LẬP TRÌNH ANDROID CĂN BẢN

Bài 5: Advanced widgets

Ths. Trần Xuân Thanh Phúc | Trường Đại học Công Nghiệp Thực Phẩm

Mục tiêu

- Hiểu rõ về ListView, GridView, RecyclerView
- Nắm bắt được các trường hợp khi nào sử dụng loại widget nào cho phù hợp

Nội dung

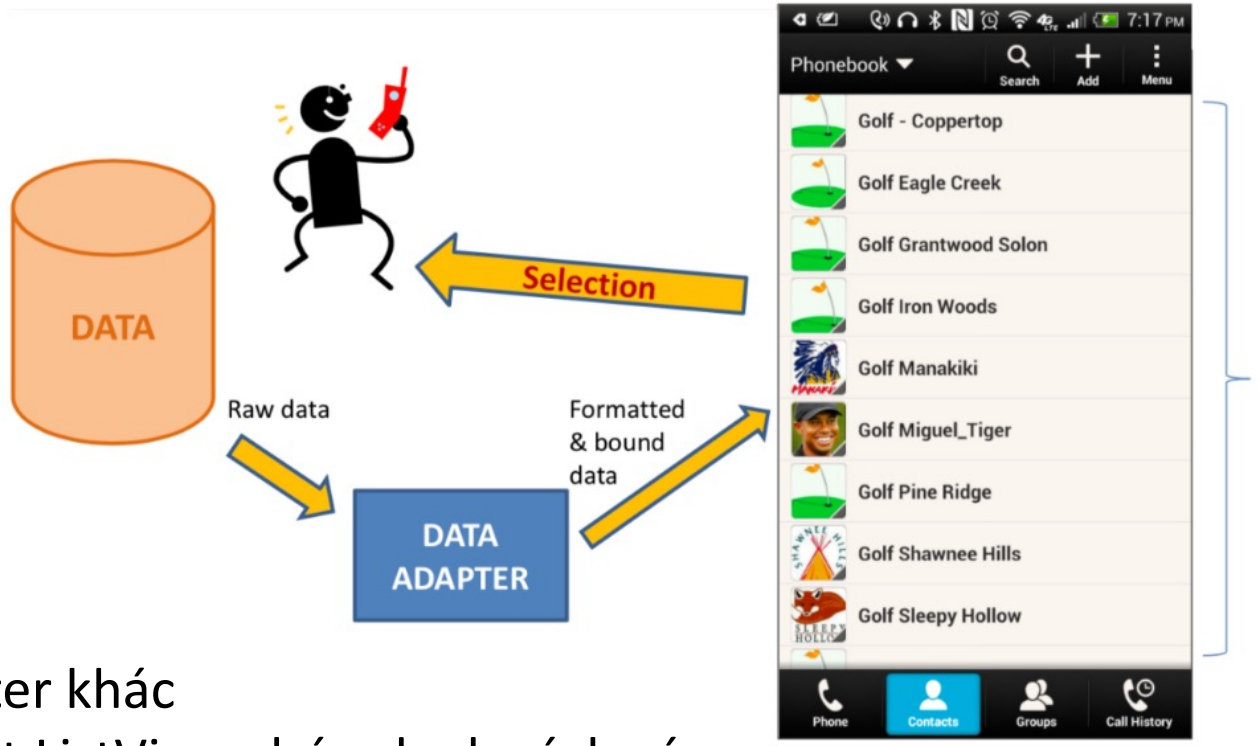
- Adapter
- ListView
- GridView
- RecyclerView

1. Adapter

- Một Adapter là một đối tượng của một lớp cài đặt giao diện Adapter, đóng vai trò như là một liên kết giữa một tập hợp dữ liệu và một Adapter View, một đối tượng của một lớp thừa kế lớp trừu tượng AdapterView
- Một Adapter có trách nhiệm lấy dữ liệu từ bộ dữ liệu và tạo ra các đối tượng View dựa trên dữ liệu đó. Các đối tượng View được tạo ra sau đó được sử dụng để gắn lên bất kỳ Adapter View mà ràng buộc với Adapter.

1. Adapter

- Data có thể đến từ nhiều nguồn khác nhau: file, database, network...



- **BaseAdapter**: Là lớp adpater cơ sở cho các Adapter khác
- **ArrayAdapter**: Một ListAdapter có thể quản lí một ListView chứa danh sách các phần tử có kiểu bất kì
- **Custom ArrayAdapter**: Thường được sử dụng để hiển thị danh sách tùy chỉnh
- **SimpleAdapter**: Nó là một Adapter đơn giản và dễ hiểu để ánh xạ dữ liệu vào những View được định nghĩa trong một tập tin XML
- **Custom SimpleAdapter**: Nó được sử dụng để hiển thị một danh sách được tùy chỉnh để truy cập các mục con của List hoặc Grid.

2. ListView

- ListView là phần tử View được dùng để hiện thị dữ liệu là một danh sách (mảng) từ một nguồn cấp gọi là Adapter
- Các bước
  - Khởi tạo ListView
  - Gán cho ListView một Adapter thông qua hàm setAdapter()
  - Khi dữ liệu trong Adapter thay đổi thì cần gọi notifyDataSetChanged()

2. ListView - Xây dựng Adapter cho ListView

```
class MyAdapter extends BaseAdapter {

        @Override
        public int getCount() {
            //Cần trả về số phần tử mà ListView hiện thị
            return 0;
        }

        @Override
        public Object getItem(int position) {
            //Cần trả về đối tượng dữ liệu phần tử ở vị trí position
            return null;
        }

        @Override
        public long getItemId(int position) {
            //Trả về một ID liên quan đến phần tử ở vị trí position
            return 0;
        }

        @Override
        public View getView(int position, View convertView, ViewGroup parent) {
            //convertView là View hiện thị phần tử, nếu là null cần tạo mới
            //(có thể nạp từ layout bằng View.inflate)

            //Cuối cùng là gán dữ liệu ở vị trí possition vào View và trả về đối
            //tượng View này

            return null;
        }
}
```

3. GridView

- GridView là một viewgroup, nó hiển thị các phần tử con trên một lưới cuộn 2 chiều.
- Một GridView được tạo từ một danh sách các GridItem. GridItem là một ô (cell) riêng lẻ trong gridview nơi mà dữ liệu sẽ được hiển thị. Bất kỳ dữ liệu nào trong gridview chỉ được hiển thị thông qua griditem.
- Một GridItem là một mảnh giao diện, nó có thể được làm bởi một số View.

3. GridView

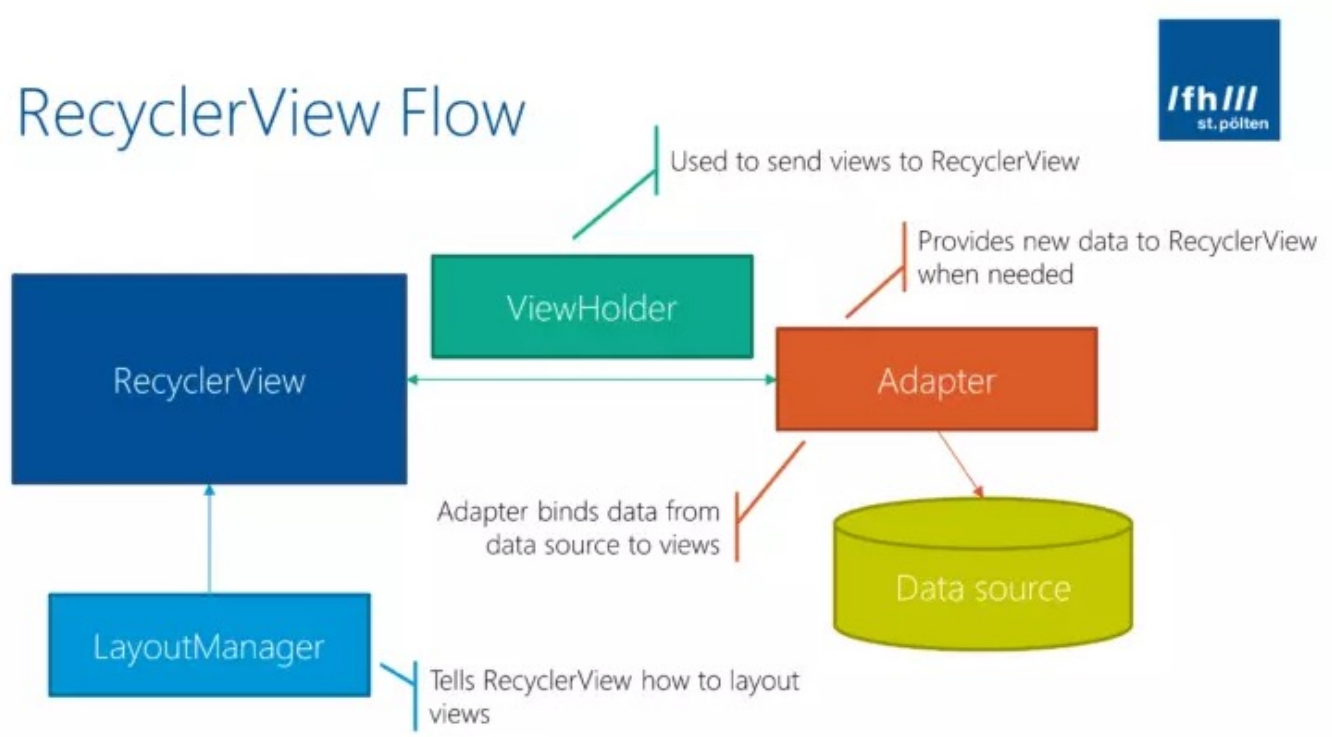
- Một số thuộc tính
  - android:numColumns: cho biếtsố lượng cột của GridView. Có thể chọn “auto_fit” để Android tự xác định số cột dựa trên không gian còn trống và các thuộc tính được liệt kê dưới đây.
  - android:verticalSpacing, android:horizontalSpacing: khoảng cách theo chiều dọc và chiều ngang giữa các items/cells của GridView
  - android:columnWidth: độ rộng của mỗi cột
  - android:stretchMode: chỉ ra cách điều chỉnh kích thước cách thành phần của GridView khi có không gian trống

4. RecyclerView

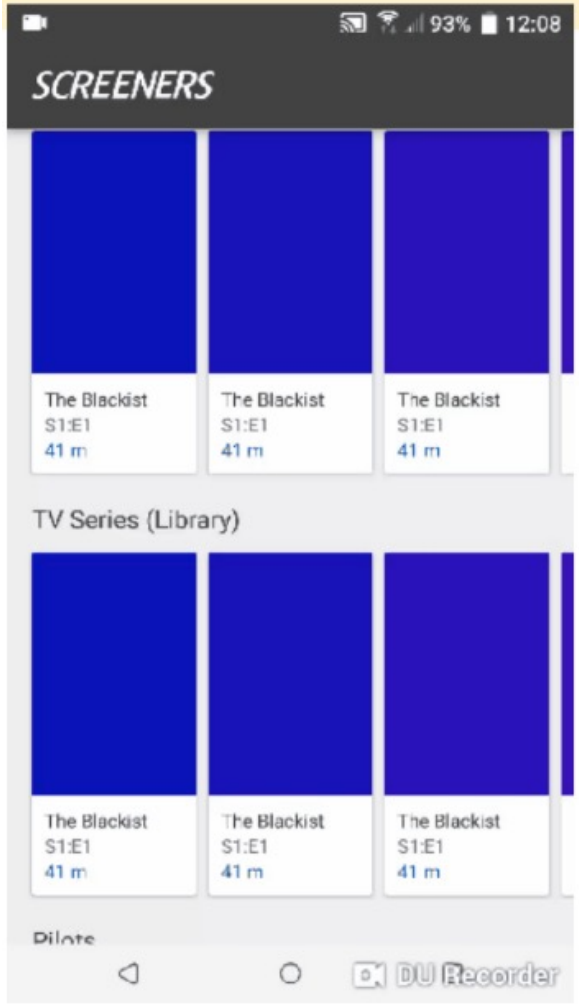
- RecyclerView là một ViewGroup nó được dùng để chuẩn bị và hiện thị các View tương tự nhau. RecyclerView được cho là sự kế thừa của ListView và GridView , và nó được giới thiệu trong phiên bản suport-v7
- Cung cấp khả năng triển khai cả bố cục Hozizontal & Vertical
- Sử dụng RecyclerView khi mà data có các thành phần thay đổi trong quá trình chạy dựa trên hành động của người dùng hoặc các dự kiện mạng.

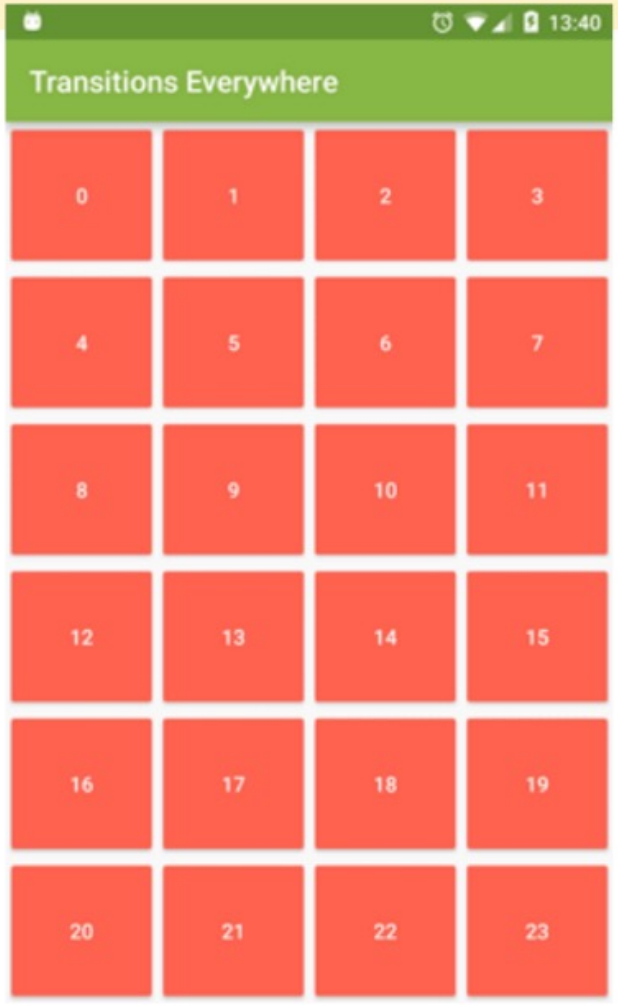
4. RecyclerView

- Khi sử dụng RecyclerView, sẽ cần làm việc với:
  - RecyclerView.Adapter: để xử lý danh sách dữ liệu và gắn dữ liệu vào từng row/cell tương ứng
  - LayoutManager: giúp bố cục vị trí các row/cell
  - ItemAnimator: giúp tạo các hiệu ứng khi thêm/xoá 1 row/cell
  - RecyclerView.Viewholder lớp dùng để gán / cập nhật dữ liệu vào các phần tử.



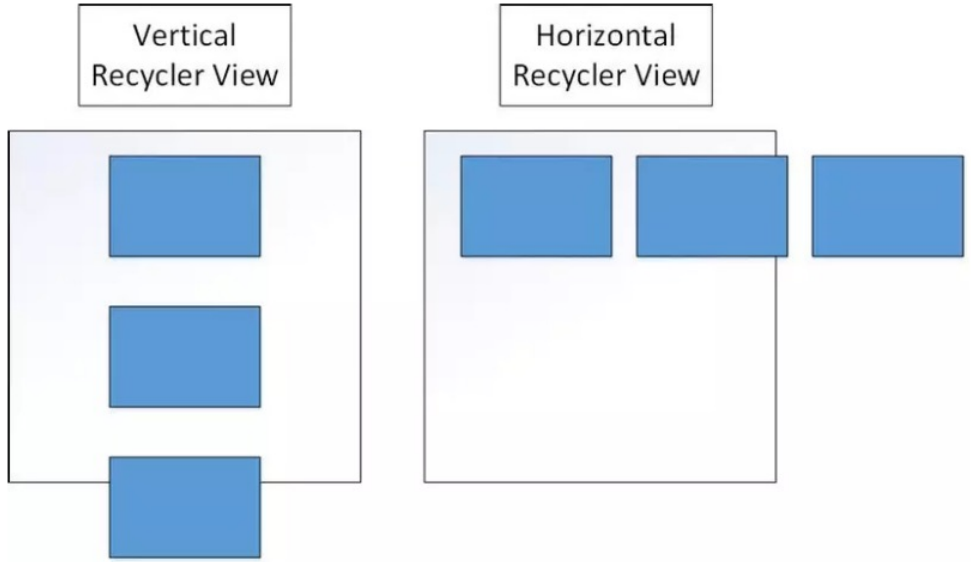
4. RecyclerView





4. RecyclerView - LayoutManagers

- Một RecyclerView cần có một layout manager và một adapter để được khởi tạo. Layout manager sẽ xác định các item bên trong RecyclerView được hiển thị như thế nào và khi nào phải tái sử dụng lại các item view (những item đã bị trượt khỏi màn hình)

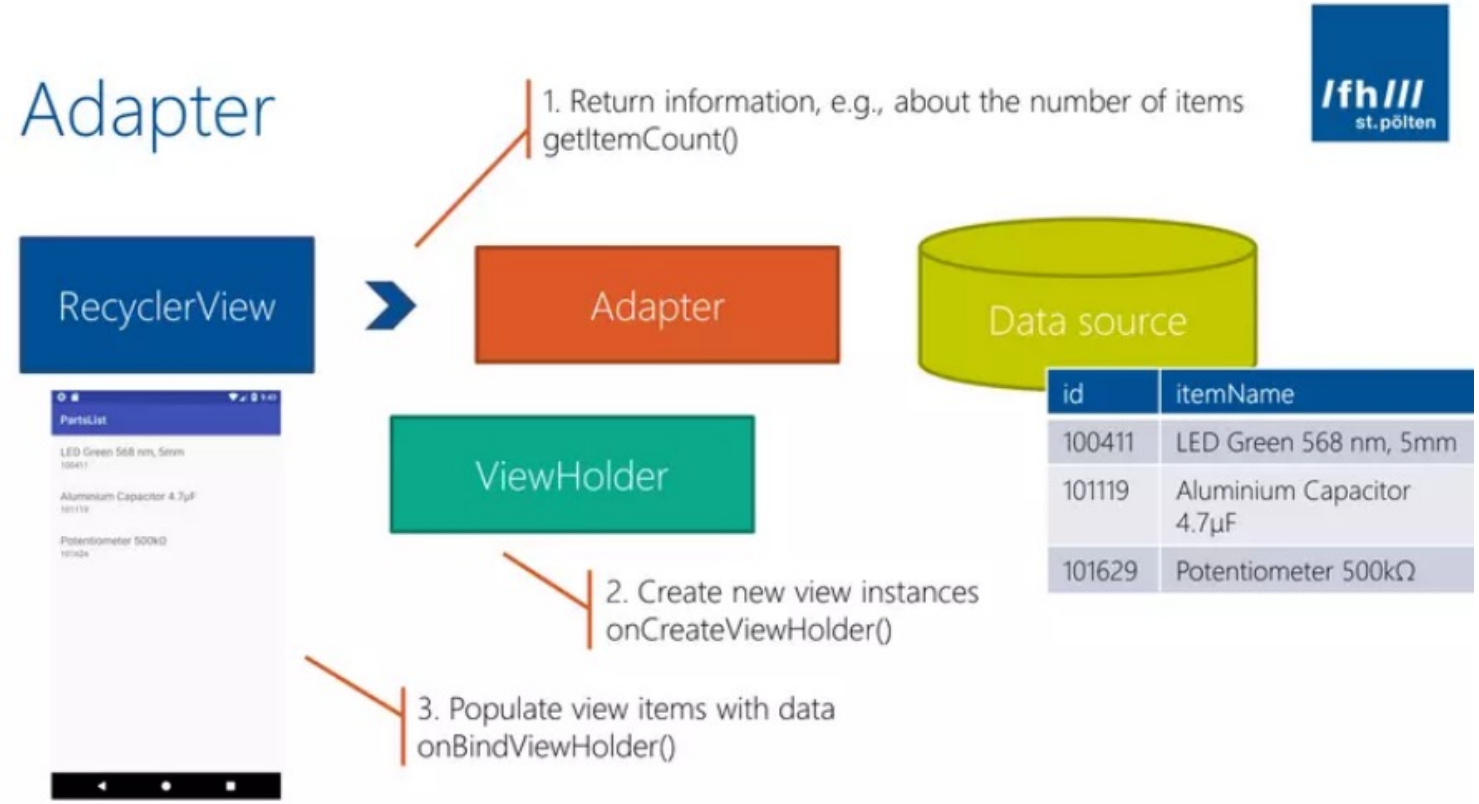


4. RecyclerView - LayoutManagers

- **LinearLayoutManager** : hiển thị các item trong danh sách có thể cuộn theo chiều dọc (horizontal) hoặc chiều ngang (Vertical).
- **GridLayoutManager** : hiển thị các item trong danh sách theo dạng lưới.
- **StaggeredGridLayoutManager** : hiển thị các item trong danh sách theo dạng lưới so le nhau
- Để tạo ra một custom layout manager, thì phải kế thừa RecyclerView.LayoutManager class.

4. RecyclerView - RecyclerView.Adapter

- Là một loại adapter mới hoạt động tương tự với các loại adapter căn bản nhưng có thêm yêu cầu ViewHolder



4. RecyclerView - RecyclerView.Adapter

- Adapter sẽ làm gần như tất cả các công việc cho RecyclerView nó kết nối Datasource vơi các View item
  - getItemCount: cho biết số phần tử của dữ liệu
  - onCreateViewHolder: tạo ra đối tượng ViewHolder, trong nó chứa View hiện thị dữ liệu
  - onBindViewHolder: chuyển dữ liệu phần tử vào ViewHolder

4. RecyclerView - ItemAnimator

- RecyclerView cung cấp các hiệu ứng trên phần tử khi phần tử đó trượt vào, di chuyển, bị xóa bằng cách sử dụng RecyclerView.ItemAnimator, hoặc SimpleItemAnimator
- Các hiệu ứng thực hiện với phần tử thêm vào bằng cách gọi **recyclerView.setItemAnimator():** như LandingAnimator, FadeInAnimator ... Hiệu ứng xảy ra khi có các thao tác thêm/xóa phần tử.

```
dependencies {
    implementation 'jp.wasabeef:recyclerview-animators:2.3.0’
}
```

```
recyclerView.setItemAnimator(new ScaleInAnimator());
```

```
adapter = new StudentAdapter(students, this);
recyclerView.setAdapter(new ScaleInAnimationAdapter(adapter));
```

4. RecyclerView – Tổng kết

- Để sử dụng RecyclerView sẽ có 7 bước chính sau đây:
  - Thêm RecyclerView support library vào gradle build file
  - Định nghĩa ra model class để sử dụng data source
  - Thêm RecyclerView vào trong activity mà bạn muốn hiển thị
  - Tạo một tệp XML để xác định một item được biểu diễn như nào
  - Tạo ra RecyclerView.Adapter và ViewHolder để gán dữ liệu cho các item
  - Gắn adapter tương ứng với data source vào RecyclerView

```
dependencies {
    ...
    implementation 'com.android.support:design:28.0.0'
}
```

LẬP TRÌNH ANDROID CĂN BẢN

Kết thúc ☺️

Ths. Trần Xuân Thanh Phúc | Trường Đại học Công Nghiệp Thực Phẩm