



## **THIẾT KẾ GIAO DIỆN ANDROID**

### **BÀI 3: ADAPTER & ADAPTERVIEW**


- ⊙ Kết thúc bài học này bạn có khả năng
  - ⊙ Hiểu được Adapter và AdapterView
  - ⊙ Biết được cách chọn lựa AdapterView cho phù hợp
  - ⊙ Kế thừa và tối ưu hóa Adapter



## Phần I: Giới thiệu về Adapter & AdapterView

 A Cây kế thừa AdapterView

 B Một số AdapterView

 C Cây kế thừa Adapter

 D Một số Adapter, Datasource

 E Dùng adapter đơn giản

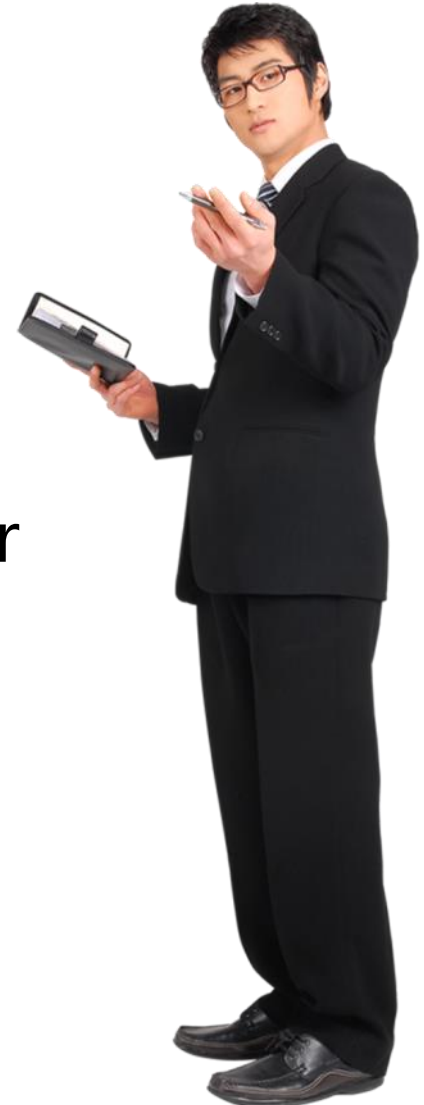
## Phần II: Tùy biến và tối ưu hóa Adapter

 D Tại sao phải tùy biến và tối ưu

 E Tùy biến adapter

 F Tối ưu adapter

 G Bắt sự kiện



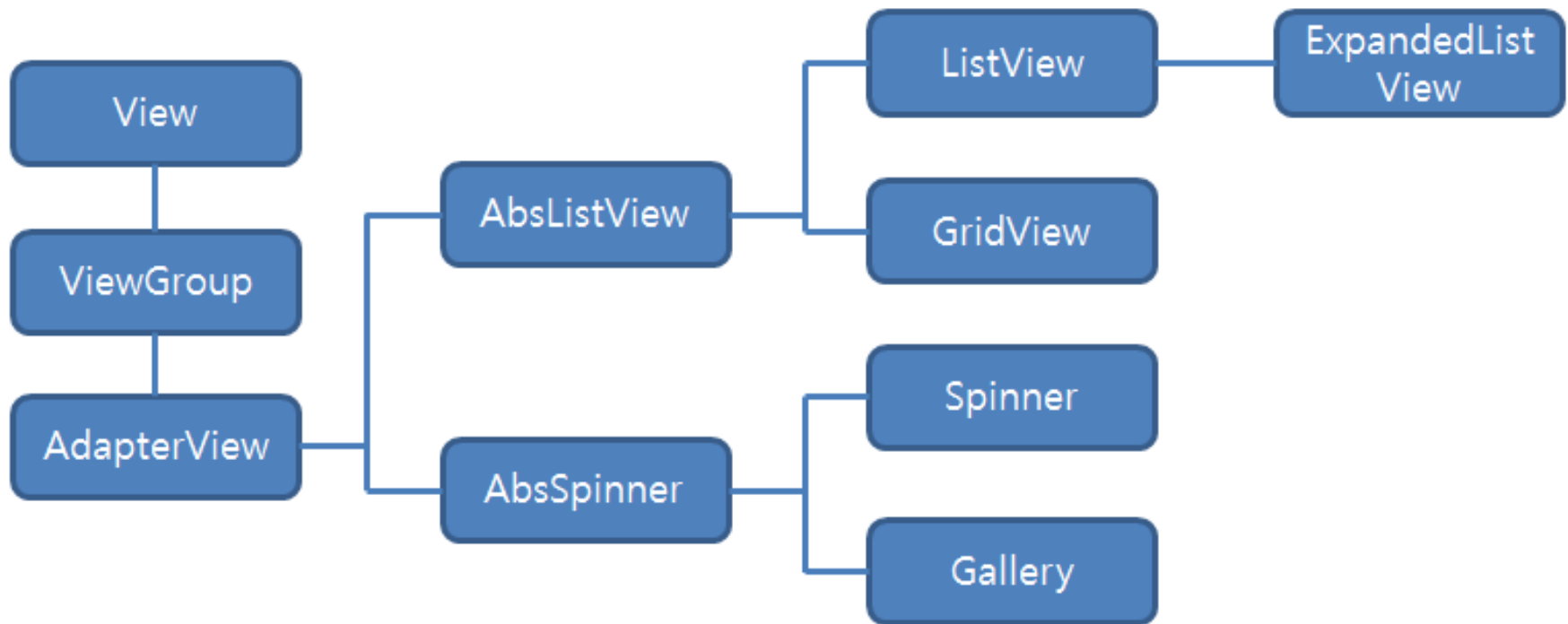


# **ADAPTER & ADAPTERVIEW**

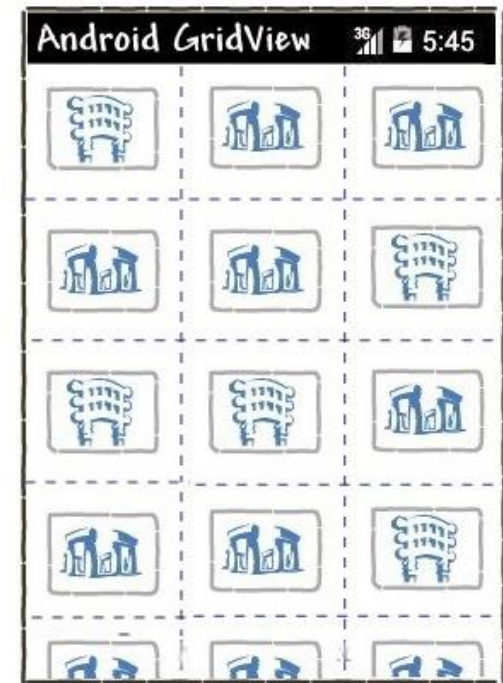
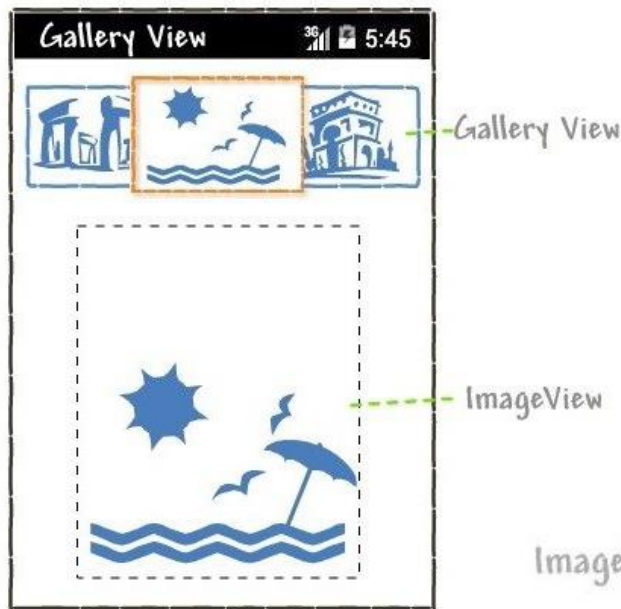
---

## **PHẦN 1: ADAPTER & ADAPTERVIEW**

- ❑ AdapterView là các view dùng để hiển thị tập hợp nhiều item trên một view.

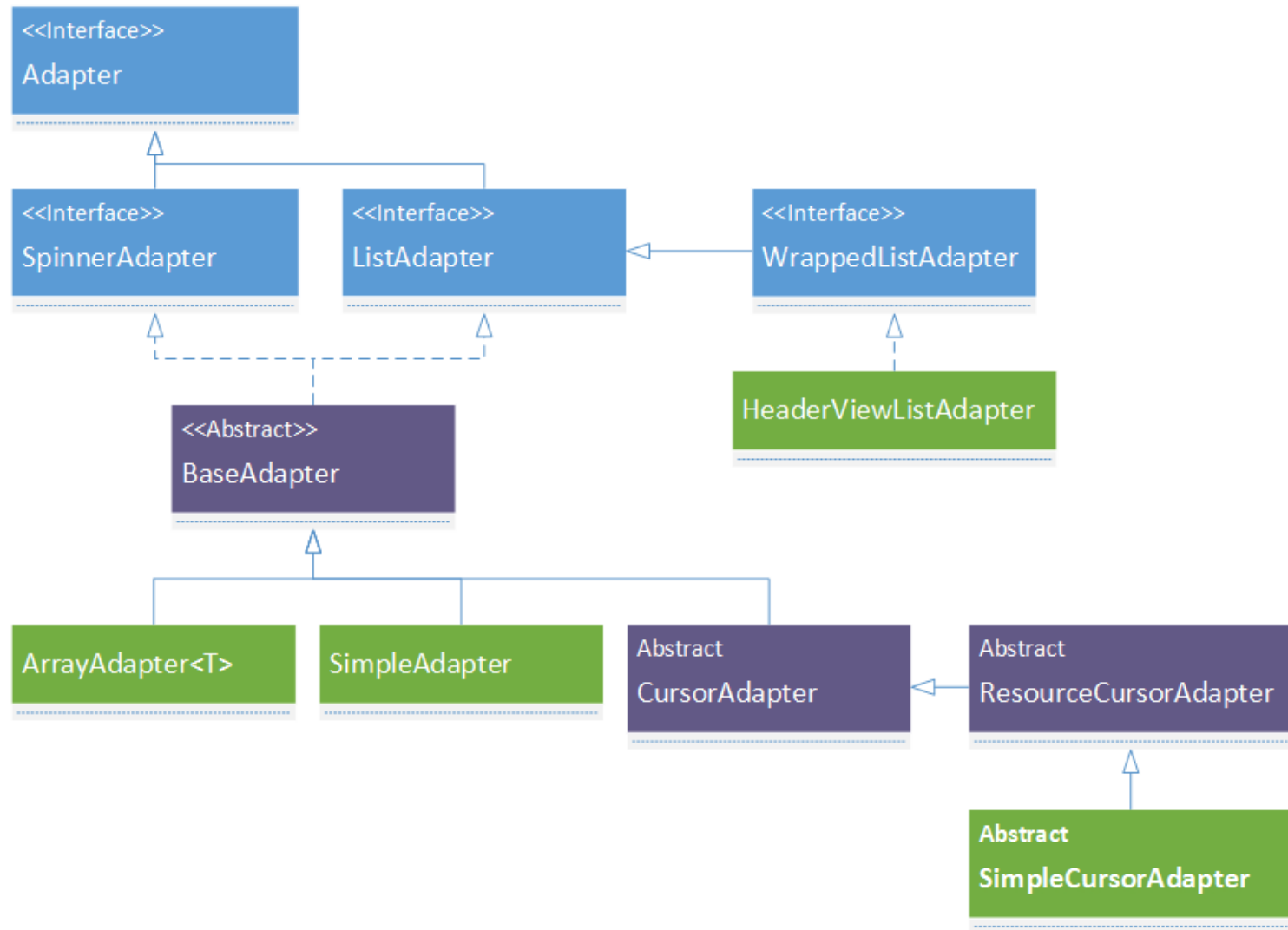


- Tùy theo hình dạng hiển thị mà ta chọn view cho phù hợp .

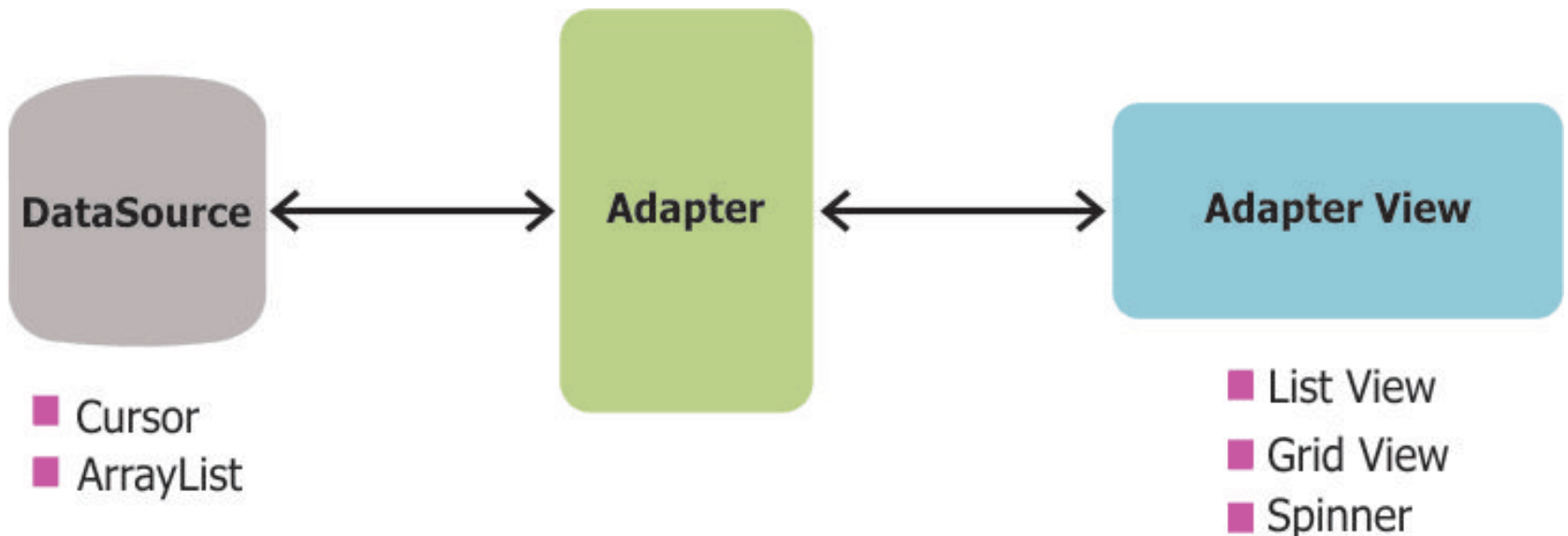


# GIỚI THIỆU VỀ ADAPTER & ADAPTERVIEW

- ❑ Adapter dùng để đổ dữ liệu lên Adapter. Tùy theo loại dữ liệu mà ta chọn lựa adapter cho phù hợp



- ❑ Adapter là cầu nối giữa DataSource(nguồn dữ liệu) và AdapterView (view hiện dữ liệu)
- ❑ DataSource có thể là: Cursor, Array, ArrayList.....
- ❑ AdapterView có thể là: ListView, GridView, Spinner, Gallery, AutoCompleteTextView.....





## ❑ Các AdapterView thông dụng

jay@gmail.com

Home

Home

Work

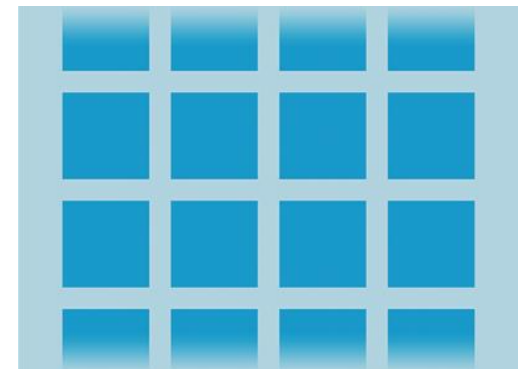
Other

Custom

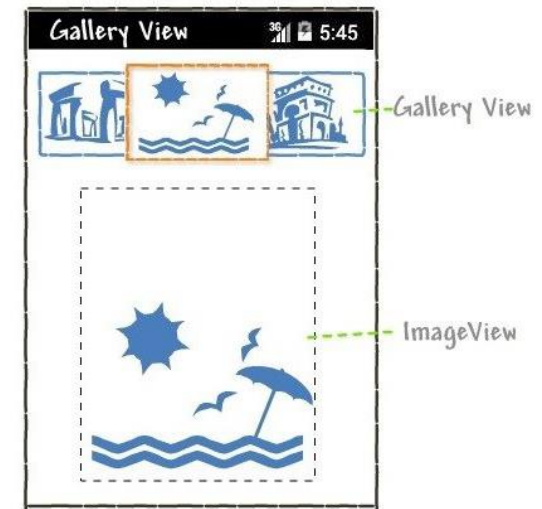
❖ Spinner : cho phép chọn 1 giá trị trong 1 tập hợp. Chọn vào 1 spinner nó sẽ hiện thị như một dropdown menu cho phép ta chọn 1 mục con

❖ ListView: hiện thị một danh sách có thể cuộn xuống

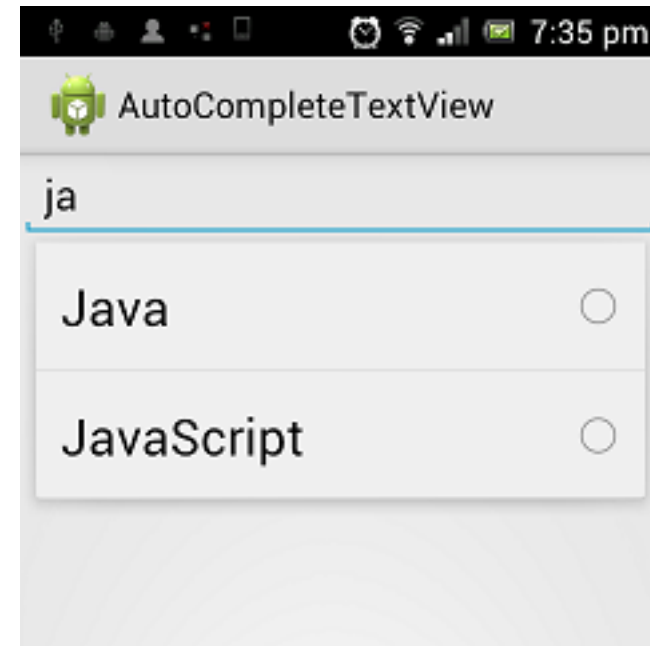
❖ GridView: hiện thị danh sách theo kiểu ô lưới 2 chiều



- ❑ Gallery: hiển thị danh sách dạng center-locked, dùng cuộn ngang để cuộn danh sách. Bị thay thế ở bản API level 16



- ❑ AutocompleteTextView: như textview nhưng khi nhập liệu sẽ được gợi ý. Các gợi ý được lấy từ nguồn dữ liệu và hiển thị kiểu dropdown



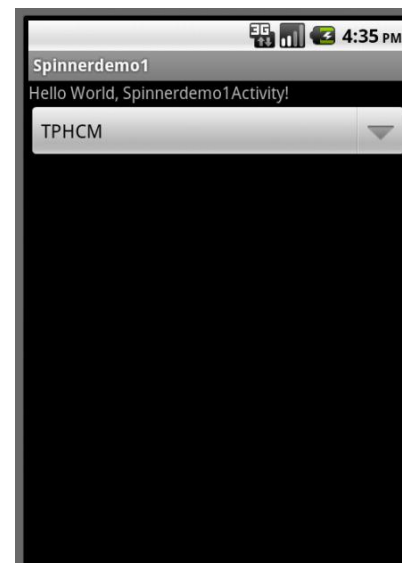
- ❑ Khi đổ dữ liệu lên AdapterView nếu dữ liệu đơn giản ta có thể dùng trực tiếp các class Adapter. Vd: ArrayAdapter, SimpleAdapter, SimpleCursorAdapter....
- ❑ Nếu dữ liệu phức tạp hoặc cần tối ưu hóa về hiệu suất ta cần tạo ra class mới kế thừa từ các class Adapter và tự triển khai nội dung. Vd: kế thừa từ ArrayAdapter, BaseAdapter....

❑ Trên giao diện kéo vào spinner, trong java ánh xạ và

```
String[] item={"saigon","vung tau", "dalat","hue"};  
  
ArrayAdapter<String> adapter=new ArrayAdapter<String>(this,  
    android.R.layout.simple_spinner_item,item);  
  
adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);  
  
spinner.setAdapter(adapter);
```

❑ ArrayAdapter nhận 3 đối số:

- ❖ Ds1: context
- ❖ Ds2: layout cho 1 item, có android ở trước R là đang dùng tài nguyên có sẵn của android.
- ❖ Ds3: nguồn dữ liệu (ở đây là một array)



- Trên giao diện kéo vào listview, trong java ánh xạ listview
- Tạo 1 file layout thiết kế:



- Tạo dữ liệu:

```
List<HashMap<String,Object>> ds=
```

```
new ArrayList<HashMap<String,Object>>();
```

```
HashMap<String,Object> hp=new HashMap<String,Object>();
```

```
hp.put("ten", "hancock");
```

```
hp.put("hình", R.drawable.hancock);
```

```
hp.put("tuổi", "18");
```

```
ds.add(hp);
```

```
hp=new HashMap<String,Object>();
```

```
hp.put("ten", "shank");
```

```
hp.put("hình", R.drawable.shank);
```

```
hp.put("tuổi", "35");
```

```
ds.add(hp);
```

## ❑ Viết tiếp code:

```
String []from={"ten","tuoi","hinh"};  
int []to ={R.id.textView,R.id.textView2,R.id.imageView};  
SimpleAdapter adapter=new SimpleAdapter(MainActivity.this,  
    ds, R.layout.layout_item, from, to);  
lv.setAdapter(adapter);
```

❑ ds1: context

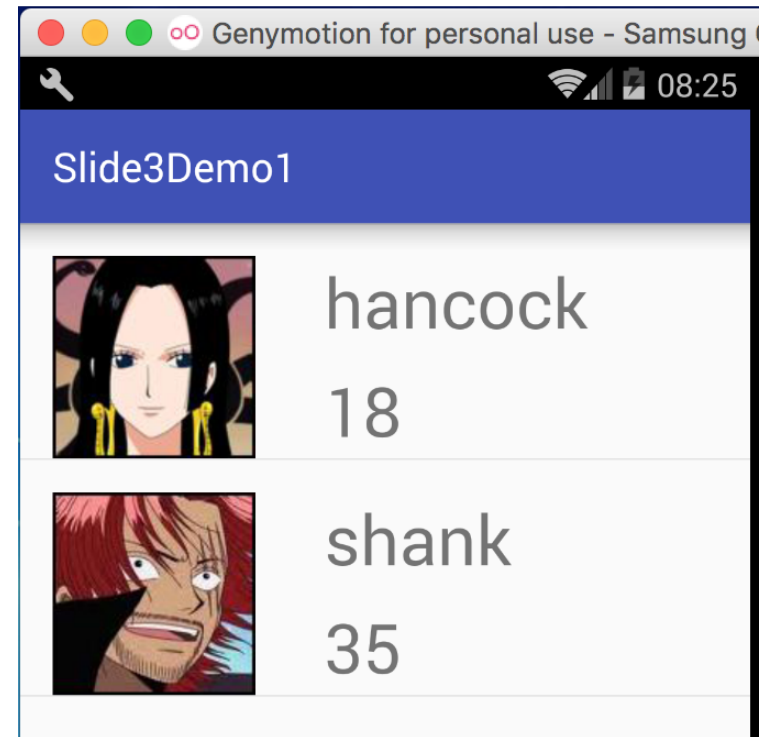
❑ ds2: dữ liệu

❑ Ds3: layout

❑ Ds4: mảng "from"

❑ Ds5: mảng "to"

❑ Dữ liệu sẽ được so khớp đồ thứ tự từ "from" sang





# DEMO

Đồ dữ liệu đơn giản





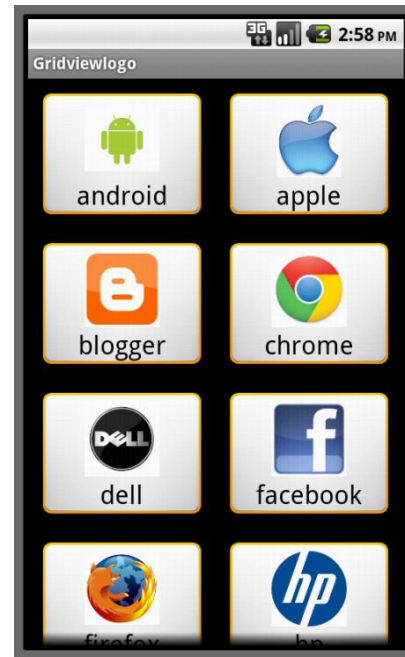
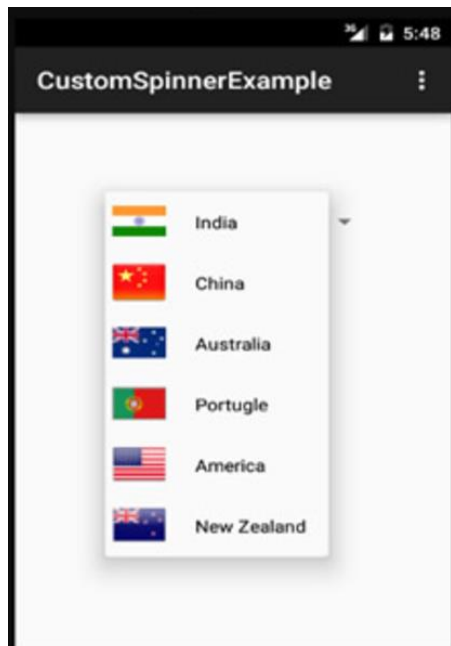
# **ADAPTER & ADAPTERVIEW**

---

## **PHẦN 2: CUSTOM ADAPTER OPTIMIZATION**

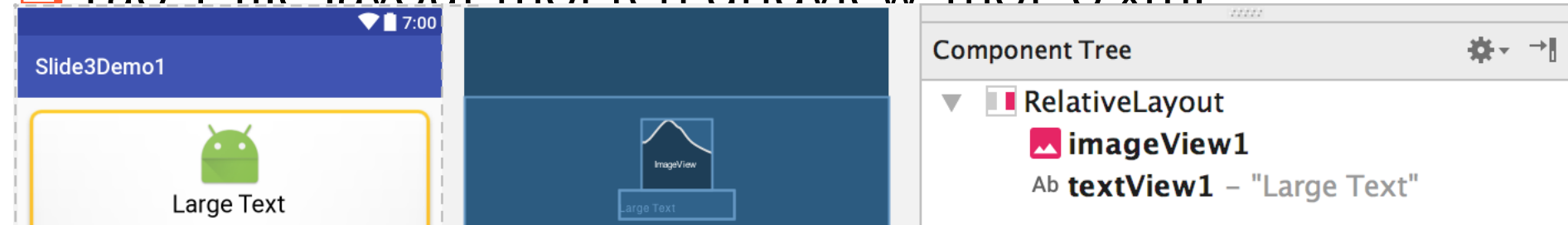


- ❑ Adapter đơn giản sẽ không có sự tùy biến giao diện và tối ưu hóa bộ nhớ. Nếu cần giao diện phức tạp và tối ưu bộ nhớ ta cần custom lại adapter
- ❑ Custom adapter được áp dụng cho tất cả các adapterview như spinner, gallery, gridview, listview....
- ❑ Thông thường nhất là kế thừa từ baseadapter và ArrayAdapter.



Thực hiện:

- ❑ Trên giao diện chính thêm vào một GridView, ánh xạ nó vào java
- ❑ Chép bộ hình vào thư mục drawable
- ❑ Tạo 1 file layout mới tên gridview một o.xml



- ❑ Trong file java tạo mảng dữ liệu

```
String[] ten={"android","apple","blogger",  
             "chrome","dell","facebook",  
             "firefox","twitter","ie",  
             "microsoft","hp","xbox"};
```

```
int[] hinh={R.drawable.android,R.drawable.apple,R.drawable.blogger  
            ,R.drawable.chrome,R.drawable.dell,R.drawable.facebook  
            ,R.drawable.firefox,R.drawable.twitter,R.drawable.ie  
            ,R.drawable.microsoft,R.drawable.hp,R.drawable.xbox};
```

# CUSTOM ADAPTER OPTIMIZATION

- ❑ Xây 1 class (nội hoặc ngoại, nếu class ngoại phải truyền đối số cho phù hợp) MyAdapter kế thừa từ BaseAdapter và override 1 số phương thức

```
public class myadapter extends BaseAdapter
{
    public int getCount() {
        // TODO Auto-generated method stub
    }

    public Object getItem(int position) {
        // TODO Auto-generated method stub
    }

    public long getItemId(int arg0) {
        // TODO Auto-generated method stub
        return 0;
    }

    public View getView(int arg0, View arg1, ViewGroup arg2) {
        // TODO Auto-generated method stub
        return arg1;
    }
}
```

- ❑ Xây thêm 1 class (nội hoặc ngoại) hỗ trợ phía giao

```
public static class View_Mot_0  
{  
    public ImageView imageview;  
    public TextView textview;  
}
```

- ❑ Trong MyAdapter, hàm getCount trả về số lượng item được hiển thị

```
public int getCount() {  
    // TODO Auto-generated method stub  
    return hình.length;  
}
```

- ❑ Trong class myadapter, hàm getView trả về 1 view, view đó sẽ được đưa vào 1 item trong gridview

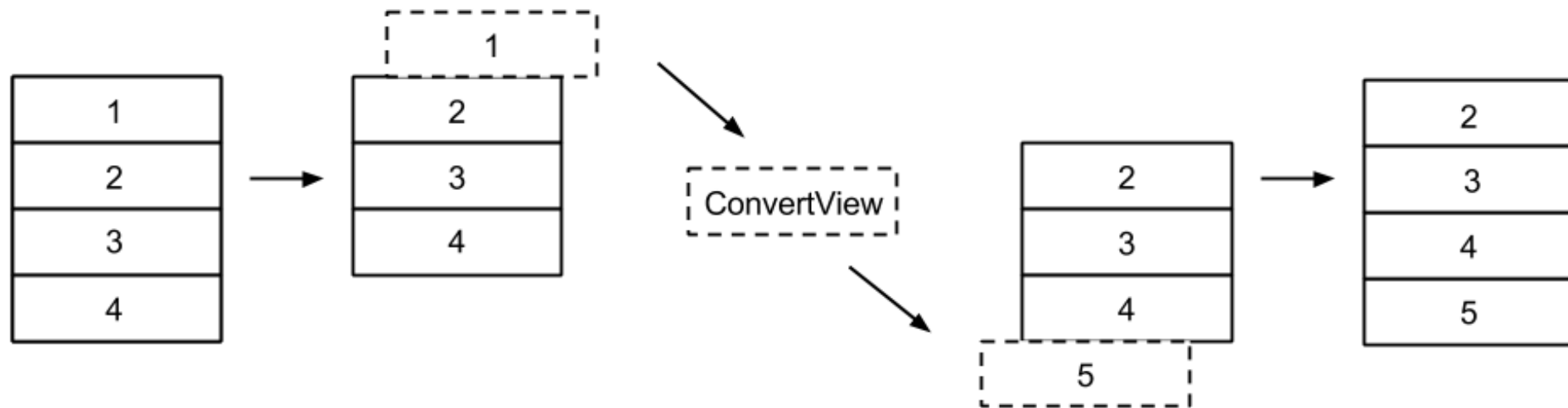
```
public View getView(int arg0, View arg1, ViewGroup arg2) {  
    //gan layout va khoi tao 1 o  
    View_Mot_0 mot_o;  
    LayoutInflater inflater= ((Activity)context).getLayoutInflater();  
    mot_o = new View_Mot_0();  
    arg1 = inflater.inflate(R.layout.gridview_mot_o, null);  
  
    //anh xa cac doi tuong tren layout va gan vao mot_o  
    mot_o.textview = (TextView) arg1.findViewById(R.id.textView1);  
    mot_o.imageview = (ImageView) arg1.findViewById(R.id.imageView1);  
  
    //cap nhat du lieu cho mot o  
    mot_o.imageview.setImageResource(hinh[arg0]);  
    mot_o.textview.setText(ten[arg0]);  
  
    return arg1;  
}
```

- ❑ Trong onCreate tạo adapter và gán vào GridView

```
myadapter adapter=new myadapter(Main2Activity.this);  
gv.setAdapter(adapter);
```

- ❑ getView sẽ chạy với các item đang hiển thị, khi cuộn getView sẽ tiếp tục chạy để có các View mới. Việc tạo liên tục các view mới sẽ là ác mộng về hiệu năng.
- ❑ Tạo 1 view mới rất tốn kém tài nguyên với các lệnh như findViewById, sau đó nó còn phải chạy inflate để điều khiển kích thước view rồi mới hiển thị lên màn hình.
- ❑ Nó còn tạo ra sự nặng nề cho cơ chế dọn rác (garbage collector) vì khi cuộn, 1 view mới được tạo, một view cũ không được tái sử dụng nên không còn được tham gia vào quá trình dọn rác.

- ❑ Để tối ưu hóa hiệu năng ta sử dụng 1 pattern tên là View Holder.
- ❑ Pattern này sẽ sử dụng một class tĩnh, tham chiếu đến view bên trong layout. Khi đó nếu ta có 1 item mới và view là null thì sẽ tạo view ra và dùng View Holder để lưu lại bộ nhớ. Khi ta scroll hàm getView sẽ chạy tiếp nhưng các view đã có rồi (!null) thì các view sẽ được tái sử dụng lại.





❑ Sửa hàm getView lại như sau:

```
View_Mot_0 mot_o;  
LayoutInflater inflater=  
    ((Activity)context).getLayoutInflater();  
  
if(arg1==null)  
{  
    mot_o = new View_Mot_0();  
    arg1 = inflater.inflate(R.layout.gridview_mot_o, null);  
    mot_o.textview = (TextView) arg1.findViewById(R.id.textView1);  
    mot_o.imageview = (ImageView) arg1.findViewById(R.id.imageView1);  
    arg1.setTag(mot_o);  
}  
else  
    mot_o=(View_Mot_0) arg1.getTag();  
  
mot_o.imageview.setImageResource(hinh[arg0]);  
mot_o.textview.setText(ten[arg0]);  
  
return arg1;
```



- ❑ Bắt sự kiện trên adapter (chú ý spinner và gallery bắt `setOnItemSelectedListener`. `ListView`, `GridView` bắt `setOnItemClickListener`):

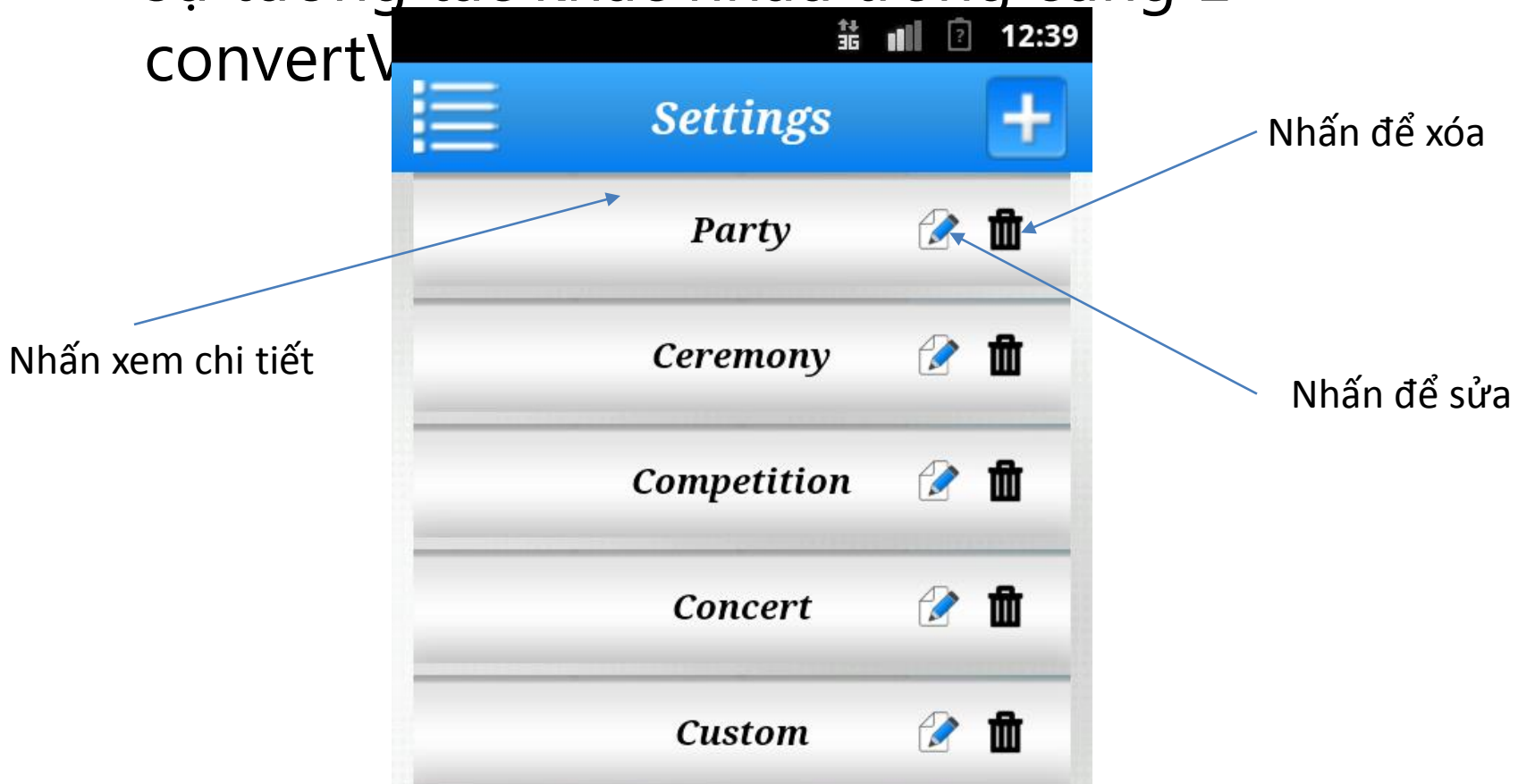
```
gv.setOnItemClickListener(new AdapterView.OnItemClickListener() {  
    @Override  
    public void onItemClick(AdapterView<?> adapterView, View view, int i, long l) {  
    }  
});
```

- ❑ Đối số thứ 3 là index của Item được click

- ❑ Ngoài ra ta có thể bắt sự kiện trong hàm getView
- ❑ Trong getView có thể bắt sự kiện trên toàn bộ convertView (1 item) hoặc có thể bắt sự kiện click vào của từng View con trong convertView
- ❑ Trong các hàm sự kiện click của view ta có thể position của item bằng đối số đầu tiên của getView, khi đó đối số đầu tiên sẽ tự sửa thành final

```
arg1.setOnClickListener(new View.OnClickListener() {  
    @Override  
    public void onClick(View view) {  
        Toast.makeText(context, arg0 + "", Toast.LENGTH_SHORT).show();  
    }  
});  
mot_o.imageView.setOnClickListener(new View.OnClickListener() {  
    @Override  
    public void onClick(View view) {  
    }  
});
```

- ❑ Với việc bắt sự kiện trên các view con của convertView ta có thể tạo ra các AdapterView với sự tương tác khác nhau trong cùng 1 convertView





# DEMO

Đổ dữ liệu và tối ưu  
hóa Adapter



- ❑ Các loại Adapter & AdapterView
- ❑ Sử dụng Adapter & AdapterView đơn giản
- ❑ Tối ưu và tùy biến Adapter





**Cảm ơn**