TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI
HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY

Phát triển ứng dụng cho Thiết bị di động
Giới thiệu RecyclerView

Giảng viên: Lê Bá Vui
Bộ môn Kỹ thuật Máy tính
Viện Công nghệ Thông tin và Truyền thông
vuilb@soict.hust.edu.vn, vui.leba@hust.edu.vn

Nội dung

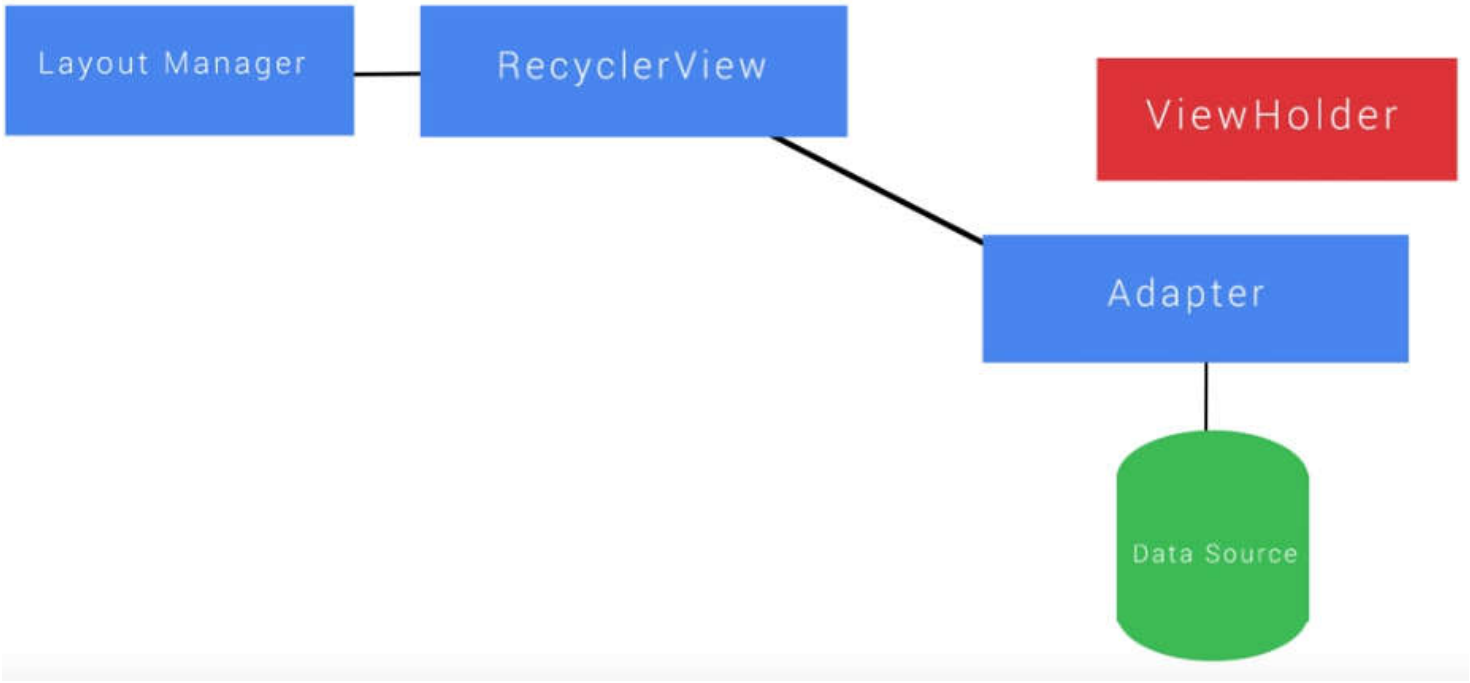
- Giới thiệu tổng quan
- Các đối tượng thành phần
- Các ví dụ lập trình với RecyclerView

Giới thiệu tổng quan

- RecyclerView là phiên bản nâng cấp của ListView và GridView:
  - Hỗ trợ các kiểu layout linh hoạt: LinearLayout (Horizontal hoặc Vertical), GridLayout, StaggerdGridLayout, hoặc kiểu layout do người lập trình tự thiết kế.
  - Áp dụng kiểu thiết kế ViewHolder để nâng cao hiệu năng.
  - Hỗ trợ các hiệu ứng animation

Các đối tượng thành phần

1. RecyclerView.Adapter
2. RecyclerView.ViewHolder
3. RecyclerView.LayoutManager
4. RecyclerView.ItemAnimator



RecyclerView.Adapter

- Khai báo các view holder khi được yêu cầu.
- Gắn các đối tượng dữ liệu với view holder tương ứng.

```
public class ContactAdapter extends RecyclerView.Adapter <RecyclerView.ViewHolder> {
    @NonNull
    @Override
    public RecyclerView.ViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
        return null;
    }

    @Override
    public void onBindViewHolder(@NonNull RecyclerView.ViewHolder holder, int position) {

    }

    @Override
    public int getItemCount() {
        return 0;
    }
}
```

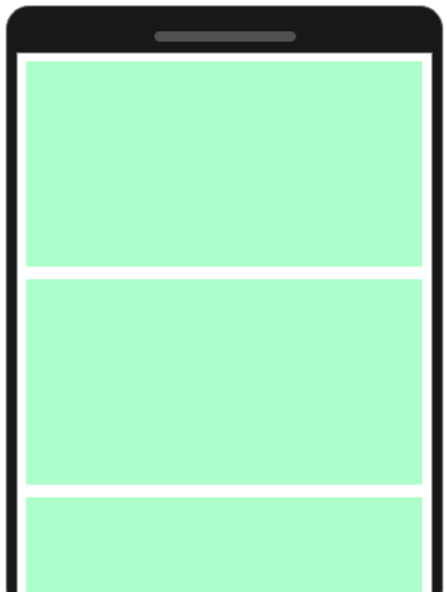
RecyclerView.ViewHolder

- Đại diện cho mỗi view item của danh sách
- Thường được khai báo bên trong lớp adapter
- Khai báo các thuộc tính là các widget trong view item
- Được RecyclerView tạo mới với số lượng tùy thuộc vào kiểu danh sách, kích thước màn hình hiển thị
- Khi item đi ra khỏi vùng hiển thị, view holder được tái sử dụng với item đi vào vùng hiển thị

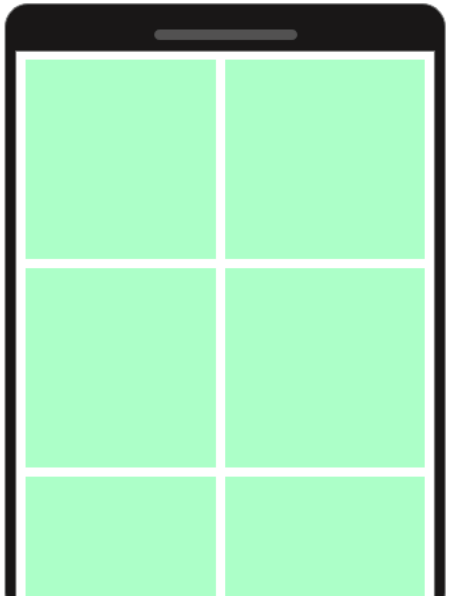
```
class MyViewHolder extends RecyclerView.ViewHolder {
    // Khai báo các widget

    MyViewHolder(View view) {
        super(view);
        // Gán các widget từ layout
        // Khai báo các thuộc tính sự kiện
    }
}
```

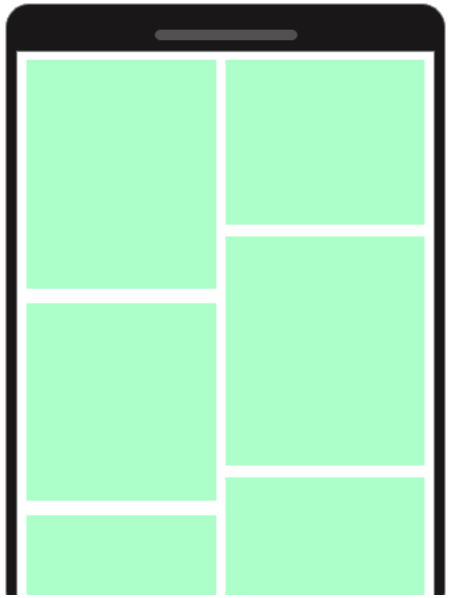
LayoutManager



LinearLayout

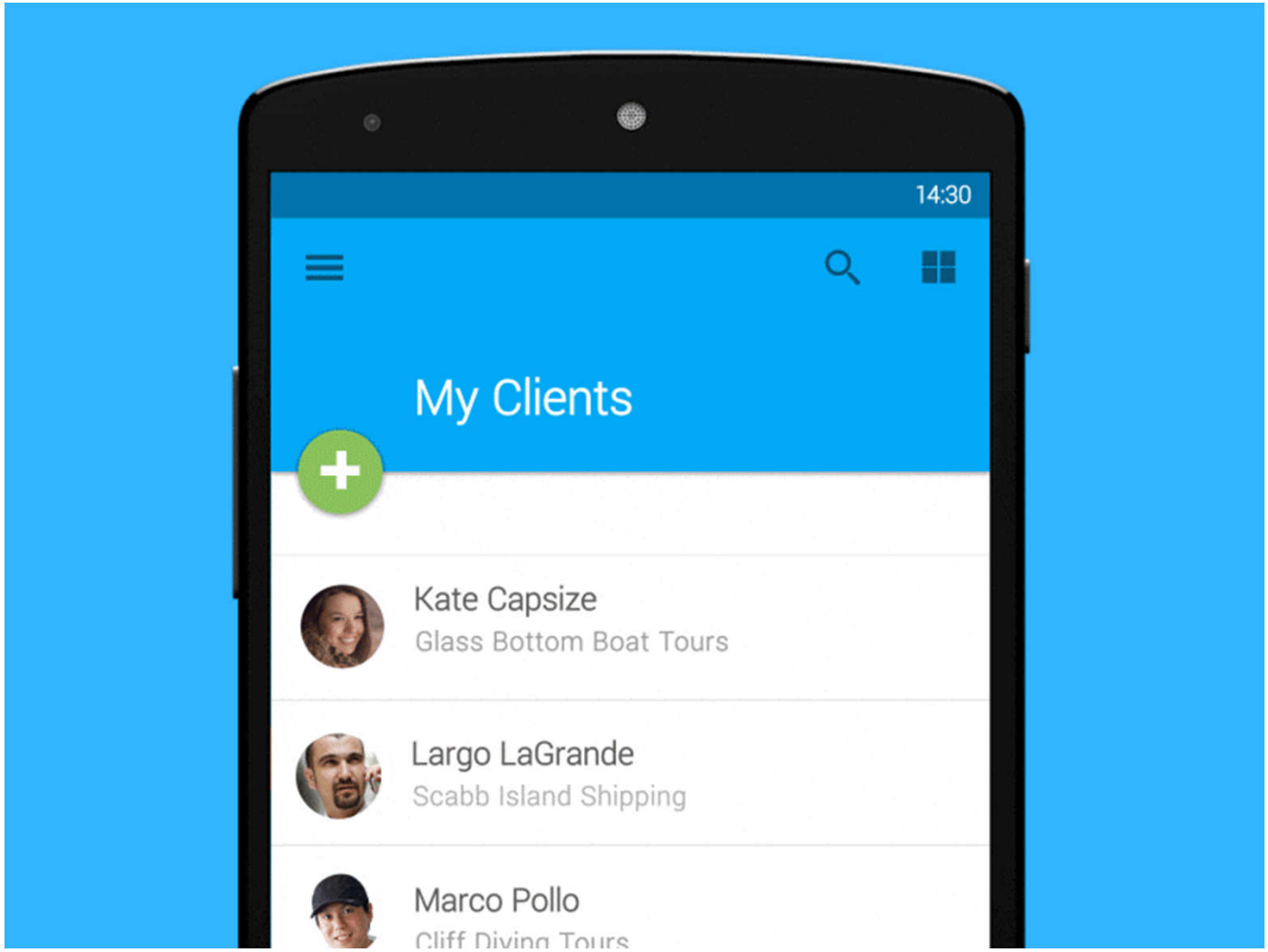


GridLayout



StaggerdGridLayout

ItemAnimator



Lập trình với RecyclerView

Bước 1: Thêm thư viện vào project:

Với android 9 trở xuống (API level 28)

```
implementation 'com.android.support:recyclerview-v7:28.0.0'
```

Với android 10 trở lên (API level 29)

```
implementation 'androidx.recyclerview:recyclerview:1.1.0'
```

Lập trình với RecyclerView

Bước 2: Thêm đối tượng RecyclerView vào layout

```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <androidx.recyclerview.widget.RecyclerView
        android:id="@+id/recycler_view"
        android:layout_width="match_parent"
        android:layout_height="match_parent"/>

</FrameLayout>
```

Lập trình với RecyclerView

Bước 3: Tạo lớp adapter mới

a. Khai báo lớp con ViewHolder có các thuộc tính là các đối tượng widget trong layout của item.

```
public class ItemAdapter extends RecyclerView.Adapter<ItemAdapter.MyViewHolder>
```

```
class MyViewHolder extends RecyclerView.ViewHolder {
    TextView textView;
    ImageView imageView;

    MyViewHolder(View view) {
        super(view);
        textView = view.findViewById(R.id.text_view);
        imageView = view.findViewById(R.id.image_view);
    }
}
```

Lập trình với RecyclerView

Bước 3: Tạo lớp adapter mới

b. Viết chồng các phương thức bắt buộc của lớp adapter

- Phương thức onCreateViewHolder được gọi khi một view holder mới được tạo
- Phương thức onBindViewHolder được gọi khi view holder trong màn hình hiển thị
- Phương thức getItemCount trả về số item trong dữ liệu

```
@NonNull
@Override
public MyViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
    View view = LayoutInflater.from(parent.getContext()).inflate(R.layout.layout_item, parent, false);
    return new MyViewHolder(view);
}

@Override
public void onBindViewHolder(@NonNull MyViewHolder holder, int position) {
    holder.textView.setText(items.get(position).getMessage());
    holder.imageView.setImageResource(items.get(position).getAvatarResource());
}

@Override
public int getItemCount() {
    return items.size();
}
```

Lập trình với RecyclerView

Bước 4: Khởi tạo trong activity

```
// Khai báo RecyclerView
RecyclerView recyclerView = (RecyclerView) findViewById(R.id.recycler_view);
recyclerView.setHasFixedSize(true);

// Khai báo kiểu layout manager
RecyclerView.LayoutManager layoutManager = new LinearLayoutManager(this);
// RecyclerView.LayoutManager layoutManager = new LinearLayoutManager(this,
LinearLayoutManager.HORIZONTAL, false);
// RecyclerView.LayoutManager layoutManager = new GridLayoutManager(this, 3);
// RecyclerView.LayoutManager layoutManager = new StaggeredGridLayoutManager(2,
StaggeredGridLayoutManager.VERTICAL);
recyclerView.setLayoutManager(layoutManager);

// Khai báo adapter
RecyclerView.Adapter mAdapter = new ItemAdapter(items);
recyclerView.setAdapter(mAdapter);
```

Lập trình với RecyclerView

Bước 5: Thiết lập ItemAnimator

Sử dụng lớp DefaultItemAnimator mặc định hoặc thư viện do các nhà phát triển cung cấp.

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

Các animation sẽ được áp dụng khi có sự thay đổi dữ liệu (thêm, bớt, thay đổi các đối tượng dữ liệu) và gọi các hàm tương ứng để cập nhật adapter.

```
items.add(new ItemModel("Message 17", R.drawable.thumb17));
mAdapter.notifyItemInserted(items.size() - 1);

items.remove(0);
mAdapter.notifyItemRemoved(0);

items.get(0).setMessage("Updated");
mAdapter.notifyItemChanged(0);
```

Lập trình với RecyclerView

Bước 6: Xử lý sự kiện khi item được nhấn

RecyclerView không sử dụng listener giống như ListView hoặc GridView nên việc xác định sự kiện liên quan đến các item được xử lý thông qua ViewHolder bằng lớp interface tự định nghĩa.

a. Khai báo interface

```
public interface ItemClickListener {
    void OnItemClick(int position);
}
```

b. Gắn interface vào lớp adapter

```
private List<ItemModel> items;
private ItemClickListener itemClickListener;

public ItemAdapter(List<ItemModel> items, ItemClickListener itemClickListener)
{
    this.items = items;
    this.itemClickListener = itemClickListener;
}
```

Lập trình với RecyclerView

Bước 6: Xử lý sự kiện khi item được nhấn

c. Gọi hàm interface khi item được nhấn

```
if (itemClickListener != null)
    view.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            itemClickListener.OnItemClick(getAdapterPosition());
        }
    });
```

d. Implement interface trong MainActivity, khai báo constructor cho adapter và viết chồng hàm OnItemClick

```
@Override
public void OnItemClick(int position) {
    Log.v("TAG", items.get(position).getMessage());
}
```

Item có nhiều kiểu giao diện khác nhau

Bước 1: Tạo layout cho mỗi kiểu giao diện

```xml
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_margin="5dp"
    android:background="#F0F0F0">

    <ImageView
        android:id="@+id/image_view"
        android:layout_width="60dp"
        android:layout_height="60dp"
        android:src="@drawable/thumb1"/>

    <TextView
        android:id="@+id/text_view"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_toRightOf="@id/image_view"
        android:gravity="center_vertical"
        android:layout_marginLeft="5dp"
        android:text="Hello World Hello World"
        android:textSize="24sp"/>

</RelativeLayout>
```

```xml
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_margin="5dp"
    android:background="#F0F0F0">

    <TextView
        android:id="@+id/text_view"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_toLeftOf="@id/image_view"
        android:gravity="center_vertical"
        android:layout_marginRight="5dp"
        android:text="Hello World Hello World"
        android:textSize="24sp"/>

    <ImageView
        android:id="@+id/image_view"
        android:layout_width="60dp"
        android:layout_height="60dp"
        android:layout_alignParentRight="true"
        android:src="@drawable/thumb1"/>
</RelativeLayout>
```

Item có nhiều kiểu giao diện khác nhau

Bước 2: Tạo view holder cho mỗi layout trong adapter

```
class LeftViewHolder extends RecyclerView.ViewHolder
{
    TextView textView;
    ImageView imageView;

    LeftViewHolder(View view) {
        super(view);
        textView = view.findViewById(R.id.text_view);
        imageView =
view.findViewById(R.id.image_view);

        if (itemClickListener != null)
            view.setOnClickListener(new
View.OnClickListener() {
                @Override
                public void onClick(View view) {

itemClickListener.OnItemClick(getAdapterPosition());
                }
            });
    }
}
```

```
class RightViewHolder extends RecyclerView.ViewHolder
{
    TextView textView;
    ImageView imageView;

    RightViewHolder(View view) {
        super(view);
        textView = view.findViewById(R.id.text_view);
        imageView =
view.findViewById(R.id.image_view);

        if (itemClickListener != null)
            view.setOnClickListener(new
View.OnClickListener() {
                @Override
                public void onClick(View view) {

itemClickListener.OnItemClick(getAdapterPosition());
                }
            });
    }
}
```

Item có nhiều kiểu giao diện khác nhau

Bước 3: Cập nhật các phương thức getItemViewType(), onCreateViewHolder(), và onBindViewHolder()

```
@Override
public int getItemViewType(int position) {
    if (items.get(position).isLeft())
        return 0;
    else
        return 1;
}
```

```
@Override
public RecyclerView.ViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
    if (viewType == 0)
    {
        View view = LayoutInflater.from(parent.getContext()).inflate(R.layout.layout_item_left,
parent, false);
        return new LeftViewHolder(view);
    }
    else {
        View view = LayoutInflater.from(parent.getContext()).inflate(R.layout.layout_item_right,
parent, false);
        return new RightViewHolder(view);
    }
}
```

Item có nhiều kiểu giao diện khác nhau

Bước 3: Cập nhật các phương thức getItemViewType(), onCreateViewHolder(), và onBindViewHolder()

```
@Override
public void onBindViewHolder(@NonNull RecyclerView.ViewHolder holder, int position) {
    if (getItemViewType(position) == 0) {
        ((LeftViewHolder)holder).textView.setText(items.get(position).getMessage());
        ((LeftViewHolder)holder).imageView.setImageResource(items.get(position).getAvatarResource());
    } else {
        ((RightViewHolder)holder).textView.setText(items.get(position).getMessage());
        ((RightViewHolder)holder).imageView.setImageResource(items.get(position).getAvatarResource());
    }
}
```

Tạo hiệu ứng “Swipe to remove”

- Sử dụng lớp ItemTouchHelper.SimpleCallback hoặc tạo lớp mới từ lớp ItemTouchHelper.Callback
- Hỗ trợ thao tác trượt (swipe) và kéo thả (drag) theo các hướng khác nhau.

```
ItemTouchHelper.SimpleCallback callback = new ItemTouchHelper.SimpleCallback(0, ItemTouchHelper.LEFT) {
    @Override
    public boolean onMove(@NonNull RecyclerView recyclerView, @NonNull RecyclerView.ViewHolder
viewHolder, @NonNull RecyclerView.ViewHolder target) {
        return false;
    }

    @Override
    public void onSwiped(@NonNull RecyclerView.ViewHolder viewHolder, int direction) {
        int position = viewHolder.getAdapterPosition();
        items.remove(position);
        adapter.notifyItemRemoved(position);
    }
};

ItemTouchHelper touchHelper = new ItemTouchHelper(callback);
touchHelper.attachToRecyclerView(recyclerView);
```

Tham khảo

- https://android.jlelse.eu/understanding-recyclerview-a-high-level-insight-part-1-dc3f81af5720
- https://developer.android.com/guide/topics/ui/layout/recyclerview


ĐẠI HỌC
BÁCH KHOA