myCursor,
          new String[ ] { "_id", "item", "money" },
          new int[ ] { R.id.id1, R.id.id2, R.id.id3 }
    );
```

Cursor myCursor
= db.rawQuery("Select * from ABC");
// 一定要有 Primary Key : _id

練習時間： SQLiteLab02

- 請匯入 LoginApp_New.zip
  - LoginApp701SQLiteLab02 專案
- 設計重點
  - 在 Activities 之間傳遞資料
    - 利用 Intent + Bundle
  - 利用 ListView 顯示資料表資料、搭配SimpleCursorAdapter

0. 連結 ListView

1. 建立資料庫物件
2. SQL statement
3. 利用 SimpleCursorAdapter 串接資料
4. 建立 ListView & Adapter 的關係
5. 事件處理 if need !

內建的包裝樣式 Android.R.layout

- 單行文字
  - android.R.layout.simple_list_item_1
- 一行文字較大(主標題)，一行文字較小(內容)
  - android.R.layout.simple_list_item_2
- 單選
  - android.R.layout.simple_list_item_single_choice
- 多選按鈕
  - android.R.layout.simple_list_item_multiple_choice
- 檢核箱
  - android.R.layout.simple_list_item_checked

Android 內建的 Layout 模組-1

也可以定義自己的 layout！

android.R.layout.simple_list_item_1

```
1.  <?xml version="1.0" encoding="UTF-8"?>
2.  <TextView xmlns:android="http://schemas.android.com/apk/res/android"
3.      android:id="@android:id/text1"
4.      style="?android:attr/listItemFirstLineStyle"
5.      android:paddingTop="2dip"
6.      android:paddingBottom="3dip"
7.      android:layout_width="fill_parent"
8.      android:layout_height="wrap_content"
9.  />
```

android.R.layout.simple_list_item_2

```
1.  <?xml version="1.0" encoding="UTF-8"?>
2.  <TextView android:id="@android:id/text1"
3.          android:layout_width="fill_parent"
4.          android:layout_height="wrap_content"
5.          style="?android:attr/listItemFirstLineStyle"/>
6.   <TextView android:id="@android:id/text2"
7.          android:layout_width="fill_parent"
8.          android:layout_height="wrap_content"
9.          android:layout_below="@android:id/text1"
10.         style="?android:attr/listItemSecondLineStyle" />
```

Android 內建的 Layout 模組-2

android.R.layout.**simple_spinner_item.xml**

```
1.  <?xml version="1.0" encoding="UTF-8"?>
2.  <TextView
3.      xmlns:android="http://schemas.android.com/apk/res/android"
4.      android:ellipsize="marquee"
5.      android:id="@id/text1"
6.      android:layout_width="fill_parent"
7.      android:layout_height="wrap_content"
8.      android:singleLine="true"
9.      style="\?android:attr/spinnerItemStyle"
10. />
```

android.R.layout.**simple_spinner_dropdown_item.xml**

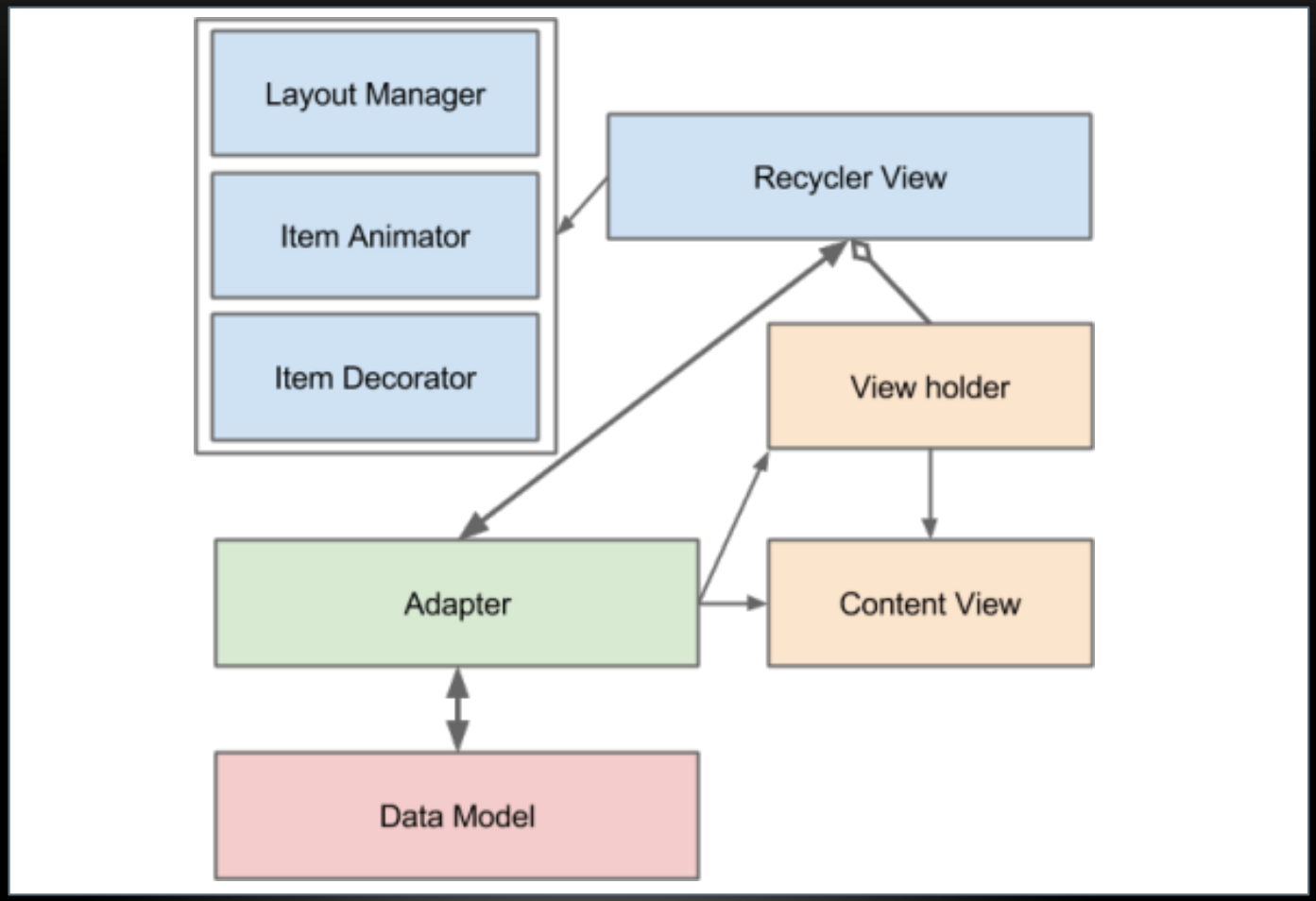
```
1.  <?xml version="1.0" encoding="UTF-8"?>
2.  <CheckedTextView
3.      xmlns:android="http://schemas.android.com/apk/res/android"
4.      android:ellipsize="marquee"
5.      android:id="@id/text1"
6.      android:layout_width="fill_parent"
7.      android:layout_height="?listPreferredItemHeight"
8.      android:singleLine="true"
9.      style="\?android:attr/spinnerDropDownItemStyle"
10.  />
```

進階元件: RecyclerView (Android 5.0+)

- 比 ListView 和 Gallery 更彈性的顯示方式，可以同時支援水平與垂直，支援滾動之過場動畫
  - 先前版本必須參考【Android support library v7】
- 利用 LayoutManager 控制顯示區域的排列方式
  - GridLayout、LinearLayout
- 利用 ViewHolder 提高整體效率：避免重複 findViewById(...)

Class Name	功能	註解
RecyclerView	主要顯示區域	
LayoutManager	規範資料顯示的排列方式	
RecyclerView.Adapter	提供顯示的資料(View)來源	
RecyclerView.ViewHolder	紀錄View內部元件以利後續顯示資料、 註冊事件處理程序...	

RecyclerView and Adapter



http://blogs.techsmith.com/inside-techsmith/devcorner-android-lollipop/
https://www.youtube.com/watch?v=Wq2o4EbM74k
https://github.com/slidenerd/materialtest

使用 RecyclerView 的關鍵步驟

- 專案須參考 v7 support library (+ Library dependency)
- **於佈局檔內放入 RecyclerView**

  <android.support.v7.widget.RecyclerView ...>
- **以 setLayoutManager(...) 指定佈局管理員**

  RecyclerView rv = (RecyclerView) findViewById(R.id.rv1);

  rv.setLayoutManager(布局管理物件);
- **指定 RecyclerView.Adapter**

  rv.setAdapter(myAdapter);

客製化 RecyclerView.Adapter

```
1.  public class MyAdapter extends
    RecyclerView.Adapter<MyAdapter.MyViewHolder> {
2.      // 定義屬性資料：如 Context, Cursor, List, …
3.      // 建構函數，根據需要定義
4.      public MyAdapter(Context context, Cursor cursor) {
5.          // 初始化
6.      }
7.      public int getItemCount( ) {
8.          // 回傳資料的數量 from Cursor, List, …
9.      }
10.     public MyViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
11.         return new MyViewHolder(mLayoutInflater.inflate(R.layout.deal_entry, parent, false));
12.     }
13.     // … 接下一頁
14. }
```

MyViewHolder

```
public void onBindViewHolder(MyViewHolder holder, int position) {
    // 取得並顯示指定位置的資料(position), 也可利用 setTag 紀錄複雜資料(ex: HashMap)
    // holder.tvSN.setText(theData);
}
```

```
class MyViewHolder extends RecyclerView.ViewHolder {
    private TextView tvSN, tvDealNo;
    public MyViewHolder(View view) {
        super(view);
        tvSN = (TextView)view.findViewById(R.id.tvSN);
        tvDealNo = (TextView)view.findViewById(R.id.tvDealNo);
        // 事件處理
        view.setOnClickListener( /*事件處理模組*/ );
    }
}
```

練習時間： RecyclerViewDemo1

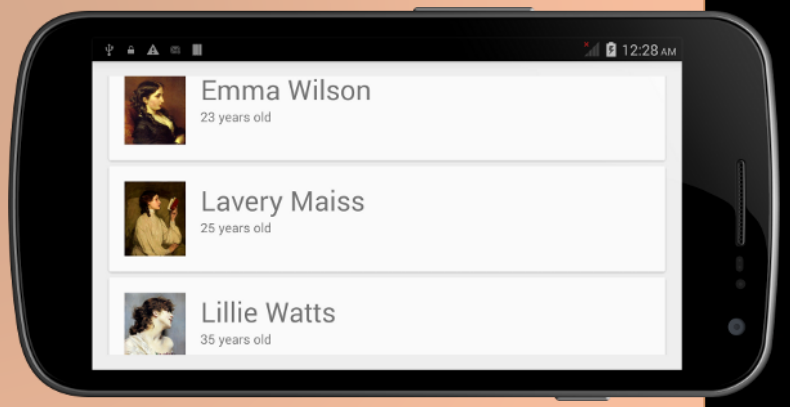
- 請匯入 RecyclerViewDemo.zip / RecyclerViewDemo1 專案
- 設計重點
  - 在佈局檔內加入 RecyclerView
    - 必須匯入 V7 RecyclerView 以及 V7 Appcomp 到工作區，
    - 專案必須引用這些程式庫!
  - 定義並產生客製化 RecyclerView.**Adapter**
    - 繼承RecyclerView.Adapter<NewsRcViewAdapter.**MyViewHolder**>
  - 顯示資料
    1. 產生資料 : ArrayList, HashMap
    2. 生成 Adapter
    3. 找到 RecyclerView物件, 設定 LayoutManager
    4. 將 Adapter 指定給 RecyclerView物件.

CardView – 卡片視圖

- CardView 繼承 FrameLayout，方便產生**陰影**(利用 elevation) 和**圓角**效果。

```
<android.support.v7.widget.CardView
    ...
    card_view:cardElevation="8dp"
    card_view:cardCornerRadius="5dp" >
    <RelativeLayout
        android:layout_width="match_parent"
        android:layout_height="100dp"
        android:padding="5dp" >
        <ImageView  ... />
        <TextView  ... />
    </RelativeLayout>

</android.support.v7.widget.CardView>
```



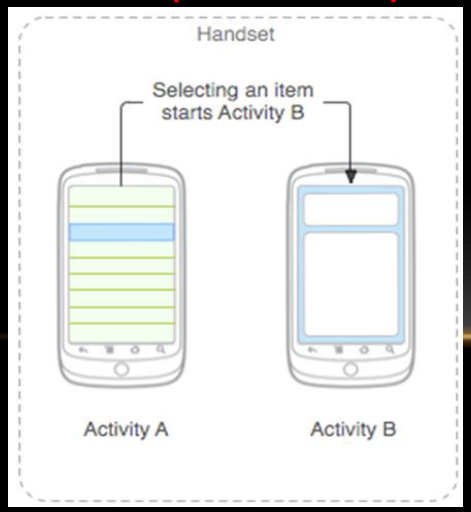
Fragment and Activity

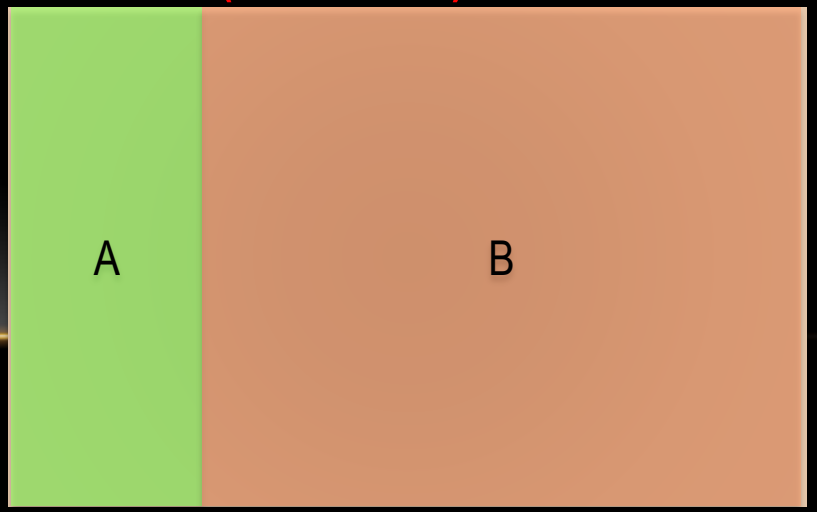
A Fragment represents a **behavior** or a **portion of user interface** in an Activity, **which has its own lifecycle.**

http://developer.android.com/guide/components/fragments.html

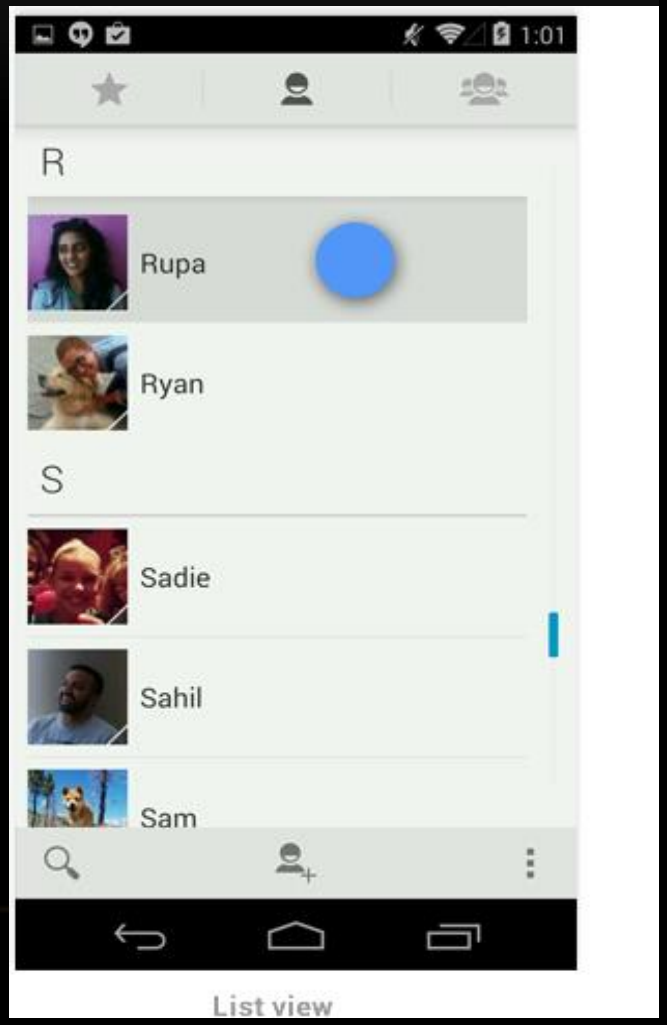
Why we need Fragment?

- 傳統架構下，在同一時間，螢幕上只能出現一個 Activity (UI) ➔ 由一組 XML + Java 共同決定畫面。
- 隨著**平板電腦**的出現(解析度更高)，**整個畫面的規劃與設計的機制必須改變。**
  - Android 3 (Honeycomb) 發展出 <Fragment> 元件
  - **讓元件(XML/Java)可以被重覆使用 (個別檔案)。**

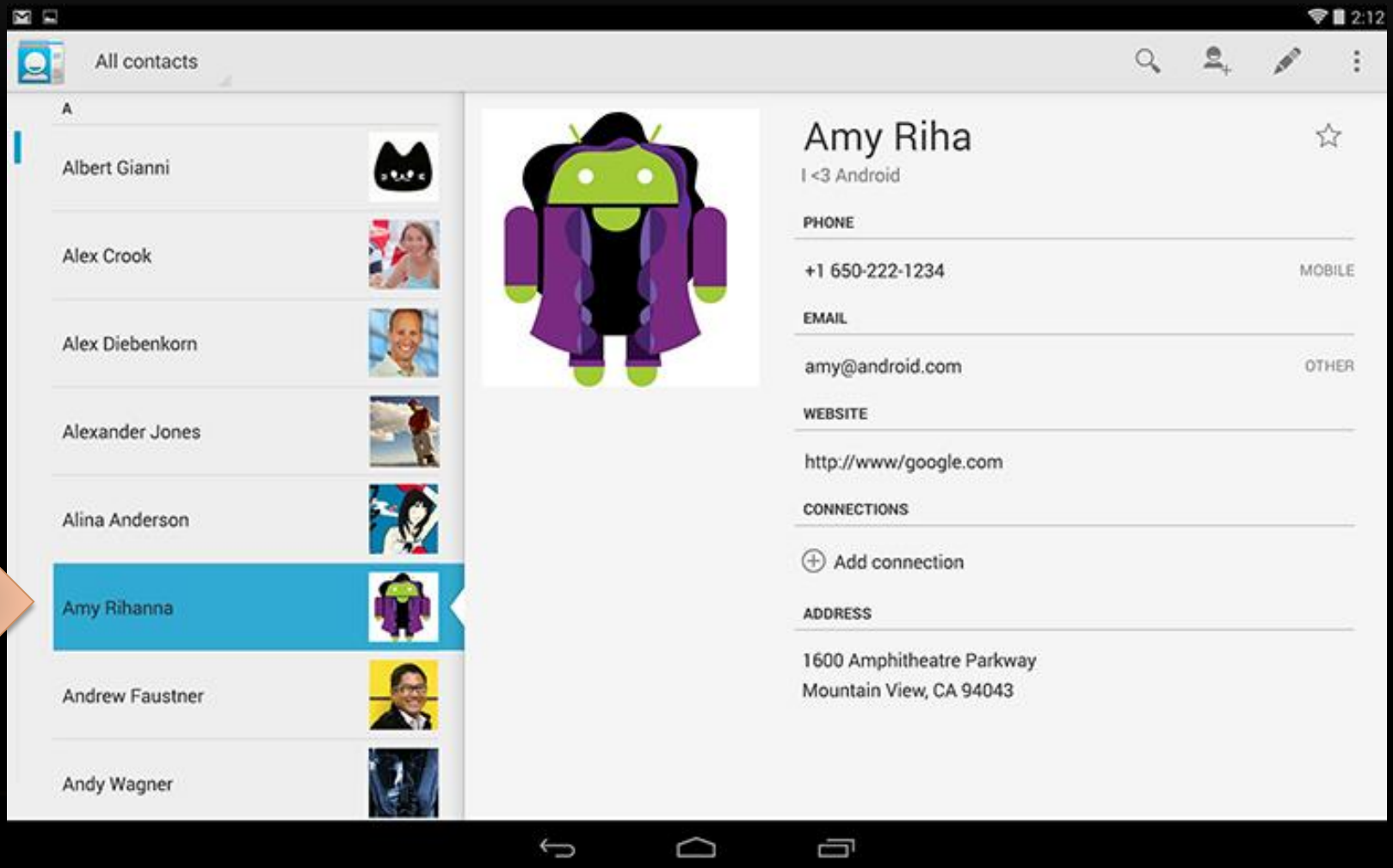




Master (List) → Detail (不同時出現)



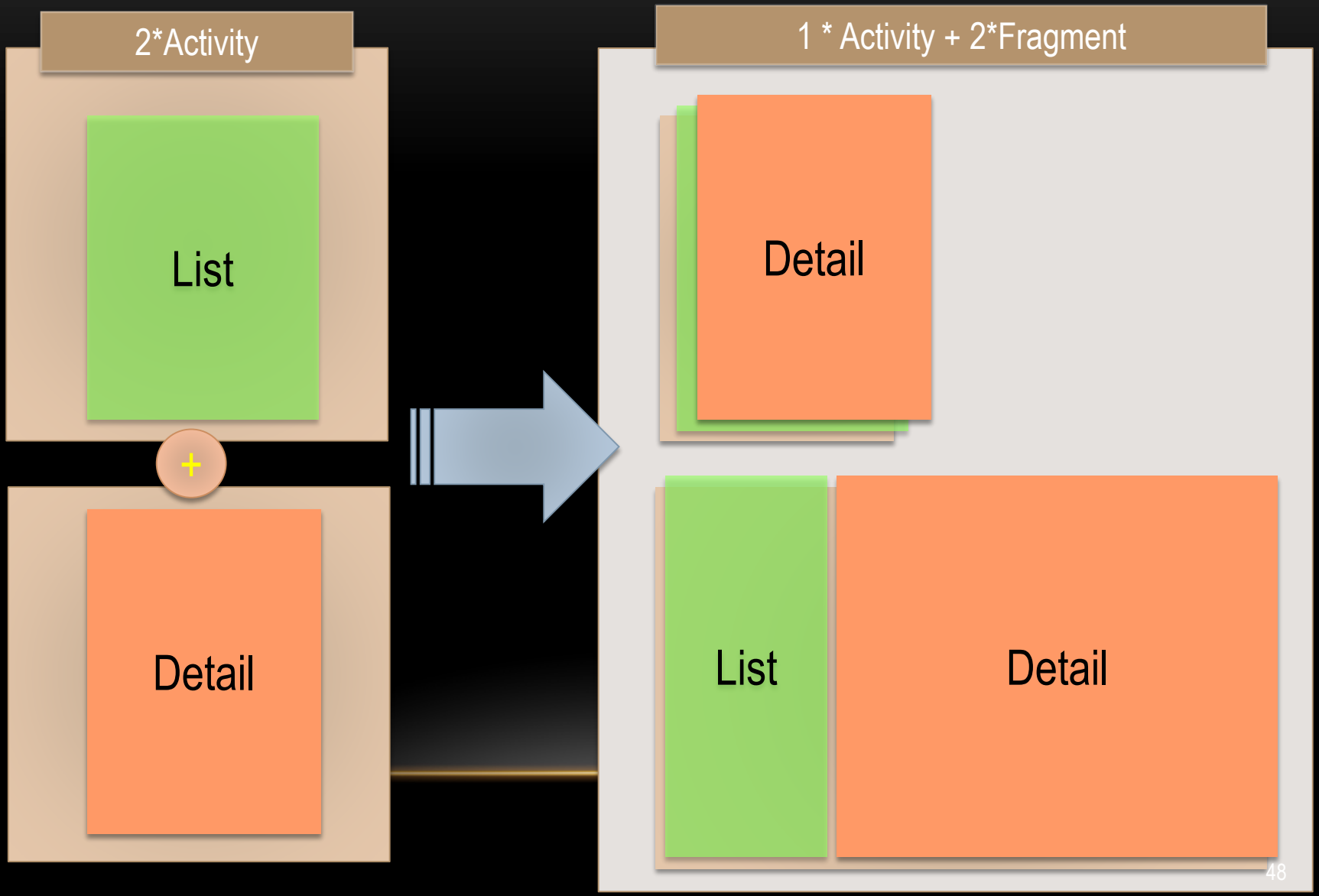
Master + Detail (同時出現)



決定雙欄顯示方式 (Master/Detail)

- **由解析度決定**
  - Layout-large :
- **由設備方向決定**
  - layout-port : 單欄
  - layout-land : 雙欄
- 設計師(程式)**根據需要**來決定

Fragment 使用情境



Fragment is …

- Fragment 與 Activity：
  - ✓ 可視為 **mini-Activity**, 提供一個動態且彈性的UI控制 ;
  - ✓ 將Activity 畫面分割為多個**區域**(Fragment)
    - ✓ **主控元件 Activity 稱為 Host Activity**
    - ✓ 個別 Fragment 擁有自己的生命周期，但仍依附在所屬的 Activity 之內、無法獨立存在。
    - ✓ **每個區域有個別的 Layout**;
  - ✓ Activity 利用 **Back Stack** 紀錄 Fragment 的歷史活動.
- **注意事項**：
  - ✓ Fragment 可不隸屬於任何 Activity(只有指令、無UI)
  - ✓ 2.x 版需加入android support library v4 程式庫

Utilize the Fragments

- **內建的 Fragment 相關類別**
  - ✓ **Fragment**：客製化(繼承)的主要基礎類別
  - ✓ **ListFragment** : 類似 ListActivity，覆寫 **onListItemClick**() 進行**事件**處理
  - ✓ **DialogFragment** : 類似浮動的 Dialog
  - ✓ **PreferenceFragment**: 類 PreferenceActivity，用於 "**設定**"功能畫面
  - ✓ …
- **客製化 Fragment -- 繼承 Fragment (或子類別) 、覆寫必要的方法** (下一頁)

Utilize the Fragments – cont.

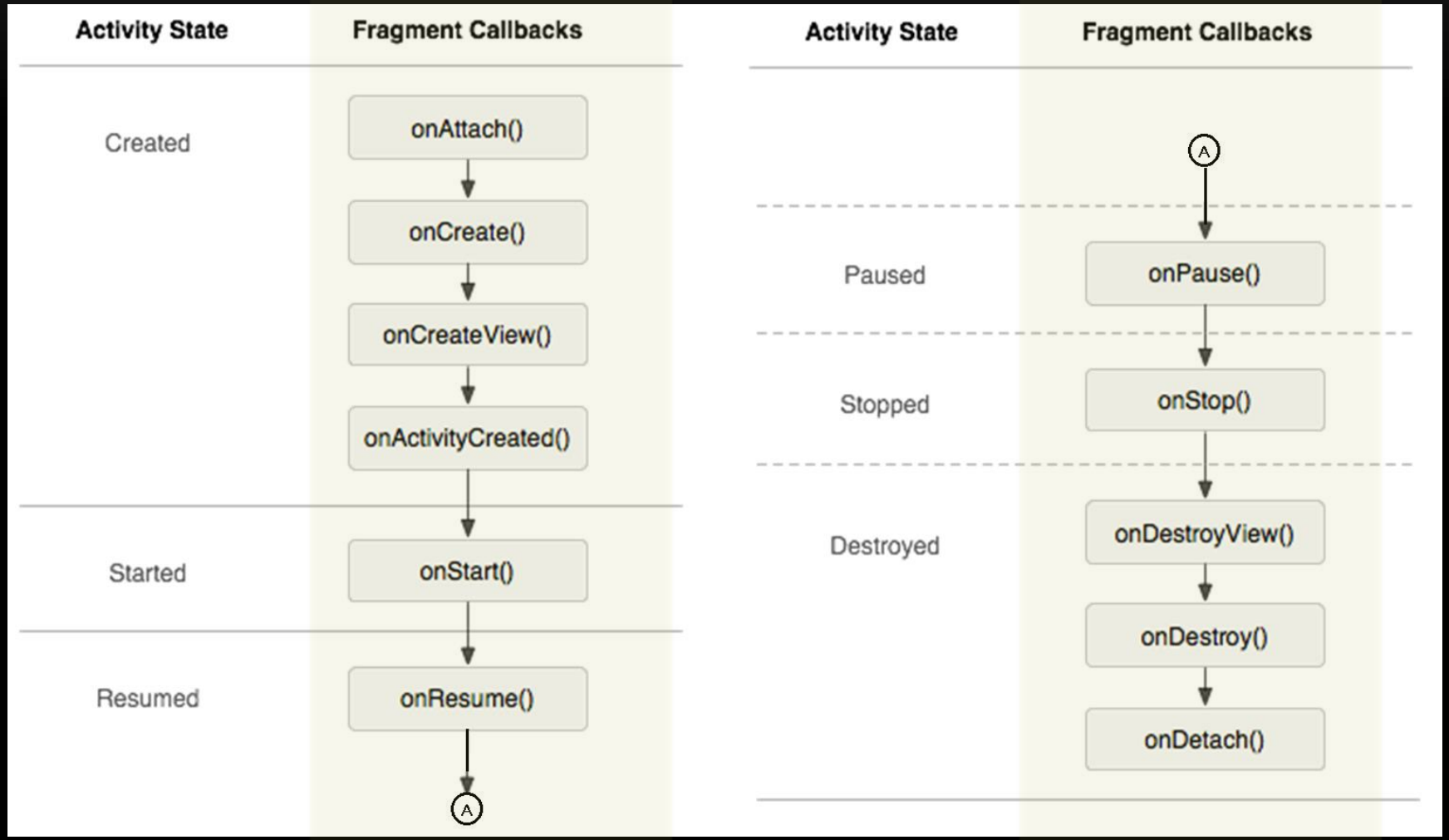
- **內建的 Fragment 相關類別**
  - **Fragment、ListFragment 、DialogFragment 、PreferenceFragment**
- **客製化 Fragment -- 繼承Fragment(或子類別) 、覆寫方法(*)**
  - ✓ **onCreate(…)** ：一般初始化
  - ✓ **onCreateView(inflater, container, savedState)** ：生成畫面

    *inflater.inflate(**R.layout.my_fragment_layout**, container, **false**);*
  - ✓ **onActivityCreated()**：當Activity 初始化完成後的額外動作
    - ✓ 例如：**註冊事件處理程序。**
  - ✓ **onPause()、 onResume()** ：暫停前與復原後的處理程序

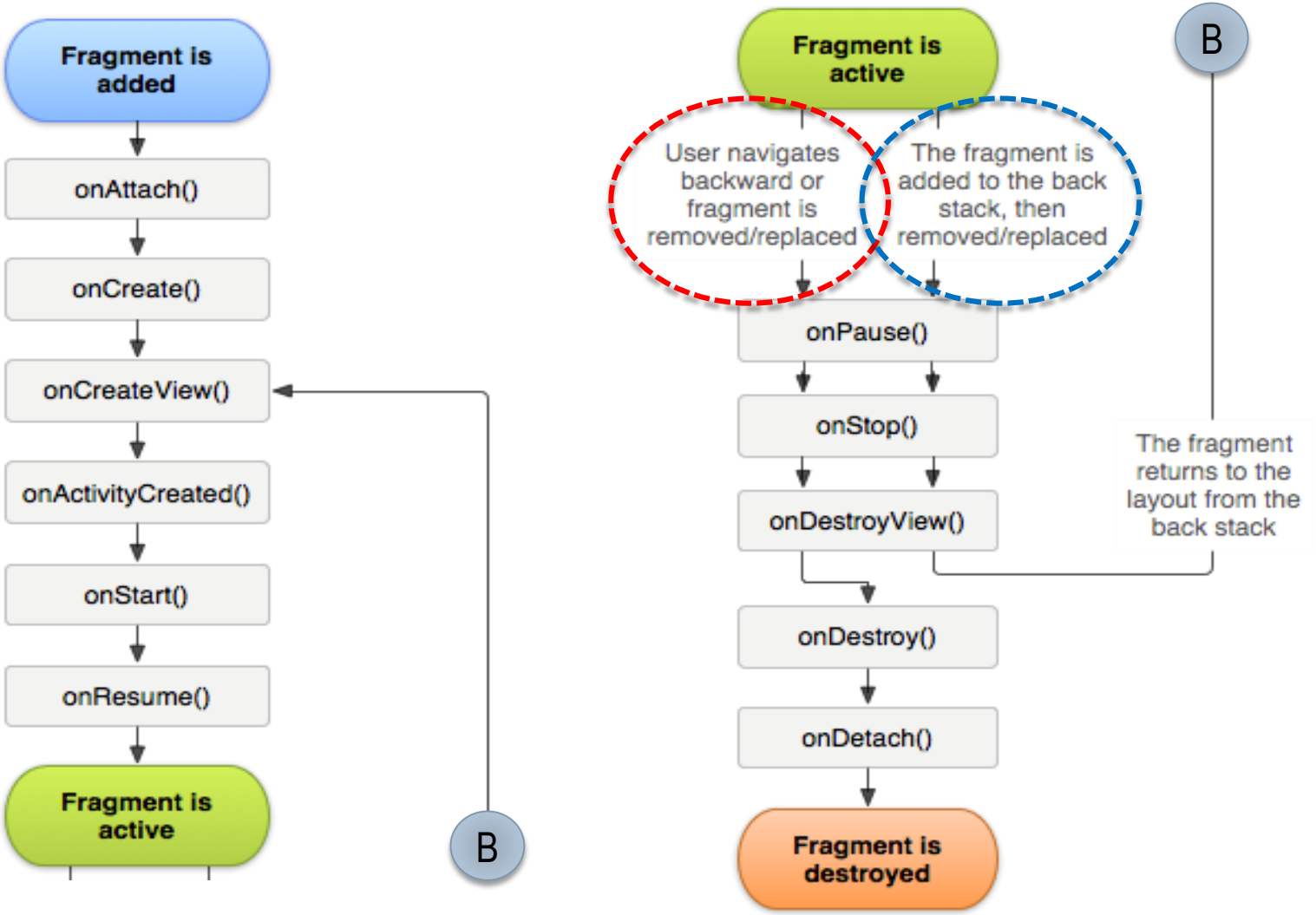
Fragment 生命週期方法的使用時機

- **onAttach(activity)**
  - Fragment 和 Host Activity 建立關聯時
- **onCreateView(inflater, container, savedState)**
  - 生成並回傳Fragment的畫面內容(通用於載入自己的佈局檔案)
- **onActivityCreated(...)**
  - Host Activity 的onCreate()執行完成後，就會接著被執行
- **onViewCreated(...)**
  - 畫面生成後，可以進行畫面中的資料更新。
- **onDestroyView()**
  - Fragment 即將被摧毀畫面內容時
- **onDetach()**
  - Fragment和Host Activity解除關聯時

Activity 與 Fragment 的依存關係 (lifecycle)



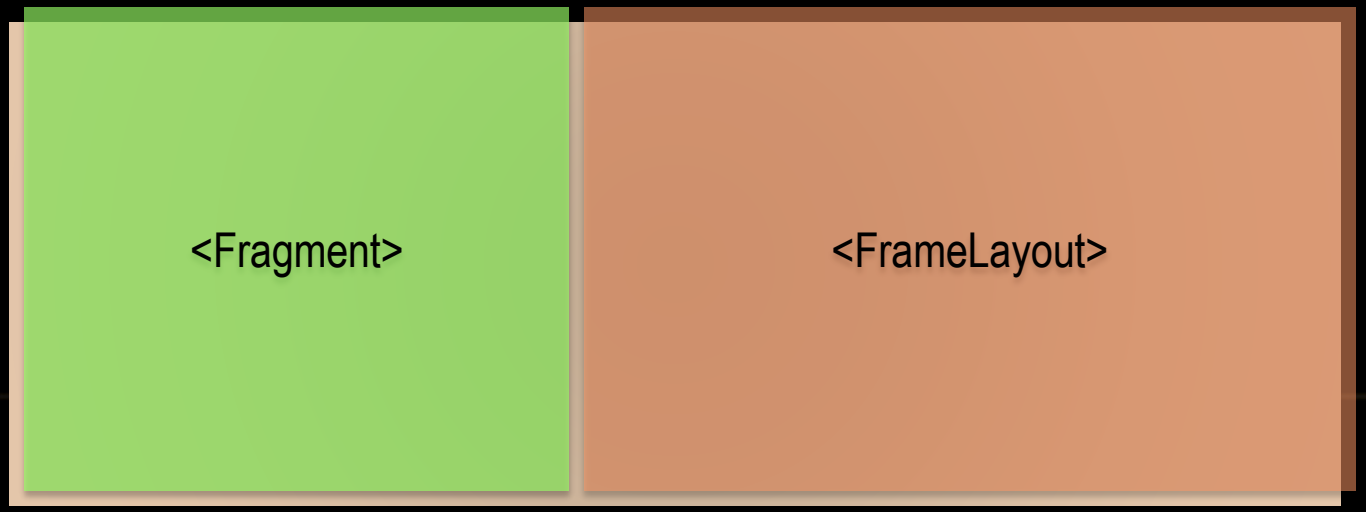
細探 Fragment 的生命週期



Fragment 的基本使用方法

- **靜態**：在布局檔案內指定**元件**為 Fragment (衍生類別)

  <**fragment** android:name="**tw.idv.james.MyFragment**" ... >
- **動態**：利用Fragment **交易控制**物件進行動態切換
  - 通常會先以 ViewGroup (如：**FrameLayout/LinearLayout** /...) 來指定Layout內的**分割區域**
  - 在程式內進行必要的內容切換。



使用 Fragment/FragmentManager 的注意事項

版本	Android 2.x	> Android 3.x
匯入模組 Import xxx;	android.support.v4.app.**FragmentActivity** android.support.v4.app.**Fragment** (須加入 v4 library jar檔，以通過編譯)	
HOST Activity X	class **X** extends **FragmentActivity**	class **X** extends **Activity**
取得 管理模組	getSupportFragmentManager()	getFragmentManager()

動態管理 Fragment 的主要觀念與方法

- **主要步驟：**
  1. 取得 FragmentManager 管理物件: getFragmentManager()
  2. 由FragmentManager物件取得**交易控制**物件 beginTransaction()
  3. 利用**交易控制物件**進行 Fragment 切換：常使用 add、replace
  4. 完成確認(交付)：commit()
- **Fragment' Transaction 主要方法：**
  - add、remove、replace:
  - attach、detach:
  - addToBackStack:
  - show、hide:
  - commit:

動態管理(增加/替換/移除) Fragment

1. 取得 Activity 的 FragmentManager **管理物件**
   - *mgr = activity.getSupportFragmentManager()*
2. 以FragmentManager啟動**交易控制物件**(FragmentTransaction)
   - *fragTrans = mgr.beginTransaction();*
3. 在ViewGroup加入Fragment ： add/replace
   - // 必要時產生新 Fragment: new XXXFragment()
   - *fragTrans.replace(container_id, fragment_obj);*
   - *fragTrans.add(contrainer_id, fragment_obj);*
4. *將該**交易**存到 **Back Stack**, 以利返回前一個fragment頁面.
   - *fragTrans.addToBackStack(null) ;*
   - *// fragTrans.popBackStack(); 模擬**Back**按鍵. 回前一畫面*
5. 設定動畫效果：*fragTrans.setTransition(...);*
6. 完成交易交付：*fragTrans.commit(); //排入UI Thread 的更新程序內*

- 在 UI Thread 執行 executePendingTransactions( )，可要求**立刻執行畫面更新**。

在Fragment 之間進行通訊

- 以 Host Activity 找元件
  - ***getActivity().findViewById(viewId)***
- 利用 FragmentManager 管理物件找到 Fragment
  - *mgr = activity.getFragmentManager()*
  - *frag = mgr.findFragmentById(fragId) // 有 UI*

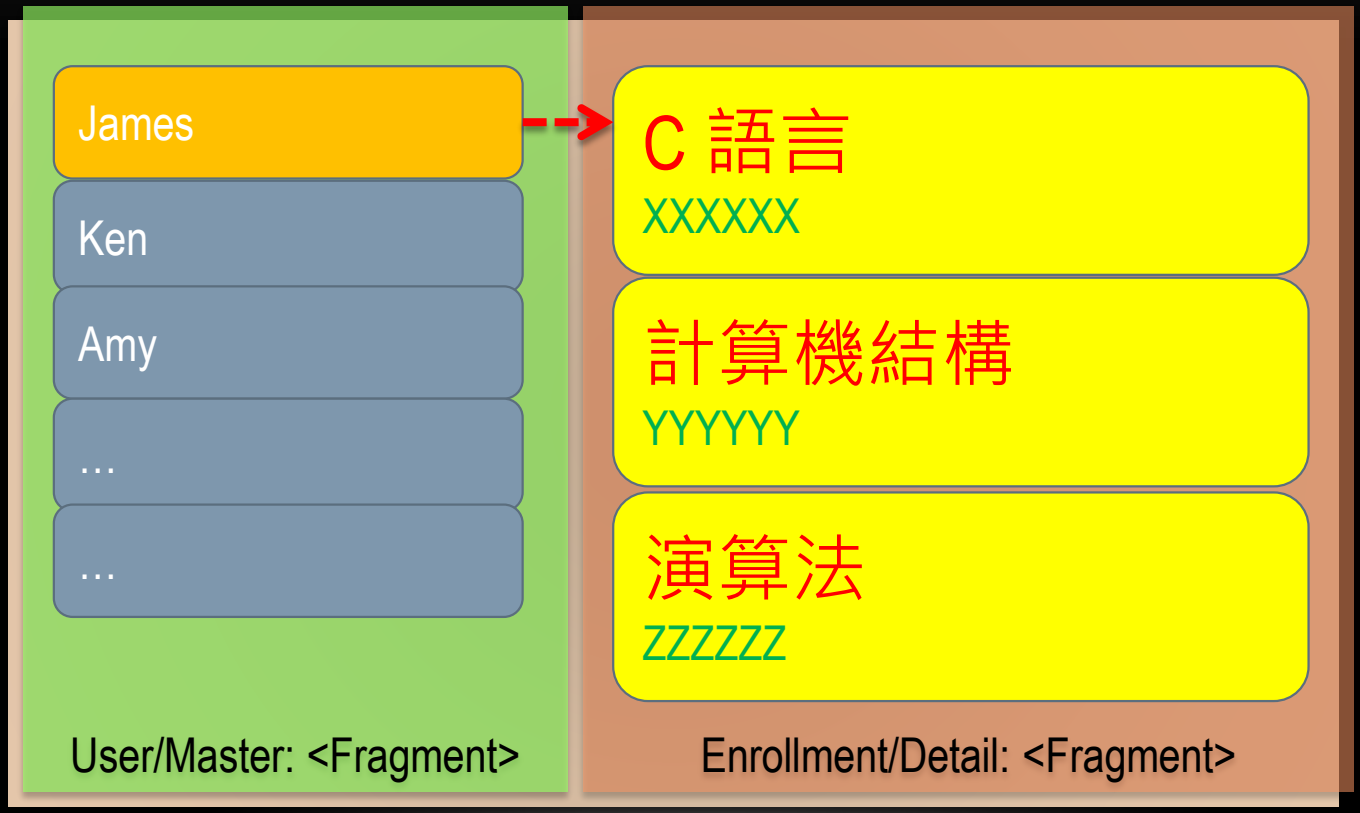
    frag = mgr.findFragmentByTag("tag") // 無 UI
  - 以 Fragment 物件來找到元件 ***frag.getView()****.findViewById(viewId)*
- Fragment 先定義 Callback Interface, 由Host Activity實作該介面；
  - 事件發生後，可由該Fragment呼叫 Host Activity 的對應方法
  - 在必要時，Host Activity 可傳遞必要參數給其他 Fragment.

Lab: 體驗 Activity 與 Fragment (略)

- **靜態 Fragment 設定(FragmentLab101)**
  - 單一Fragment 的畫面
  - 多個Fragment 的畫面
- **動態 Fragment 切換(FragmentLab102)**
  - 利用 FragmentManager 與 FragmentTransaction 動態切換顯示的內容
  - 使用情境與畫面設計：直立、橫放各有不同畫面呈現。
- **重覆使用模組的設計架構 (FragmentLab103)**
  - 使用情境與畫面設計：直立、橫放各有不同畫面呈現。
  - 重覆使用處理模組：一個區域對應一個獨立的 Java 處理程式
- **生命週期的過程 (FragmentLab104)**
  - 了解 Host Activity 和 Fragment 之間的關聯.
- **生命週期的過程 (FragmentLab105)**
  - 練習使用內建 Master-Detail 精靈產生器.

請開啟另一個 pdf 檔案

整合： SQLite + Fragment + RecyclerView



練習時間： SQLiteLab03

- 請匯入 LoginApp_New.zip
  - LoginApp702SQLiteLab03 專案
- 設計重點
  - 觀察程式基本框架, MainActivity
    - .getAccountInfo( ) : ArrayList<HashMap<String, String>>
    - .getEntrollmentBySid(int sid) : Cursor
  - 利用 2 * Fragment 切割主畫面(左:1右:2)
  - 左邊顯示 Master (ListView + ArrayList + HashMap)
  - 右邊顯示 Detail (RecyclerView + Cursor)
  - 點擊左方項目(學生) ➔ 及時更新右邊內容(選課)。
    - FragmentManager.findFragment(...)

主畫面設計 main.xml

```
1.  <!-- Master Fragment -->
2.  <fragment
3.      android:id="@+id/fragmentMaster"
4.      android:name="tw.idv.jameschen.sqlitelab.MasterFragment"
5.      android:layout_weight="1"
6.      android:layout_width="0dp"
7.      android:layout_height="fill_parent" />
8.  <!-- Detail Fragment -->
9.  <fragment
10.     android:id="@+id/fragmentDetail"
11.     android:name="tw.idv.jameschen.sqlitelab.DetailFragment"
12.     android:layout_weight="2"
13.     android:layout_width="0dp"
14.     android:layout_height="fill_parent" />
```

顯示學生名單（左）master.xml

```
1.  <!-- 學生清單 : ListView -->
2.  <ListView
3.      android:id="@+id/lvMaster"
4.      android:layout_width="match_parent"
5.      android:layout_height="match_parent" >
6.  </ListView>
```

顯示選課清單(右)　detail.xml

```
1.  <!-- 課程清單 : RecyclerView -->
2.  <android.support.v7.widget.RecyclerView
3.      android:id="@+id/rvItemList"
4.      android:layout_width="match_parent"
5.      android:layout_height="match_parent"
6.      android:scrollbarAlwaysDrawHorizontalTrack="true"
7.      android:scrollbarStyle="outsideInset"
8.      android:scrollbars="vertical" />
```

補齊 MasterFragment.java - 1

```
1.  public View onCreateView(LayoutInflater inflater, ViewGroup
    container, Bundle savedInstanceState) {
2.      //
3.      mainActivity = (MainActivity) getActivity();
4.      View view = inflater.inflate(R.layout.master, null);
5.      lvMaster = (ListView) view.findViewById(R.id.lvMaster);
6.      return view;
7.  }
```

補齊 MasterFragment.java - 2

```
1.  public void onActivityCreated(Bundle savedInstanceState) {
2.      super.onActivityCreated(savedInstanceState);
3.      accountInfos = mainActivity.getAccountInfo();
4.      accounts = new ArrayList<String>();
5.      for (int i=0; i<accountInfos.size(); i++) {
6.              HashMap<String, String> acc = accountInfos.get(i);
7.              accounts.add( acc.get("name"));
8.      }
9.      ArrayAdapter<String> adapter = new ArrayAdapter<String>(
10.             mainActivity,
11.             android.R.layout.simple_list_item_1,
12.             accounts);
13.     lvMaster.setAdapter(adapter);
14.     lvMaster.setOnItemClickListener(this);
15. }
```

補齊 MasterFragment.java - 3

```
1. public void onItemClick(AdapterView<?> parent, View
   view, int position, long id) {
2.     HashMap<String, String> item = accountInfos.get(position);
3.     int sid = Integer.parseInt(item.get("sid"));
4.     // 切換選課清單
5.     mainActivity.switchStudentEnrollment(sid); // ??
6. }
```

補齊 DetailFragment.java - 1

```
1.    public View onCreateView(LayoutInflater inflater, ViewGroup container,
2.                                        Bundle savedInstanceState) {
3.        mainActivity = (MainActivity) getActivity();
4.        View view = inflater.inflate(R.layout.detail, null);
5.        adapter = new EnrollmentRcViewAdapter(mainActivity, inflater, null );
6.        rvDetailList = (RecyclerView) view.findViewById(R.id.rvItemList);
7.        LinearLayoutManager layoutManager1 =
                  new LinearLayoutManager(
                        mainActivity, LinearLayoutManager.VERTICAL, false);
8.        rvDetailList.setLayoutManager(layoutManager1);
9.        rvDetailList..setAdapter(adapter);
10.       return view;
11.   }
```

補齊 DetailFragment.java - 2

```
1. Cursor cursor = null;
2. public void switchStudentEnrollment(int sid) {
3.         if (cursor != null)
4.                 cursor.close();
5.         cursor = mainActivity.getEntrollmentBySid(sid);
6.         adapter.changeCursor(cursor);
7. }
```

補齊 MainActivity.java

```
1. public void switchStudentEnrollment(int sid) {
2.     // (*) 利用學號切換選課清單
3.     DetailFragment fragDetail = (DetailFragment)
4.         getFragmentManager().findFragmentById(R.id.fragmentDetail);
5.     fragDetail.switchStudentEnrollment(sid);
6. }
```

網路連線與資料交換

Socket, HTTP

Android 網路連線的基本資訊

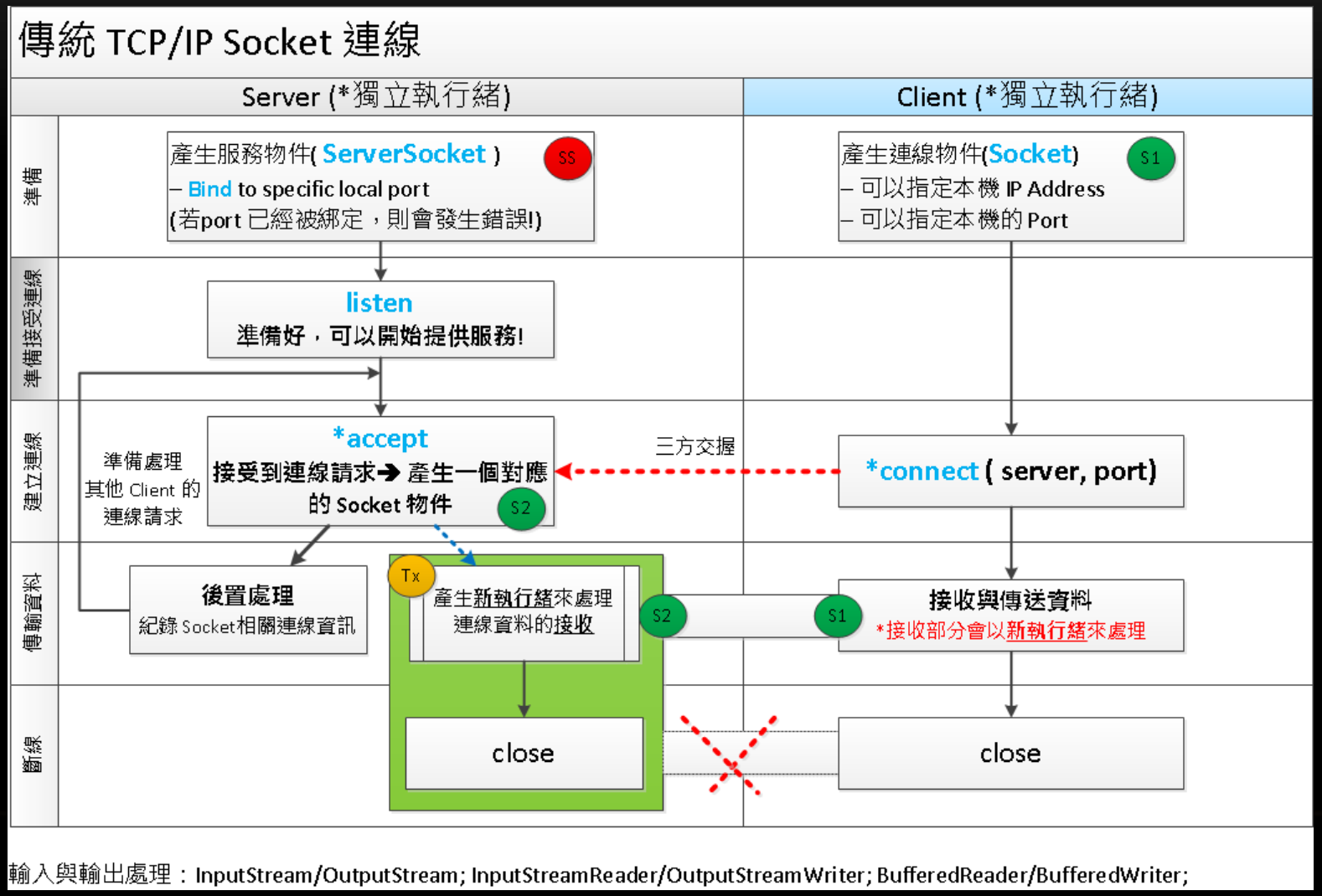
- **Socket Programming**
  - **TCP**: Socket/ServerSocket
  - **UDP**: DatagramSocket/DatagramPacket
- **Customized Components**
  - **HTTP Client**: HttpURLConnection(light-weight), DefaultHttpClient(heavy)
  - **Others**…
- **Common Topics:**
  - **I/O**: InputStream/OutputStream; Reader/Writer/…
  - **ANR**: Multi-Thread, Updating UI, …
  - **JSON/XML**: JSONObject(…) , JSONArray(…) …

http://android-developers.blogspot.tw/2011/09/androids-http-clients.html

利用網路交換資料：Socket? HTTP?

- 原生TCP/IP的方法: **Socket**
  - 以Thread 建立連線 new Socket(ServerIpAddr, Port);
  - 取得輸出/輸入串流、進行資料的傳送與接收(Thread)
  - 關閉連線
- 利用既有協定 (**HTTP**, FTP, …) : 省事!
  - 建立連線(HTTP): URL, URLConnection/HttpURLConnection/…
    - GET? POST?
  - 取得輸出/輸入串流：進行資料的傳送與接收(Thread)
  - 關閉連線

TCP/IP -- Socket/ServerSocket



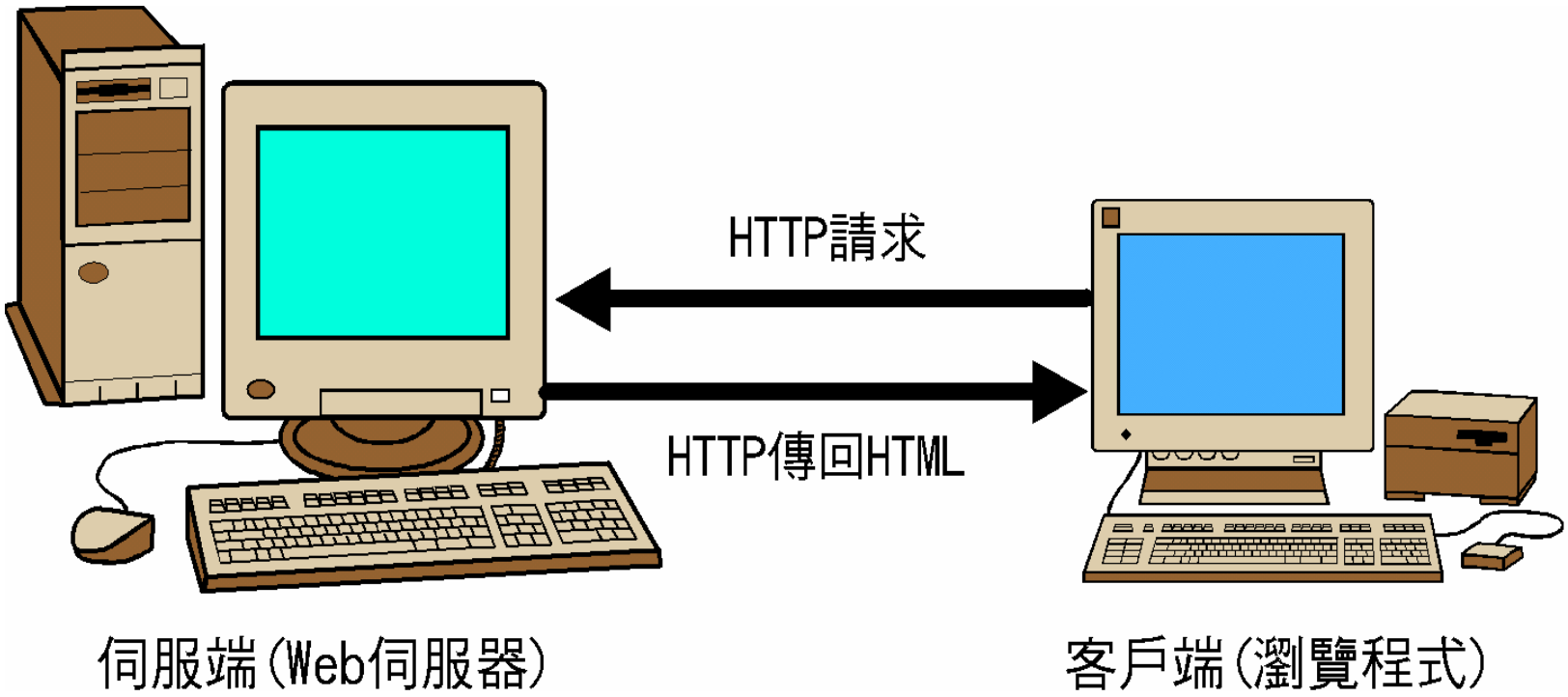
練習實務開發內容

- **需加入網路的權限使用宣告**

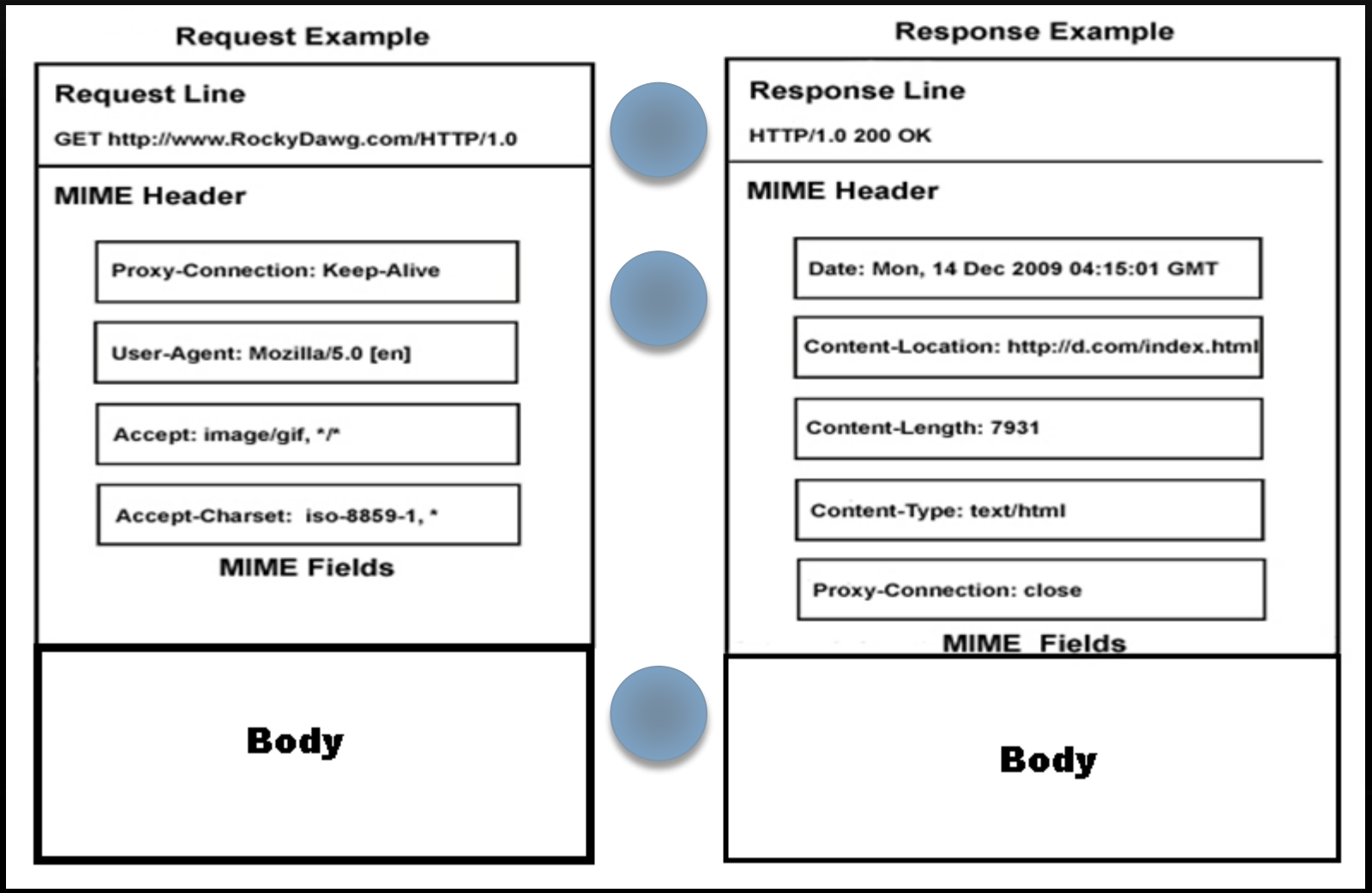
  android.permission.INTERNET

- **開發 EchoClient**
  - 建立 Socket 連線到 Echo Server,
  - **練習傳送與接收資料**
- **開發 ChatClient**
  - 建立 Socket 連線到 Chat Server (ChatServer.jar),
  - 練習傳送與接收聊天室的資料
- **開發 ChatServer**
  - 開發一個【多對一】的服務端程式
  - ?? 其他設備(手機)如何跟你連線??

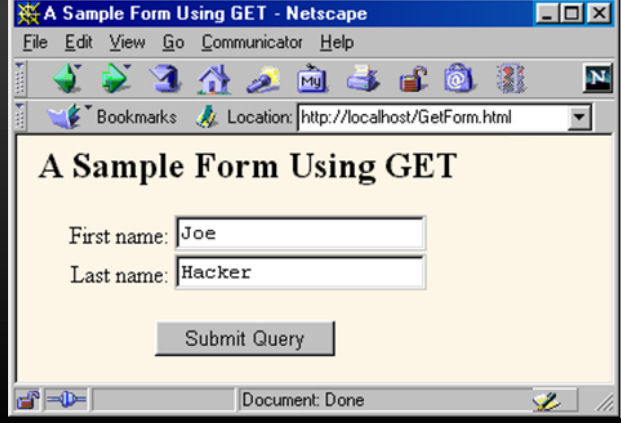
WWW / HTTP Protocol : Request+Response



HTTP Request/Response 封包格式



表單: Posting Data with GET

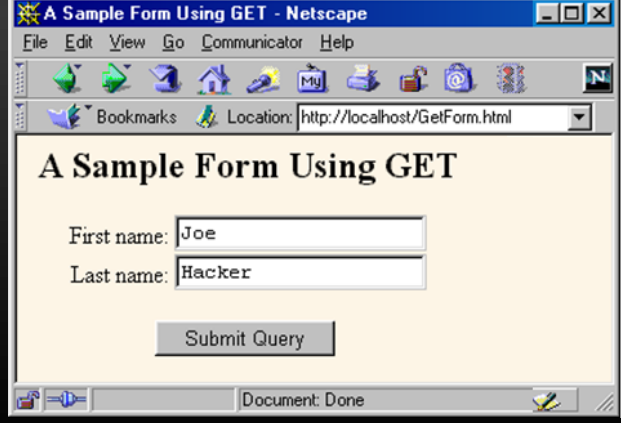



```
...
<BODY BGCOLOR="#FDF5E6">
<H2 ALIGN="CENTER">A Sample Form Using GET</H2>

<FORM ACTION=http://somehost/SomeProgram METHOD="GET">
  <CENTER>
  First name:
  <INPUT TYPE="TEXT" NAME="firstName" VALUE="Joe"><BR>
  Last name:
  <INPUT TYPE="TEXT" NAME="lastName" VALUE="Hacker"><P>
  <INPUT TYPE="SUBMIT">
  </CENTER>
</FORM>

</BODY></HTML>
```

表單: Posting Data with POST




```
...
<BODY BGCOLOR="#FDF5E6">
<H2 ALIGN="CENTER">A Sample Form Using POST</H2>

<FORM ACTION=http://somehost/SomeProgram METHOD="POST">
  <CENTER>
  First name:
  <INPUT TYPE="TEXT" NAME="firstName" VALUE="Joe"><BR>
  Last name:
  <INPUT TYPE="TEXT" NAME="lastName" VALUE="Hacker"><P>
  <INPUT TYPE="SUBMIT">
  </CENTER>
</FORM>

</BODY></HTML>
```

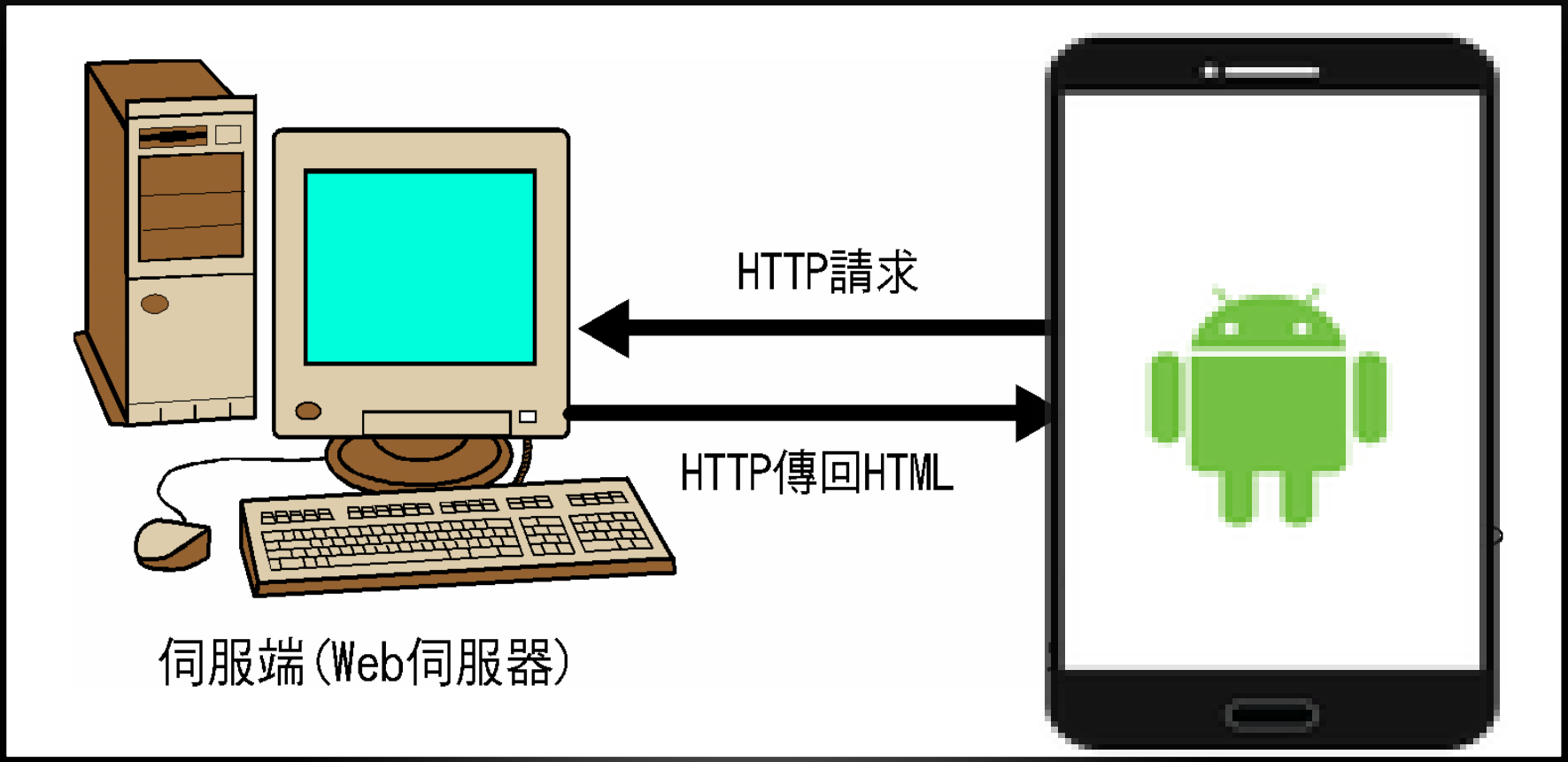
GET 與 POST 傳送內容的主要差異

```
GET /SomeProgram?firstName=Joe&lastName=Hacker HTTP/1.0
Referer: http://localhost/GetForm.html
Connection: Keep-Alive
User-Agent: Mozilla/4.7 [en] (Win98; U)
Host: localhost:8088
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, image/png, */*
Accept-Encoding: gzip
Accept-Language: en
Accept-Charset: iso-8859-1,*,utf-8
```

```
POST /SomeProgram HTTP/1.0
Referer: http://localhost/PostForm.html
Connection: Keep-Alive
User-Agent: Mozilla/4.7 [en] (Win98; U)
Host: localhost:8088
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, image/png, */*
Accept-Encoding: gzip
Accept-Language: en
Accept-Charset: iso-8859-1,*,utf-8
Content-type: application/x-www-form-urlencoded
Content-length: 29

firstName=Joe&lastName=Hacker
```

利用 HTTP 開發 App網路應用



使用 HTTP 連線的可能方案 – part 1

- **HttpClient** (Apache) : SDK 23(含) 以後不再支援

  需修改 build.gradle, 加入設定(紅色)

```
android {
    useLibrary 'org.apache.http.legacy'
}
```

- **URL**/**URLConnection**/**HttpURLConnection** : SDK >= 23
- **Volley**/**RequestQueue**/**XXXRequest** 輕量級 HTTP Client
  - 若需大量下載資料, 則建議使用 **DownloadManager**

```
dependencies {
    compile 'com.android.volley:volley:1.1.0'
}
```

- 直接下載 HttpClient.jar 套件包 http://hc.apache.org/downloads.cgi
- https://developer.android.com/training/volley/index.html

使用 HTTP 連線的可能方案 – part 2

- **OkHttp** http://square.github.io/okhttp/

```
dependencies {
    compile 'com.squareup.okhttp3:okhttp:3.10.0'
}
```

- **NoHttp** https://github.com/yanzhenjie/NoHttp
- **OkGo** https://github.com/jeasonlzy/okhttp-OkGo
  - 支援 RxJava/RxJava2 : Reactive Extensions for the JVM

RxJava ➔ https://yongjhih.gitbooks.io/feed/content/RxJava.html

HttpURLConnection/HttpsURLConnection

- **建立HTTP連線**

```
URL url = new URL("your_URL");
HttpURLConnection conn=(HttpURLConnection)url.openConnection();
```

- **設定必要參數**(Header meta-data /content-type, cookie, …)

```
conn.setRequestMethod("GET"); // POST ➔ 上傳
conn.setRequestProperty("Accept-Encoding", "identity");
conn.setDoInput(true);
// 預設上傳為關閉, 開啟上傳: conn.setDoOutput(true)
```

- **建立連線、取得回應**(Header meta-data/content-type, length)

```
conn.connect();          // 顯式呼叫，立刻建立連線。可不呼叫此方法!
conn.getResponseCode( ); conn.getContentLength( ); …
conn.getHeaderField(String key) …
```

- **處理輸出與輸入： I/O Stream**

```
InputStream is = conn.getInputStream();
OutputStream os = conn.getOutputStream();
```

- **關閉連線**

```
conn.disconnect();
```

URL

HttpURLConnection

InputStream

OutputStream

URLEncoder

HttpResponse

http://developer.android.com/reference/java/net/HttpURLConnection.html
https://docs.oracle.com/javase/tutorial/networking/urls/connecting.html

練習時間： LoginApp800 (遠端檢查帳密)

- 請匯入 LoginApp_New.zip
  - LoginApp800Remote 專案
- 修改 MainActivity 的帳密檢查
  - 無法**立刻**知道結果！
  - 將後續動作（切換畫面）置入 onPostExecute(...)
- 完成 LoginAsyncTask 設計
  - 非同步處理技巧 (**Thread**)
  - **前景**與**背景**程式碼**集中管理**
- 修改 *AndroidManifest.xml*
  - *uses_permission* → ***INTERNET***

測試網址 http://163.17.83.53/demo01/checkAccount.php?name=amy&pw=111222

LoginAsyncTask 程式框架

```
1. class LoginAsyncTask extends AsyncTask<String, Integer, Boolean> {
2.     protected void onPreExecute() {
3.         super.onPreExecute();        // 若執行時間無法確定，可以顯示 ProgressDailog
4.     }
5.     protected void onPostExecute(Boolean result) {
6.         super.onPostExecute(result); // 若執行前有顯示 ProgressDailog, 此時要取消！
7.         // 根據 result 處理後續動作!!
8.     }
9.     protected void onProgressUpdate(Integer... values) {
10.        super.onProgressUpdate(values); // 根據 values 更新 ProgressDailog 進度值
11.    }
12.    protected Boolean doInBackground(String... params) {
13.        acc = params[0];                    //取得帳號與密碼
14.        String pw = params[1];
15.        // 進行遠程登入，最後將結果回傳 !!
16.        // 若執行工作較久（ex: 迴圈），可以適時地更新進度: publishProgress( p );
17.        return true;
18.    }
19.}
```

例：利用 HTTP/GET 進行帳密確認

```
1.  URL url = new URL( "http://....?acc=james&pw=123");
2.  URLConnection conn = url.openConnection();
3.  String encoding = conn.getContentEncoding();
4.  InputStream is = (InputStream)conn.getContent();
5.  BufferedReader br = new BufferedReader(
6.                          new InputStreamReader( is, "UTF-8" ));
7.  String result = br.readLine();
8.  // 判別 result 內容
9.  conn.close();
```

範例：ProgressDialog 的簡易使用

```
1.  ProgressDialog pDialog = new ProgressDialog(context);
2.  pDialog.setMax(100);
3.  pDialog.setProgressStyle(ProgressDialog.STYLE_HORIZONTAL);
4.  pDialog.setMessage(downloadMsg);
5.  pDialog.setCanceledOnTouchOutside(false);
6.  pDialog.setCancelable(true);
7.  pDialog.setButton(
8.          ProgressDialog.BUTTON_NEGATIVE,
9.          "Cancel",
10.         new DialogInterface.OnClickListener() {
11.             public void onClick(DialogInterface dialog, int which) {
12.                 // cancelJob();
13.             }
14.         });
15. pDialog.show();
```

網路資料處理之相關議題

JSON、SQLite

XML - a Real Case

```
1. <?xml version="1.0" encoding="UTF-8"?>
2. <menu id="file" value="File">
3.     <popup>
4.         <menuitem value="New" onclick="CreateNewDoc()" />
5.         <menuitem value="Open" onclick="OpenDoc()" />
6.          <menuitem value="Close" onclick="CloseDoc()" />
7.     </popup>
8. </menu>
```

如何處理與解析 XML

- **相關元件與資訊**
  - DOM: Document Object Model
    - DocumentBuilderFactory/DocumentBuilder + Document/Element/NodeList
  - **SAX**: Simple API for XML
    - http://en.wikipedia.org/wiki/Simple_API_for_XML
    - SAXParserFactory/SAXParser + XMLReader/DefaultHandler
    - android.sax package
  - **XmlPull**: SAX-like
    - http://www.xmlpull.org
    - XmlPullParserFactory/XmlPullParser
  - 動態產生簡易的 XML 檔案
    - Xml/XmlSerializer/…

http://www.anddev.org/parsing_xml_from_the_net_-_using_the_saxparser-t353.html

JSON –JSONObject, JSONArray

```
1.  {
2.    "menu": {
3.      "id": "file",
4.      "value": "File",
5.      "popup": {
6.          "menuitem": [
7.              {"value": "New", "onclick": "CreateNewDoc()"},
8.              {"value": "Open", "onclick": "OpenDoc()"},
9.              {"value": "Close", "onclick": "CloseDoc()"}
10.           ]
11.       }
12.   }
13. }
```

http://json.parser.online.fr/

JSONObject 的重要方法

- **取得屬性的數量**

```
int num = json.length();
```

- **取得 Keys (Iterator)**

```
Iterator<String> keys = json.keys();
```

- **利用 key 取得 value**

```
String value = json.getString( "STR" );
```

- **放入資料 < key, value>**

```
json.put( "STR", "001" );
json.put( "extras", JSONObject.NULL); // null value
```

- **移除資料 (key)**

```
json.remove( "extras");
```

- **轉為字串**

```
jasonArray.toString();
```

JSONArray的重要方法

- **取得元素/JSONObject的數量**

```
int numOfItems = jsonArray.length();
```

- **取得資料值/JSONObject**

```
JSONObject json = jsonArray.getJSONObject(i);
int num = jsonArray.getInt(i); // for 基本資料型別
```

- **放入資料值/JSONObject (json)**

```
jsonArray.put(value/json );              // append
jsonArray.put( index, value/json );    // replace?
```

- **移除特定位置的資料**

```
jsonArray.remove( index );
```

- **轉為字串**

```
jasonArray.toString();
```

使用 Apache **HttpClient** 的程式框架

- **產生 HTTP Client 連線物件**

```
HttpClient client = new DefaultHttpClient();
```

- **產生 GET/POST 物件與資料(Entity)**

```
HttpGet get = new HttpGet("URL?name=abc&pw=123");
HttpPost post= new HttpPost("URL");
ArrayList<NameValuePair> dataPair = new ArrayList<NameValuePair>();
dataPair.add( new BasicNameValuePair("name", "abc") );
dataPair.add( new BasicNameValuePair("pw", "123") );
post.setEntity(new UrlEncodedFormEntity(dataPair));
```

- **執行動作(Request)並等待回傳結果(Response)**

```
HttpRespone res = client.execute(post);    // or get 物件
```

- **讀取回傳狀態與結果**

```
int statusCode = res.getStatusLine().getStatusCode();
HttpEntity entity = res.getEntity();
// String result = EntityUtils.toString(entity);
BufferedReader br = new BufferedReader( new InputStreamReader( entity.getContent() ) );
```

HttpClient

HttpPost/Get

NameValuePair

HttpEntity/EntityUtils

StringEntity

BinaryEntity

UrlEncoder

HttpResponse

InputStream

傳送 JSON 資料➔ POST/HttpEntity

- **產生 JSON 物件/陣列，並放入資料**

```
JSONObject json = new JSONObject(); // or JSONArray(…);
json.put("account", "jameschen");
json.put("email", "jameschen@abc");
```

- 將 json 資料轉為 HttpEntity (字串, 二進位)

```
StringEntity entity = new StringEntity( json.toString());
ByteArrayEntity entity = new ByteArrayEntity(jason.toString().getBytes());//中文
```

- **設定 HTTP 上傳資料格式**

```
entity.setContentType("application/json;charset=UTF-8");
entity.setContentEncoding(
        new BasicHeader(HTTP.CONTENT_TYPE,"application/json;charset=UTF-8"));
```

- **將資料塞入 HttpPost 物件內**

```
post.setEntity(entity);
```

解析 收到的 JSON 資料

- **從 HTTP 連線取得一筆 JSON 資料(物件、陣列)**

```
String jsonString = …..;
```

- **將字串解析為 JSON 物件/陣列**

```
JSONObject json = new JSONObject( jsonString );
```

或

```
JSONArray jsonArray = new JSONArray( jsonString );
int numOfObjects = jsonArray.length();
JSONObject json = jsonArray.getJSONObject(i);
```

將 JSONObject (Record) 新增到資料表

```
1.  ContentValues cv = new ContentValues();
2.  JSONObject rec = newRecords.getJSONObject(i); // Array
3.  Iterator<String> keys = rec.keys();
4.  // Insert the record into some Table
5.  cv.clear();
6.  for (int j = 0; j < rec.length(); j++) {
7.      String key = keys.next();
8.      cv.put(key, rec.getString(key));
9.  }
10. long rowid = db.insertOrThrow( tablename, "", cv);
```

將資料 (Record) 轉為 JSONObject

```
1.JSONObject jsonObject = new JSONObject();
2.for (int i = 0; i < cursor.getColumnCount(); i++) {
3.     if (cursor.getColumnName(i) != null) {
4.         switch ( cursor.getType(i) ) {
5.             case Cursor.FIELD_TYPE_STRING:
6.                 jsonObject.put( cursor.getColumnName(i), cursor.getString(i)); break;
7.             case Cursor.FIELD_TYPE_INTEGER:
8.                 jsonObject.put( cursor.getColumnName(i), cursor.getInt(i));  break;
9.             case Cursor.FIELD_TYPE_FLOAT:
10.                jsonObject.put( cursor.getColumnName(i), cursor.getFloat(i)); break;
11.            case Cursor.FIELD_TYPE_NULL:
12.                jsonObject.put( cursor.getColumnName(i), JSONObject.NULL); break;
13.            default:
14.                jsonObject.put( cursor.getColumnName(i), cursor.getString(i)); break;
15.        }
16.    }
17.}
18.// 回傳 JSONObject 做進一步處理!
```

練習時間： LoginApp801 HTTP/JSON

- 以特定帳密登入遠端 (HttpURLConnection, LoginApp800)
- 利用 AsyncTask 將遠端資料 (todolist) 同步到本機 SQLite 資料表 (參考下一頁定義)
  - **下載與同步資料 : Boolean doInBackground(…)**
    - downloadData(url) : 利用 HttpClient 取得 JSON 封包
    - **insertData(json)** : 將 JSONArray 轉為 ContentValues、利用 SQLiteDatabase.insert(…) 將資料新增到表格內(todolist)
  - **顯示結果 : onPostExecute(Boolean result)**

http://163.17.83.53/demo01/getTodoList.php?sid=1

TodoList 表格定義、JSON 封包

Name	Type
tid	INTEGER
todo	VARCHAR2
due	DATE
sid	INTEGER

1


```
[
    {
        "tid":"1",
        "todo":"TODO-011",
        "due":"2015-08-26",
        "sid":"1"
    },
    {
        "tid":"2",
        "todo":"TODO-012",
        "due":"2015-08-29",
        "sid":"1"
    },
    {
        "tid":"3",
        "todo":"TODO-013",
        "due":"2015-09-01",
        "sid":"1"
    }
]
```

補充：SQLite 交易控制的基本框架

```
try {
      // 開始進入交易模式
      db.beginTransaction();
      // 處理資料維護
      db.execSQL("SQL_Statement");
      // 設定交易成功!!
      db.setTransactionSuccessful();   // Clear status ; 否則 rollback!
}
catch (Exception ex) {
      // 例外處理
}
finally {
     // 結束交易
     db.endTransaction();
}
```

資料處理(交換)

- SharedPreferences
- File I/O
- SQLite Database
- Handler + Message
- Networking: Socket, HTTP,
- ContentProvider + ContentResolver
- BroadcastReceiver + sendBroadcast(…)
- Service(Binder) + Parcel
- Service(Binder) + Handler + Messenger
- …

二、硬體與感知應用

- 各種感測器的使用
- GPS 與 Google Maps

三、社群與雲端整合開發

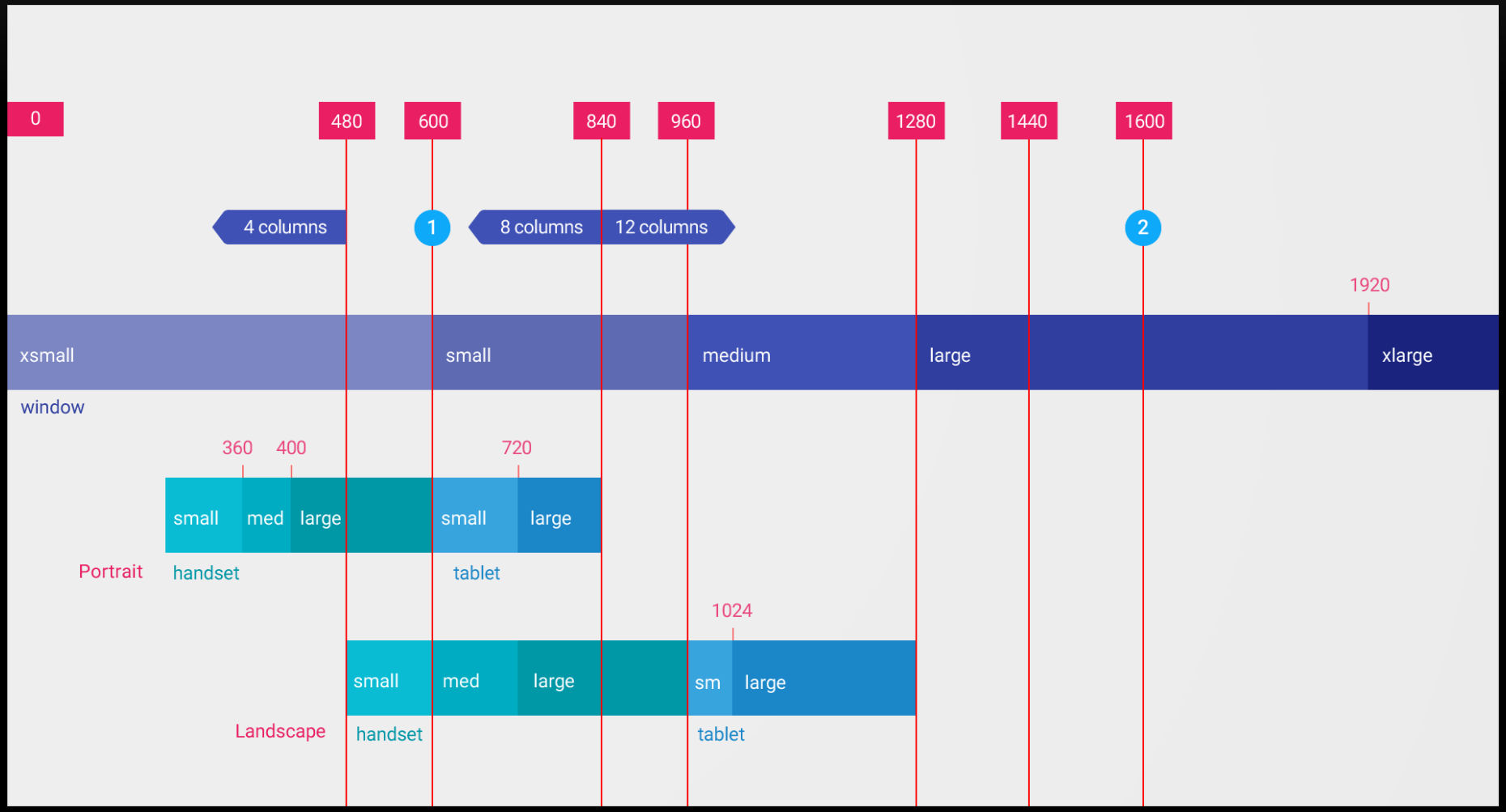
- Google Cloud Messaging (GCM)
- Google Calendar, Google Drive,
- FB, Line, Youtube

Material Design -- Android 5.0

- Material Design 新增很多內容，主要分為四部份：
  - 主題和佈局：自訂狀態列、導覽條..
  - 視圖和陰影： View 加入高度(z)參數，決定陰影大小
  - UI 控制項：RecyclerView, CardView
  - 動畫：增加不少新的動畫效果

http://www.google.com/design/spec/material-design/introduction.html

螢幕解析度之關係



http://www.google.com/design/spec/layout/adaptive-ui.html#adaptive-ui-breakpoints