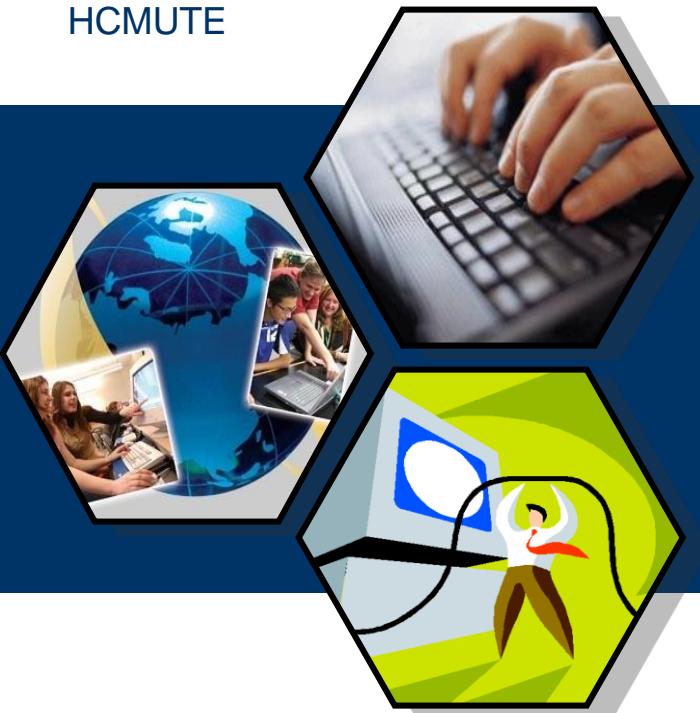




HCMUTE

KHOA CÔNG NGHỆ THÔNG TIN

UTEx



GIỚI THIỆU ANDROID

Khoa Công nghệ Thông tin
Đại học Sư phạm Kỹ thuật TP.HCM

ThS. Nguyễn Hữu Trung



ANDROID LÀ GÌ?

4

- Android là một hệ điều hành mã nguồn mở và là một hệ điều hành dựa trên Linux. Ban đầu Android được phát triển bởi Công ty Android với sự hỗ trợ tài chính từ Google, sau đó được Google mua lại vào năm 2005.
- Phiên bản beta của Android Software Development Kit (SDK) được công bố bởi Google vào năm 2007, trong khi phiên bản thương mại đầu tiên Android 1.0 được công bố 9/2008.
- Vào 27/6/2012, tạo hội nghị Google I/O, Google công bố phiên bản Android tiếp theo là 4.1 Jelly Bean. Jelly Bean là một bản cập nhật với mục đích đầu tiên là cải thiện giao diện người dùng (User Interface), cả về tính năng lẫn hiệu suất.
- Mã nguồn cho Android là miễn phí. Google công bố hầu hết các code dưới Apache License version 2.0, và phần còn lại, các thay đổi Linux Kernel dưới GNU General Public License version 2.

- **Android** là mã nguồn mở
- Bất kỳ ai cũng tùy biến lại **Android Platform**
- Người sử dụng có thể lựa chọn nhiều ứng dụng trên **Android**.
- **Android** cung cấp nhiều tính năng thú vị như: thời tiết, mở màn hình, RSS (Really Simple Syndication) vv
- **Android** cũng hỗ trợ cho các dịch vụ nhắn tin (SMS và MMS), trình duyệt web, lưu trữ dữ liệu (SQLite), kết nối (GSM, CDMA, BlueTooth, Wi-Fi vv), media....

- Android chiếm hơn 87,7% (2017), 69.74% (2022) thị phần điện thoại thông minh trên toàn thế giới với khoảng hơn 2 tỷ thiết bị đã được kích hoạt và hơn 1,3 triệu lượt kích hoạt mỗi ngày.



Music



News



Multimedia



Sports



Lifestyle



Food & Drink



Travel



Weather



Books



Business



Reference



Navigation



Social Media



Utilities



Finance

- Hiện tại, dãy Codename của Android từ A tới T, như Astro, Blender, Cupcake, Donut, Eclair, Froyo, Gingerbread, Honeycomb, Ice Cream Sandwitch, Jelly Bean, KitKat và Lollipop,..., Pie 2018, Android 10 Q (2019), Red velvet cake(2020), Snow Cone(2021), Tiramisu (2022).



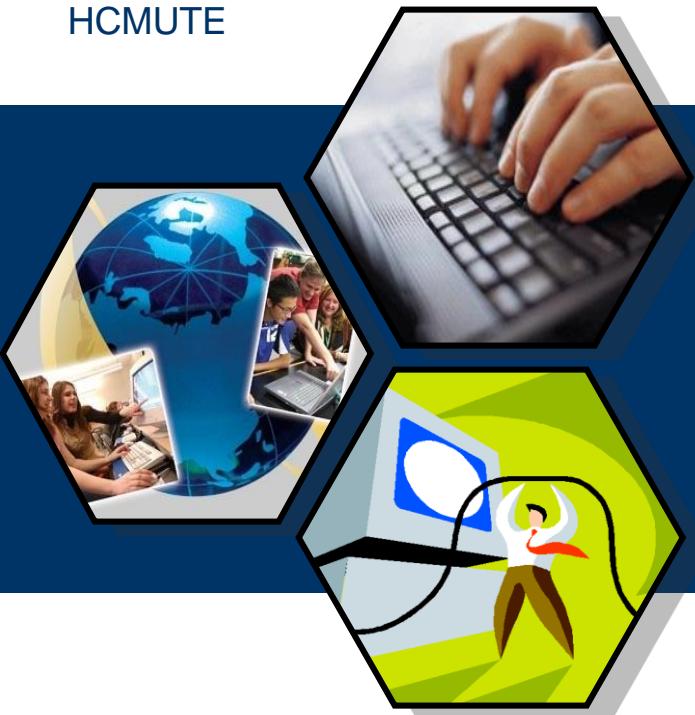


HCMUTE

KHOA CÔNG NGHỆ THÔNG TIN

UTEx

CÀI ĐẶT JDK và ANDROID



Khoa Công nghệ Thông tin
Đại học Sư phạm Kỹ thuật TP.HCM

ThS. Nguyễn Hữu Trung



9

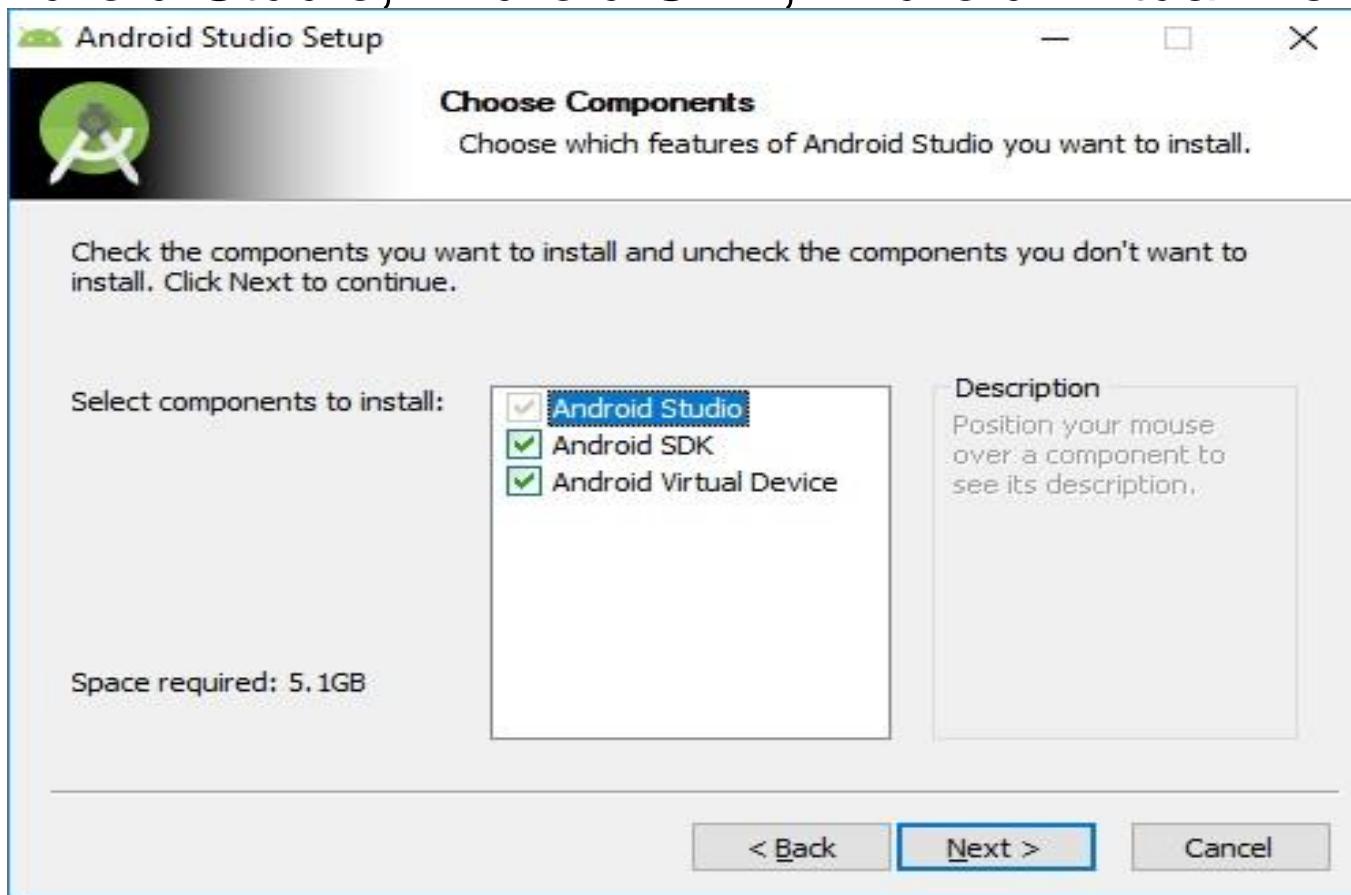
- ❑ Tải JDK 8 trở lên và cài đặt vào máy.
- ❑ Cấu hình môi trường JDK trên máy tính.

- Bạn có thể phát triển ứng dụng Android ở môi trường Windows, Mac OS và cả Linux. Ngoài ra có rất nhiều công cụ để xây dựng các ứng dụng Android phải kể đến như Android Studio, IntelliJ IDEA, Eclipse, Xamarin...
 - Thiết lập JDK (Java Development Kit)
 - Thiết lập Android IDEs

- **Bước 1:** Bạn nhấp đúp chuột vào tập tin Android Studio vừa tải về, sau đó nhấn Next để xác nhận cài đặt Android Studio.



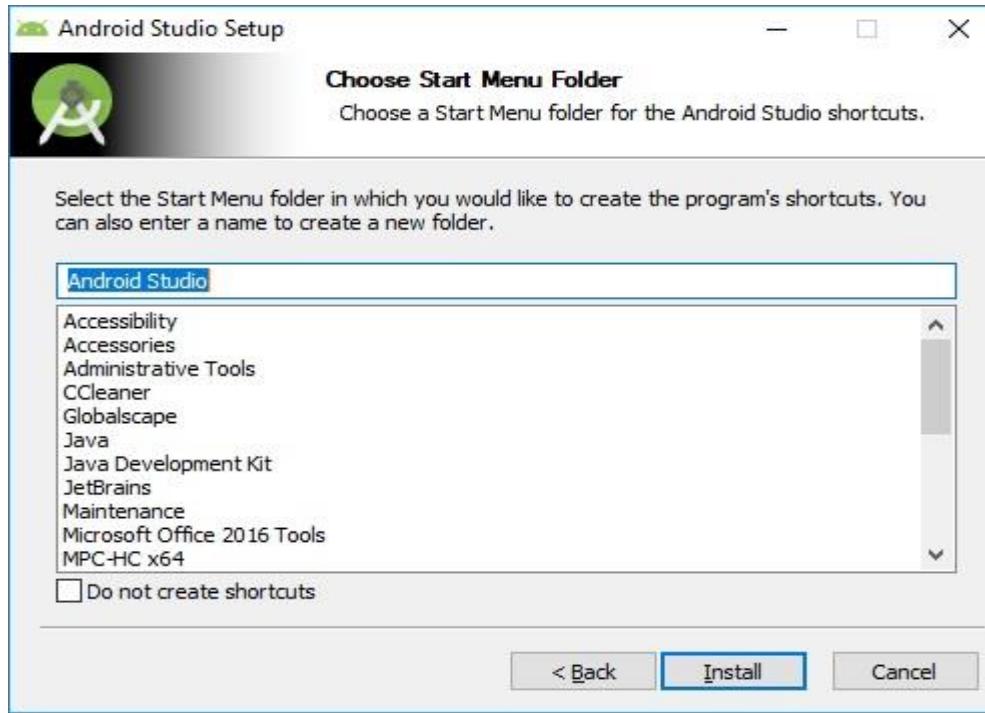
- Bạn hãy tích chọn tất cả các tùy chọn hiện có bao gồm Android Studio, Android SDK, Android Virtual Device.



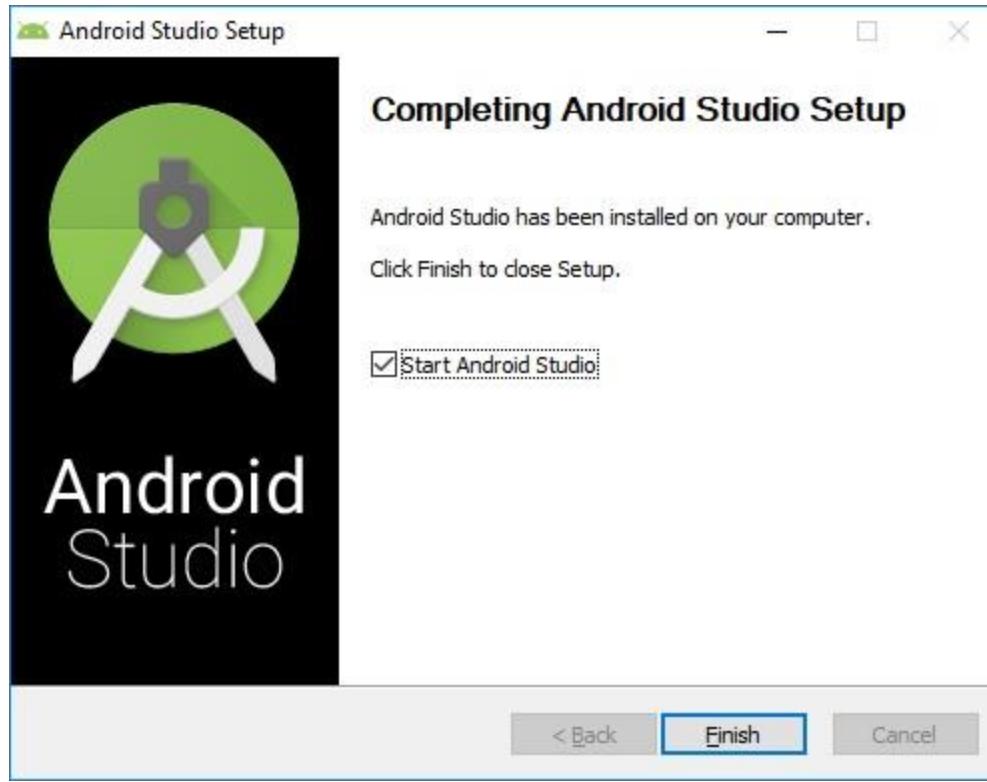
- Đọc xong phần giấy phép, điều khoản xong thì nhấn chọn **I Agree** để đồng ý cài đặt Android Studio.



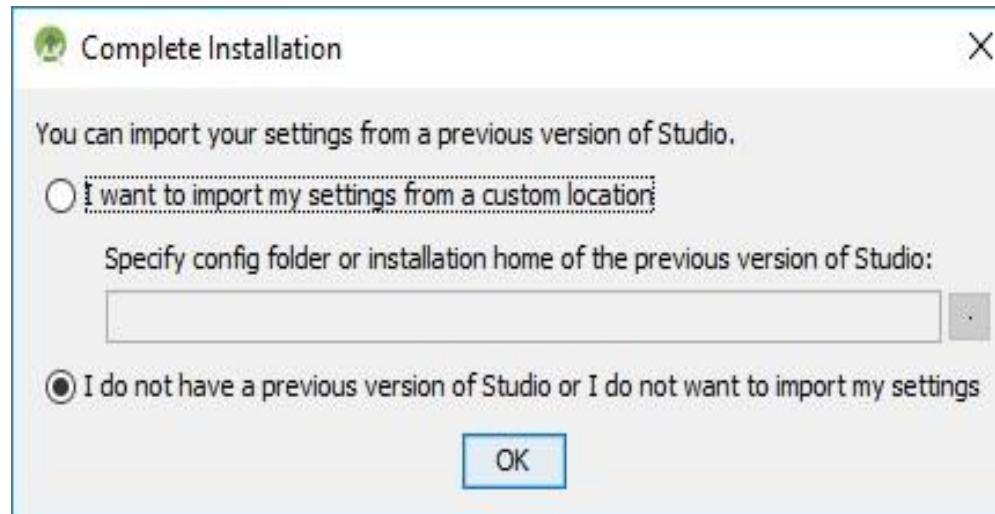
- Màn hình tiếp theo là lựa chọn nơi lưu trữ, bạn có thể để mặc định và nhấn **Next** để qua bước kế tiếp.
- Bây giờ hãy nhấn chọn **Install** để bắt đầu tiến hành cài đặt Android Studio.



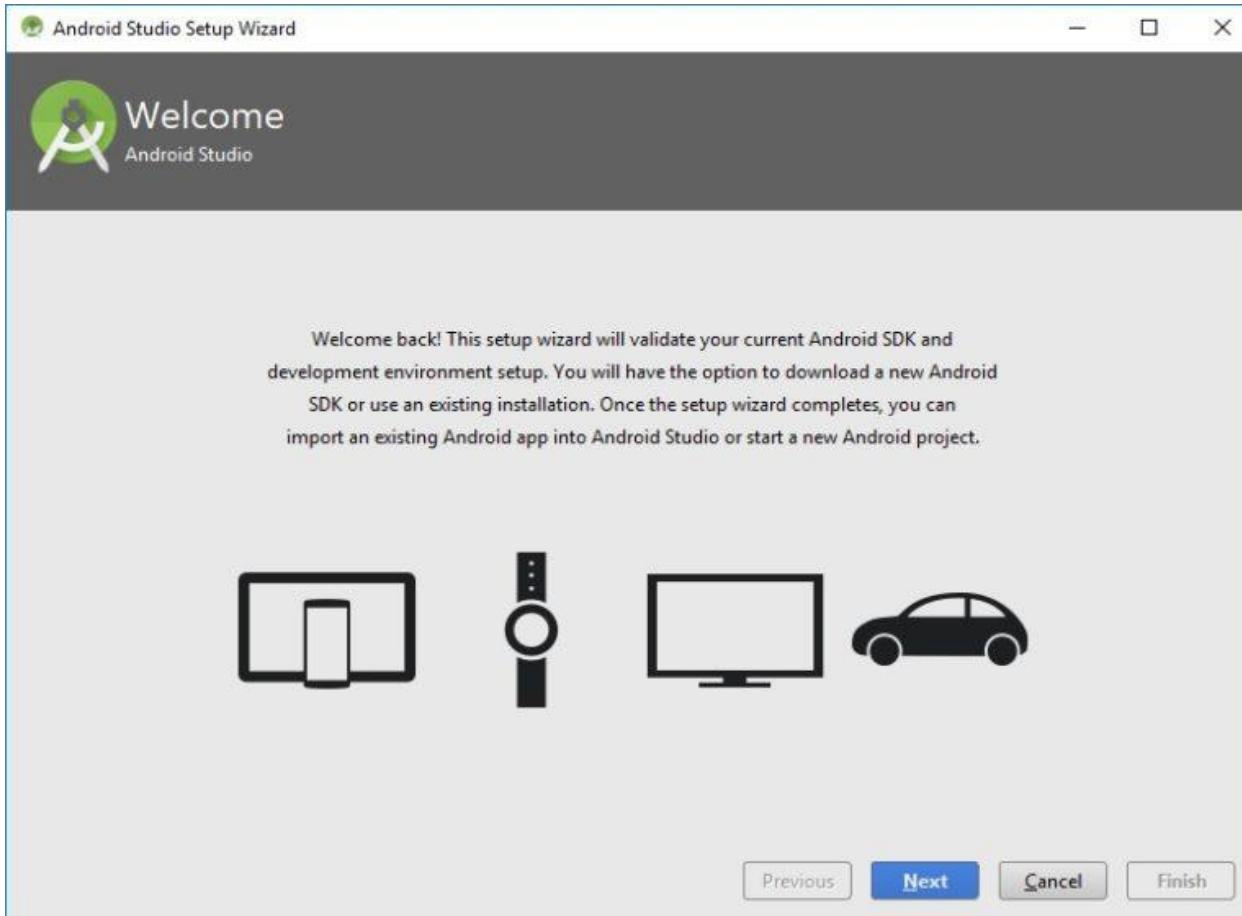
- Hãy chờ đợi quá trình cài đặt diễn ra tùy cấu hình từng máy, sau có thông báo cài đặt thành công bạn nhấn **Finish**.



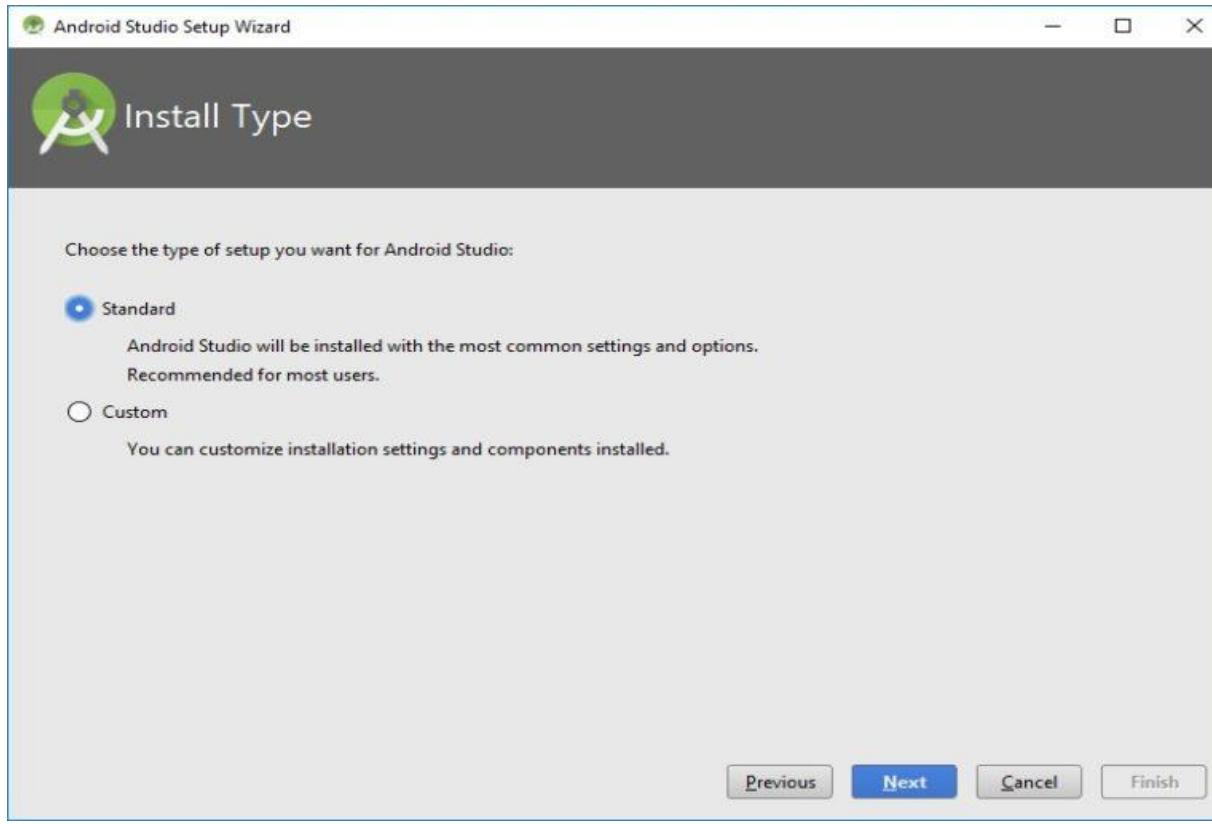
- **Bước 2:** Bây giờ bạn hãy nhấn chọn các bước như sau đây.



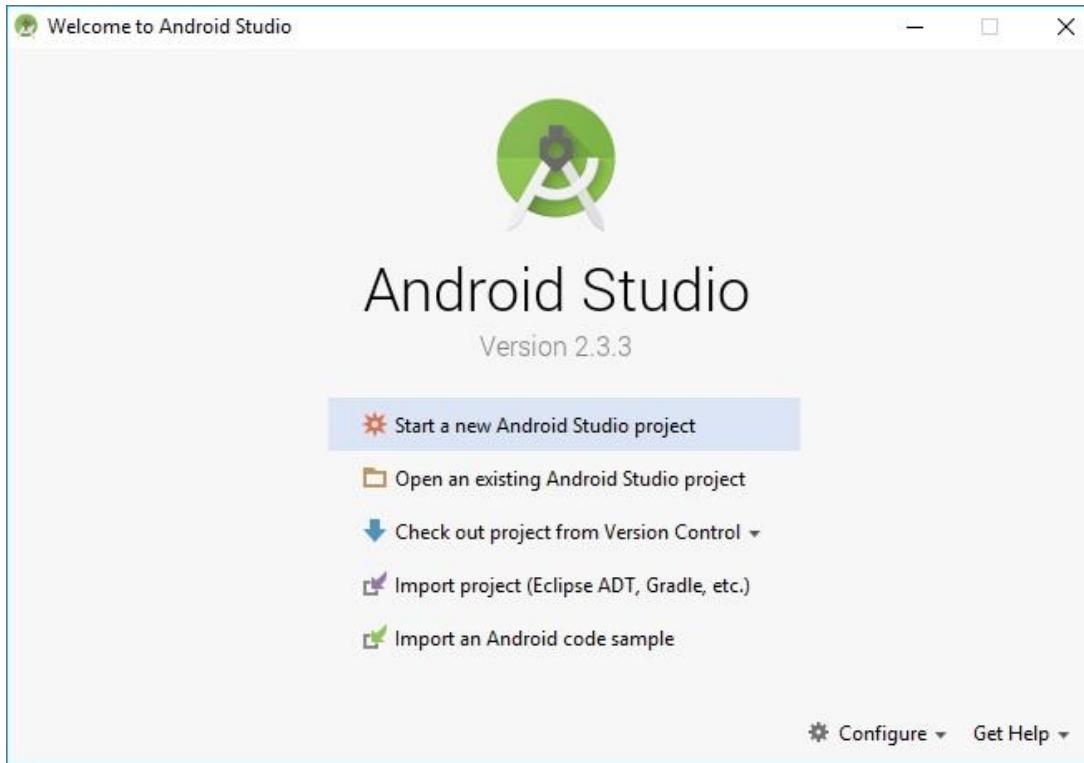
- Nhấn **Next** để qua bước kế tiếp.



- Nhấn **Next** để qua bước kế tiếp. Rồi nhấn **Finish** để qua bước kế tiếp



- **Bước 3:** Tiếp tục chờ đợi để quá trình cài đặt diễn ra, sau khi cài đặt xong nhấn **Finish** để hoàn thành quá trình.



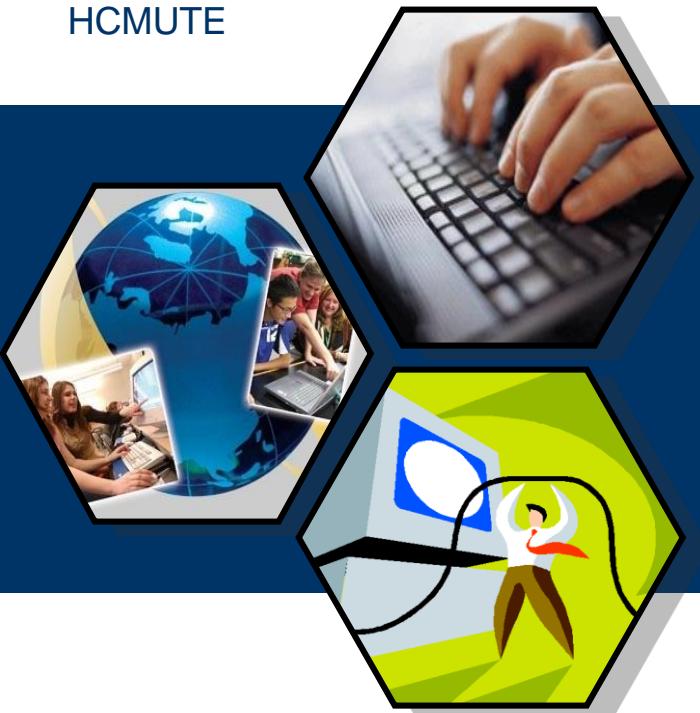


HCMUTE

KHOA CÔNG NGHỆ THÔNG TIN



TẠO PROJECT ANDROID



**Khoa Công nghệ Thông tin
Đại học Sư phạm Kỹ thuật TP.HCM**

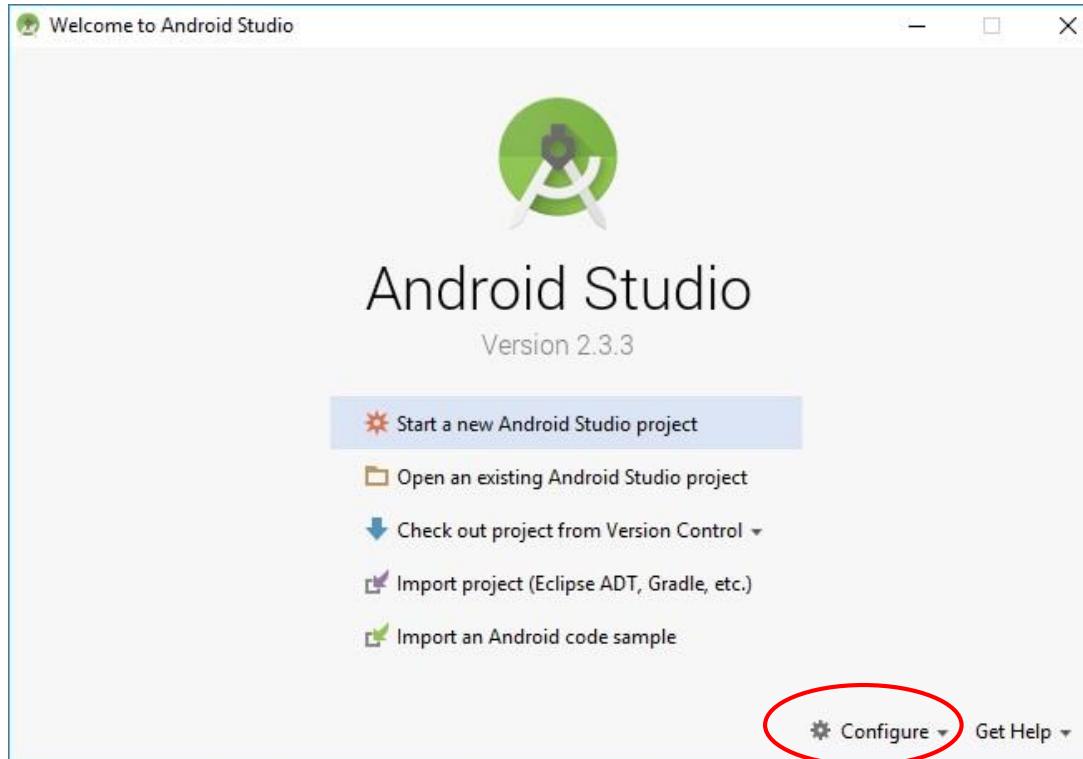
ThS. Nguyễn Hữu Trung



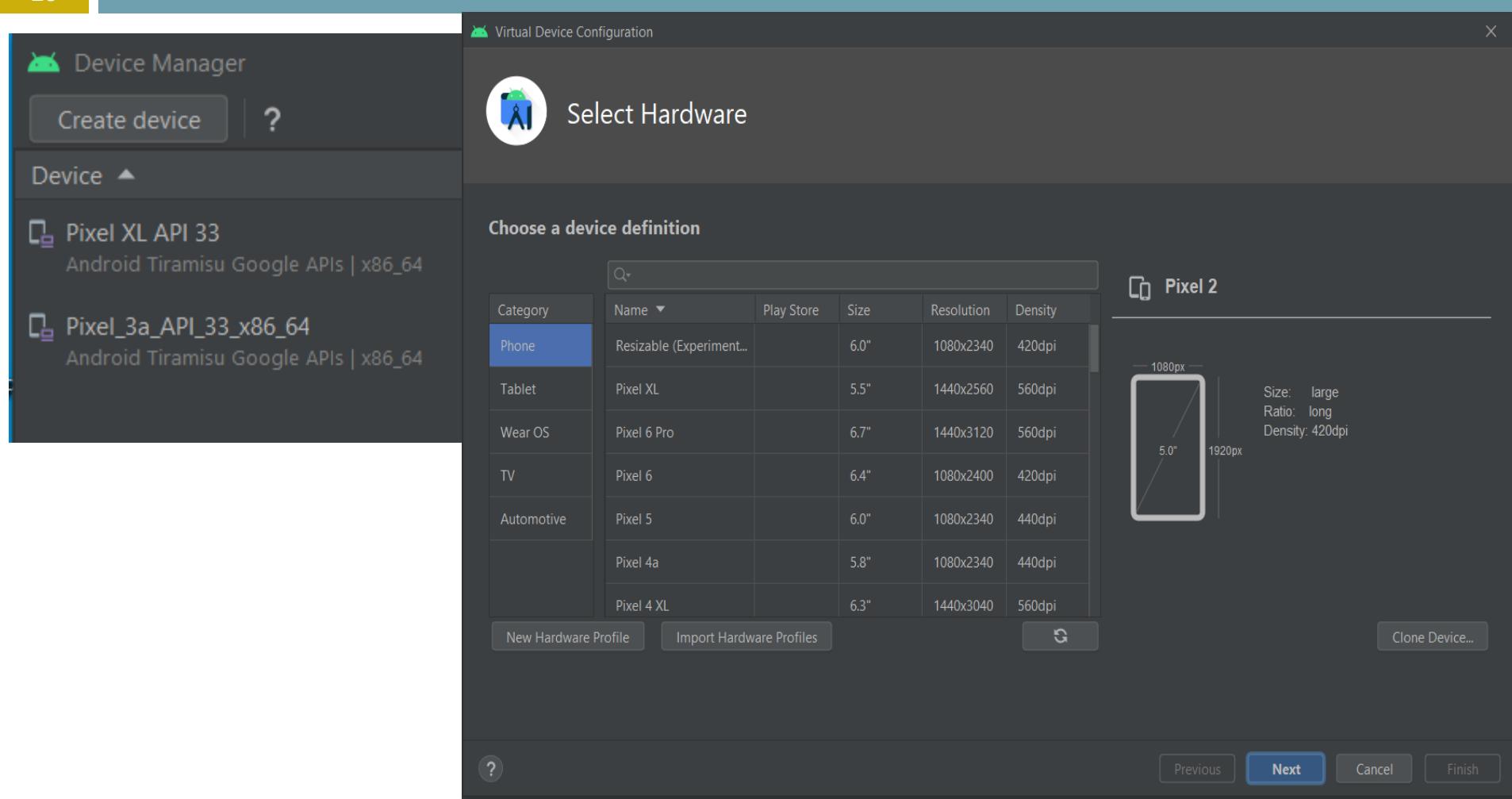
- Khởi động Android Studio lần đầu tiên, bạn chọn Configure để cấu hình máy ảo mô phỏng.



- Khởi động Android Studio lần đầu tiên, bạn chọn Configure để cấu hình máy ảo mô phỏng (Virtual Device manager).



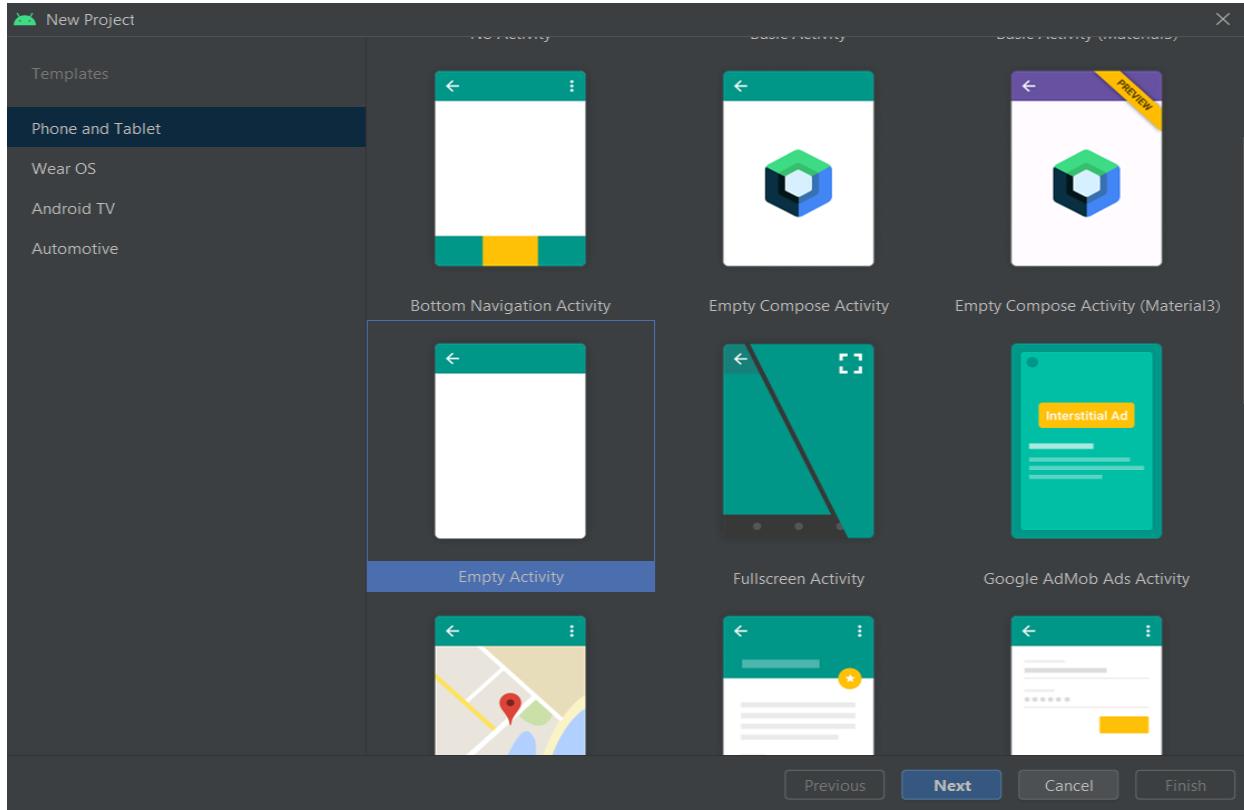
23

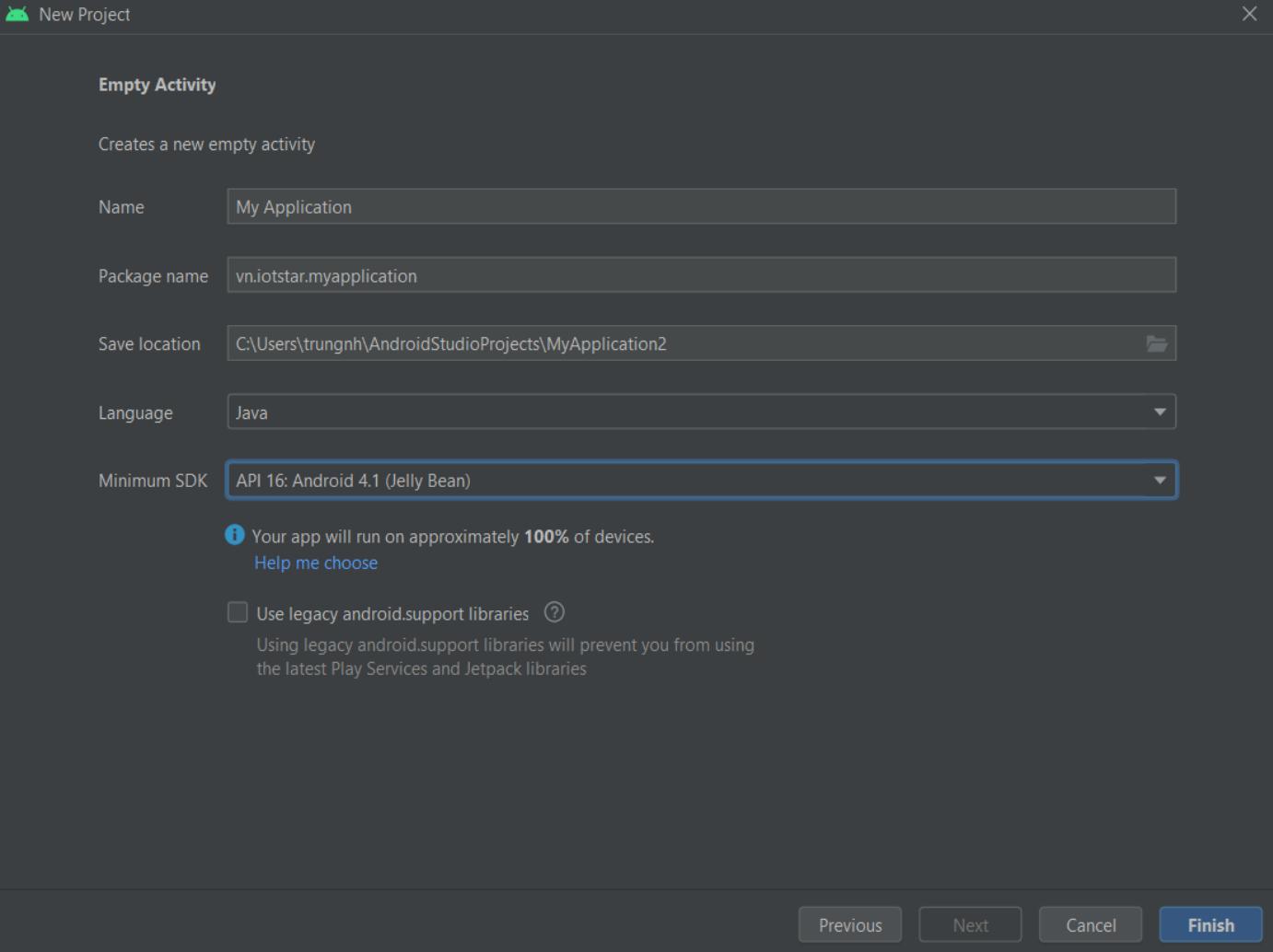


The screenshot shows the "Virtual Device Configuration" window. On the left, the "Device Manager" sidebar lists two devices: "Pixel XL API 33" and "Pixel_3a_API_33_x86_64". The main window is titled "Select Hardware" and displays a table of device definitions under the heading "Choose a device definition". A search bar is at the top of the table. The table has columns for Category, Name, Play Store, Size, Resolution, and Density. The "Phone" category is selected. A row for "Pixel 2" is highlighted, showing its dimensions: 1080px width, 5.0" diagonal, and 1920px height. The table also lists other devices like Pixel XL, Pixel 6 Pro, Pixel 6, Pixel 5, Pixel 4a, and Pixel 4 XL. At the bottom of the table are buttons for "New Hardware Profile", "Import Hardware Profiles", and a refresh icon. Below the table are buttons for "?", "Previous", "Next", "Cancel", and "Finish".

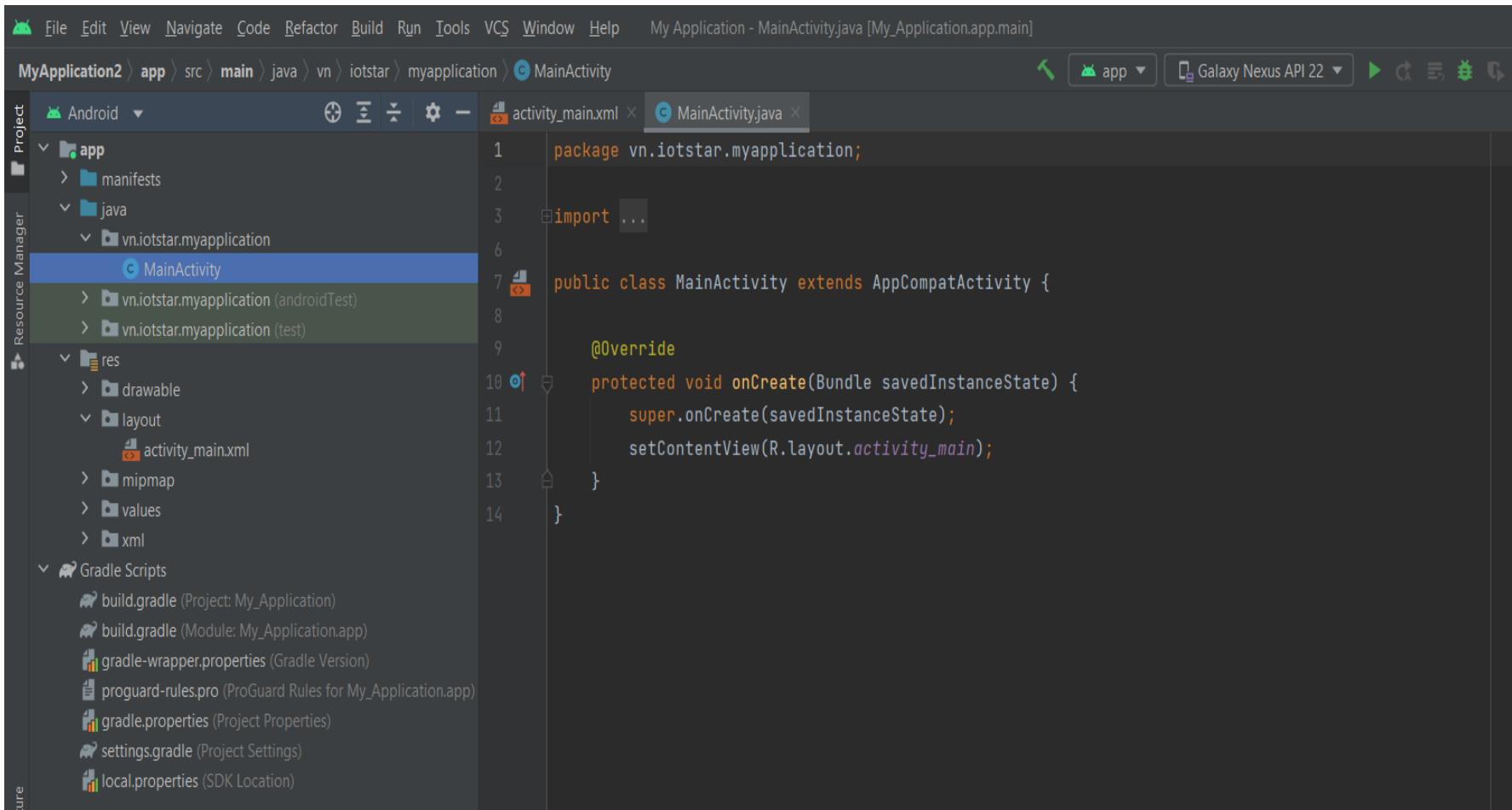
Category	Name	Play Store	Size	Resolution	Density
Phone	Resizable (Experiment...)		6.0"	1080x2340	420dpi
Tablet	Pixel XL		5.5"	1440x2560	560dpi
Wear OS	Pixel 6 Pro		6.7"	1440x3120	560dpi
TV	Pixel 6		6.4"	1080x2400	420dpi
Automotive	Pixel 5		6.0"	1080x2340	440dpi
	Pixel 4a		5.8"	1080x2340	440dpi
	Pixel 4 XL		6.3"	1440x3040	560dpi

- Khởi động Android Studio chọn New project rồi chọn Empty Activity





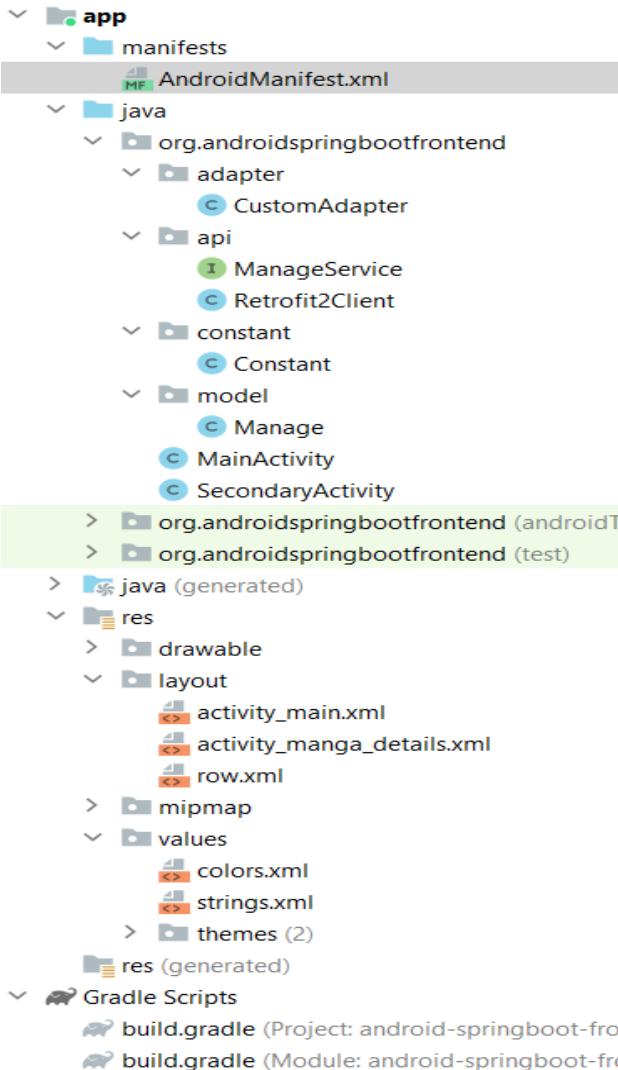
□ Chọn
các
thông tin
rồi bấm
Finish



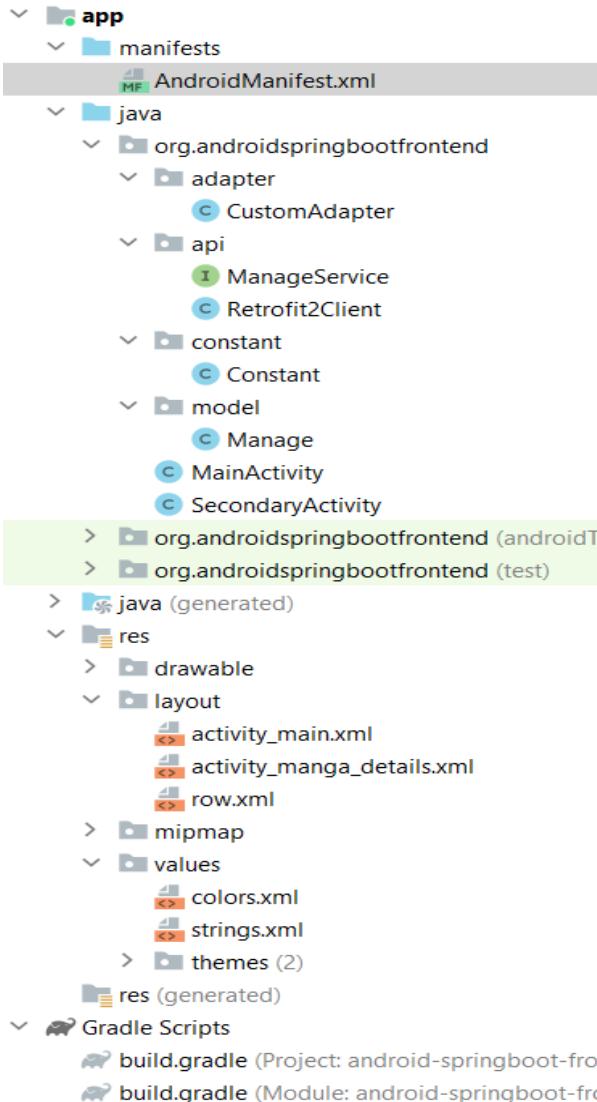
The screenshot shows the Android Studio interface with the following details:

- File Bar:** File Edit View Navigate Code Refactor Build Run Tools VCS Window Help
- Title Bar:** My Application - MainActivity.java [My_Application.app.main]
- Project Tree (Project Manager):**
 - app
 - manifests
 - java
 - vn.iotstar.myapplication
 - MainActivity
 - vn.iotstar.myapplication (androidTest)
 - vn.iotstar.myapplication (test)
 - res
 - drawable
 - layout
 - activity_main.xml
 - mipmap
 - values
 - xml
 - Gradle Scripts
 - build.gradle (Project: My_Application)
 - build.gradle (Module: My_Application.app)
 - gradle-wrapper.properties (Gradle Version)
 - proguard-rules.pro (ProGuard Rules for My_Application.app)
 - gradle.properties (Project Properties)
 - settings.gradle (Project Settings)
 - local.properties (SDK Location)

- Code Editor:** activity_main.xml and MainActivity.java
- Toolbar:** Includes icons for Undo, Redo, Run, Stop, and others.



- **Thư mục manifests:** chứa thông tin cấu hình của ứng dụng. **AndroidManifest.xml:** tập tin XML chứa tất cả các thông tin cấu hình dùng để build ứng dụng và các thành phần của ứng dụng (activity, service, broadcast receivers, content providers v.v.). Mỗi ứng dụng đều có một tập tin AndroidManifest.xml. Trong ứng dụng, Activity nào muốn sử dụng đều bắt buộc phải có khai báo trong AndroidManifest.xml



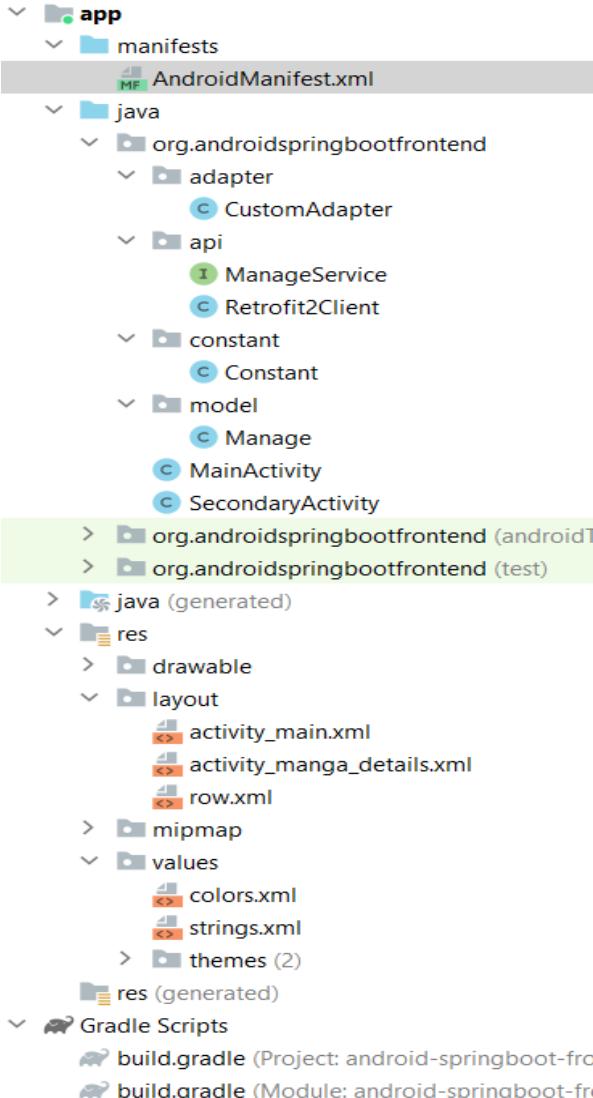
The screenshot shows the Android Studio project structure on the left and the content of the `AndroidManifest.xml` file on the right.

Project Structure:

- app**:
 - manifests**: `AndroidManifest.xml`
 - java**:
 - org.androidspringbootfrontend**:
 - adapter**: `CustomAdapter`
 - api**:
 - `ManageService`
 - `Retrofit2Client`
 - constant**: `Constant`
 - model**:
 - `Manage`
 - `MainActivity`
 - `SecondaryActivity`
 - org.androidspringbootfrontend** (androidTest)
 - org.androidspringbootfrontend** (test)
 - java (generated)**
 - res**:
 - drawable**
 - layout**:
 - `activity_main.xml`
 - `activity_manga_details.xml`
 - `row.xml`
 - mipmap**
 - values**:
 - `colors.xml`
 - `strings.xml`
 - themes (2)**
 - res (generated)**
- Gradle Scripts**:
 - `build.gradle` (Project: android-springboot-fro)
 - `build.gradle` (Module: android-springboot-frc)

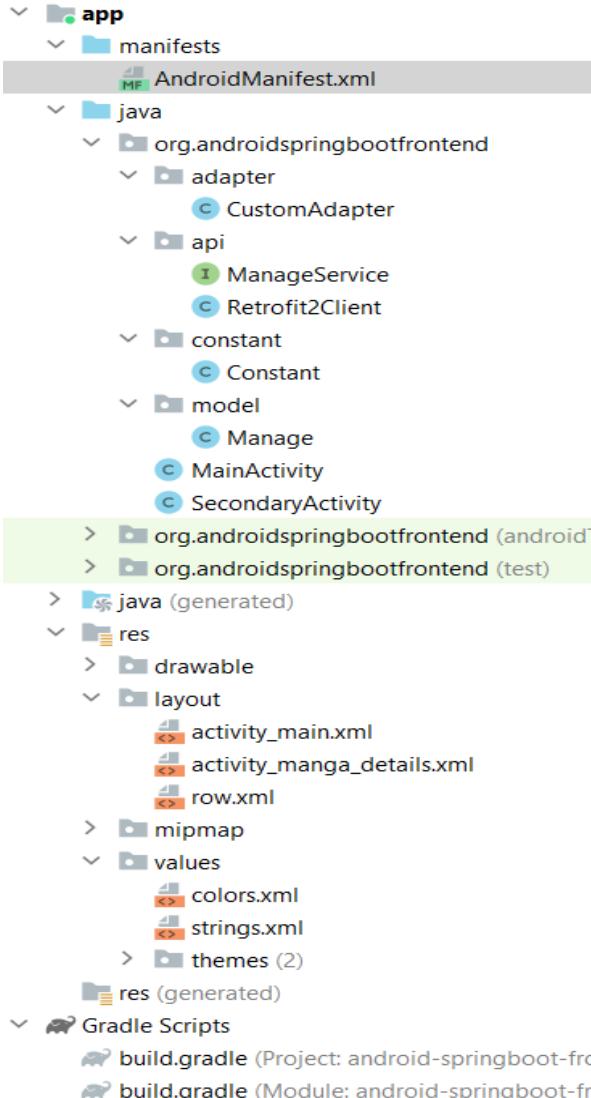
```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android">
    <uses-permission android:name="android.permission.INTERNET" />
    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="android-springboot-frontend"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.Androidspringbootfrontend"
        android:usesCleartextTraffic="true">
        <activity
            android:name=".MainActivity"
            android:label="android-springboot-frontend"
            android:theme="@style/Theme.AppCompat.NoActionBar"
            android:exported="true">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity android:name=".SecondaryActivity" />
    </application>
</manifest>
```

Cấu trúc project



- **Thư mục java:** chứa tất cả các file mã nguồn .java của ứng dụng. Tương ứng với mỗi **Activity** thì file mã nguồn sẽ chứa các xử lý trên Activity đó. Activity nào được khởi chạy đầu tiên khi ứng dụng hoạt động sẽ được khai báo đầu tiên trong tập tin **AndroidManifest.xml**.
- **Thư mục res:** chứa các tài nguyên của ứng dụng, bao gồm các tập tin hình ảnh, các thiết kế giao diện, thực đơn,... của ứng dụng.

Cấu trúc project



- **Thư mục res:** chứa các tài nguyên của ứng dụng, bao gồm các tập tin hình ảnh, các thiết kế giao diện, thực đơn,... của ứng dụng.

Thư mục	Mô tả
<code>drawable/</code>	Thư mục chứa các hình ảnh của ứng dụng
<code>layout/</code>	Thư mục chứa các tập tin XML dùng để thiết kế giao diện của ứng dụng. Hiện giờ đang có tập tin <i>activity_main.xml</i> chính là phần thiết kế giao diện của ứng dụng Hello world .
<code>mipmap/</code>	Thư mục thường chứa các hình ảnh dạng icon của ứng dụng.
<code>values/</code>	Thư mục quản lý các giá trị, đó có thể là các giá trị chuỗi (<i>strings.xml</i>), màu sắc (<i>colors.xml</i>),...

Cấu trúc project

31

app

- manifests
- AndroidManifest.xml
- java

 - org.androidspringbootfrontend

 - adapter
 - CustomAdapter

 - api

 - ManageService
 - Retrofit2Client

 - constant
 - Constant
 - model

 - Manage
 - MainActivity
 - SecondaryActivity

org.androidspringbootfrontend (androidT)

org.androidspringbootfrontend (test)

java (generated)

res

- drawable
- layout

 - activity_main.xml
 - activity_manga_details.xml
 - row.xml

- mipmap
- values

 - colors.xml
 - strings.xml

- themes (2)

res (generated)

Gradle Scripts

- build.gradle (Project: android-springboot-fro)
- build.gradle (Module: android-springboot-frc)

Thiết lập và truy xuất Android Resources

```
res/
    drawable/
        icon.png
    layout/
        activity_main.xml
        info.xml
    values/
        strings.xml
```

Thư mục **res/** chứa tất cả các thư mục con. Trong ví dụ trên chúng ta có 1 **image resource**, 2 **layout resource**, và 1 **string resource**.

- anim/**: Chứa các file xml định nghĩa các hiệu ứng. Chúng được truy xuất từ class **R.anim**.
- color/**: Là các file xml định nghĩa về màu sắc. Chúng được truy xuất từ class **R.color**.
- drawable/**: Chứa các file hình ảnh **.png**, **.jpg**, **.gif** hoặc các file xml đã được biên dịch thành bitmaps. Chúng được truy xuất từ class **R.drawable**.
- layout/**: Chứa các file xml định nghĩa giao diện người dùng. Chúng được truy xuất từ class **R.layout**.

app

- manifests
- AndroidManifest.xml**
- java
 - org.androidspringbootfrontend
 - adapter
 - CustomAdapter
 - api
 - ManageService
 - Retrofit2Client
 - constant
 - Constant
 - model
 - Manage
 - MainActivity
 - SecondaryActivity
 - org.androidspringbootfrontend (androidTest)
 - org.androidspringbootfrontend (test)
- java (generated)
- res
 - drawable
 - layout
 - activity_main.xml
 - activity_manga_details.xml
 - row.xml
 - mipmap
 - values
 - colors.xml
 - strings.xml
 - themes (2)
- res (generated)

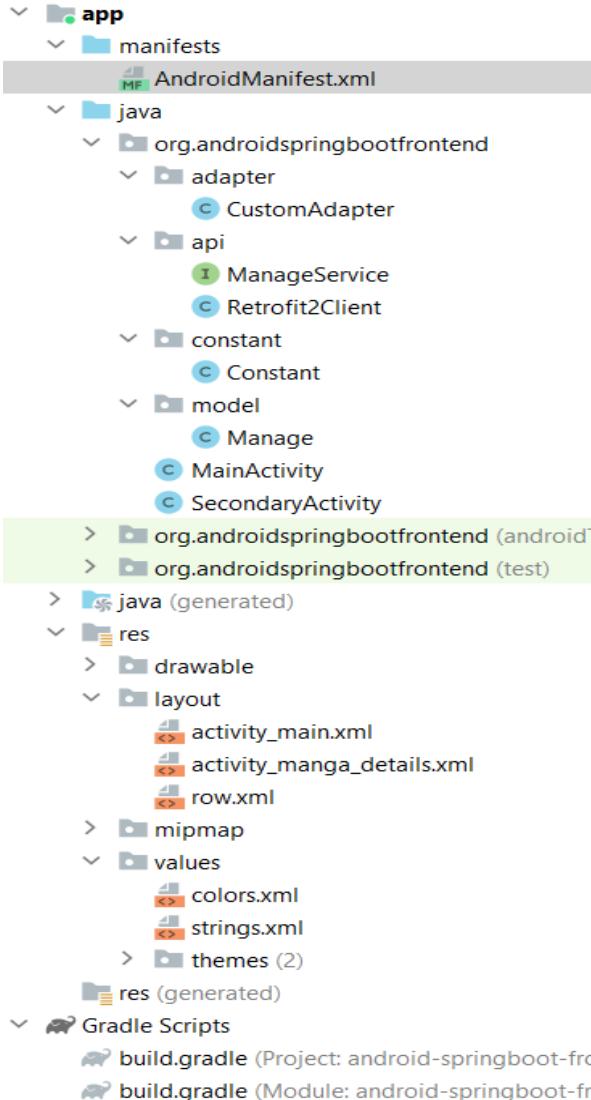
- Gradle Scripts
- build.gradle (Project: android-springboot-fro)
- build.gradle (Module: android-springboot-frc)

Thiết lập và truy xuất Android Resources

```
res/
    drawable/
        icon.png
    layout/
        activity_main.xml
        info.xml
    values/
        strings.xml
```

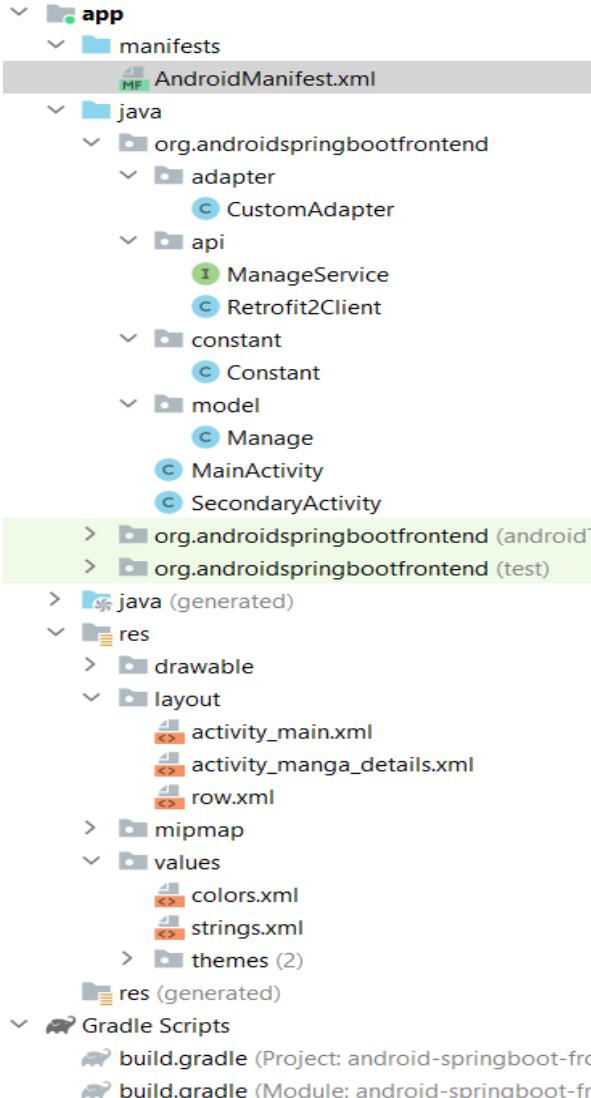
Thư mục **res/** chứa tất cả các thư mục con. Trong ví dụ trên chúng ta có 1 **image resource**, 2 **layout resource**, và 1 **string resource**.

- menu/**: Chứa các file xml định nghĩa menu của ứng dụng, có thể là Options Menu, Context Menu, Sub Menu. Chúng được truy xuất từ class **R.menu**.
- raw/**: Chứa các file bất kỳ, tùy ý. Cần phải gọi **Resources.openRawResource()**. Với **resource ID**, nó là **R.raw.filename** để mở 1 file **raw**.

A screenshot of the Android Studio project structure. The 'app' module is expanded, showing its contents. The 'res' folder is also expanded, showing 'drawable', 'layout', 'mipmap', 'values', and 'themes'. The 'values' folder contains 'colors.xml' and 'strings.xml'. The 'res' folder is marked as '(generated)'. Other sections shown include 'Gradle Scripts' with 'build.gradle' files for the project and module, and 'java' sections for the main application and test classes.

Thiết lập và truy xuất Android Resources

- **values/**: Chứa các file xml để lưu các giá trị đơn giản như string, integers, colors,...
 - + **arrays.xml**: Dùng để định nghĩa các mảng dữ liệu, và được truy xuất từ class **R.array**.
 - + **integers.xml**: Dùng để định nghĩa các số integer và được truy xuất từ class **R.integer**.
 - + **bools.xml**: Dùng để định nghĩa kiểu dữ liệu bool và được truy xuất từ class **R.bool**.
 - + **colors.xml**: Dùng để định nghĩa các giá trị color và được truy xuất từ class **R.color**.



Thiết lập và truy xuất Android Resources

- **values/**: Chứa các file xml để lưu các giá trị đơn giản như string, integers, colors,...
 - + **dimens.xml**: Dùng để định nghĩa các giá trị về kích thước, chiều. Được truy xuất từ class **R.dimen**.
 - + **strings.xml**: Dùng để định nghĩa các string và được gọi từ class **R.string**.
- **styles.xml**: Dùng để định nghĩa các styles và được gọi từ class **R.style**.
- **xml/**: Chứa các file xml bất kỳ và có thể được đọc bằng cách gọi hàm Resources.getXML(). Bạn có thể lưu các file XML cấu hình ở đây và nó sẽ được dùng khi ứng dụng chạy.

Thiết lập và truy xuất Android Resources

Ví dụ: Truy xuất `res/drawable/myImage.png`

```
ImageView imageView = (ImageView)  
findViewById(R.id.myimageview);  
  
imageView.setImageResource(R.drawable.myimage);
```

Ví dụ: Truy xuất `res/values/strings.xml`

```
<?xml version="1.0" encoding="utf-8"?>  
<resources>  
    <string name="hello">Hello, World!</string>  
</resources>
```

Để sử dụng string: '`hello`'

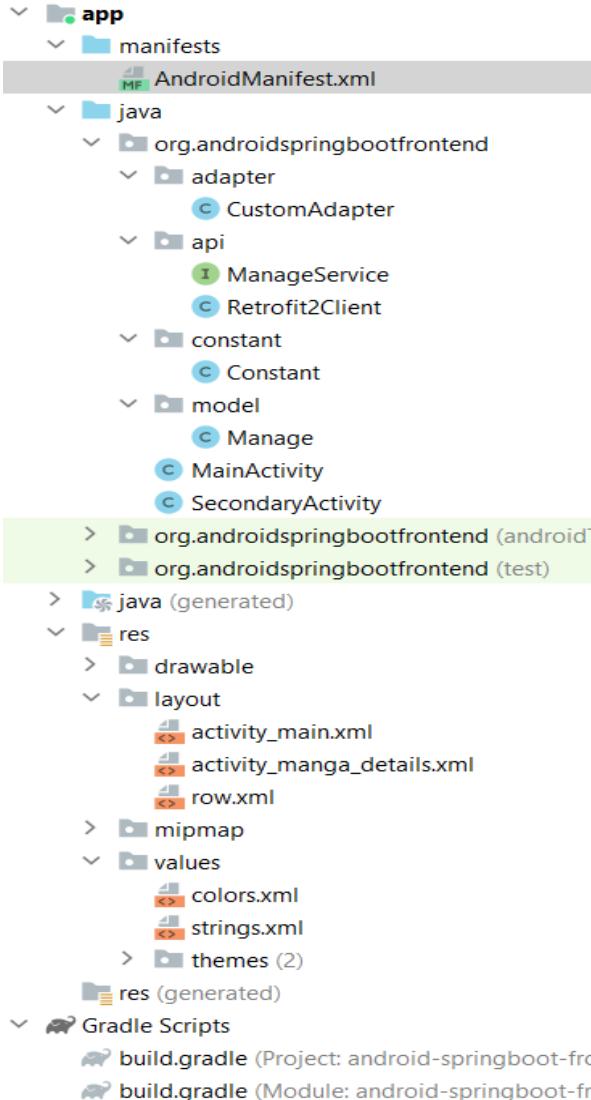
```
TextView msgTextView = (TextView) findViewById(R.id.msg);  
msgTextView.setText(R.string.hello);
```

Truy xuất resources trong XML

Ví dụ: Truy xuất `res/values/strings.xml`

```
<?xml version="1.0" encoding="utf-8"?>  
  
<resources>  
  
    <color name="opaque_red">#f00</color>  
  
    <string name="hello">Hello!</string>  
  
</resources>
```

```
        android:textColor="@color  
        /opaque_red"  
        android:text="@string/hel  
        lo" />
```

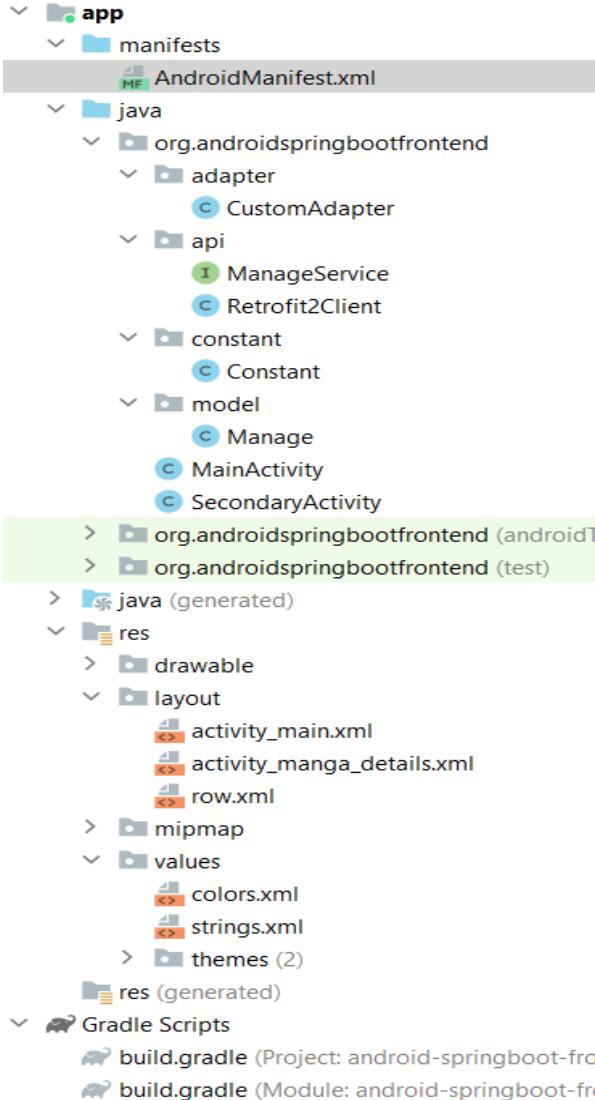


Android R.java là một tập tin được tạo tự động bởi aapt (Android Asset Packaging Tool) . Nó chứa tất cả các ID trong thư mục res /. Nó chứa nhiều class tĩnh : **menu, id, layout, attr, drawable, string .v.v**

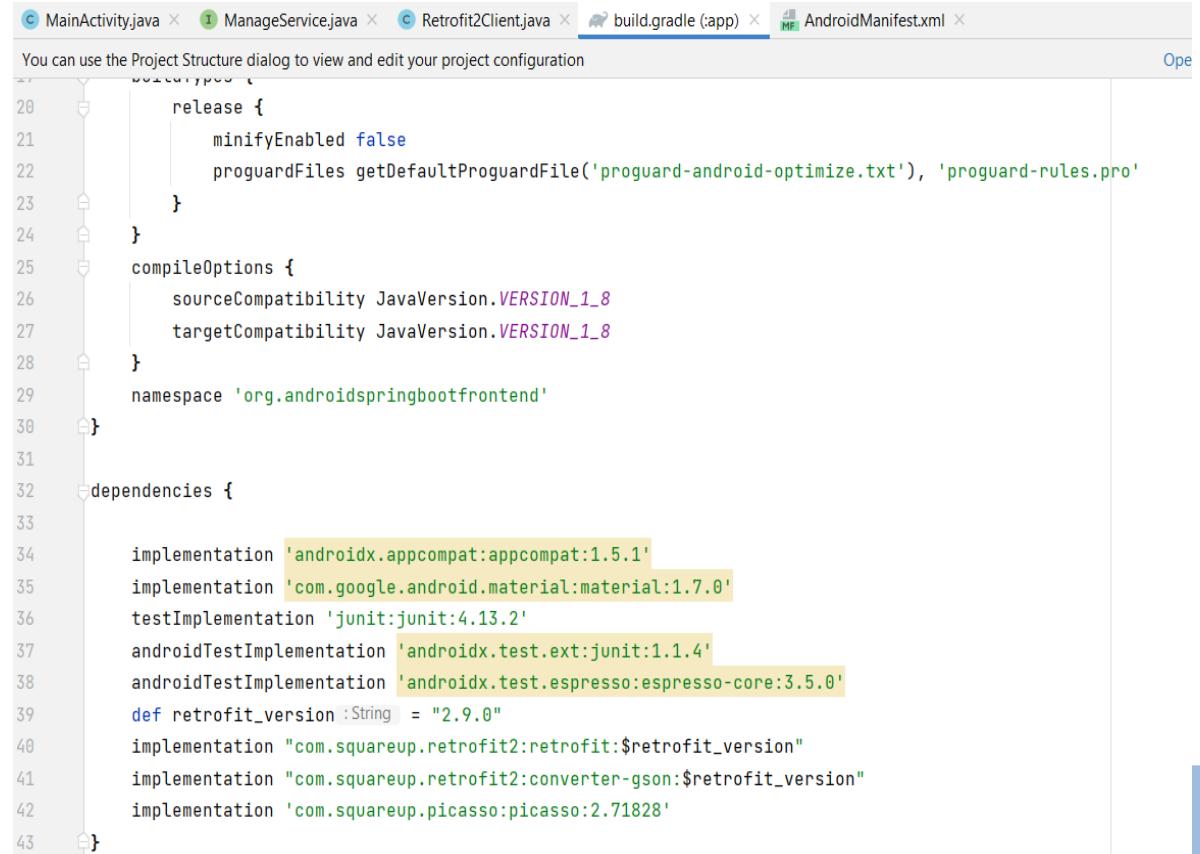
Truy xuất: **R.id, R.layout, R.menu,...**

```
setContentView(R.layout.activity_main);
```

```
Toolbar toolbar = findViewById(R.id.toolbar);
```



- **Build.gradle:** thông tin cấu hình project, các dependency của project.



The screenshot shows the Android Studio code editor with the 'build.gradle (app)' tab selected. The code in the file is as follows:

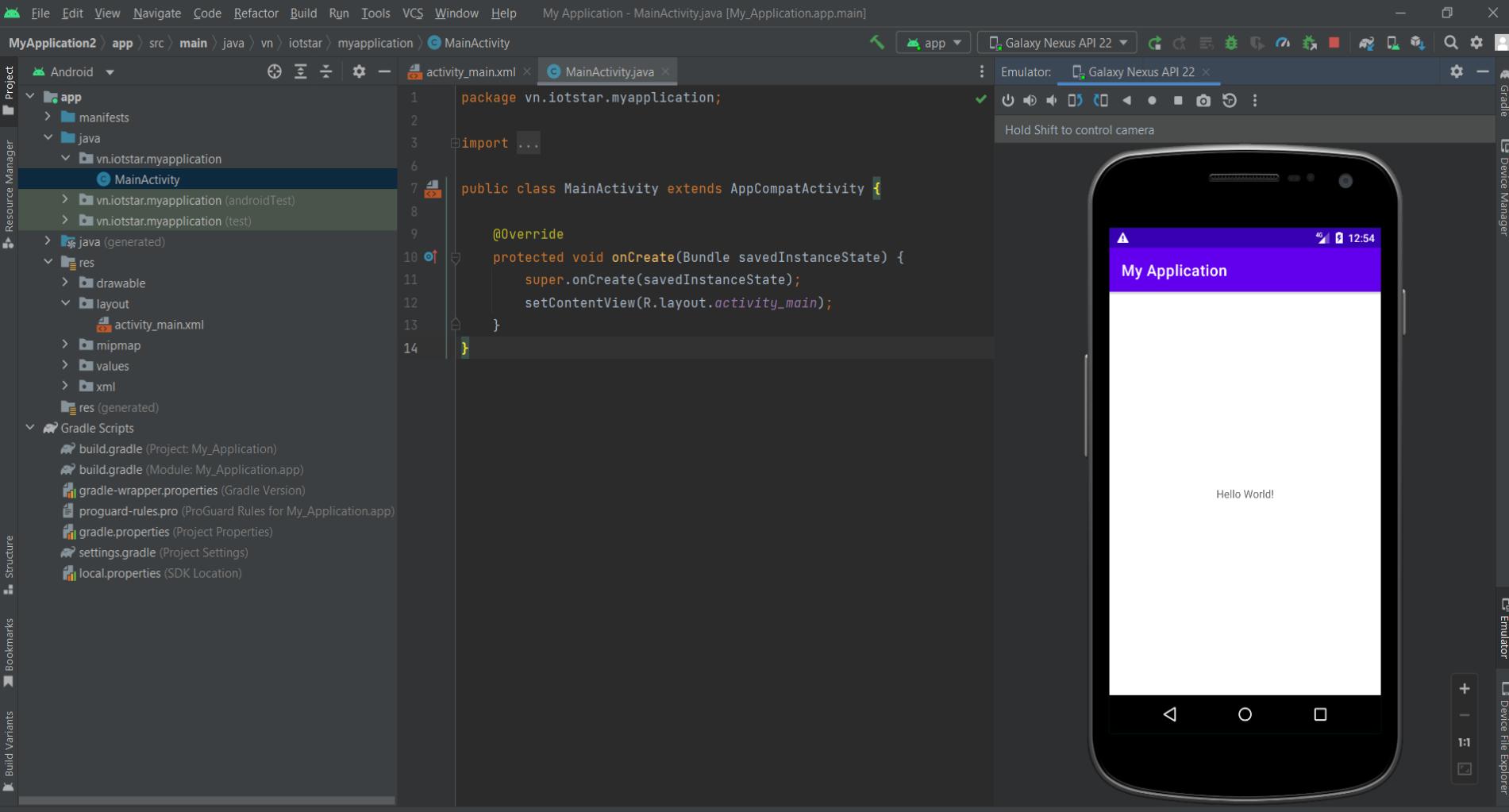
```
release {
    minifyEnabled false
    proguardFiles getDefaultProguardFile('proguard-android-optimize.txt'), 'proguard-rules.pro'
}

compileOptions {
    sourceCompatibility JavaVersion.VERSION_1_8
    targetCompatibility JavaVersion.VERSION_1_8
}

namespace 'org.androidspringbootfrontend'

dependencies {

    implementation 'androidx.appcompat:appcompat:1.5.1'
    implementation 'com.google.android.material:material:1.7.0'
    testImplementation 'junit:junit:4.13.2'
    androidTestImplementation 'androidx.test.ext:junit:1.1.4'
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.5.0'
    def retrofit_version :String = "2.9.0"
    implementation "com.squareup.retrofit2:retrofit:$retrofit_version"
    implementation "com.squareup.retrofit2:converter-gson:$retrofit_version"
    implementation 'com.squareup.picasso:picasso:2.71828'
}
```



The screenshot shows the Android Studio interface with the following details:

- Project Structure:** The project is named "My Application2". The `MainActivity.java` file is selected in the Project tree.
- Code Editor:** The `MainActivity.java` file contains the following Java code:

```
package vn.iotstar.myapplication;

import ...

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
}
```
- Emulator:** An Android emulator for a "Galaxy Nexus API 22" device is running, displaying the application's UI.
- UI Preview:** The emulator screen shows a purple header bar with the text "My Application" and a white content area with the text "Hello World!".
- Toolbars and Panels:** The top bar includes standard Android Studio icons for File, Edit, View, Navigate, Code, Refactor, Build, Run, Tools, VCS, Window, and Help. The bottom right corner features the CN logo.

```
@Override
```

```
protected void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);
```

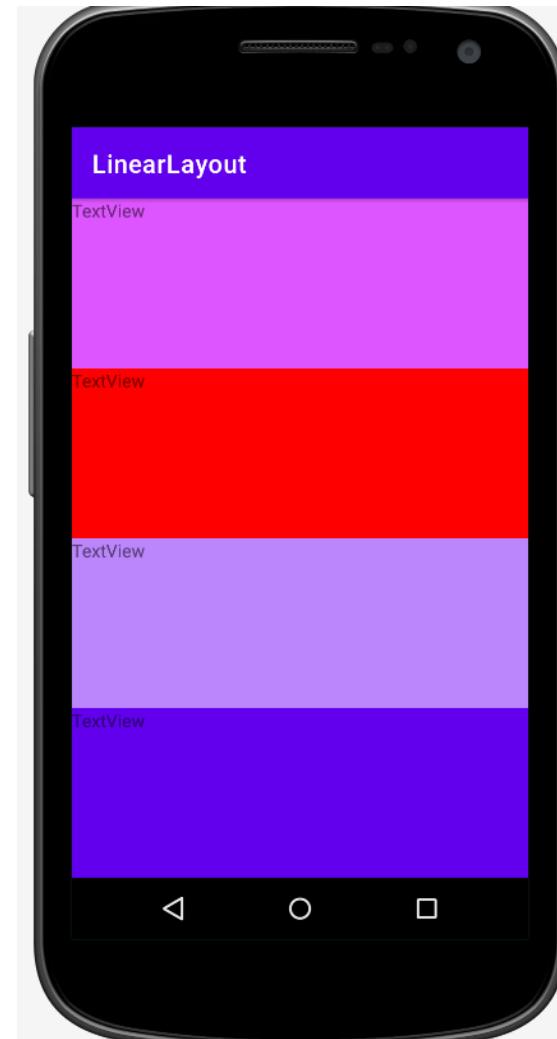
```
requestWindowFeature(Window.FEATURE_NO_TITLE); //will hide  
the title not the title bar
```

```
this.getWindow().setFlags(WindowManager.LayoutParams.FLAG_F  
ULLSCREEN,
```

```
    WindowManager.LayoutParams.FLAG_FULLSCREEN); //int  
flag, int mask
```

```
    setContentView(R.layout.activity_main);
```

```
}
```



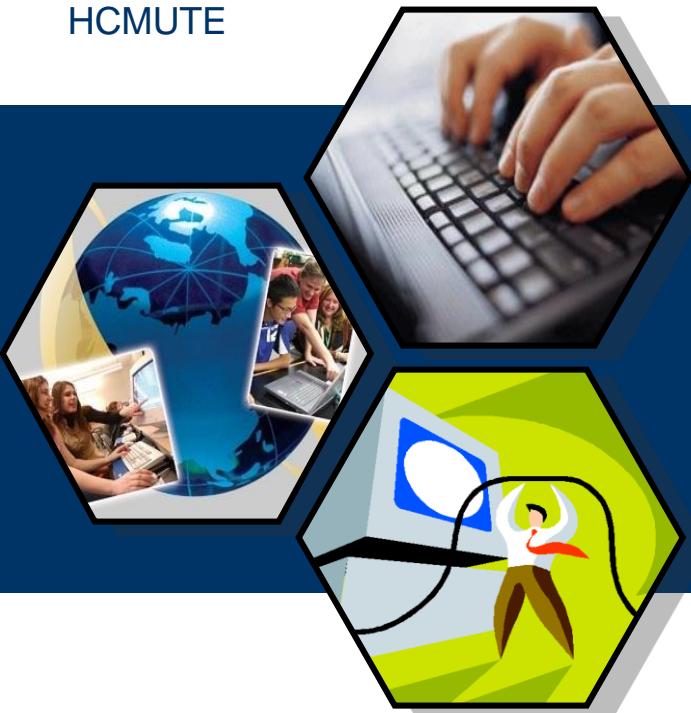


HCMUTE

KHOA CÔNG NGHỆ THÔNG TIN

UTEx

QUY TRÌNH THIẾT KẾ ỨNG DỤNG

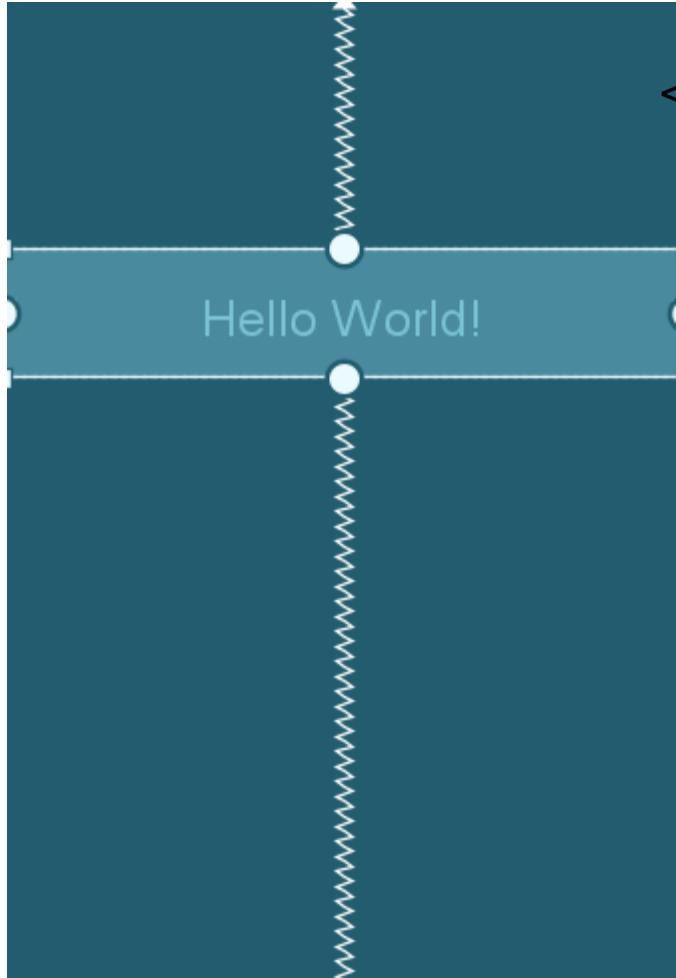


Khoa Công nghệ Thông tin
Đại học Sư phạm Kỹ thuật TP.HCM

ThS. Nguyễn Hữu Trung



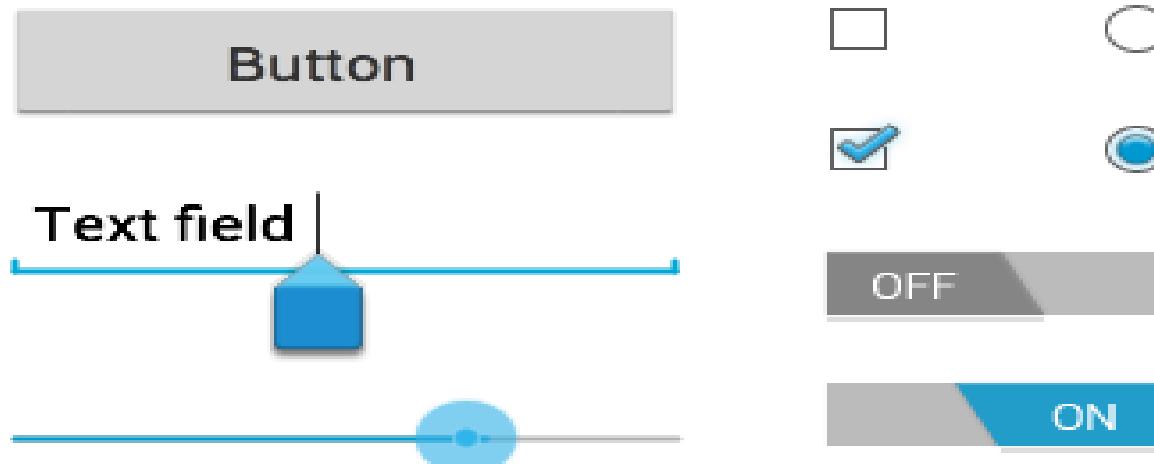
- **Bước 1 Kéo thả: thiết kế giao diện (Layout, Control)**
- **Bước 2 Ánh xạ: các view trên giao diện**
- **Bước 3 Viết code: điều khiển các view và bài toán.**



```
<TextView  
    android:id="@+id/textView1"  
    android:layout_width="0dp"  
    android:layout_height="wrap_content"  
    android:padding="16dp"  
    android:text="Hello World!"  
    android:textAlignment="center"  
    android:textColor="#FF9800"  
    android:textSize="34sp"  
    app:layout_constraintBottom_toBottomOf="parent"  
    app:layout_constraintEnd_toEndOf="parent"  
    app:layout_constraintHorizontal_bias="0.44"  
    app:layout_constraintStart_toStartOf="parent"  
    app:layout_constraintTop_toTopOf="parent"  
    app:layout_constraintVertical_bias="0.234" />
```

```
public class MainActivity extends AppCompatActivity {  
  
    TextView txtNoiDung1; //khai báo toàn cục  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
  
        //ánh xạ dựa vào Id  
        txtNoiDung1 = (TextView) findViewById(R.id.textView1);  
  
        //viết code điều khiển textView1  
        txtNoiDung1.setText("Chào bạn");  
    }  
}
```

- Input Control là các thành phần có tính tương tác trong giao diện UI của ứng dụng. Android cung cấp nhiều control đa dạng để bạn có thể sử dụng trong UI như button, text field, seek bar, checkbox, zoom button, toggle button, ...



- **TextView** được sử dụng để hiển thị text tới người dùng.

```
<TextView android:id="@+id/text_id"  
         android:layout_width="wrap_content"  
         android:layout_height="wrap_content"  
         android:text="Hello World" />
```

```
TextView myText = (TextView) findViewById(R.id.text_id);
```

- EditText Là một lớp con được định nghĩa trước của TextView mà bao gồm các khả năng chỉnh sửa đa dạng.

```
<EditText  
    android:id="@+id/textview"  
    android:layout_height="wrap_content"  
    android:layout_width="match_parent"  
    android:inputType="text"  
    android:maxLines="1"  
    android:lines="1"/>
```

Thuộc tính **android:inputType** dùng để thiết lập kiểu nhập liệu, hiện thị

- **AutoCompleteTextView** trong Android hỗ trợ cho người dùng những gợi ý liên quan khi bạn nhập vào trường EditText.

```
<AutoCompleteTextView  
    android:id="@+id/autoCompleteTextView1"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:layout_alignParentLeft="true"  
    android:layout_below="@+id/textView1"  
    android:layout_marginLeft="36dp"  
    android:layout_marginTop="17dp"  
    android:ems="10"  
    android:text="">  
    <requestFocus />  
</AutoCompleteTextView>
```

```
java.lang.Object  
↳ android.view.View  
    ↳ android.widget.TextView  
        ↳ android.widget.EditText  
            ↳ android.widget.AutoCompleteTextView
```

<Button

```
    android:id="@+id	btnThucHien"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginEnd="180dp"
    android:text="Thực hiện"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/textView1"
    app:layout_constraintVertical_bias="0.353" />
```

//viết code điều khiển btnClick

```
btnClick.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        //viết code điều khiển textView1
        txtNoiDung1.setText("Chào bạn");
    }
});
```

UserName

Password

ĐĂNG NHẬP

<EditText

```
    android:id="@+id/editTextName"
    android:layout_width="345dp"
    android:layout_height="51dp"
    android:layout_marginTop="32dp"
    android:ems="10"
    android:hint="UserName"
    android:inputType="textPersonName"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"/>
```

<EditText

```
    android:id="@+id/editTextPassword"
    android:layout_width="345dp"
    android:layout_height="51dp"
    android:layout_marginTop="12dp"
    android:ems="10"
    android:hint="Password"
    android:inputType="textPersonName"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/editTextName" />
```

UserName

Password

ĐĂNG NHẬP

```
<Button  
    android:id="@+id/btnDangNhap"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:layout_marginTop="32dp"  
    android:text="Đăng nhập"  
    android:textColor="#fff"  
    app:layout_constraintEnd_toEndOf="parent"  
    app:layout_constraintHorizontal_bias="0.86"  
    app:layout_constraintStart_toStartOf="parent"  
    app:layout_constraintTop_toBottomOf=  
    "@+id/editTextPassword" />
```

```
btnDangnhap =(Button) findViewById(R.id.btnDangNhaph);  
editTextname = (EditText) findViewById(R.id.editTextName);  
editTextpass = (EditText) findViewById(R.id.editTextPassword);
```

```
btnDangnhap.setOnClickListener(new View.OnClickListener() {  
    @Override  
    public void onClick(View view) {  
        //viết code  
        if(editTextname.getText().toString().equals("admin")  
            && editTextpass.getText().toString().equals("admin")){  
            txtNoiDung1.setText("Đăng nhập thành công!");  
        }else{  
            txtNoiDung1.setText("Đăng nhập thất bại" +  
editTextpass.getText().toString());  
        }  
    }  
});
```

My Application

8

SINH SỐ NGẪU NHIÊN

```
<TextView  
    android:id="@+id/textViewSo"  
    android:layout_width="match_parent"  
    android:layout_height="40dp"  
    android:textAlignment="gravity"  
    android:gravity="center"  
  
    android:text="" />
```

```
<Button  
    android:id="@+id/buttonRnd"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:layout_marginTop="92dp"  
    android:text="Sinh số ngẫu nhiên"
```

My Application

8

SINH SỐ NGẪU NHIÊN

```
//ánh xạ
txtSoN = (TextView)
findViewById(R.id.textViewSo);
btnRnd = (Button)
findViewById(R.id.buttonRnd);

//viết code sinh ngẫu nhiên
btnRnd.setOnClickListener(new
View.OnClickListener() {
    @Override
    public void onClick(View view) {
        //tạo số ngẫu nhiên
        Random random = new Random();
        int number = random.nextInt(10);
        txtSoN.setText(number + ""); //number +
        ép kiểu
    }
});
```

My Application

Nhập số thứ nhất

Nhập số thứ hai

SINH SỐ NGẪU NHIÊN

My Application

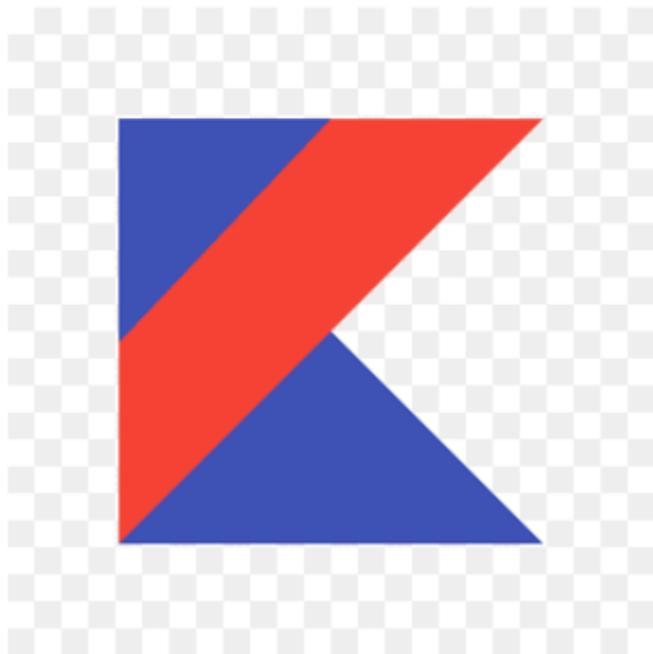
5

3

7

SINH SỐ NGẪU NHIÊN

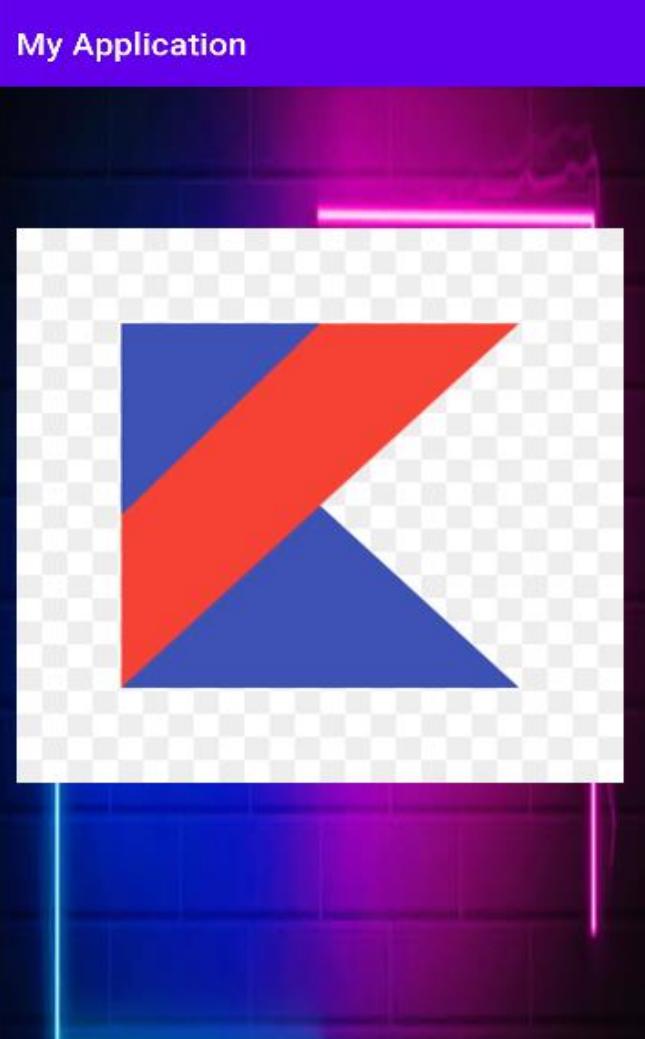
My Application



```
<ImageView  
    android:scaleType="center"  
    android:id="@+id/imageView1"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:layout_marginTop="92dp"  
    android:src="@drawable/kotlin"  
    app:layout_constraintEnd_toEndOf="parent"  
    app:layout_constraintStart_toStartOf="parent"  
    app:layout_constraintTop_toTopOf="parent" />
```

```
ImageView img1= (ImageView)  
findViewById(R.id.imageView1);  
  
img1.setImageResource(R.drawable.kotlin);
```

My Application

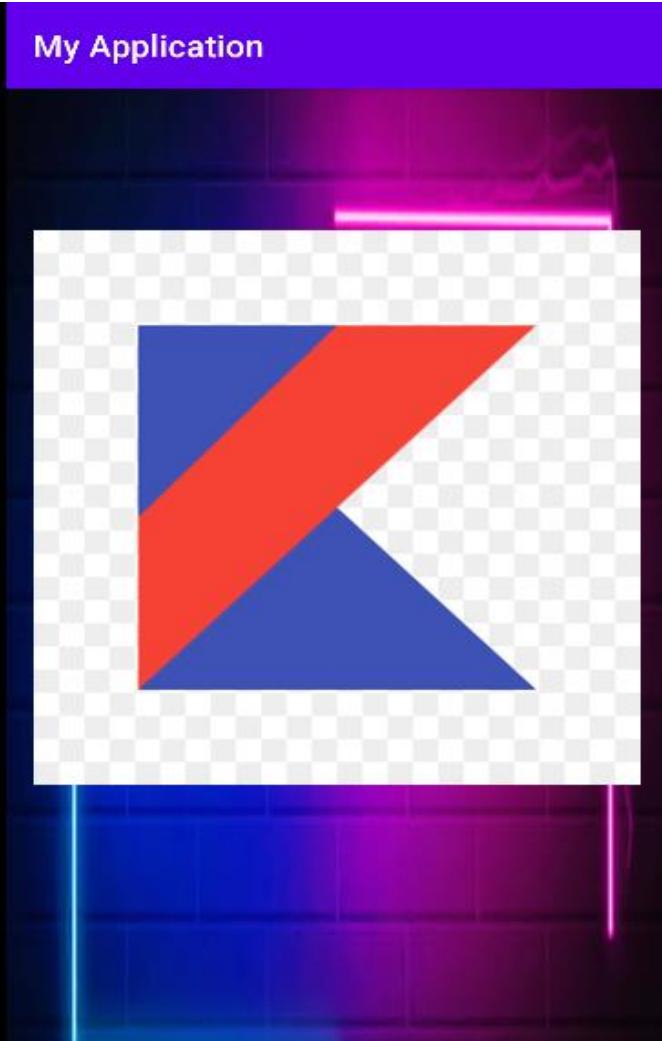


```
<androidx.constraintlayout.widget.ConstraintLayout  
    xmlns:android="http://schemas.android.com/apk/res/  
    android"  
        xmlns:app="http://schemas.android.com/apk/res-  
        auto"  
        xmlns:tools="http://schemas.android.com/tools"  
        android:layout_width="match_parent"  
        android:layout_height="match_parent"  
        android:background="@drawable/bg1"  
        android:id="@+id/constraintLayout1"  
    >
```

```
ConstraintLayout bg = (ConstraintLayout)  
findViewById(R.id.constraintLayout1);
```

```
bg.setBackgroundColor(Color.BLUE);
```

```
bg.setBackgroundResource(R.drawable.bg2);
```



Random background

```
ArrayList<Integer> arrayList = new ArrayList<>();  
arrayList.add(R.drawable.bg1);  
arrayList.add(R.drawable.bg2);  
arrayList.add(R.drawable.bg3);  
arrayList.add(R.drawable.bg4);
```

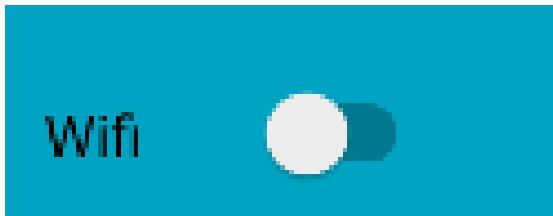
```
Random random = new Random();  
int vitri = random.nextInt(arrayList.size());
```

```
bg.setBackgroundResource(arrayList.get(vitri));
```



```
<ImageButton  
    android:id="@+id/imageButton1"  
    android:layout_width="80dp"  
    android:layout_height="80dp"  
    android:background="@drawable/on"  
    app:layout_constraintBottom_toBottomOf="parent"  
    app:layout_constraintEnd_toEndOf="parent"  
    app:layout_constraintHorizontal_bias="0.45"  
    app:layout_constraintStart_toStartOf="parent"  
    app:layout_constraintTop_toTopOf="parent"  
    app:layout_constraintVertical_bias="0.732" />
```

```
ImageButton img2 = (ImageButton)  
findViewById(R.id.imageButton1);  
img2.setOnClickListener(new View.OnClickListener() {  
    @Override  
    public void onClick(View v) {  
        mg1.setImageResource(R.drawable.dart);  
        img1.getLayoutParams().width=550;  
        img1.getLayoutParams().height=550;  
    }  
});
```



```
<Switch  
    android:id="@+id/switch1"  
    android:layout_width="97dp"  
    android:layout_height="38dp"  
    android:text="Wifi"/>
```

```
//switch  
Switch sw = (Switch) findViewById(R.id.switch1);  
sw.setOnCheckedChangeListener(new CompoundButton.OnCheckedChangeListener() {  
    @Override  
    public void onCheckedChanged(CompoundButton buttonView, boolean isChecked) {  
        if(isChecked){ //isChecked = true  
            Toast.makeText(ControlActivity.this,"Wifi đang bật",Toast.LENGTH_LONG).show()  
        }else{  
            Toast.makeText(ControlActivity.this,"Wifi đang tắt",Toast.LENGTH_LONG).show();  
        }  
    }  
});
```



```
<CheckBox  
    android:id="@+id/checkBox"  
    android:layout_width="151dp"  
    android:layout_height="66dp"  
    android:text="Background 1"
```

```
CheckBox ck1 = (CheckBox) findViewById(R.id.checkBox2);  
ck1.setOnCheckedChangeListener(new  
CompoundButton.OnCheckedChangeListener() {  
    @Override  
    public void onCheckedChanged(CompoundButton buttonView, boolean isChecked)  
{  
    if(isChecked){  
        bg.setBackgroundResource(R.drawable.bg3);  
    }else{  
        bg.setBackgroundResource(R.drawable.bg4);  
    }  
}  
});
```

CheckBox

61



```
//RadioGroup
RadioGroup radioGroup = (RadioGroup)
findViewById(R.id.radioGroup1);
radioGroup.setOnCheckedChangeListener(new
RadioGroup.OnCheckedChangeListener() {
    @Override
    public void onCheckedChanged(RadioGroup group,
    int checkedId) {
        //checkID trả về ID radio
        switch (checkedId){
            case R.id.radioButton:
                bg.setBackgroundResource(R.drawable.bg3);
                break;
            case R.id.radioButton2:
                bg.setBackgroundResource(R.drawable.bg4);
                break;
        }
    }
});
```

```
<RadioGroup
    android:id="@+id/radioGroup1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent" >
    <RadioButton
        android:id="@+id radioButton"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:checked="true"
        android:text="Background 1" />
    <RadioButton
        android:id="@+id radioButton2"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Background 2" />
</RadioGroup>
```



```
<ProgressBar  
    android:id="@+id/progressBar2"  
    android:progress="40"  
    android:max="100"  
    android:secondaryProgress="60"  
    style="@android:style/Widget.ProgressBar.Horizontal"  
    android:layout_width="0dp"  
    android:layout_height="wrap_content"/>
```

```
//progressbar  
ProgressBar progressBar = (ProgressBar) findViewById(R.id.progressBar2);  
img2.setOnClickListener(new View.OnClickListener() {  
    @Override  
    public void onClick(View v) {  
        int current = progressBar.getProgress();  
        progressBar.setProgress(current + 10);  
    }  
});
```

ProgressBar

63



```
<ProgressBar
    android:id="@+id/progressBar2"
    android:progress="40"
    android:max="100"
    android:secondaryProgress="60"
    style="@android:style/Widget.ProgressBar.Horizontal"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
```

//progrebar

```
ProgressBar progressBar = (ProgressBar) findViewById(R.id.progressBar2);
img2.setOnClickListener(new View.OnClickListener() {
```

@Override

```
public void onClick(View v) {
```

```
    int current = progressBar.getProgress();
```

//chạy đều đều >100 quay về 0

```
    if (current>= progressBar.getMax()) {
```

```
        current=0;
```

```
}
```

```
    progressBar.setProgress(current + 10); //thiết lập progress
```

```
}
```

```
};
```

```
//progressbar
ProgressBar progressBar = (ProgressBar) findViewById(R.id.progressBar2);
img2.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        //tự đếm progress
        CountDownTimer countDownTimer = new CountDownTimer(10000,1000) {
            @Override
            public void onTick(long millisUntilFinished) {
                int current = progressBar.getProgress();
                //chạy đều đều >100 quay về 0
                if (current>= progressBar.getMax()){
                    current=0;
                }
                progressBar.setProgress(current + 10); //thiết lập progress
            }
            @Override
            public void onFinish() {
                Toast.makeText(ControlActivity.this,"Hết giờ",Toast.LENGTH_LONG).show();
            }
        };
        countDownTimer.start();
    }
});
```



```
<SeekBar  
    android:id="@+id/seekBar"  
    style="@android:style/Widget.DeviceDefault.Light.SeekBar"  
    android:layout_width="0dp"  
    android:layout_height="wrap_content"  
    android:max="100"  
    android:progress="40"  
    android:secondaryProgress="70"  
    android:thumb="@android:mipmap/sym_def_app_icon"
```

//SeekBar

```
SeekBar seekBar = (SeekBar) findViewById(R.id.seekBar);  
seekBar.setOnSeekBarChangeListener(new SeekBar.OnSeekBarChangeListener() {  
    @Override  
    public void onProgressChanged(SeekBar seekBar, int progress, boolean fromUser) {  
        //progress: giá trị của seekbar  
        Log.d("AAA","Giá trị:" + progress);  
    }  
    @Override  
    public void onStartTrackingTouch(SeekBar seekBar) {  
        Log.d("AAA","Start");  
    }  
    @Override  
    public void onStopTrackingTouch(SeekBar seekBar) {  
        Log.d("AAA","Stop");  
    }});
```

- Bước 1: Vẽ button lên ViewGroup
- Bước 2: Tạo Drawble Resource File

```
<shape xmlns:android="http://schemas.android.com/apk/res/android"  
    android:shape="rectangle">  
    <solid android:color="#ff00" />  
    <size android:height="550dp"  
        android:width="200dp"/>  
    <corners android:radius="30dp" />  
    <gradient android:centerColor="@color/purple_200"/>  
</shape>
```

- Bước 3: Gọi Drawble

```
    android:background="@drawable/button_custom"
```

- Bước 1: Vẽ button lên ViewGroup
- Bước 2: Tạo Drawble Resource File

```
<shape xmlns:android="http://schemas.android.com/apk/res/android"  
    android:shape="rectangle">  
    <solid android:color="#ff00" />  
    <size android:height="550dp"  
        android:width="200dp"/>  
    <corners android:radius="30dp" />  
    <gradient android:centerColor="@color/purple_200"/>  
</shape>
```

- Bước 3: Gọi Drawble

```
    android:background="@drawable/button_custom"
```

- Bước 1: Tạo thư mục menu trong Resources
- Bước 2: Tạo Menu Resource File

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:title="Setting"/>
    <item android:title="Share"/>
    <item android:title="Logout"/>
</menu>
```

- Bước 3: Gọi menu

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    getMenuInflater().inflate(R.menu.menu_setting,menu);
    return super.onCreateOptionsMenu(menu);
}
```

□ Bước 1: Bắt sự kiện menu

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto">
    <item android:title="Setting"
        android:id="@+id/menuSetting"
        android:icon="@android:drawable/ic_menu_set_as"
        app:showAsAction="always"/>
    <item android:title="Share"
        android:orderInCategory="1"
        android:id="@+id/menuShare"/>
    <item android:title="Logout"
        android:orderInCategory="2"
        android:id="@+id/menuLogout"/>
    <item android:title="Menu con"
        android:orderInCategory="3">
        <menu>
            <item android:title="Menu con 1"/>
            <item android:title="Menu con 2"/>
        </menu>
    </item>
</menu>
```

□ Bước1: Bắt sự kiện menu

@Override

```
public boolean onOptionsItemSelected(@NonNull MenuItem item) {  
    switch (item.getItemId()) {  
        case R.id.menuSetting:  
            //lệnh  
            break;  
        case R.id.menuShare:  
            break;  
        case R.id.menuLogout:  
            break;  
    }  
  
    return super.onOptionsItemSelected(item);  
}
```

□ Bước1: Gọi Popup menu

//popup menu

```
private void ShowPopupMenu(){  
    PopupMenu popupMenu = new PopupMenu(this,btnButton);  
  
    popupMenu.getMenuInflater().inflate(R.menu.menu_setting,popupMenu.getMen  
    u());  
    popupMenu.show();  
}
```

□ Bước 2: Bắt sự kiện Popup menu

```
//popup menu
private void ShowPopupMenu(){
    PopupMenu popupMenu = new PopupMenu(this,btnButton);
    popupMenu.getMenuInflater().inflate(R.menu.menu_setting,popupMenu.getMenu());
    //bắt sự kiện
    popupMenu.setOnMenuItemClickListener(new PopupMenu.OnMenuItemClickListener() {
        @Override
        public boolean onMenuItemClick(MenuItem item) {
            switch (item.getItemId()){
                case R.id.menuSetting:
                    //lệnh
                    Toast.makeText(ControlActivity2.this,"Bạn đang chọn Setting",Toast.LENGTH_LONG).show();
                    break;
                case R.id.menuShare:
                    btnButton.setText("Chia sẻ");
                    break;
                case R.id.menuLogout:
                    break;
            }
            return false;
        }
    });
    popupMenu.show();
}
```

□ Bước 1: Gọi Context menu

```
@Override  
public void onCreateContextMenu(ContextMenu menu, View v,  
ContextMenu.ContextMenuItemInfo menuInfo) {  
    getMenuInflater().inflate(R.menu.menu_setting,menu);  
    menu.setHeaderTitle("Context Menu");  
    menu.setHeaderIcon(R.mipmap.ic_launcher);  
    super.onCreateContextMenu(menu, v, menuInfo);  
}
```

//đăng ký view cho context menu trong onCreate
registerForContextMenu(btnButton);

□ Bước 2: bắt sự kiện Context menu

//bắt sự kiện Context Menu

@Override

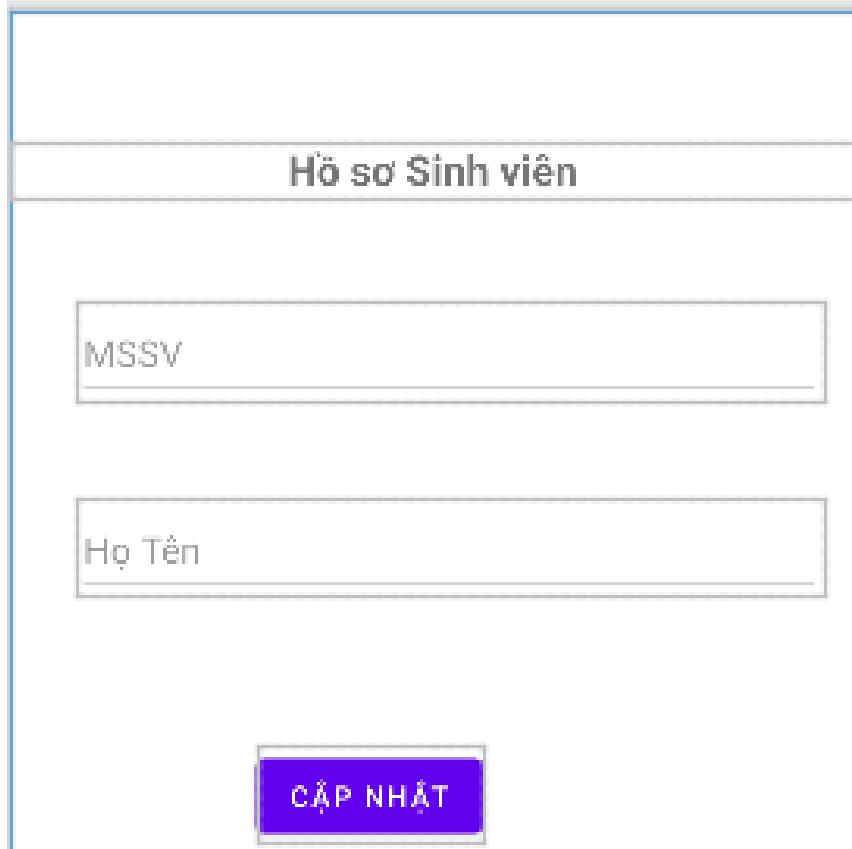
```
public boolean onContextItemSelected(@NonNull MenuItem item) {  
    switch (item.getItemId()) {  
        case R.id.menuSetting:  
            //lệnh  
            Toast.makeText(ControlActivity2.this, "Bạn đang chọn  
Setting", Toast.LENGTH_LONG).show();  
            break;  
        case R.id.menuShare:  
            btnButton.setText("Chia sẻ");  
            break;  
        case R.id.menuLogout:  
            break;  
    }  
    return super.onContextItemSelected(item);  
}
```

Alert Dialog

75

```
private void XacNhanXoa( final int vitri){  
    AlertDialog.Builder alert = new AlertDialog.Builder(this);  
    alert.setTitle("Thông báo");  
    alert.setMessage("Bạn có muốn đăng xuất không");  
    alert.setPositiveButton("Có", new DialogInterface.OnClickListener() {  
        @Override  
        public void onClick(DialogInterface dialog, int which) {  
            //lệnh nút có  
        }  
    });  
    alert.setNegativeButton("Không", new DialogInterface.OnClickListener() {  
        @Override  
        public void onClick(DialogInterface dialog, int which) {  
            //lệnh nút không  
        }  
    });  
    alert.show();  
}
```

□ Bước 1: Tạo layout DiaLog



```
//tạo hàm Dialog
private void DiaLog1(){
    Dialog dialog = new Dialog(this);

    dialog.requestWindowFeature(Window.FEATURE_NO_TITLE);
    dialog.setContentView(R.layout.dialog);

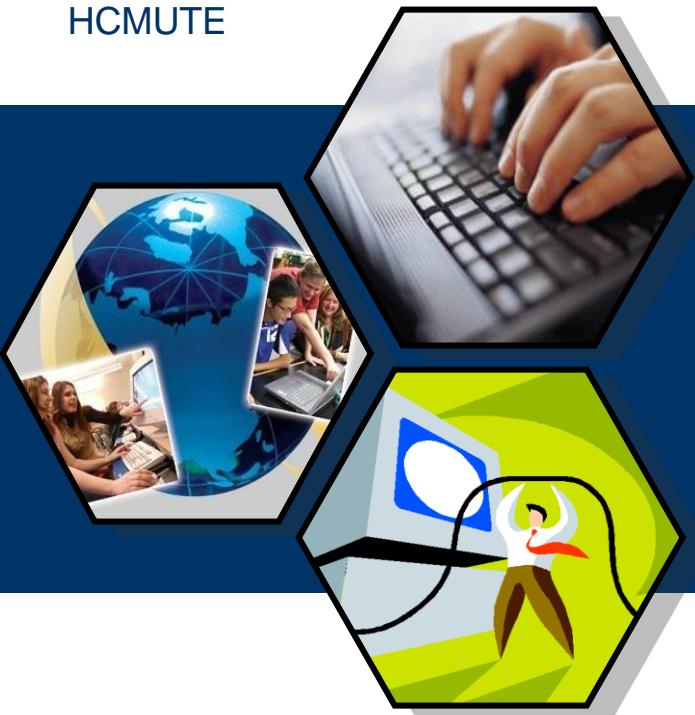
    dialog.setCanceledOnTouchOutside(false);
    //Ánh xạ
    EditText editText1 = (EditText)
    dialog.findViewById(R.id.editNumber1);
    //viết code sự kiện
    //bắt sự kiện Dialog
    dialog.show(); //hủy gọi dialog.dismiss();
}
```



HCMUTE

KHOA CÔNG NGHỆ THÔNG TIN

UTEx



LAYOUT TRONG ANDROID

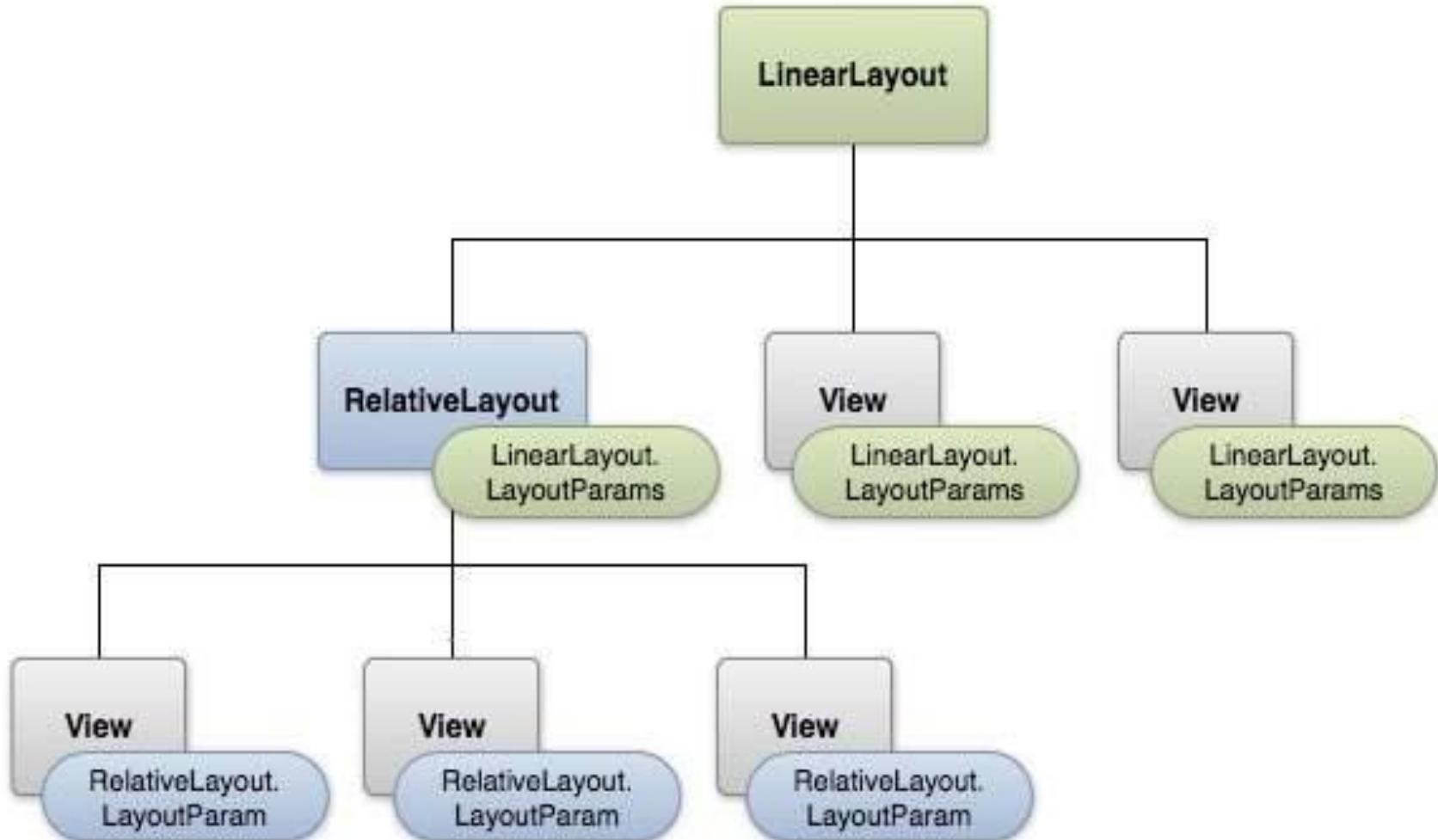
Khoa Công nghệ Thông tin
Đại học Sư phạm Kỹ thuật TP.HCM

ThS. Nguyễn Hữu Trung



- Kiến trúc nền tảng cho giao diện UI là một đối tượng View, view là lớp cơ sở cho Widget để tạo các thành phần UI có tính tương tác như button, các trường text, ...
- **ViewGroup** là một lớp cung cấp Container giữ các View khác hoặc các ViewGroup khác và định nghĩa các thuộc tính Layout của chúng.

- ❑ Layout đặc trưng định nghĩa cấu trúc nhìn thấy cho một giao diện UI trong Android và có thể được tạo bởi sử dụng các đối tượng **View**/**ViewGroup** hoặc bạn có thể khai báo Layout của bạn bởi sử dụng XML file đơn giản là **main_Layout.xml**, được đặt trong thư mục res/layout của Project.



Layout	Miêu tả
Linear Layout	Linear Layout là một view group mà căn chỉnh các view con theo một hướng nào đó: chiều dọc hay chiều ngang . Thuộc tính: android:orientation
Relative Layout	Là Layout hiển thị các View con với các vị trí tương đối. Vị trí của mỗi View có thể được xác định so với các View khác hoặc với thành phần cha của chúng (thông qua id). Bạn có thể sắp xếp View sang bên phải, bên dưới một View khác, giữa màn hình, v.v.. Để định nghĩa vị trí cho mỗi View bạn sử dụng nhiều thuộc tính có sẵn từ RelativeLayout.LayoutParams .
Table Layout	Table Layout là một view mà nhóm tất cả các view vào trong các hàng và các cột .

Layout**Công dụng****Absolute
Layout**

Là layout cho phép bạn chỉ định vị trí chính xác (tọa độ x / y) của các đối tượng. Absolute layouts thường ít linh hoạt và khó duy trì hơn các loại layouts khác.

**Frame
Layout**

FrameLayout là một trong những layout hữu ích được cung cấp bởi hệ thống Android, cho phép các đối tượng giao diện người dùng được xếp chồng chéo với nhau.

Layout	Công dụng
	<p>Constraint layout được giới thiệu lần đầu tiên tại sự kiện Google I/O 2016, Constraint Layout sẵn dùng với bản Android 2.3 (API level 9)</p> <p>match_constraint: xác định width và height của một View</p>
Constraint layout	<p>VD : TextView dưới được set nằm giữa view parent</p> <pre><TextView android:layout_width="wrap_content" android:layout_height="wrap_content" android:text="Text View 1" app:layout_constraintBottom_toBottomOf="parent" app:layout_constraintLeft_toLeftOf="parent" app:layout_constraintRight_toRightOf="parent" app:layout_constraintTop_toTopOf="parent" /></pre>

Thuộc tính	Ý nghĩa
<code>android:id</code> <code>android:id="@+id/mybutton"</code>	ID nhận diện duy nhất View, Button myButton = (Button) findViewById(R.id.mybutton);
<code>android:layout_width</code>	Độ rộng của Layout, dp (Density-independent pixel), sp (Scale-independent pixel), pt (các point là 1/72 inch), px (pixel), mm (mili met), và in (inch).
<code>android:layout_height</code>	Chiều cao của Layout, wrap_content theo nội dung, fill_parent/match_parent theo view cha.
<code>android:layout_marginTop</code>	Lề cạnh trên của Layout
<code>android:layout_marginBottom</code>	Lề cạnh dưới của Layout
<code>android:layout_marginLeft</code>	Lề cạnh trái của Layout
<code>android:layout_marginRight</code>	Lề cạnh phải Layout
<code>android:layout_gravity</code>	Xác định cách các view con được đặt tại đâu

Thuộc tính	Ý nghĩa
android:layout_weight	Xác định có bao nhiêu phần trong Layout được cấp phát tới View đó
android:layout_x	Xác định tọa độ x của Layout
android:layout_y	Xác định tọa độ y của Layout
android:paddingLeft	Left padding được điền cho Layout
android:paddingRight	Right padding được điền cho Layout
android:paddingTop	Top padding được điền cho Layout
android:paddingBottom	Bottom padding được điền cho Layout

```
    android:layout_marginLeft="40dp"  
  
<?xml version="1.0" encoding="utf-8"?>  
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"  
    xmlns:tools="http://schemas.android.com/tools"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:layout_marginLeft="40dp"  
    android:layout_marginTop="30dp"  
    android:background="#72acdf"  
    android:paddingBottom="50dp"  
    android:paddingLeft="50dp"  
    android:paddingRight="50dp"  
    android:paddingTop="50dp"  
    tools:context="dev4u.com.helloworld.MainActivity">  
  
    <Button  
        android:id="@+id	btnMyButton"  
        android:layout_width="wrap_content"  
        android:layout_height="wrap_content"  
        android:text="Button Demo" />  
  
    <!--<TextView-->  
    <!--android:layout_width="wrap_content"-->  
    <!--android:layout_height="wrap_content"-->
```





The image shows a screenshot of the Android Studio interface. On the left, the Project Navigational Drawer displays the project structure:

- LayoutDemo (selected)
- src
 - com.example.layoutdemo
 - MainActivity.java
- gen [Generated Java Files]
- Android 7.1.1
- Android Private Libraries
- Android Dependencies
- assets
- bin
- libs
- res
 - drawable-hdpi
 - drawable-ldpi
 - drawable-mdpi
 - drawable-xhdpi
 - drawable-xxhdpi
 - layout
 - activity_main.xml
 - menu
 - values
 - values-v11
 - values-v14
 - values-w820dp
- AndroidManifest.xml
- ic_launcher-web.png
- proguard-project.txt
- project.properties

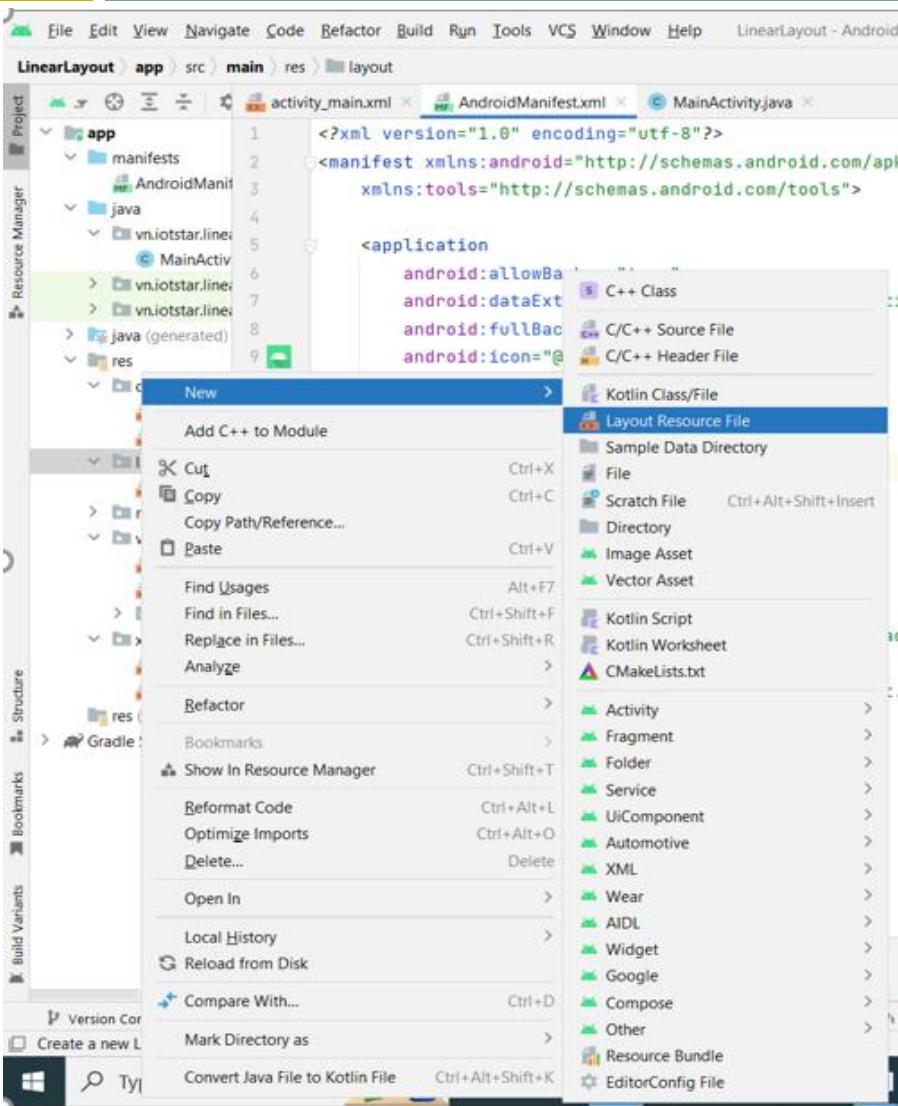
On the right, the code editor shows the `AndroidManifest.xml` file with the following content:

```
4     android:versionCode="1"
5     android:versionName="1.0" >
6
7     <uses-sdk
8         android:minSdkVersion="8"
9         android:targetSdkVersion="21" />
10
11    <application
12        android:allowBackup="true"
13        android:icon="@drawable/ic_launcher"
14        android:label="@string/app_name"
15        android:theme="@style/AppTheme" >
16        <activity
17            android:name=".MainActivity"
18            android:label="@string/app_name" >
19            <intent-filter>
20                <action android:name="android.intent.action.MAIN" />
21
22                <category android:name="android.intent.category.LAUNCHER" />
23            </intent-filter>
24        </activity>
25    </application>
26
27 </manifest>
```

The code editor has several annotations with orange circles and numbers:

- An orange circle with the number "1" is placed over the `<uses-sdk>` block.
- An orange circle with the number "2" is placed over the `activity_main.xml` file in the `res/layout` folder.
- An orange circle with the number "3" is placed over the `AndroidManifest.xml` file.
- An orange circle with the number "4" is placed over the `android:label` attribute in the `<activity>` block.

Đổi Layout mặc định bằng một Layout khác bất kỳ



Đặt tên my_new_layout.xml

Bây giờ vào lại MainActivity.java. Sửa lại dòng lệnh setContentView thành:

```
protected void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.my_new_layout);  
}
```

89

```
✓ LayoutDemo
  ✓ src
    ✓ com.example.layoutdemo
      > MainActivity.java
  > gen [Generated Java Files]
  > Android 7.1.1
  > Android Private Libraries
  > Android Dependencies
  ✓ assets
  ✓ bin
  ✓ libs
  ✓ res
    ✓ drawable-hdpi
    ✓ drawable-ldpi
    ✓ drawable-mdpi
    ✓ drawable-xhdpi
    ✓ drawable-xxhdpi
  ✓ layout
    activity_main.xml
  ✓ menu
  ✓ values
  ✓ values-v11
  ✓ values-v14
  ✓ values-w820dp
  AndroidManifest.xml
  ic_launcher-web.png
  proguard-project.txt
  project.properties
```

```
5 import android.view.Window;
6 import android.view.MenuItem;
7
8 public class MainActivity extends Activity {
9
10    @Override
11    protected void onCreate(Bundle savedInstanceState) {
12        super.onCreate(savedInstanceState);
13        setContentView(R.layout.activity_main);
14    }
15
16    @Override
17    public boolean onCreateOptionsMenu(Menu menu) {
18        // Inflate the menu; this adds items to the action bar if it is present.
19        getMenuInflater().inflate(R.menu.main, menu);
20        return true;
21    }
22
23    @Override
24    public boolean onOptionsItemSelected(MenuItem item) {
25        // Handle action bar item clicks here. The action bar will
26        // automatically handle clicks on the Home/Up button, so long
27        // as you specify a parent activity in AndroidManifest.xml.
28        int id = item.getItemId();
29        if (id == R.id.action_settings) {
30            return true;
31        }
32        return super.onOptionsItemSelected(item);
33    }
34}
```

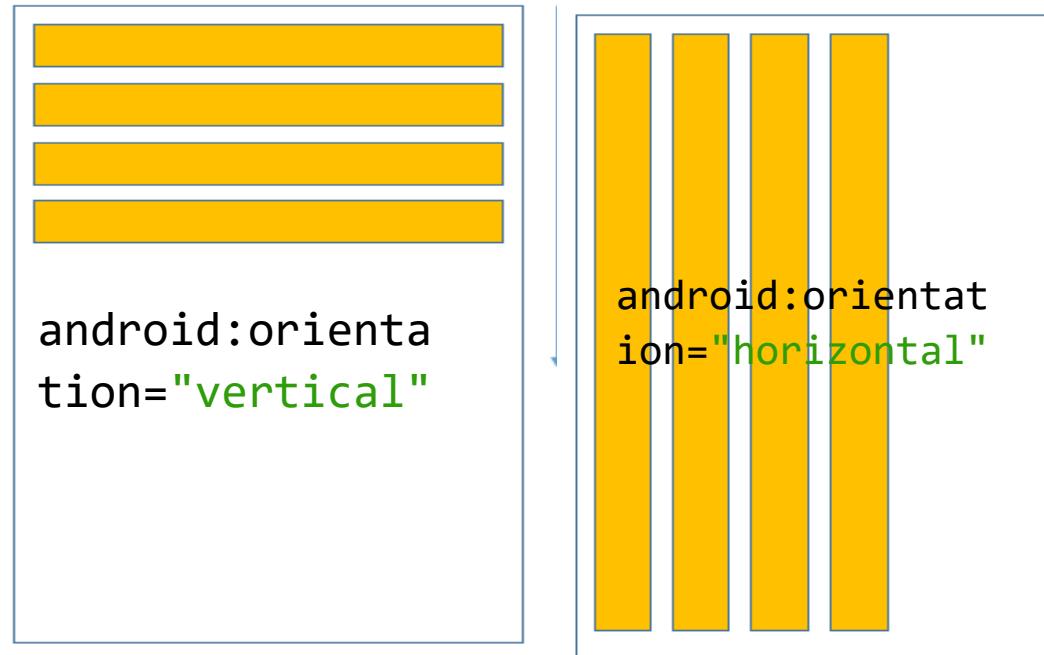
FrameLayout là một ViewGroup được sử dụng rất nhiều trong android. Bởi vì nó là ViewGroup đơn giản nhất,

FrameLayout được định nghĩa bắt đầu bởi thẻ <FrameLayout> và thẻ đóng </FrameLayout>. Ở giữa thẻ đóng và thẻ mở chính là các view con của nó.

Quy tắc layout các view con trong FrameLayout là các view sẽ nắn chồng lên nhau, view thêm vào sau sẽ nằm đè lên view nằm phía dưới

```
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/FrameLayout1"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity" >
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello World"
        android:textColor="#2c3e50"
        android:textSize="32sp" />
    <ImageView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:src="@drawable/ic_launcher_background" />
</FrameLayout>
```

- LinearLayout là một view group việc căn chỉnh các view con theo một hướng nào đó: chiều dọc hay chiều ngang.
- Khi Layout được tạo, bạn có thể tải layout resource từ phần code, trong phương thức callback là **Activity.onCreate()**, như sau:



```
public void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.activity_main);  
}
```

Linear Layout

92

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">
```

```
<TextView android:id="@+id/text"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="This is a TextView" />
```

```
<Button android:id="@+id/button"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="This is a Button" />
```

```
<!-- More GUI components go here -->
```

```
</LinearLayout>
```

```
<TextView  
    android:background="#ff0"  
    android:fontFamily="sans-serif"  
    android:textColor="#FF5722"  
    android:textSize="30sp"  
    android:id="@+id/textView5"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:text="Hello World"/>
```

```
<TextView  
    android:layout_width="0dp"  
    android:layout_height="match_parent"  
    android:layout_weight="1"  
    android:background="#af5656" />
```

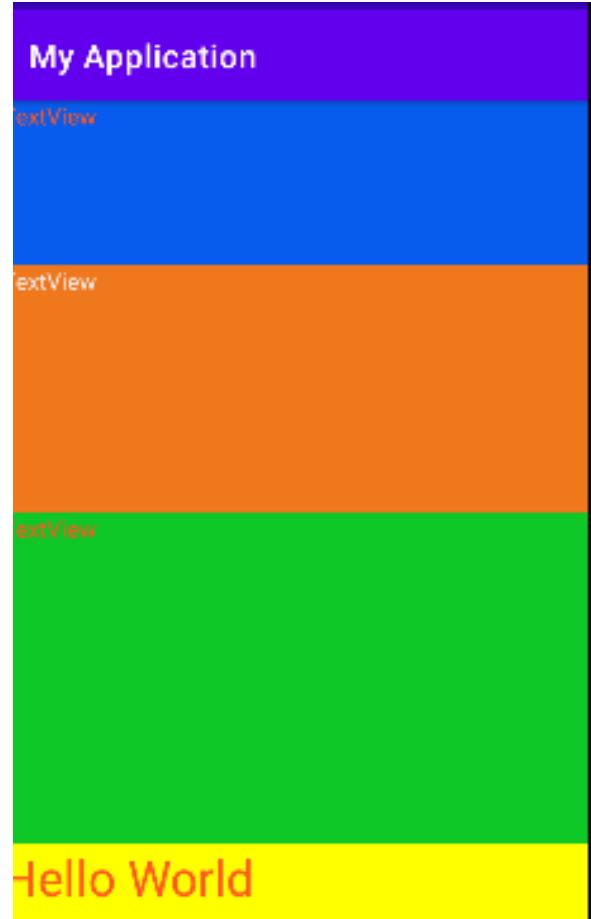
```
<TextView  
    android:layout_width="0dp"  
    android:layout_height="match_parent"  
    android:layout_weight="4"  
    android:background="#28c6a6" />
```

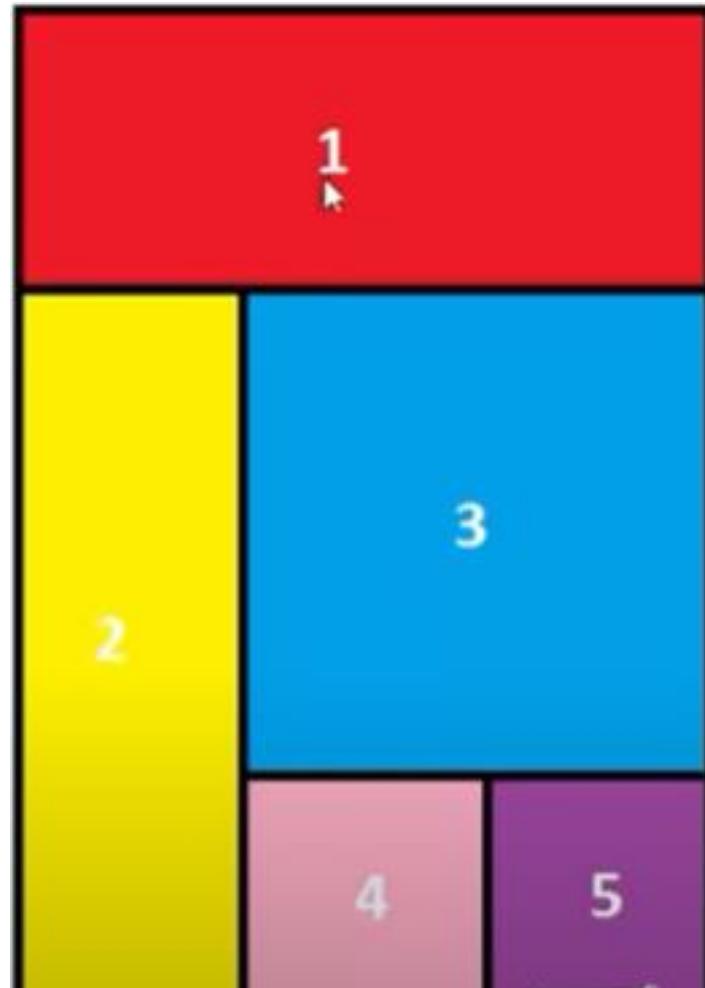
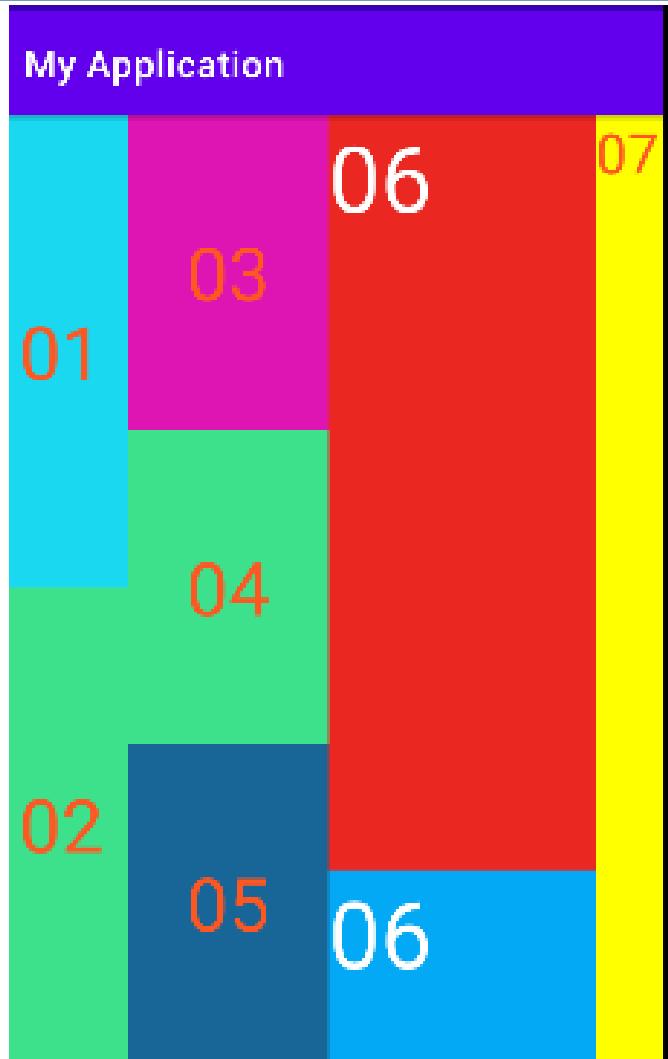
Chia tỉ lệ layout

Ngoài cách sử dụng **LinearLayout** thông thường thì **LinearLayout** thường được sử dụng để phân chia tỉ lệ layout bằng cách sử dụng thuộc tính **layout_weight** và **weightSum**.

- **weightSum**: Xác định trọng số của **LinearLayout** hiện tại.
- **layout_weight**: Trọng số mà view con trong **LinearLayout** chiếm giữ.

```
<TextView  
    android:background="#ff0"  
    android:fontFamily="sans-serif"  
    android:textColor="#FF5722"  
    android:textSize="30sp"  
    android:id="@+id/textView5"  
    android:layout_width="match_parent"  
    android:layout_height="100dp"  
    android:text="Hello World"  
    android:layout_weight="1" //chia trọng số, tỉ lệ  
/>  
<LinearLayout  
    android:weightSum="10" /> <!-- 2 - 3 - 4 - 1 -->
```





Relative Layout

96

- RelativeLayout cho phép sắp xếp các control theo vị trí tương đối giữa các control khác trên giao diện (kể cả control chứa nó). Thường nó dựa vào Id của các control khác để sắp xếp theo vị trí tương đối. Do đó khi làm RelativeLayout bạn phải chú ý là đặt Id control cho chuẩn xác, nếu sau khi Layout xong mà bạn lại đổi Id của các control thì giao diện sẽ bị xáo trộn (do đó nếu đổi ID thì phải đổi luôn các tham chiếu khác sao cho khớp với Id bạn mới đổi).
- RelativeLayout là ViewGroup cũng được sử dụng khá nhiều trong android, RelativeLayout được định nghĩa trong xml bởi cặp thẻ đóng mở <RelativeLayout> và thẻ đóng </RelativeLayout>.

Relative Layout

Quy tắc layout của RelativeLayout khá giống như FrameLayout, nhưng có một số điểm đặc biệt là các view có thể xác định bằng vị trí **tương đối (relative)** với các view khác thông qua id.

Các vị trí tương đối này như sau:

- **android:layout_above="id_name"**: view hiện tại sẽ nằm phía trên view có thuộc tính id là id_name.
- **android:layout_below="id_name"**: view hiện tại sẽ nằm phía dưới view có thuộc tính id là id_name.
- **android:layout_toLeftOf="id_name"**: view hiện tại sẽ nằm bên trái dưới view có thuộc tính id là id_name.
- **android:layout_toRightOf="id_name"**: view hiện tại sẽ nằm phía bên phải view có thuộc tính id là id_name.

Relative Layout

```
1. <?xml version="1.0" encoding="utf-8"?>
2. <RelativeLayout
3.     xmlns:android="http://schemas.android.com/apk/res/android"
4.     android:layout_width="match_parent"
5.     android:layout_height="match_parent">
6.
7.     <TextView
8.         android:id="@+id/tv_username"
9.         android:layout_width="wrap_content"
10.        android:layout_height="wrap_content"
11.       android:text="username" />
12.
13.     <EditText
14.         android:id="@+id/edt_username"
15.         android:layout_width="match_parent"
16.         android:layout_height="wrap_content"
17.         android:layout_below="@+id/tv_username" />
18.
19.     <TextView
20.         android:id="@+id/tv_password"
21.         android:layout_width="wrap_content"
22.         android:layout_height="wrap_content"
23.         android:layout_below="@+id/edt_username"
24.         android:text="password" />
25.
26.     <EditText
27.         android:id="@+id/edt_password"
28.         android:layout_width="match_parent"
29.         android:layout_height="wrap_content"
30.         android:layout_below="@+id/tv_password"
31.         android:inputType="textPassword" />
32. </RelativeLayout>
```

Một số thuộc tính khác thường xuyên sử dụng với RelativeLayout:

- **android:layout_alignParentBottom="boolean"**: Căn dưới phần tử hiện tại theo phần tử cha nếu set là true.
- **android:layout_alignParentTop="boolean"**: Căn trên phần tử hiện tại theo phần tử cha nếu set là true.
- **android:layout_alignParentRight="boolean"**: Căn phải phần tử hiện tại theo phần tử cha nếu set là true.
- **android:layout_alignParentLeft="boolean"**: Căn trái phần tử hiện tại theo phần tử cha nếu set là true.

Table Layout

99

- Table Layout là một view mà nhóm tất cả các view vào trong các hàng và các cột.

```
<?xml version="1.0" encoding="utf-8"?>
<TableLayout
    android:stretchColumns="*"
    android:layout_height="match_parent"
    android:layout_width="match_parent"
    xmlns:android="http://schemas.android.com/apk/res/android">
    <TableRow>
        <ImageView
            android:layout_width="100dp"
            android:layout_height="100dp"
            android:src="@drawable/android"/>
        <TextView
            android:layout_span="3"
            android:layout_gravity="center"
            android:textSize="30sp"
            android:text="dong 1"/>
    </TableRow>
</TableLayout>
```

Các thuộc tính quan trọng:

- android:stretchColumns
- android:shrinkColumns
- android:collapseColumns

Dùng **layout_span** để trộn các cột

```
<TableRow>
    <TextView android:text="URL:" />
    <EditText
        android:id="@+id/entry"
        android:layout_span="3" />
</TableRow>
```

Constraint Layout

100

- Constraint layout được giới thiệu lần đầu tiên tại sự kiện **Google I/O 2016**, Constraint Layout sẵn dùng với bản Android 2.3 (API level 9)
- **match_constraint**: xác định width và height của một View
 - VD : TextView dưới được set nằm giữa view parent

```
<TextView  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:text="Text View 1"  
    app:layout_constraintBottom_toBottomOf="parent"  
    app:layout_constraintLeft_toLeftOf="parent"  
    app:layout_constraintRight_toRightOf="parent"  
    app:layout_constraintTop_toTopOf="parent" />
```

Constraint Layout

101

- Thực hiện 02 điểm neo: ở trên bên trái và ở trên bên phải (hoặc ở dưới bên trái và ở dưới bên phải)

```
<androidx.constraintlayout.widget.ConstraintLayout  
    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">  
    <TextView  
        android:id="@+id/textView"  
        android:layout_width="wrap_content"  
        android:layout_height="wrap_content"  
        android:text="Hello World!"  
        app:layout_constraintBottom_toBottomOf="parent"  
        app:layout_constraintEnd_toEndOf="parent"  
        app:layout_constraintStart_toStartOf="parent"  
        app:layout_constraintTop_toTopOf="parent"  
        app:layout_constraintVertical_bias="0.206" />
```

```
<Button  
    android:id="@+id/button"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:layout_marginTop="88dp"  
    android:text="Button"  
    app:layout_constraintEnd_toEndOf="parent"  
    app:layout_constraintHorizontal_bias="0.233"  
    app:layout_constraintStart_toStartOf="parent"  
    app:layout_constraintTop_toBottomOf  
    = "@+id/textView" />  
  
</androidx.constraintlayout.widget.ConstraintLayout  
>
```

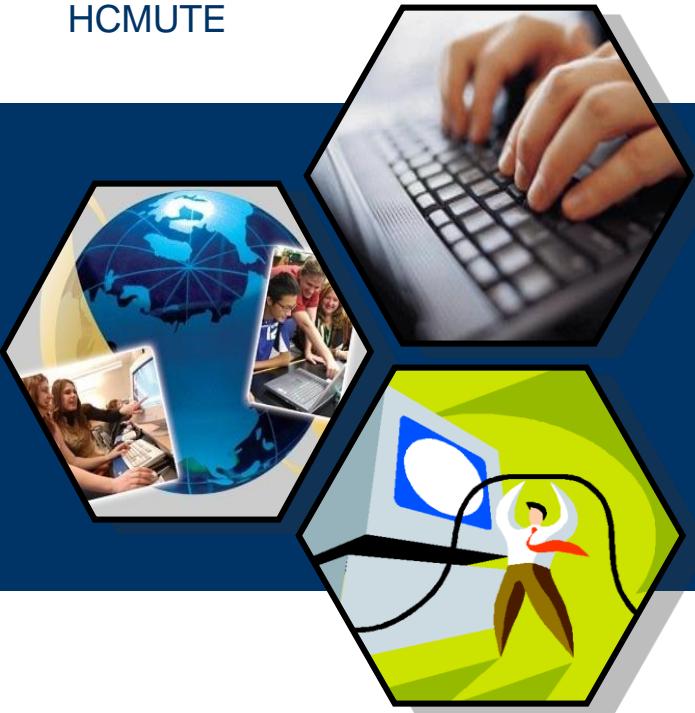


HCMUTE

KHOA CÔNG NGHỆ THÔNG TIN



Điều khiển sự kiện (Event - Handling)



**Khoa Công nghệ Thông tin
Đại học Sư phạm Kỹ thuật TP.HCM**

ThS. Nguyễn Hữu Trung



Có 3 khái niệm liên quan đến quản lý sự kiện:

- **Event Listeners:** Là một giao diện trong một View, View này chứa một phương thức duy nhất là callback. Phương thức này được gọi bởi Framework Android, khi View đã được đăng ký sự kiện và người dùng tác động lên View trong UI (giao diện người dùng).
- **Event Listeners Registration:** Event Registration là một quá trình mà một Event Handler đã được đăng ký với Event Listener để mà Handler này được gọi khi Event Listener kích hoạt sự kiện.
- **Event Handlers:** Khi một Event xảy ra, và chúng ta đã đăng ký một Event Listener cho sự kiện, thì Event Listener gọi Event Handler, là phương thức mà thực sự xử lý sự kiện đó

Event Handler	Event Listener Name
onClick()	OnClickListener() Đăng ký sự kiện khi người dùng hoặc click hoặc chạm (touche) hoặc focus trên bất kỳ widget như button, text, image vv. Chúng ta sẽ sử dụng phương onClick() để xử lý sự kiện.
onLongClick()	OnLongClickListener() Đăng ký sự kiện khi người dùng hoặc click hoặc chạm (touche) hoặc focus trên bất kỳ widget như button, text, image vv. trong một hoặc nhiều giây. Chúng ta sẽ sử dụng phương onLongClick() để xử lý sự kiện.
onFocusChange()	OnFocusChangeListener() Sự kiện phát sinh khi widget mất focus.

Event Handler	Event Listener Name
onKey()	OnFocusChangeListener() Sự kiện phát sinh khi người dùng focus trên widget và nhấn (presse) hoặc thả (release) một phím trên thiết bị.
onTouch()	OnTouchListener() Sự kiện phát sinh khi người dùng nhấn phím, nhả phím, hoặc bất kỳ cử chỉ chuyển động trên màn hình.
onMenuItemClick()	OnMenuItemClickListener() Sự kiện phát sinh khi người dùng chọn một mục trong menu.
onCreateContextMenu()	OnCreateContextMenuListener() Sự kiện phát sinh khi người dùng chọn một mục trong menu ngữ cảnh (Context Menu)

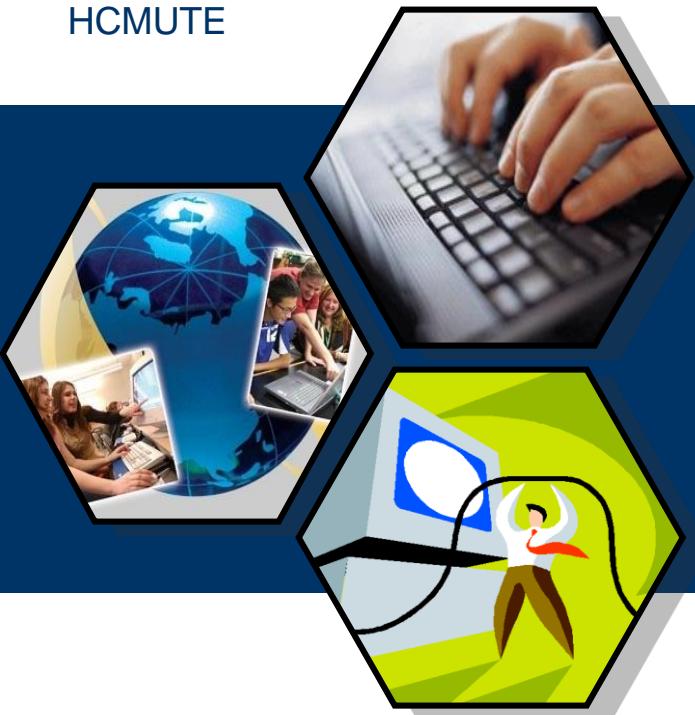
1. Xử lý sự kiện trong Layout (Handle event in Layout)
2. Xử lý sự kiện bằng lớp nặc danh (Inline anonymous listener)
3. Kế thừa Interface OnClickListener (Implements OnClickListener Interface)
4. Sử dụng biến (Event Listener using Variable)
5. Bắt sự kiện thông qua lớp lắng nghe (Listener Class)
6. PerformClick method



HCMUTE

KHOA CÔNG NGHỆ THÔNG TIN

UTEx

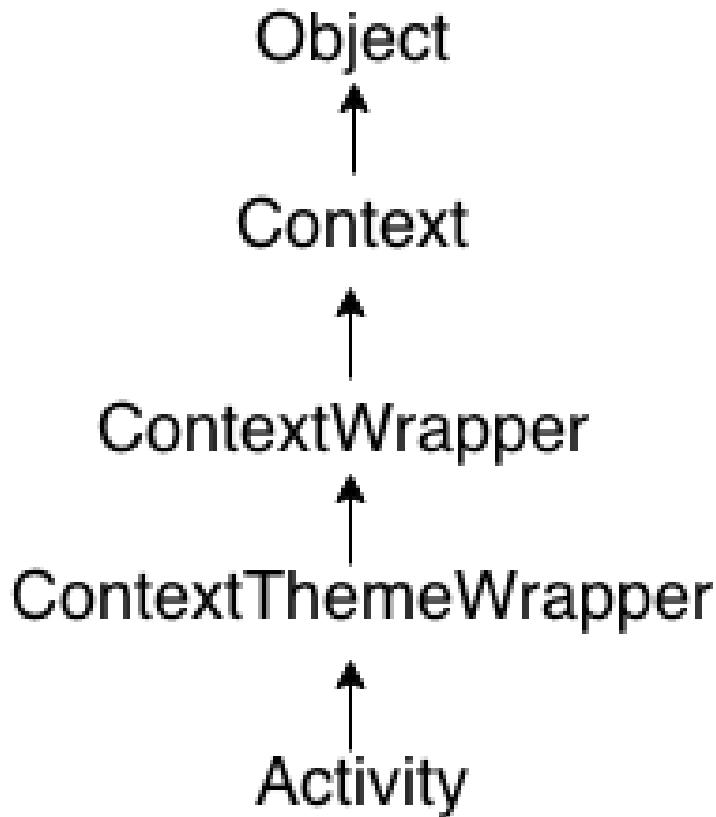


ACTIVITY TRONG ANDROID

Khoa Công nghệ Thông tin
Đại học Sư phạm Kỹ thuật TP.HCM

ThS. Nguyễn Hữu Trung



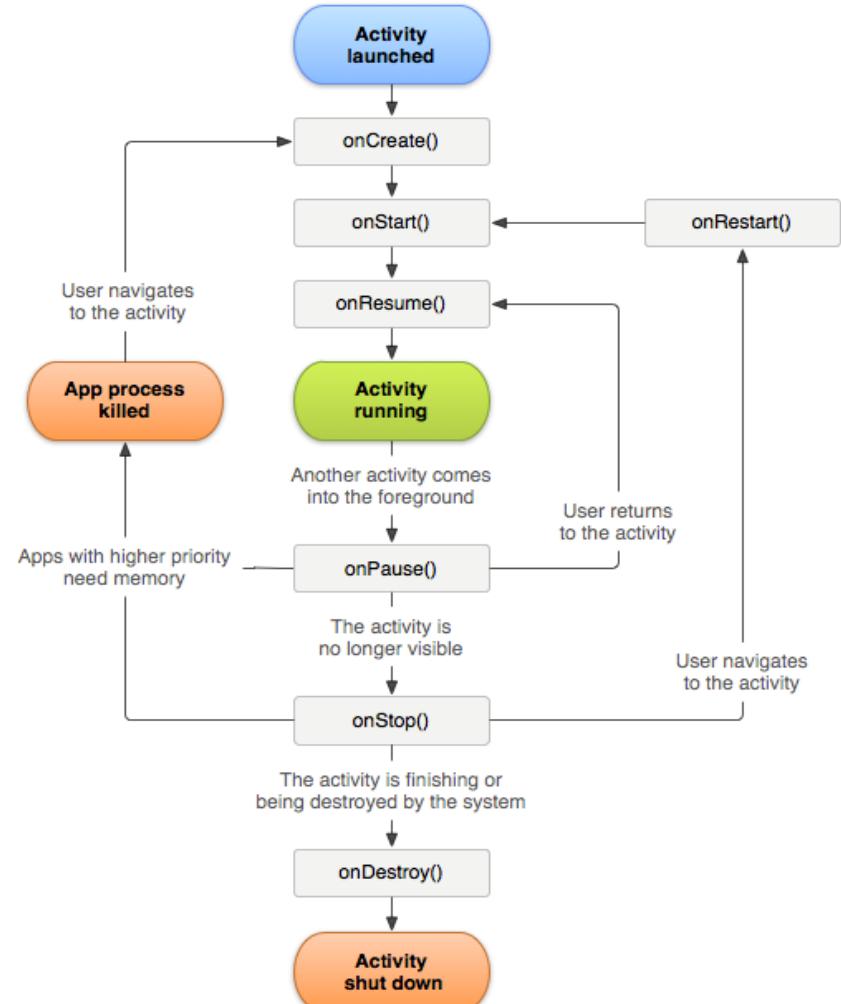


- Một **Activity** đại diện cho một màn hình với một giao diện người dùng. Giống như window hay Frame trong Java, Bạn có thể đặt các Widget vào một màn hình đơn giản

Vòng đời của một Activity

108

- Vòng đời của một Activity trong Android gồm có 7 phương thức, mỗi phương thức thể hiện hành vi khác nhau của **Activity**.
 - Chú ý: Phương thức onCreate() và onDestroy() được gọi một lần trong suốt vòng đời của Activity



Phương thức	Ý nghĩa
onCreate()	Nó là phương thức đầu tiên được gọi dùng để tạo một activity vào lần đầu tiên activity được gọi.
onStart()	Sẽ được gọi khi nó hiện hữu với người dùng.
onResume()	Sẽ được gọi khi người dùng tương tác với các ứng dụng.
onPause()	Tạm dừng một activity, không nhận dữ liệu do người dùng nhập vào và không thể thực thi lệnh nào. Phương thức này được gọi khi activity hiện tại đang được tạm dừng, và activity trước đó đang được tiếp tục.
onStop()	Được gọi khi một activity đã không được nhìn thấy trong thời gian dài.
onRestart()	Được gọi khi activity cần được dùng trở lại sau khi bị gọi onStop();
onDestroy()	Được gọi trước khi hệ thống hủy activity.

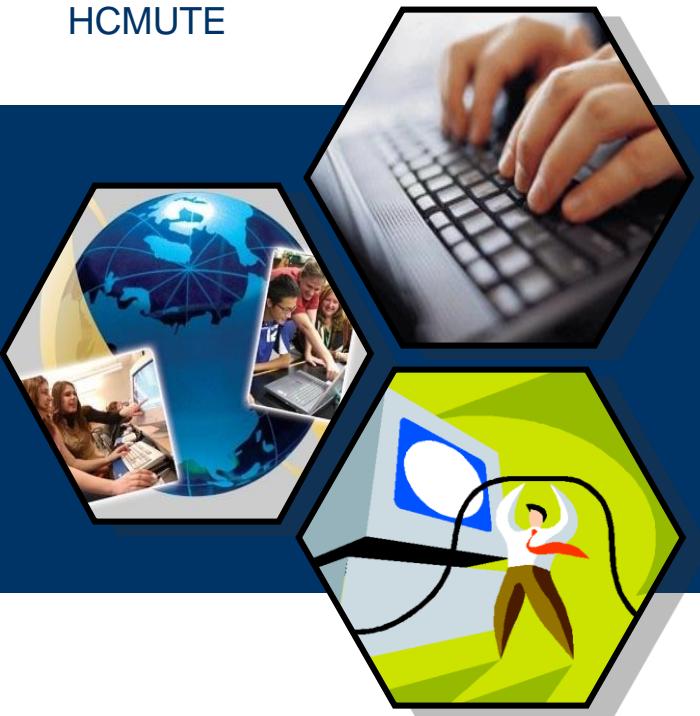
Ví dụ

110



HCMUTE

KHOA CÔNG NGHỆ THÔNG TIN



Intents trong Android



- Intents là một thành phần quan trọng trong android. Nó cho phép các thành phần ứng dụng có thể yêu cầu các hàm từ các thành phần ứng dụng android khác. Ví dụ một activity có thể chạy một activity khác ở bên ngoài để chụp ảnh.
- Để chạy một activity, broadcast receivers thì sử dụng phương thức **startActivity(intent)**. Phương thức này được định nghĩa trong đối tượng **context**.
- Android intents thường được sử dụng chính:
 - Start dịch vụ
 - Gọi một activity
 - Hiển thị một trang web
 - Hiển thị danh sách liên hệ
 - Gởi một tin nhắn
 - Gọi điện thoại.
- Có 2 loại intent trong Android: implicit và explicit.

- Đối tượng dữ liệu của Intent là một cấu trúc dữ liệu được dùng bởi các component trong ứng dụng hoặc hệ thống nhận được và có những xử lý thích hợp.

Đối tượng dữ liệu của Intent có thể chứa các thuộc tính:

- **Action**

- Tên của các action mà Intent sẽ được thực hiện.
- Action có thể được định nghĩa sẵn Android cung cấp hoặc do người lập trình tự định nghĩa cho riêng ứng dụng (như dùng trong BroadcastReceiver).

- **Data**

- Dữ liệu mà thành phần được gọi (activity, service,...) sẽ xử lý.
- Được định dạng là đối tượng URI (tham khảo class Uri class).
- Data truyền vào sẽ được xử lý và hành động tùy theo “the MIME type” mà dữ liệu sử dụng.

□ Category

- Là chuỗi ký tự chứa thông tin về nhóm phân loại các đối tượng để xử lý các intent.
- Có 2 category được thấy trong các file manifest:
 - **CATEGORY_BROWSABLE**: Cho phép start một activity bằng Web browser để hiển thị dữ liệu định dạng là một liên kết ví dụ như một e-mail hay một bức hình trên mạng.
 - **CATEGORY_LAUNCHER**: Khai báo để chỉ định activity sẽ được start khi bắt đầu mở ứng dụng.

□ Extras

- Chứa tất cả các cặp key-value pairs chứa các thông tin dữ liệu bổ sung truyền qua.
- Thông số extras sẽ được gán và đọc bằng phương thức **putExtras()/getExtras()** methods tương ứng với thông số là đối tượng có cấu trúc Bundle.

□ Flags

- Những giá trị của flag này sẽ hướng dẫn hệ thống Android cách để start một activity và kết thúc một activity.

Intent Filter là gì?

115

- Activity, Service và BroadCast Receiver sử dụng Intent Filter để thông báo cho hệ thống biết các dạng Implicit Intent mà nó có thể xử lý. Nói cách khác, Intent Filter là bộ lọc Intent, chỉ cho những Intent được phép đi qua nó.
- Ví dụ chúng ta thiết lập 1 Activity với bộ lọc Intent cho phép bắt và xử lý các Intent gửi SMS.

```
<activity android:name=".MainActivity"
    android:label="@string/activity_name">
    <intent-filter>
        <action
            android:name="android.intent.action.SENDTO" />
        <category
            android:name="android.intent.category.DEFAULT" />
        <data android:scheme="sms" />
    </intent-filter>
</activity>
```

- Explicit Intents là những intent đã chỉ rõ thành phần sẽ nhận và xử lý. Thông thường những intent này sẽ không cần gán bổ sung thêm các thuộc tính khác như action, data. Explicit Intent thường được sử dụng để khởi chạy các activity, hoặc service trong cùng 1 ứng dụng

```
Intent intent = new Intent(getApplicationContext(),  
SecondActivity.class);  
  
startActivity(intent);  
  
finish();
```

Implicit Intent

117

- Implicit Intent: là loại Intent có các Action được Android xây dựng sẵn, nó không chỉ rõ các Component xử lý (các class xử lý) mà nó sẽ cung cấp cho hệ điều hành một loạt các thông tin yêu cầu sau đó hệ điều hành sẽ đổi chiều xem trong hệ thống có bao nhiêu phần mềm khác có thể đáp ứng xử lý yêu cầu này rồi hiện ra một Dialog chứa tên của các phần mềm có thể xử lý để người dùng chọn
- Ví dụ, dưới đây sẽ yêu cầu hệ thống android để xem một trang web. Tất cả các trình duyệt web phải được đăng ký tương ứng vào dữ liệu intent thông qua bộ lọc intent.

```
Intent intent=new Intent(Intent.ACTION_VIEW);  
  
intent.setData(Uri.parse("https://google.com"));  
  
startActivity(intent);
```

Ví dụ: Gọi điện thoại trong android

```
Uri uri = Uri.parse("tel:" + "900");
```

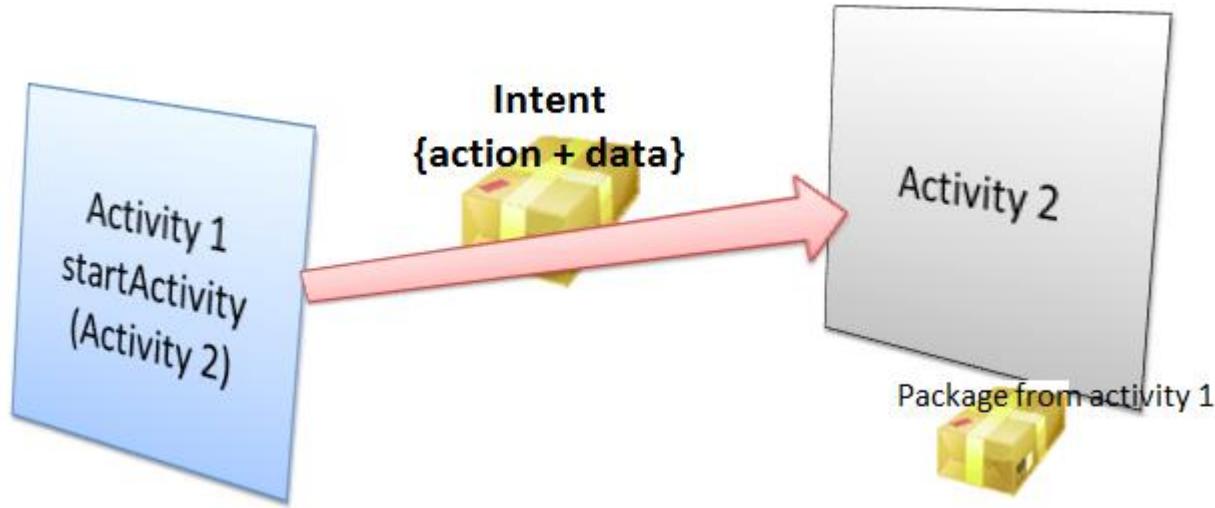
```
Intent intent = new Intent(Intent.ACTION_CALL, uri);
```

// Để có thể call được thì trong AndroidManifest.xml phải khai báo

```
// <uses-permission android:name="android.permission.CALL_PHONE"
/>
```

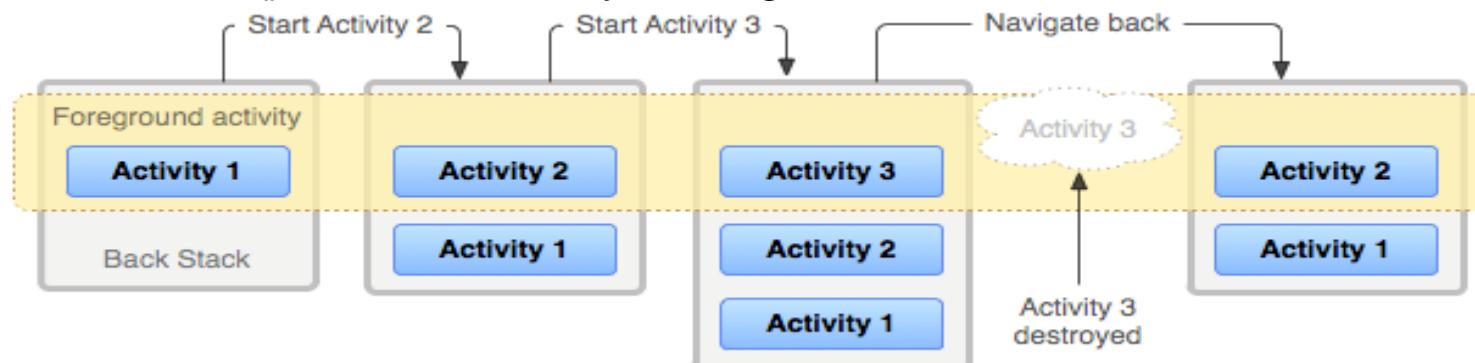
```
startActivity(intent);
```

Định dạng	Action	Mô tả
tel:phone_number	ACTION_VIEW	Mở một Dial form
tel:phone_number	ACTION_CALL	Thực hiện gọi tới số
http://web_address	ACTION_VIEW	Mở trình duyệt web
https://web_address		theo địa chỉ
http://web_address	ACTION_WEB_SEARCH	Thực hiện tìm kiếm
https://web_address		
sms://	ACTION_SENDTO	Gửi tin nhắn
geo:latitude,longitude		
geo:latitude,longitude?z=zoom	ACTION_VIEW	Mở ứng dụng Bản đồ
geo:0,0?q=my+street+address		và trả tới vị trí được
geo:0,0?q=business+near+city		xác định



□ Không gửi kèm dữ liệu:

- Khi chuyển sang KetQuaActivity thì MainActivity được đẩy xuống dưới Back Stack và KetQuaActivity sẽ ở trên đầu của Back Stack. Vì vậy khi nhấn trở lại từ KetQuaActivity thì MainActivity được đẩy lên và onResume() của MainActivity được gọi.



```
Intent intent = new Intent(MainActivity.this, KetQuaActivity.class);
startActivity(intent);
```

Mọi Activity tạo ra phải được khai báo trong file **AndroidManifest.xml**, nếu không khai báo mà vẫn sử dụng Activity này thì hệ thống sẽ ném ra ngoại lệ, điều này cũng được áp dụng khi sử dụng Service trong Android.

```
<activity android:name=".KetQuaActivity"></activity>
```

□ Gửi kèm dữ liệu

- Khi tạo Intent phải đặt các dữ liệu kèm theo qua đối tượng intent này, sử dụng các phương thức có tiền tố là put của đối tượng intent để gửi dữ liệu, mỗi dữ liệu đặt vào sẽ có 1 key đi kèm.
- Char, boolean, int, float, long, String hay kiểu mảng boolean[], char[], int[] , kiểu dữ liệu object, tuy nhiên những kiểu dữ liệu này phải implement interface Parcelable hoặc interface Serializable.

```
Intent intent = new Intent(MainActivity.this, KetQuaActivity.class);
intent.putExtra("int_key", 4);
intent.putExtra("char_key", 'r');
intent.putExtra("boolean_key", true);
intent.putExtra("long_key", 323L);
intent.putExtra("float_key", 3.2f);
intent.putExtra("string_key", "Chuyen Activity trong Android");
intent.putExtra("double", 32D);
intent.putExtra("int_array_key", new int[]{1, 2, 9});
intent.putExtra("boolean_array_key", new boolean[]{true, false, true, true});
intent.putExtra("char_array_Key", new char[] {'e', 'i', 't', 'g', 'u', 'i', 'd', 'e'});
intent.putExtra("rect_key", new Rect(0,0, 200, 200));
```

□ Gửi dữ liệu đi sử dụng Bundle

- Thay vì sử dụng bằng cách đặt dữ liệu vào intent thì sử dụng 1 đối tượng gọi là **Bundle**, đặt các dữ liệu vào đối tượng này và sử dụng phương thức **putExtras()** để đặt hết vào intent.

```
Intent intent = new Intent(MainActivity.this, KetQuaActivity.class);
Bundle bundle = new Bundle();
bundle.putChar("char", 'e');
bundle.putInt("int", 3);
bundle.putString("string", "Nguyễn Hữu Trung");
bundle.putFloat("float", 5.2f);
bundle.putDouble("double", 843D);
bundle.putLong("long", 55343L);
bundle.putParcelable("parcelable", new Rect(0, 0, 300, 300));
intent.putExtras(bundle);
startActivity(intent);
```

□ Nhận dữ liệu

- Sau khi startActivity(), ở Activity B (KetQuaActivity) có được dữ liệu mà Activity A (MainActivity) gửi đến như sau:
 - Lấy ra đối tượng Intent mà Activity A gửi qua bằng phương thức getIntent().
 - Từ đối tượng intent có được từ getIntent(), lấy những dữ liệu trong đối tượng này bằng phương thức getXExtra() (với X là kiểu dữ liệu muốn lấy).

```
// Get data from MainActivity
int intValue = intent.getIntExtra("int_key", 0);
char charValue = intent.getCharExtra("char_key", 'a');
float floatValue = intent.getFloatExtra("float_key", 0f);
boolean booleanValue = intent.getBooleanExtra("boolean_key", false);
String stringValue = intent.getStringExtra("string_key");
int[] intArrayValue = intent.getIntArrayExtra("int_array_key");
char[] charArrayValue = intent.getCharArrayExtra("char_array_key");
Rect rect = intent.getParcelableExtra("rect_key");
```

□ Nhận dữ liệu

- Sau khi startActivity(), ở Activity B (KetQuaActivity) có được dữ liệu mà Activity A (MainActivity) gửi đến như sau:
 - Lấy ra đối tượng Intent mà Activity A gửi qua bằng phương thức getIntent().
 - Từ đối tượng intent có được từ getIntent(), lấy những dữ liệu trong đối tượng này bằng phương thức getXExtra() (với X là kiểu dữ liệu muốn lấy).

Nếu gửi dữ liệu bằng **Bundle** thì làm như sau:

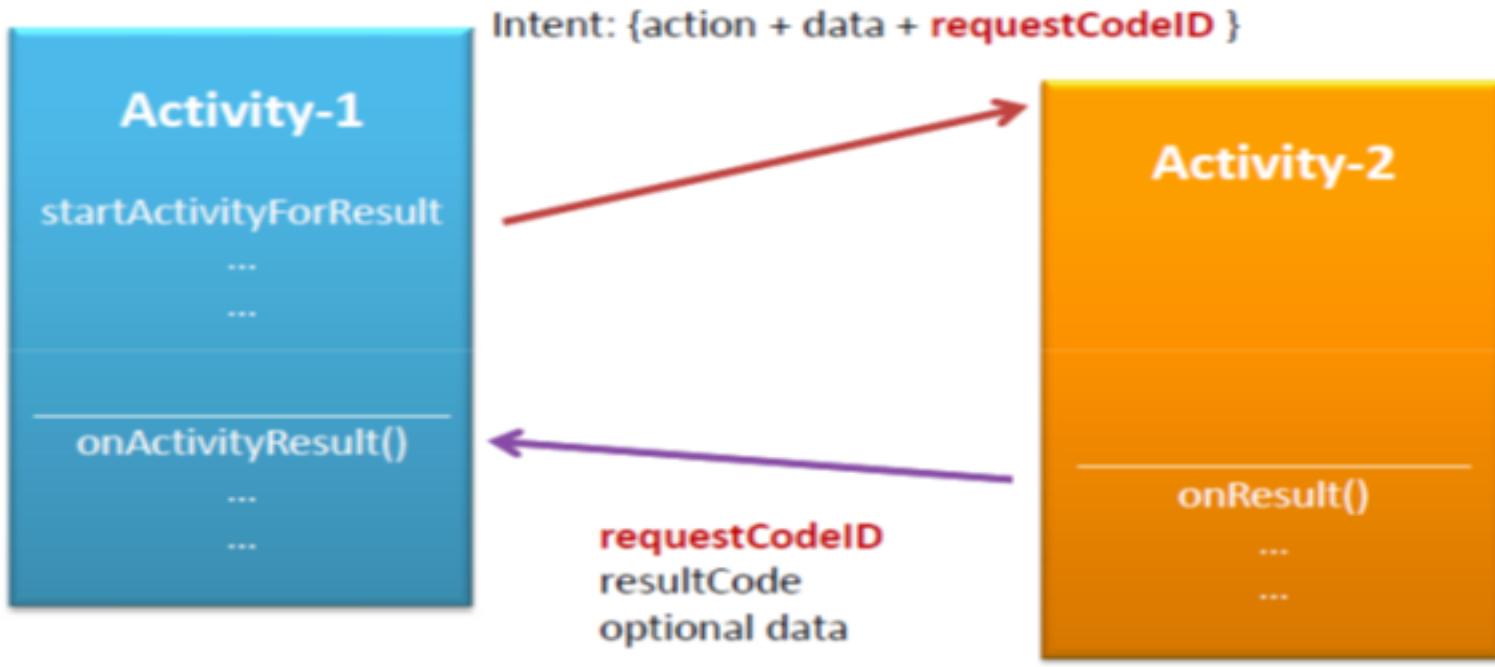
```
Intent intent = getIntent();
Bundle bundle = intent.getExtras();
char charValue = bundle.getChar("char");
int intValue = bundle.getInt("int");
String stringValue = bundle.getString("string");
float floatValue = bundle.getFloat("float");
double doubleValue = bundle.getDouble("double");
long longValue = bundle.getLong("long");
Rect rect = bundle.getParcelable("parcelable");
```

Truyền Đối Tượng qua lại giữa các Activity, các đối tượng này phải được Serialize

```
class Person implements Serializable
{
    private int id;
    private String name;
    public Person(int id, String name)
    {
        this.id=id;
        this.name=name;
    }
    public String toString()
    {
        return this.id+" - "+this.name;
    }
}
```

```
Bundle bundle=new Bundle();
Person p=new Person(1, "teo");
bundle.putSerializable("t1", p);
//...
Person t1=(Person) bundle.getSerializable("t1");
```

- Bằng cách sử dụng `startActivityForResult(Intent intent, int requestCode)` thay vì dùng `startActivity()` bạn có thể start một Activity và sau đó nhận kết quả trả về từ Activity đó thông qua phương thức `onActivityResult()`.



- Chúng ta sẽ dựa vào requestCodeID và resultCode để xử lý.

```
startActivityForResult ( Intent, requestCodeID )
```



- Việc tạo Intent trong trường hợp này cũng như trường hợp trước. Nó chỉ khác hàm gọi :

```
onActivityResult ( requestCodeID, resultCode, Intent )
```



```
public static final int REQUEST_CODE_INPUT=113;
```

```
//Mở Activity với REQUEST_CODE_INPUT  
  
Intent intent = new  
Intent(getApplicationContext(),ActivityInput.class);  
  
//gọi startActivityForResult  
  
startActivityForResult(intent, REQUEST_CODE_INPUT);
```

@Override

```
protected void onActivityResult(int requestCode, int resultCode, Intent data)
    super.onActivityResult(requestCode, resultCode, data);

    //Kiểm tra có đúng requestCode =REQUEST_CODE_INPUT hay không
    //Vì ta có thể mở Activity với những RequestCode khác nhau
    if(requestCode==REQUEST_CODE_INPUT)

    {
        //Kiểm tra ResultCode trả về, cái này ở bên InputDataActivity truyền về
        //có nó rồi thì xử lý trả lên thông thường
        switch(resultCode)
        {
            case RESULT_CODE_SAVE1:
                //giá trị từ InputDataActivity
                int v1= data.getIntExtra("data", 0);}
```

Ví dụ

131

```
11 public class FirstActivity extends AppCompatActivity {
12     public static final int REQ_CODE = 100;
13     TextView textViewReceivedText;
14
15     @Override
16     protected void onCreate(Bundle savedInstanceState) {
17         super.onCreate(savedInstanceState);
18         setContentView(R.layout.activity_first);
19         setTitle("First Activity");
20
21         textViewReceivedText = findViewById(R.id.textViewReceivedText);
22
23         findViewById(R.id.buttonGotoSecond).setOnClickListener(new View.OnClickListener() {
24             @Override
25             public void onClick(View view) {
26                 Intent intent = new Intent(getApplicationContext(), SecondActivity.class);
27                 startActivityForResult(intent, REQ_CODE);
28             }
29         });
30     }
31
32     @Override
33     protected void onActivityResult(int requestCode, int resultCode, @Nullable Intent data) {
34         super.onActivityResult(requestCode, resultCode, data);
35         if(requestCode == REQ_CODE){
36             if(resultCode == RESULT_OK){
37                 if(data != null & data.getStringExtra(SecondActivity.KEY_NAME) != null){
38                     textViewReceivedText.setText(data.getStringExtra(SecondActivity.KEY_NAME));
39                 }
40             }
41         }
42     }
43 }
```

- Từ androidx.appcompat 1.3.1 trở lên thì `startActivityForResult` không được hỗ trợ nữa.

```
public class FirstActivity extends AppCompatActivity {
    public static final int REQ_CODE = 100;
    TextView textViewReceivedText;

    ActivityResultLauncher<Intent> startForResult = registerForActivityResult(new ActivityResultContracts.StartActivityForResult(), new ActivityResultCallback<ActivityResult>() {
        @Override
        public void onActivityResult(ActivityResult result) {
            if(result != null && result.getResultCode() == RESULT_OK){
                if(result.getData() != null && result.getData().getStringExtra(SecondActivity.KEY_NAME) != null){
                    textViewReceivedText.setText(result.getData().getStringExtra(SecondActivity.KEY_NAME));
                }
            }
        }
    });

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_first);
        setTitle("First Activity");

        textViewReceivedText = findViewById(R.id.textViewReceivedText);

        findViewById(R.id.buttonGotoSecond).setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Intent intent = new Intent(packageContext: FirstActivity.this, SecondActivity.class);
                startForResult.launch(intent);
            }
        });
    }
}
```

Xử lý lỗi Deprecated for startActivityForResult

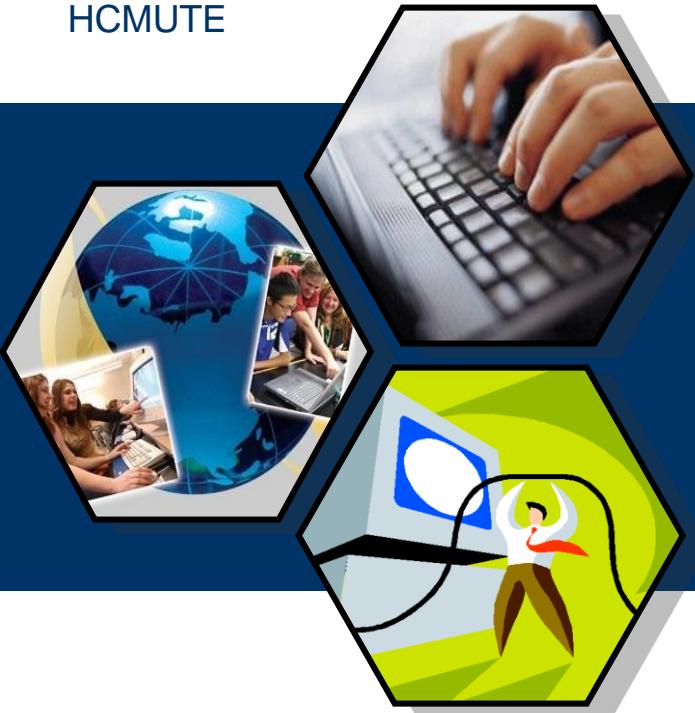
133

```
10 public class SecondActivity extends AppCompatActivity {
11     EditText editTextToSend;
12     public static final String KEY_NAME = "NAME";
13
14     @Override
15     protected void onCreate(Bundle savedInstanceState) {
16         super.onCreate(savedInstanceState);
17         setContentView(R.layout.activity_second);
18         setTitle("Second Activity");
19
20         editTextToSend = findViewById(R.id.editTextToSend);
21
22         findViewById(R.id.buttonSend).setOnClickListener(new View.OnClickListener() {
23             @Override
24             public void onClick(View view) {
25                 String enteredText = editTextToSend.getText().toString();
26                 Intent intent = new Intent();
27                 intent.putExtra(KEY_NAME, enteredText);
28                 setResult(RESULT_OK, intent);
29                 finish();
30             }
31         });
32     }
33 }
34 }
```



KHOA CÔNG NGHỆ THÔNG TIN

UTEx



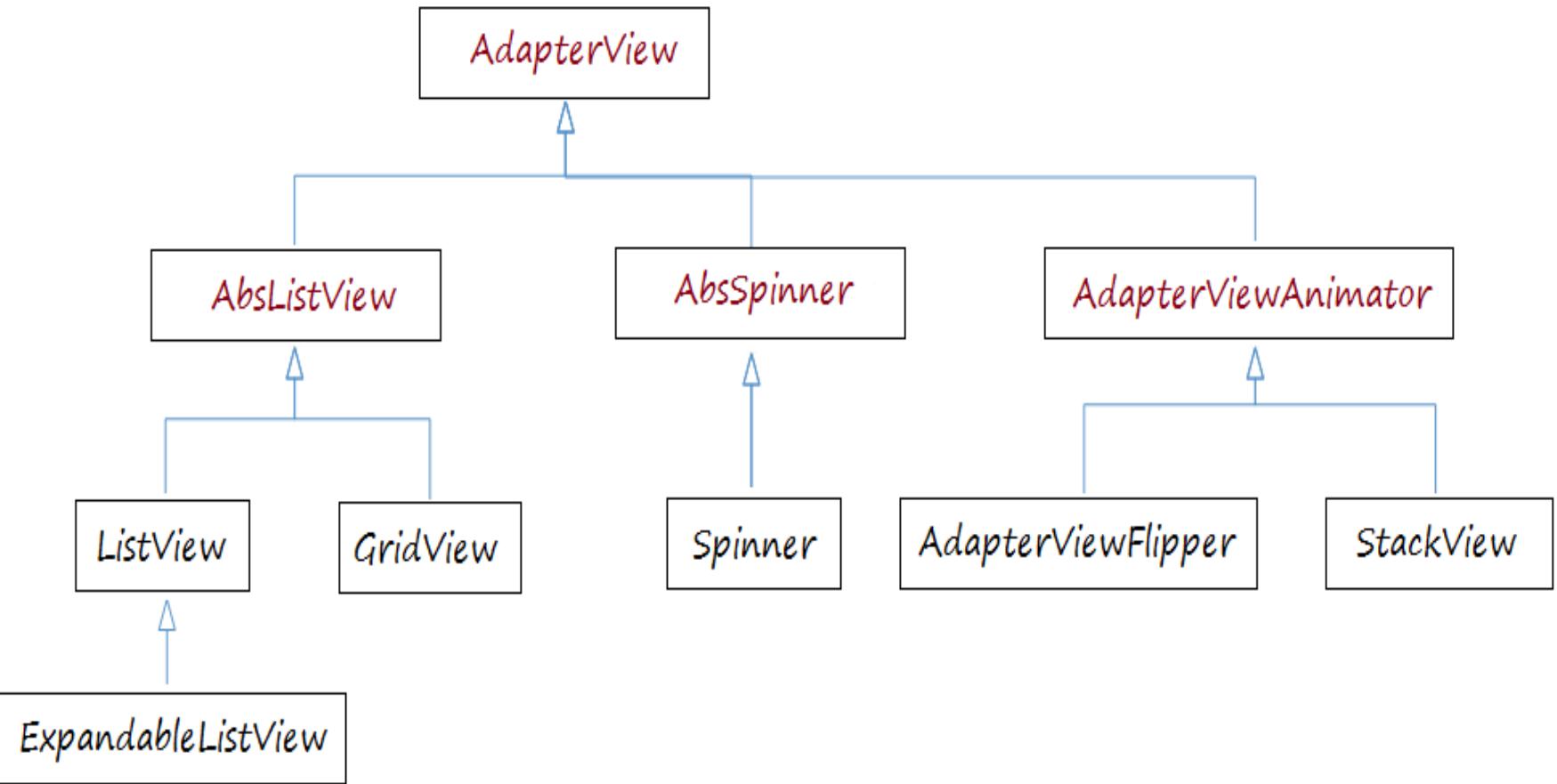
LISTVIEW

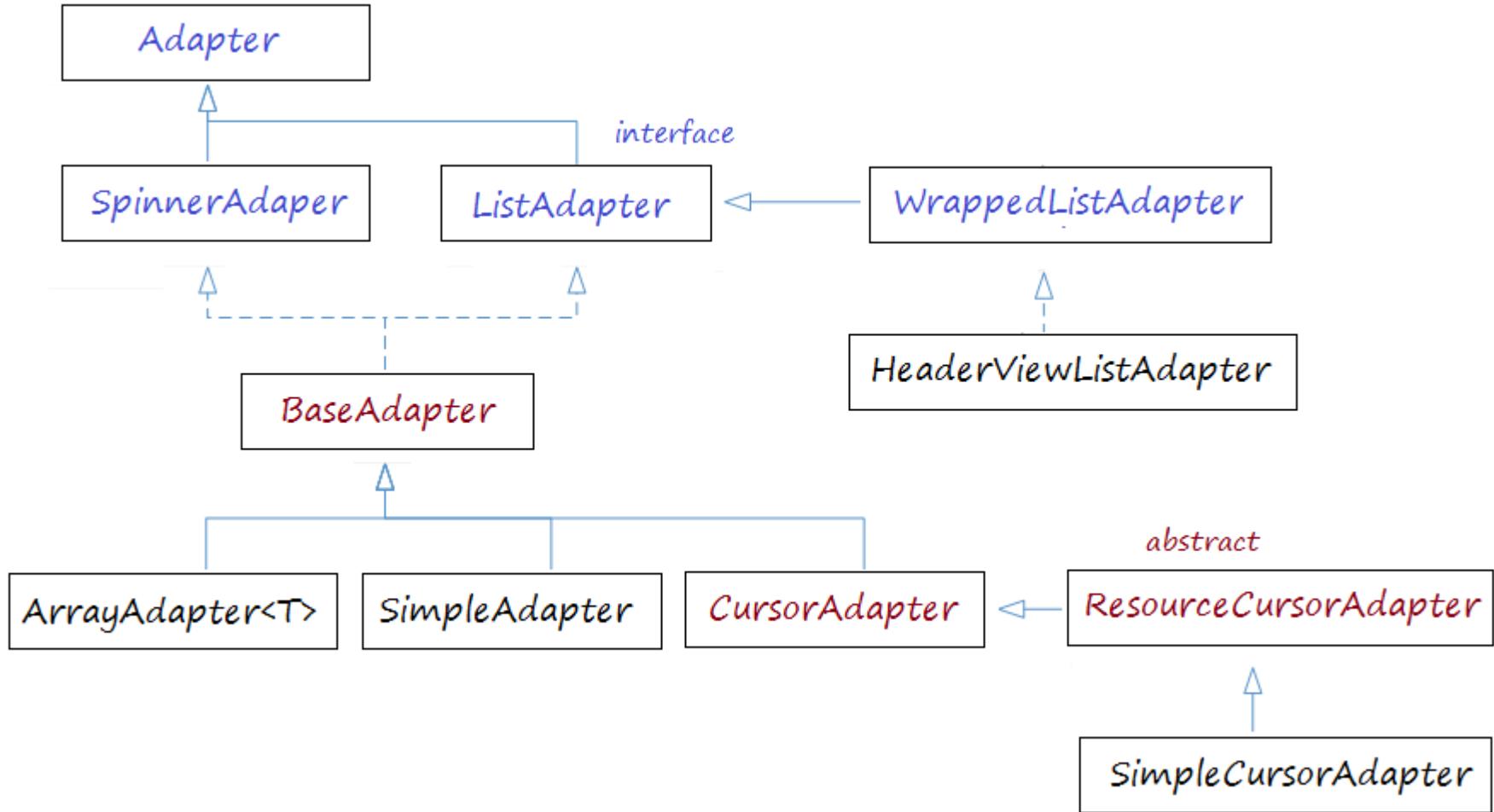
**Khoa Công nghệ Thông tin
Đại học Sư phạm Kỹ thuật TP.HCM**

ThS. Nguyễn Hữu Trung



- Có nhiều View cần tới Android Adapter để quản lý dữ liệu hiển thị, các View này là con của class AdapterView

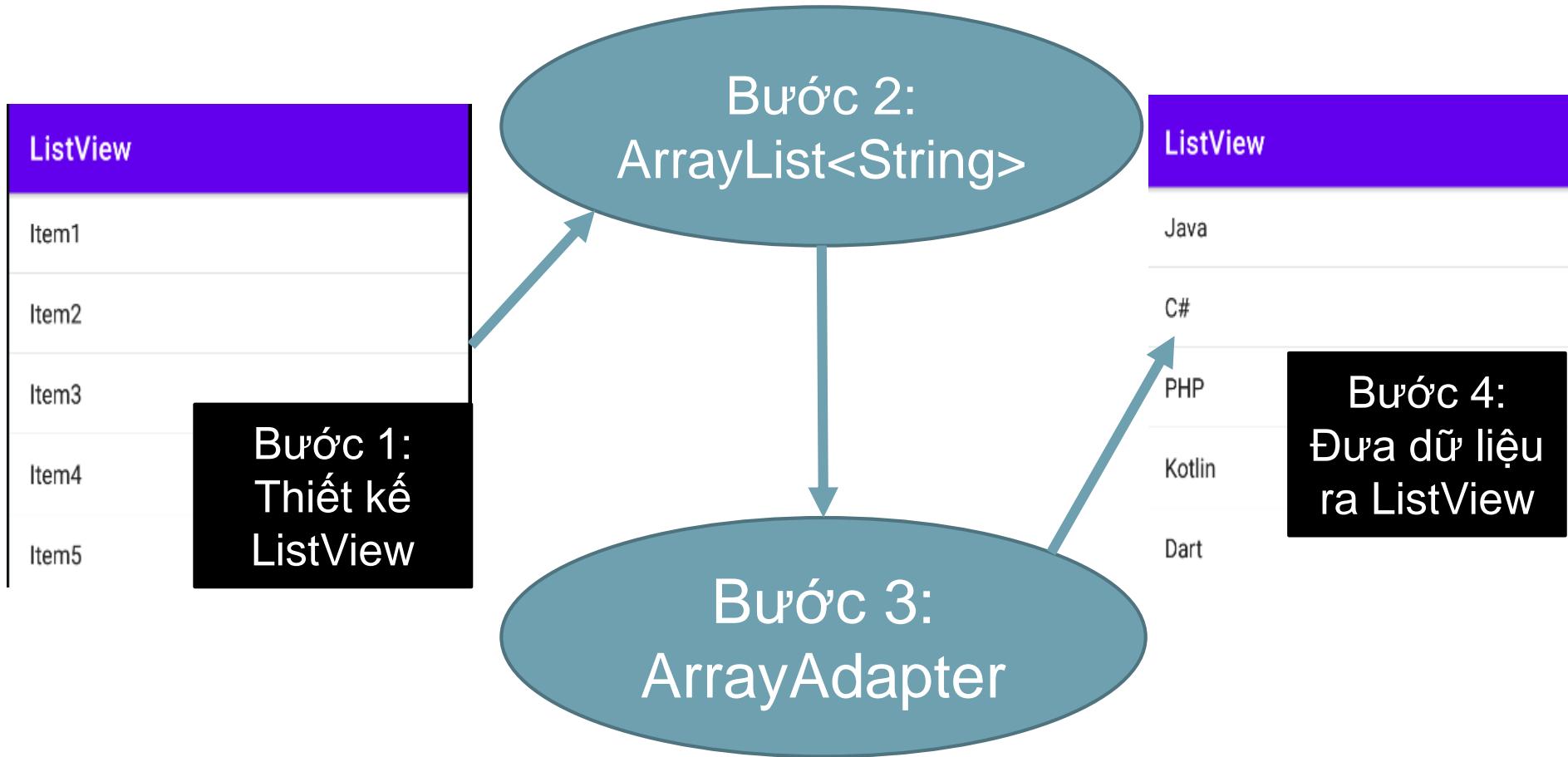




- ❑ **ListView** trong Android là một view group, hiển thị các item theo một danh sách có thể cuộn được theo chiều thẳng đứng.

```
<ListView  
    android:id="@+id/listview"  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content">  
</ListView>
```

Thuộc tính	Mô tả
android:id	ID là duy nhất để nhận diện Layout
android:divider	Nó có thể vẽ hoặc tô màu giữa các item trong danh sách.
android:dividerHeight	Nó xác định chiều cao của dải phân cách. Điều này có thể là px, dp, sp, in hoặc mm.
android:entries	Chỉ định tham chiếu đến một tài nguyên mảng sẽ điền vào ListView.
android:footerDividersEnabled	Khi được đặt thành false, ListView sẽ không vẽ dải phân cách trước mỗi chế độ xem chân trang. Giá trị mặc định là true.
android:headerDividersEnabled	Khi được đặt thành false, ListView sẽ không vẽ dải phân cách sau mỗi chế độ xem tiêu đề. Giá trị mặc định là true.



ListView

Item1

Item2

Item3

Item4

Item5

Bước 1:
Thiết kế
ListView

Mở file .xml thiết kế ListView

<ListView

```
    android:id="@+id/listview1"  
    android:layout_width="409dp"  
    android:layout_height="354dp"  
    tools:layout_editor_absoluteX="1dp"  
    tools:layout_editor_absoluteY="1dp"  
    tools:ignore="MissingConstraints" />
```

Bước 2:
ArrayList<String>

```
public class MainActivity extends  
AppCompatActivity {  
    //khai báo  
    ListView listView;  
    ArrayList<String> arrayList;
```

```
//ánh xạ  
listView = (ListView) findViewById(R.id.listview1);  
//Thêm dữ liệu vào List  
arrayList = new ArrayList<>();  
arrayList.add("Java");  
arrayList.add("C#");  
arrayList.add("PHP");  
arrayList.add("Kotlin");  
arrayList.add("Dart");
```

Bước 3: ArrayAdapter

ArrayAdapter có 03 tham số:

- Context: màn hình hiển thị
- Dạng Layout muốn đổ vào
- Dữ liệu List đổ vào

```
//Tạo ArrayAdapter
ArrayAdapter adapter = new ArrayAdapter(
    MainActivity.this,
    android.R.layout.simple_list_item_1,
    arrayList
);
```

ListView

Java

C#

PHP

Kotlin

Dart

Bước 4:
Đưa dữ liệu
ra ListView

//truyền dữ liệu từ adapter ra listview

`listView.setAdapter(adapter);`

ListView

Java

C#

PHP

Kotlin

Dart

2

//bắt sự kiện click nhanh trên từng dòng của Listview

```
listView.setOnItemClickListener(new AdapterView.OnItemClickListener() {
    @Override
    public void onItemClick(AdapterView<?> adapterView, View view, int i, long l) {
        //code yêu cầu
        //i: trả về vị trí click chuột trên ListView -> i ban đầu =0
        Toast.makeText(MainActivity.this,""+i,
        Toast.LENGTH_SHORT).show();
    }
});
```

Lấy nội dung: arrayList.get(i)

ListView

Java

C#

PHP

Kotlin

Dart

2

```
//bắt sự kiện click giữ trên từng dòng của ListView
listView.setOnItemLongClickListener(new
AdapterView.OnItemLongClickListener() {
    @Override
    public boolean
onItemLongClick(AdapterView<?> adapterView,
View view, int i, long l) {
        //code yêu cầu
        //i: trả về vị trí click chuột trên ListView -> i
        ban đầu =0
        Toast.makeText(MainActivity.this,"Bạn đang
nhấn giữ "+ i + "-" + arrayList.get(i) ,
Toast.LENGTH_SHORT).show();
        return false;
    }
});
```

ListView

Thêm môn học:

THÊM

```
EditText editText1;  
Button btnNhaph;
```

ASP.NET

Java

C#

PHP

Kotlin

Dart

```
editText1 = (EditText) findViewById(R.id.editText1);  
btnNhaph = (Button) findViewById(R.id.btnNhaph);
```

```
btnNhaph.setOnClickListener(new View.OnClickListener() {  
    @Override  
    public void onClick(View view) {  
        String name = editText1.getText().toString();  
        arrayList.add(name);  
        adapter.notifyDataSetChanged();  
    }  
});
```

ListView

Thêm môn học:

```
//Khai báo  
Button btnCapnhat;  
int vitri = -1;
```

THÊM

CẬP NHẬT

Java

C#

PHP

```
//ánh xạ  
btnCapNhat = (Button) findViewById(R.id.btnCapNhat);
```

```
//bắt sự kiện trên từng dòng của Listview  
listView.setOnItemClickListener(new  
AdapterView.OnItemClickListener() {  
    @Override  
    public void onItemClick(AdapterView<?> adapterView, View view,  
    int i, long l) {  
        //lấy nội dung đưa lên edittext  
        editText1.setText(arrayList.get(i));  
        vitri = i;  
    }  
});
```

ListView

Thêm môn học:

THÊM

CẬP NHẬT

Java

C#

PHP

```
btnCapNhat.setOnClickListener(new  
View.OnClickListener() {  
    @Override  
    public void onClick(View view) {  
        arrayList.set(vitri, editText1.getText().toString());  
        adapter.notifyDataSetChanged();  
    }  
});
```

```
//Xóa item  
arrayList.remove(i);  
adapter.notifyDataSetChanged();
```

Bước 1:
Tạo Class
MonHoc

Bước 3:
ArrayList<MonHoc>

Bước 4:
MonHocAdapter

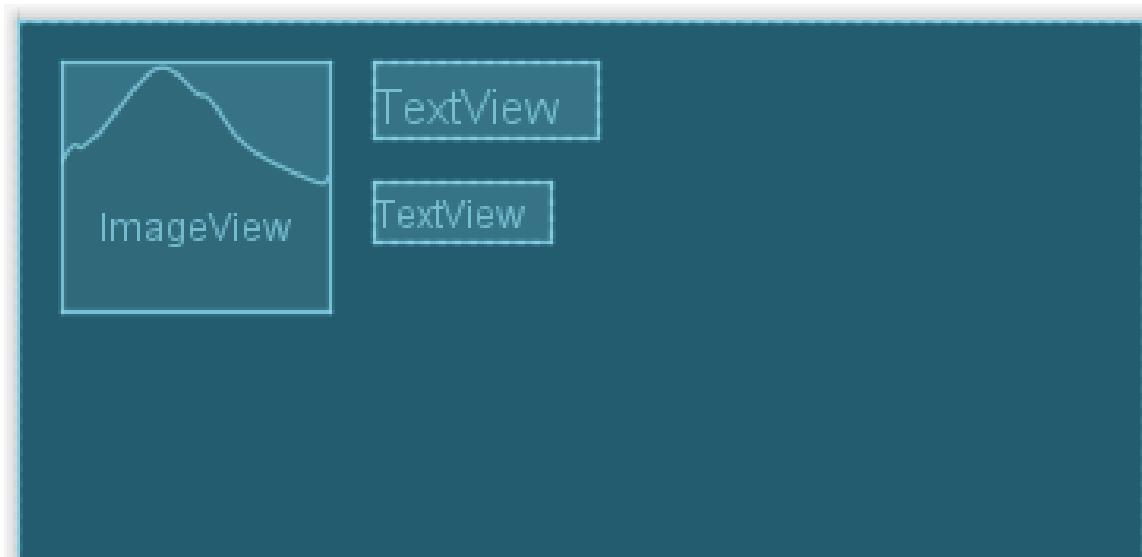
Bước 2:
row_monhoc.xml

Bước 5:
Đưa dữ liệu
ra ListView

Bước 1:
Tạo Class
MonHoc

```
package vn.iotstar.listview;  
  
public class MonHoc {  
    private String name;  
    private String desc;  
    private int pic;  
    //kích phải chọn Generate.. Chọn  
    constructor, getter and setter
```

Bước 2: tạo layout
row_monhoc.xml



Bước 4: MonHocAdapter

```
//trả về số dòng  
@Override  
public int getCount() {  
    return monHocList.size(); //lấy  
    kích thước monhoclist  
}
```

```
public class MonhocAdapter extends  
BaseAdapter {  
    //khai báo  
    private Context context;  
    private int layout;  
    private List<MonHoc> monHocList;  
  
    //tạo Constructors  
  
    public MonhocAdapter(Context context,  
    int layout, List<MonHoc> monHocList) {  
        this.context = context;  
        this.layout = layout;  
        this.monHocList = monHocList;  
    }
```

@Override

```
public View getView(int i, View view, ViewGroup viewGroup) {  
    //lấy context  
    LayoutInflater inflater = (LayoutInflater)  
        context.getSystemService(Context.LAYOUT_INFLATER_SERVICE);  
    //gọi view chưa layout  
    view = inflater.inflate(layout,null);  
    //ánh xạ view  
    TextView textName = (TextView) view.findViewById(R.id.textName);  
    TextView textDesc = (TextView) view.findViewById(R.id.textDesc);  
    ImageView imagePic = (ImageView) view.findViewById(R.id.imagePic);  
  
    //gán giá trị  
    MonHoc monHoc = monHocList.get(i);  
    textName.setText(monHoc.getName());  
    textDesc.setText(monHoc.getDesc());  
    imagePic.setImageResource(monHoc.getPic());  
    //trả về view  
    return view;  
}
```



Java

Java 1

C#

C# 1

PHP

PHP 1

Bước 5:
Đưa dữ liệu
ra ListView

```
public class MainActivity extends AppCompatActivity {  
    //khai báo  
    ListView listView;  
    ArrayList<MonHoc> arrayList;  
    MonhocAdapter adapter;  
  
    //ánh xạ  
    AnhXa();  
  
    //Tạo Adapter  
    adapter = new MonhocAdapter(MainActivity.this,  
        R.layout.row_monhoc,  
        arrayList  
    );  
    //truyền dữ liệu từ adapter ra listview  
    listView.setAdapter(adapter);
```



Java

Java 1

C#

C# 1

PHP



PHP 1

Bước 5:
Đưa dữ liệu
ra ListView

```
private void AnhXa() {
    listView = (ListView) findViewById(R.id.listview1);

    editText1 = (EditText) findViewById(R.id.editText1);
    btnNhaph = (Button) findViewById(R.id.btnNhaph);
    btnCapNhat = (Button) findViewById(R.id.btnCapNhat);

    //Thêm dữ liệu vào List
    arrayList = new ArrayList<>();
    arrayList.add(new MonHoc("Java","Java 1",R.drawable.java1));
    arrayList.add(new MonHoc("C#","C# 1",R.drawable.c));
    arrayList.add(new MonHoc("PHP","PHP 1",R.drawable.php));
    arrayList.add(new MonHoc("Kotlin","Kotlin
1",R.drawable.kotlin));
    arrayList.add(new MonHoc("Dart","Dart 1",R.drawable.dart));
}
```

```
Mở MonHocAdapter lên  
//tạo class viewholder  
private class ViewHolder{  
    TextView textView, textDesc;  
    ImageView imagePic;  
}
```

ViewHolder trong ListView

157

```
@Override
public View getView(int i, View view, ViewGroup viewGroup) {
    //khởi tạo viewholder
    ViewHolder viewHolder;
    //lấy context
    if (view==null){
        LayoutInflater inflater = (LayoutInflater)
context.getSystemService(Context.LAYOUT_INFLATER_SERVICE);
        //gọi view chưa layout
        view = inflater.inflate(layout,null);
        //ánh xạ view
        viewHolder = new ViewHolder();
        viewHolder.textName = (TextView) view.findViewById(R.id.textName);
        viewHolder.textDesc = (TextView) view.findViewById(R.id.textDesc);
        viewHolder.imagePic = (ImageView) view.findViewById(R.id.imagePic);
        view.setTag(viewHolder);
    }else{
        viewHolder= (ViewHolder) view.getTag();
    }
}
```

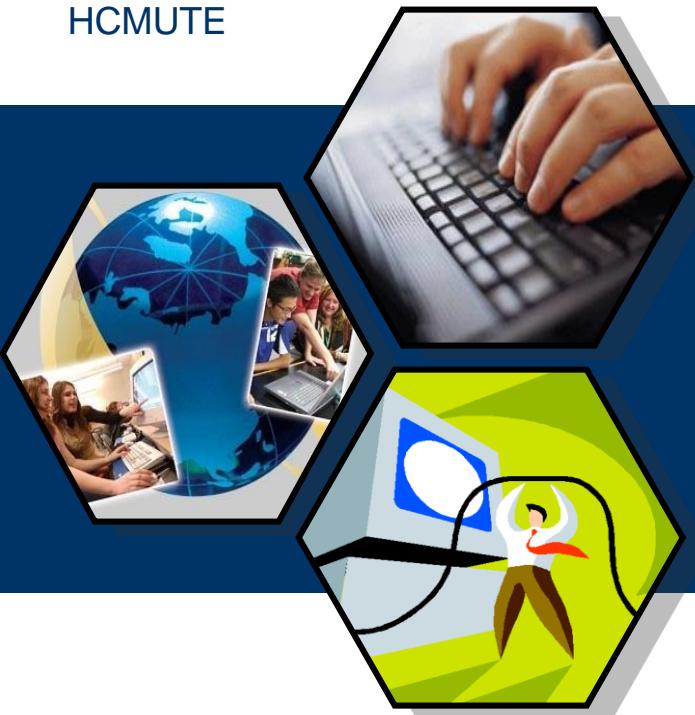
```
//gán giá trị  
MonHoc monHoc = monHocList.get(i);  
viewHolder.textName.setText(monHoc.getName());  
viewHolder.textDesc.setText(monHoc.getDesc());  
viewHolder.imagePic.setImageResource(monHoc.getPic());  
//trả về view  
return view;  
}
```



HCMUTE

KHOA CÔNG NGHỆ THÔNG TIN

UTEx



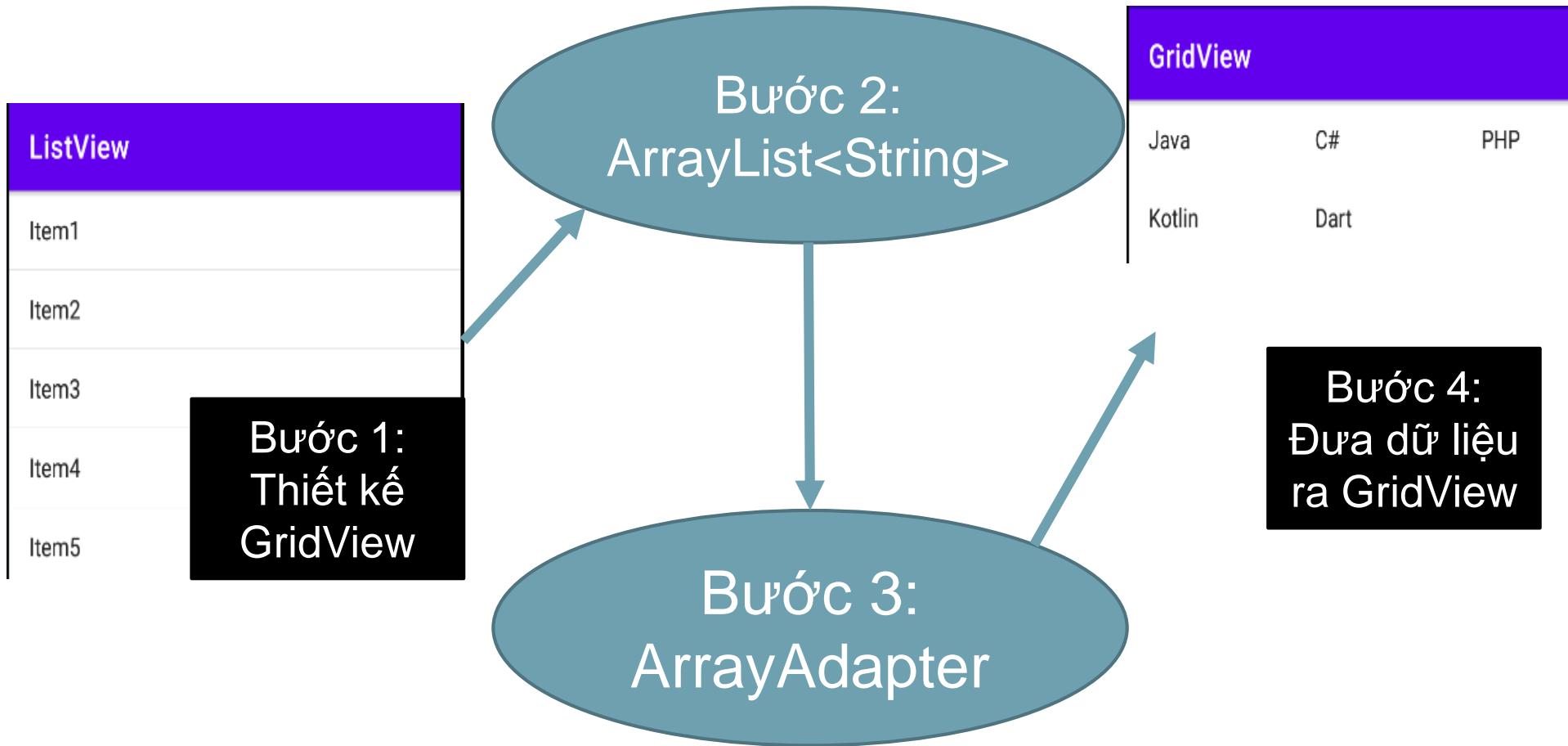
GRIDVIEW

Khoa Công nghệ Thông tin
Đại học Sư phạm Kỹ thuật TP.HCM

ThS. Nguyễn Hữu Trung



- GridView trong Android hiển thị các item trong mảng lưới hai chiều có thể scroll và các item này không cần thiết phải được định nghĩa trước, nhưng chúng tự động chèn vào Layout bởi sử dụng một ListAdapter.
- Các thuộc tính của GridView:
 - **android:verticalSpacing** -> Định nghĩa khoảng cách mặc định theo chiều dọc giữa các hàng. Có thể là trong px, dp, sp, in, hoặc mm.
 - **android:horizontalSpacing** -> Định nghĩa khoảng cách mặc định theo chiều ngang giữa các cột. Có thể là trong px, dp, sp, in, hoặc mm.
 - **android:numColumns** -> Xác định có bao nhiêu cột để hiển thị



GridView

Java

C#

PHP

Kotlin

Dart

Bước 1:
Thiết kế
ListView

Mở file .xml thiết kế GridView

```
<GridView  
    android:numColumns="3"  
    android:id="@+id/gridview1"  
    android:layout_width="409dp"  
    android:layout_height="729dp"  
    tools:layout_editor_absoluteX="1dp"  
    tools:layout_editor_absoluteY="1dp"  
    tools:ignore="MissingConstraints" />
```

Bước 2:
ArrayList<String>

```
public class MainActivity extends  
AppCompatActivity {  
    //khai báo  
    GridView gridView;  
    ArrayList<String> arrayList;
```

```
//ánh xạ  
gridView = (GridView) findViewById(R.id.gridview1);  
//Thêm dữ liệu vào List  
arrayList = new ArrayList<>();  
arrayList.add("Java");  
arrayList.add("C#");  
arrayList.add("PHP");  
arrayList.add("Kotlin");  
arrayList.add("Dart");
```

Bước 3: ArrayAdapter

ArrayAdapter có 03 tham số:

- Context: màn hình hiển thị
- Dạng Layout muốn đổ vào
- Dữ liệu List đổ vào

```
//Tạo ArrayAdapter
ArrayAdapter adapter = new ArrayAdapter(
    MainActivity.this,
    android.R.layout.simple_list_item_1,
    arrayList
);
```

GridView

Java

C#

PHP

Kotlin

Dart

//truyền dữ liệu từ adapter ra gridview
`gridView.setAdapter(adapter);`

Bước 4:
Đưa dữ liệu
ra ListView

ListView

Java

C#

PHP

Kotlin

Dart

2

//bắt sự kiện click nhanh trên từng dòng của Gridview

```
gridView.setOnItemClickListener(new AdapterView.OnItemClickListener() {
    @Override
    public void onItemClick(AdapterView<?> adapterView, View view, int i, long l) {
        //code yêu cầu
        //i: trả về vị trí click chuột trên GridView -> i ban đầu =0
        Toast.makeText(MainActivity.this, "" + i,
        Toast.LENGTH_SHORT).show();
    }
});
```

Lấy nội dung: arrayList.get(i)

ListView

Java

C#

PHP

Kotlin

Dart

2

//bắt sự kiện click giữ trên từng dòng của Gridview
gridView.setOnItemLongClickListener(new
AdapterView.OnItemLongClickListener() {
 @Override
 public boolean
onItemLongClick(AdapterView<?> adapterView,
View view, int i, long l) {
 //code yêu cầu
 //i: trả về vị trí click chuột trên GridView -> i
 ban đầu =0
 Toast.makeText(MainActivity.this,"Bạn đang
nhấn giữ "+ i + "-" + arrayList.get(i) ,
Toast.LENGTH_SHORT).show();
 return false;
}
});

ListView

Thêm môn học:

THÊM

```
EditText editText1;  
Button btnNhaph;
```

ASP.NET

Java

C#

PHP

Kotlin

Dart

```
editText1 = (EditText) findViewById(R.id.editText1);  
btnNhaph = (Button) findViewById(R.id.btnNhaph);
```

```
btnNhaph.setOnClickListener(new View.OnClickListener() {  
    @Override  
    public void onClick(View view) {  
        String name = editText1.getText().toString();  
        arrayList.add(name);  
        adapter.notifyDataSetChanged();  
    }  
});
```

ListView

Thêm môn học:

```
//Khai báo  
Button btnCapnhat;  
int vitri = -1;
```

THÊM

CẬP NHẬT

Java

C#

PHP

```
//ánh xạ  
btnCapNhat = (Button) findViewById(R.id.btnCapNhat);
```

```
//bắt sự kiện trên từng dòng của Gridview  
gridView.setOnItemClickListener(new  
AdapterView.OnItemClickListener() {  
    @Override  
    public void onItemClick(AdapterView<?> adapterView, View view,  
    int i, long l) {  
        //lấy nội dung đưa lên edittext  
        editText1.setText(arrayList.get(i));  
        vitri = i;  
    }  
});
```

ListView

Thêm môn học:

THÊM

CẬP NHẬT

Java

C#

PHP

```
btnCapNhat.setOnClickListener(new  
View.OnClickListener() {  
    @Override  
    public void onClick(View view) {  
        arrayList.set(vitri, editText1.getText().toString());  
        adapter.notifyDataSetChanged();  
    }  
});
```

```
//Xóa item  
arrayList.remove(i);  
adapter.notifyDataSetChanged();
```

Bước 1:
Tạo Class
MonHoc

Bước 3:
ArrayList<MonHoc>

Bước 2:
row_monhoc.xml

Bước 4:
MonHocAdapter

GridView



Java

Java 1



Kotlin

Kotlin 1



C#

C# 1



Dart



PHP

PHP 1



Dart 2

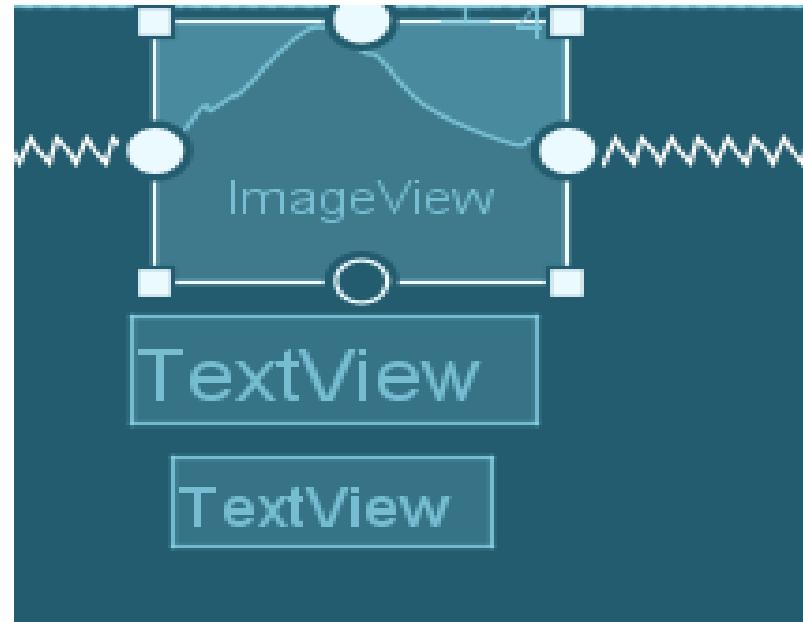
Bước 5:
Đưa dữ liệu
ra GridView

Bước 1:
Tạo Class
MonHoc

```
package vn.iotstar.gridview;

public class MonHoc {
    private String name;
    private String desc;
    private int pic;
    //kích phải chọn Generate.. Chọn
    constructor, getter and setter
```

Bước 2: tạo layout
row_monhoc.xml



Bước 4: MonHocAdapter

```
//trả về số dòng  
@Override  
public int getCount() {  
    return monHocList.size(); //lấy  
    kích thước monhoclist  
}
```

```
public class MonhocAdapter extends  
BaseAdapter {  
    //khai báo  
    private Context context;  
    private int layout;  
    private List<MonHoc> monHocList;  
  
    //tạo Constructors  
  
    public MonhocAdapter(Context context,  
    int layout, List<MonHoc> monHocList) {  
        this.context = context;  
        this.layout = layout;  
        this.monHocList = monHocList;  
    }
```

monHocList == null ? 0 : monHocList.size()

@Override

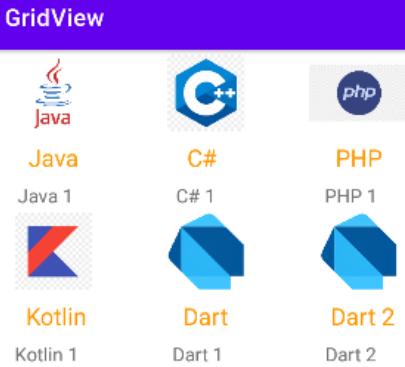
```
public View getView(int i, View view, ViewGroup viewGroup) {  
    //lấy context  
    LayoutInflater inflater = (LayoutInflater)  
        context.getSystemService(Context.LAYOUT_INFLATER_SERVICE);  
    //gọi view chưa layout  
    view = inflater.inflate(layout,null);  
    //ánh xạ view  
    TextView textName = (TextView) view.findViewById(R.id.textName);  
    TextView textDesc = (TextView) view.findViewById(R.id.textDesc);  
    ImageView imagePic = (ImageView) view.findViewById(R.id.imagePic);  
  
    //gán giá trị  
    MonHoc monHoc = monHocList.get(i);  
    textName.setText(monHoc.getName());  
    textDesc.setText(monHoc.getDesc());  
    imagePic.setImageResource(monHoc.getPic());  
    //trả về view  
    return view;  
}
```

GridView



Bước 5:
Đưa dữ liệu
ra ListView

```
public class MainActivity extends AppCompatActivity {  
    //khai báo  
    ListView listView;  
    ArrayList<MonHoc> arrayList;  
    MonhocAdapter adapter;  
  
    //ánh xạ  
    AnhXa();  
  
    //Tạo Adapter  
    adapter = new MonhocAdapter(MainActivity.this,  
        R.layout.row_monhoc,  
        arrayList  
    );  
    //truyền dữ liệu từ adapter ra listview  
    listView.setAdapter(adapter);
```



Bước 5:
Đưa dữ liệu
ra ListView

```
private void AnhXa() {  
    gridView = (GridView) findViewById(R.id.gridview1);  
  
    editText1 = (EditText) findViewById(R.id.editText1);  
    btnNhaph = (Button) findViewById(R.id.btnNhaph);  
    btnCapNhat = (Button) findViewById(R.id.btnCapNhat);  
  
    //Thêm dữ liệu vào List  
    arrayList = new ArrayList<>();  
    arrayList.add(new MonHoc("Java", "Java 1", R.drawable.java1));  
    arrayList.add(new MonHoc("C#", "C# 1", R.drawable.c));  
    arrayList.add(new MonHoc("PHP", "PHP 1", R.drawable.php));  
    arrayList.add(new MonHoc("Kotlin", "Kotlin  
1", R.drawable.kotlin));  
    arrayList.add(new MonHoc("Dart", "Dart 1", R.drawable.dart));  
}  
}
```

Mở MonHocAdapter lên
//tạo class viewholder
private class ViewHolder{
 TextView textView, textDesc;
 ImageView imagePic;
}

ViewHolder trong GridView

179

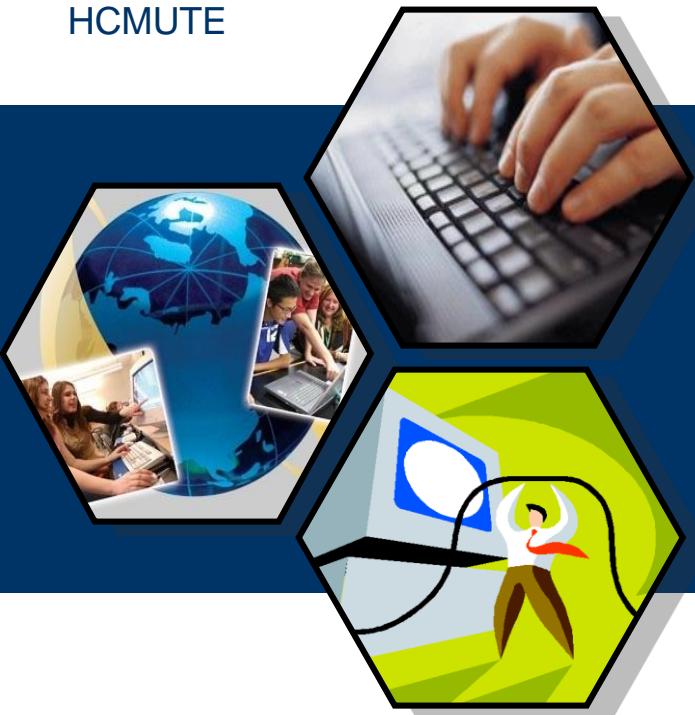
```
@Override
public View getView(int i, View view, ViewGroup viewGroup) {
    //khởi tạo viewholder
    ViewHolder viewHolder;
    //lấy context
    if (view==null){
        LayoutInflater inflater = (LayoutInflater)
context.getSystemService(Context.LAYOUT_INFLATER_SERVICE);
        //gọi view chưa layout
        view = inflater.inflate(layout,null);
        //ánh xạ view
        viewHolder = new ViewHolder();
        viewHolder.textName = (TextView) view.findViewById(R.id.textName);
        viewHolder.textDesc = (TextView) view.findViewById(R.id.textDesc);
        viewHolder.imagePic = (ImageView) view.findViewById(R.id.imagePic);
        view.setTag(viewHolder);
    }else{
        viewHolder= (ViewHolder) view.getTag();
    }
}
```

```
//gán giá trị  
MonHoc monHoc = monHocList.get(i);  
viewHolder.textName.setText(monHoc.getName());  
viewHolder.textDesc.setText(monHoc.getDesc());  
viewHolder.imagePic.setImageResource(monHoc.getPic());  
//trả về view  
return view;  
}
```



KHOA CÔNG NGHỆ THÔNG TIN

UTEx



RECYCLEVIEW

Khoa Công nghệ Thông tin
Đại học Sư phạm Kỹ thuật TP.HCM

ThS. Nguyễn Hữu Trung



- ❑ RecyclerView là 1 ViewGroup, được giới thiệu trong Android L (Android API 21), đây là 1 ViewGroup, có chức năng tương tự ListView nhưng nó mạnh mẽ, linh hoạt hơn rất nhiều, ListView chỉ hỗ trợ cuộn các item trong ListView theo chiều dọc (vertical) mà không hỗ trợ cuộn theo chiều ngang (horizontal), RecyclerView hỗ trợ được tất cả và hơn thế nữa.

- So với ListView thì RecyclerView có những điểm vượt trội hơn như sau:
 - **ListView** không cần sử dụng ViewHolder để cải thiện hiệu suất của ListView. Nhưng ngược lại, khi tạo 1 Adapter sử dụng với **RecyclerView**, bắt buộc phải sử dụng ViewHolder để cải thiện hiệu suất.
 - Mục đích sử dụng **ViewHolder**, để tái sử dụng View, nhằm tránh việc tạo View mới và **findViewById** quá nhiều.
 - ListView chỉ hỗ trợ danh sách dạng cuộn dọc, RecyclerView cung cấp **RecyclerView.LayoutManager** cho phép layout các item trong ListView theo các kiểu khác nhau (ngang, dọc, dạng lưới, dạng staggered grid).
 - Xử lý animation cho các item trong ListView không dễ dàng nhưng RecyclerView có hỗ trợ ItemAnimator giúp xử lý animation khi thêm vào hay xóa 1 item ra khỏi Recycler 1 cách dễ dàng. Mặc định RecyclerView, sử dụng DefaultItemAnimator.
 - ListView sử dụng divider không được linh hoạt nhưng với RecyclerView có hỗ trợ ItemDecoration, cho phép vẽ divider 1 cách tùy thích.
 - ListView hỗ trợ các phương thức `setOnItemClickListener()` và `setOnLongItemClickListener()` để chọn 1 item trong ListView. RecyclerView chỉ hỗ trợ 1 phương thức `onItemTouchListener()`.

- RecyclerView.Adapter

- Đây là thành phần xử lý dữ liệu collection (dữ liệu kiểu danh sách) và bind (gắn) những dữ liệu này, lên các item của RecyclerView.
- Khi tạo custom Adapter phải override lại 2 phương thức chính là:
 - onCreateViewHolder(): dùng để tạo View mới cho RecyclerView, nếu RecyclerView đã cached lại View thì phương thức này sẽ không được gọi.
 - onBindViewHolder(): dùng gắn dữ liệu vào View.

- LayoutManager

- Là thành phần có chức năng sắp xếp các item trong RecyclerView. Các item cuộn dọc hay ngang đều phụ thuộc vào đặt LayoutManager cho RecyclerView.
- Các lớp con của LayoutManager:
 - LinearLayoutManager: hỗ trợ cuộn các item theo chiều ngang hay chiều dọc.
 - GridLayoutManager: layout các item trong RecyclerView dưới dạng Grid giống như khi sử dụng GridView.
 - StaggeredGridLayoutManager: layout các item trong ListView dưới dạng lưới so le.
 - ItemAnimator: là thành phần hỗ trợ animation khi thêm vào hay xóa 1 item ra khỏi RecyclerView. Để tìm hiểu rõ phần này, cần tìm hiểu các lớp sau:
 - ItemAnimator: là lớp đại diện cho khung sườn của animation trong RecyclerView.
 - SimpleItemAnimator: là lớp wrapper lại ItemAnimator.
 - DefaultItemAnimator: lớp xử lý animation mặc định sử dụng trong RecyclerView.

1. Tạo model lớp để chứa dữ liệu.
2. Thêm RecyclerView vào **main_activity.xml**.
3. Tạo giao diện cho 1 dòng.
4. Tạo custom Adapter và gán dữ liệu cho từng dòng trong Adapter.
5. Cài đặt RecyclerView trong **MainActivity.java**.

□ Bước 1: tạo model để chứa dữ liệu

```
public class SongModel implements Serializable {  
    private String mCode;  
    private String mTitle;  
    private String mLyric;  
    private String mArtist;  
    //TẠO CONSTRUCTOR, GETTER VÀ SETTER BẰNG  
    CÁCH KÍCH PHẢI CHUỘT CHỌN GENERATE..  
}
```

□ Bước 2: thêm RecyclerView vào main_activity.xml

```
<androidx.recyclerview.widget.RecyclerView  
    android:id="@+id/rv_songs"  
    android:layout_width="match_parent"  
    android:layout_height="match_parent"  
    app:layout_constraintBottom_toBottomOf="parent"  
    app:layout_constraintEnd_toEndOf="parent" />
```



□ **Bước 3: tạo giao diện cho 1 row để hiển thị lên RecyclerView**

File **row_item_song.xml** trong thư mục layout của project, được thiết kế để hiển thị dòng như hình dưới đây:

```
<TextView  
    android:id="@+id/tv_title"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:ellipsize="end"  
    android:textColor="#2c3e50"  
    android:textSize="16dp"  
    android:textStyle="bold" />  
  
<TextView  
    android:id="@+id/tv_lyric"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:ellipsize="end"  
    android:textColor="#34495e"  
    android:textSize="14dp" />  
  
<TextView  
    android:id="@+id/tv_artist"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:textColor="#7f8c8d"  
    android:textSize="14dp" />
```

```
<TextView  
    android:id="@+id/tv_code"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:textColor="#19b395"  
    android:textSize="18dp" />
```

□ Bước 4: tạo custom Adapter và gắn dữ liệu cho từng dòng trong Adapter

```
public class SongAdapter extends RecyclerView.Adapter<SongAdapter.SongViewHolder> {  
    private static final String TAG = "SongAdapter";  
    private List<SongModel> mSongs;  
    private Context mContext;  
    private LayoutInflator mLayoutInflater;  
    public SongAdapter(Context context, List<SongModel> datas) {  
        mContext = context;  
        mSongs = datas;  
        mLayoutInflater = LayoutInflator.from(context);  
    }  
    @Override  
    public SongViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {  
        // Inflate view from row_item_song.xml  
        View itemView = mLayoutInflater.inflate(R.layout.row_item_song, parent, attachToRoot: false);  
        return new SongViewHolder(itemView);  
    }  
    @Override  
    public void onBindViewHolder(SongViewHolder holder, int position) {  
        // Get song in mSong via position  
        SongModel song = mSongs.get(position);  
        //bind data to viewholder  
        holder.tvCode.setText(song.getmCode());  
        holder.tvTitle.setText(song.getmTitle());  
        holder.tvLyric.setText(song.getmLyric());  
        holder.tvArtist.setText(song.getmArtist());  
    }  
}
```

□ Bước 4: tạo custom Adapter và gắn dữ liệu cho từng dòng trong Adapter

```
@Override  
public int getItemCount() {  
    return mSongs.size();  
}  
  
class SongViewHolder extends RecyclerView.ViewHolder {  
    private TextView tvCode;  
    private TextView tvTitle;  
    private TextView tvLyric;  
    private TextView tvArtist;  
  
    public SongViewHolder(View itemView) {  
        super(itemView);  
        tvCode = (TextView) itemView.findViewById(R.id.tv_code);  
        tvTitle = (TextView) itemView.findViewById(R.id.tv_title);  
        tvLyric = (TextView) itemView.findViewById(R.id.tv_lyric);  
        tvArtist = (TextView) itemView.findViewById(R.id.tv_artist);  
    }  
}
```

Tạo **Custom Adapter** trong **RecyclerView** dễ dàng hơn so với **ListView**, chỉ cần tạo 1 lớp kế thừa **RecyclerView**. Còn **ViewHolder** chỉ việc xử lý trong **onCreateViewHolder** để tạo mới **ViewHolder**. Và **onBindViewHolder** để gán dữ liệu lấy từ collection vào **ViewHolder**.

□ Bước 5: cài đặt RecyclerView trong MainActivity.java

Không giống như **ListView** chỉ cần đặt Adapter cho ListView là đủ. Đối với **RecyclerView** phải đặt 2 thành phần bắt buộc dưới đây:

- Đặt **adapter** cho **RecyclerView** sử dụng phương thức **setAdapter()**.
- Đặt **layout manager** để muôn layout các item như các kiểu dọc, ngang, lưới, lưới so le bằng cách sử dụng phương thức **setLayoutManager()**.

```
rvSongs.setAdapter(mSongAdapter);
LinearLayoutManager linearLayoutManager = new
LinearLayoutManager(this, LinearLayoutManager.VERTICAL, false);
rvSongs.setLayoutManager(linearLayoutManager);
```

Vì muốn RecyclerView cuộn theo chiều dọc nên sử dụng `LinearLayoutManager.VERTICAL`. Nếu muốn cuộn theo chiều ngang thì truyền vào `LinearLayoutManager.HORIZONTAL`:

```
LinearLayoutManager linearLayoutManager = new LinearLayoutManager(this, LinearLayoutManager.HORIZONTAL,
false);
```

Sử dụng những lớp sau để `setLayoutManager()` cho RecyclerView:

- `GridLayoutManager`: layout các item trong RecyclerView theo dạng lưới.
- `StaggeredLayoutManager`: layout các item trong RecyclerView theo dạng lưới so le nhau.

□ Bước 5: cài đặt RecyclerView trong MainActivity.java

```
public class MainActivity extends AppCompatActivity {

    private RecyclerView rvSongs;
    private SongAdapter mSongAdapter;
    private List<SongModel> mSongs;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        rvSongs = (RecyclerView) findViewById(R.id.rv_songs);

        // Create song data
        mSongs = new ArrayList<>();
        mSongs.add(new SongModel( mCode: "60696", mTitle: "NẾU EM CÒN TỒN TẠI", mLyric: "Khi anh bắt đầu 1 tình yêu Là lúc anh tự thay", mArtist: "Trịnh Đình Quang"));
        mSongs.add(new SongModel( mCode: "60701", mTitle: "NGỐC", mLyric: "Có rất nhiều những câu chuyện Em dấu riêng mình em biết", mArtist: "Khắc Việt"));
        mSongs.add(new SongModel( mCode: "60650", mTitle: "HÃY TIN ANH LẦN NỮA", mLyric: "Dẫu cho ta đã sai khi ở bên nhau Cô yêu thương", mArtist: "Thiên Dũng"));
        mSongs.add(new SongModel( mCode: "60610", mTitle: "CHUỖI NGÀY VẮNG EM", mLyric: "Từ khi em bước ra đi cõi lòng anh ngập tràn bao", mArtist: "Duy Cường"));
        mSongs.add(new SongModel( mCode: "60656", mTitle: "KHI NGƯỜI MÌNH YÊU KHÓC", mLyric: "Nước mắt em đang rơi trên những ngón tay Nước mắt em", mArtist: "Phạm Mạnh Quỳnh"));
        mSongs.add(new SongModel( mCode: "60685", mTitle: "MỎ", mLyric: "Anh mơ gặp em anh mơ được ôm anh mơ được gần", mArtist: "Trịnh Thăng Bình"));
        mSongs.add(new SongModel( mCode: "60752", mTitle: "TÌNH YÊU CHẮP VÁ", mLyric: "Muốn đi xa nơi yêu thương mình từng có Để không nghe", mArtist: "Mr. Siro"));
        mSongs.add(new SongModel( mCode: "60608", mTitle: "CHỞ NGÀY MÙA TAN", mLyric: "1 ngày mưa và em khuất xa nơi anh bóng dáng cứ", mArtist: "Trung Đức"));
        mSongs.add(new SongModel( mCode: "60603", mTitle: "CÂU HỎI EM CHUA TRÀ LỜI", mLyric: "Cần nơi em 1 lời giải thích thật lòng Đừng lặng im", mArtist: "Yuki Huy Nam"));
        mSongs.add(new SongModel( mCode: "60720", mTitle: "QUA ĐÌ LĂNG LỄ", mLyric: "Đôi khi đến với nhau yêu thương chẳng được lâu nhưng khi", mArtist: "Phan Mạnh Quỳnh"));
        mSongs.add(new SongModel( mCode: "60856", mTitle: "QUÊN ANH LÀ ĐIỀU EM KHÔNG THỂ - REMIX", mLyric: "Cần thêm bao lâu để em quên đi niềm đau Cần thêm", mArtist: "Thiện Ngôn"));
        mSongAdapter = new SongAdapter( context: this, mSongs);
        rvSongs.setAdapter(mSongAdapter);
        LinearLayoutManager linearLayoutManager = new LinearLayoutManager( context: this, LinearLayoutManager.VERTICAL, reverseLayout: false);
        rvSongs.setLayoutManager(linearLayoutManager);
    }
}
```

- Chọn item trong RecyclerView, bằng cách setOnClickListener cho itemView, trong constructor của SongViewHolder như sau:

```
//thiết lập sự kiện
itemView.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        SongModel song = mSongs.get(getAdapterPosition());
        Toast.makeText(mContext, song.getmTitle(),
        Toast.LENGTH_SHORT).show();
    }
});
```

- Giới thiệu và cách tạo RecyclerView nâng cao, RecyclerView với nhiều dòng khác nhau hay còn được gọi là RecyclerView Multiple View Type, có thể nhận thấy tính năng News Feed của Facebook.
- Việc xây dựng RecylerView có nhiều dòng item khác nhau cũng giống như RecyclerView chỉ có 1 dòng item nhưng khác biệt ở custom adapter như sau:
 - Tạo n ViewHolder ứng với n dòng item.
 - Override lại phương thức getItemViewType(int position) để lấy về kiểu tương ứng với từng vị trí trong collection.
 - Phương thức onCreateViewHolder() phải dựa vào kiểu để tạo ViewHolder tương ứng.
 - Phương thức onBindViewHolder() phải dựa vào kiểu để gắn dữ liệu tương ứng vào ViewHolder.

- Hướng dẫn tạo Custom Adapter
- Tạo 3 file XML tương ứng với 3 dòng item như sau:
 - ▣ **row_text.xml**: có 1 TextView để hiển thị văn bản.
 - ▣ **row_image.xml**: hiển thị 1 hình ảnh ImageView.
 - ▣ **row_user.xml**: có 2 TextView hiển thị tên và địa chỉ của user.

- Hướng dẫn tạo Custom Adapter
- Tạo 3 file XML tương ứng với 3 dòng item như sau:
 - ▣ **row_text.xml**: có 1 TextView để hiển thị văn bản.
 - ▣ **row_image.xml**: hiển thị 1 hình ảnh ImageView.
 - ▣ **row_user.xml**: có 2 TextView hiển thị tên và địa chỉ của user.

row_text.xml

```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
        android:layout_width="match_parent"
        android:layout_marginTop="3dp"
        android:layout_height="wrap_content">
    <TextView
        android:background="#ecf0f1"
        android:padding="5dp"
        android:textColor="#2c3e50"
        android:id="@+id/tv_text"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />
</FrameLayout>
```

row_image.xml

```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="3dp">
    <ImageView
        android:id="@+id/imv_image"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="#95a5a6" />
</FrameLayout>
```

row_user.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="3dp"
    android:background="#bdc3c7"
    android:orientation="vertical">
    <TextView
        android:id="@+id/tv_name"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:textColor="#34495e"
        android:textSize="18dp" />
    <TextView
        android:id="@+id/tv_address"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:textColor="#16a085"
        android:textSize="16dp" />
</LinearLayout>
```

Tạo UserModel.java

```
public class UserModel implements Serializable {  
    private String name;  
    private String address;  
    //tạo Contructor, getter và setter bằng cách nhấp chuột phải  
    //chọn Generate...  
}
```

Tạo UserModel.java

```
public class UserModel implements Serializable {  
    private String name;  
    private String address;  
    //tạo Contructor, getter và setter bằng cách nhấp chuột phải  
    //chọn Generate...  
}
```

Tiếp theo, tiến hành tạo 1 CustomAdapter.

```
public class CustomAdapter extends RecyclerView.Adapter<RecyclerView.ViewHolder> {  
    private Context mContext;  
    private List<Object> mObjects;  
    public static final int TEXT = 0;  
    public static final int IMAGE = 1;  
    public static final int USER = 2;  
    public CustomAdapter(Context context, List<Object> objects) {  
        mContext = context;  
        mObjects = objects;  
    }
```

- Có 3 ViewHolder như sau:

```
public class TextViewHolder extends RecyclerView.ViewHolder {  
    private TextView tvText;  
  
    public TextViewHolder(View itemView) {  
        super(itemView);  
        tvText = (TextView) itemView.findViewById(R.id.tv_text);  
  
        itemView.setOnClickListener(new View.OnClickListener() {  
            @Override  
            public void onClick(View view) {  
                Toast.makeText(mContext, mObjects.get(getAdapterPosition()).toString(), Toast.LENGTH_SHORT).show();  
            }  
        });  
    }  
}
```

□ Có 3 ViewHolder như sau:

```
public class ImageViewHolder extends RecyclerView.ViewHolder {  
    private ImageView imvImage;  
  
    public ImageViewHolder(View itemView) {  
        super(itemView);  
  
        imvImage = (ImageView) itemView.findViewById(R.id.imv_image);  
        itemView.setOnClickListener(new View.OnClickListener() {  
            @Override  
            public void onClick(View view) {  
                Toast.makeText(mContext, mObjects.get(getAdapterPosition()).toString(), Toast.LENGTH_SHORT).show();  
            }  
        });  
    }  
}
```

□ Có 3 ViewHolder như sau:

```
public class UserViewHolder extends RecyclerView.ViewHolder {  
    private TextView tvName;  
    private TextView tvAddress;  
    public UserViewHolder(View itemView) {  
        super(itemView);  
        tvName = (TextView) itemView.findViewById(R.id.tv_name);  
        tvAddress = (TextView) itemView.findViewById(R.id.tv_address);  
  
        itemView.setOnClickListener(new View.OnClickListener() {  
            @Override  
            public void onClick(View view) {  
                UserModel user = (UserModel) mObjects.get(getAdapterPosition());  
                Toast.makeText(mContext, text: user.getName() + ", " + user.getAddress(), Toast.LENGTH_SHORT).show();  
            }  
        });  
    }  
}
```

❑ Override lại phương thức getItemViewType():

```
@Override  
public int getItemViewType(int position) {  
    if (mObjects.get(position) instanceof String)  
        return TEXT;  
    else if (mObjects.get(position) instanceof Integer)  
        return IMAGE;  
    else if (mObjects.get(position) instanceof UserModel)  
        return USER;  
    return -1;  
}
```

□ Override lại phương thức onCreateViewHolder():

```
@Override  
public RecyclerView.ViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {  
    LayoutInflater li = LayoutInflater.from(mContext);  
    switch (viewType) {  
        case TEXT:  
            View itemView0 = li.inflate(R.layout.row_text, parent, attachToRoot: false);  
            return new TextViewHolder(itemView0);  
        case IMAGE:  
            View itemView1 = li.inflate(R.layout.row_image, parent, attachToRoot: false);  
            return new ImageViewHolder(itemView1);  
        case USER:  
            View itemView2 = li.inflate(R.layout.row_user, parent, attachToRoot: false);  
            return new UserViewHolder(itemView2);  
        default:  
            break;  
    }  
    return null;  
}
```

❑ Override lại phương thức onBindViewHolder():

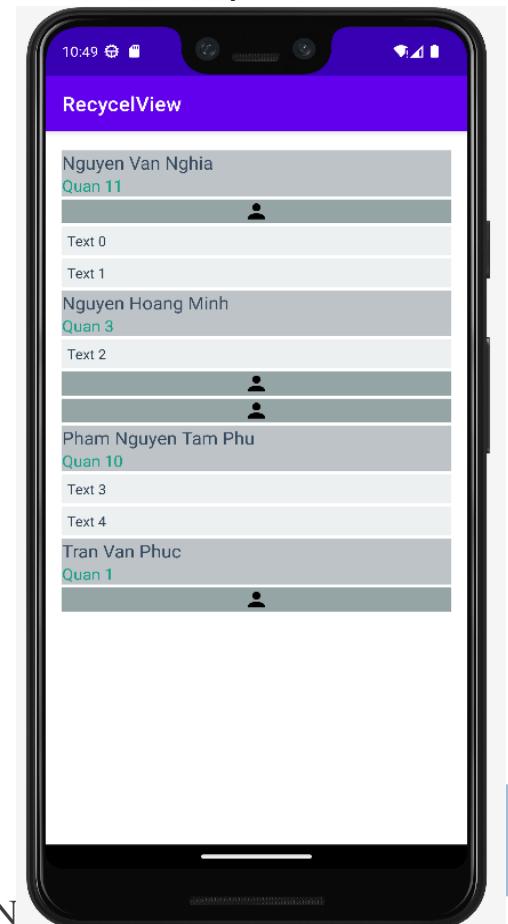
```
@Override  
public void onBindViewHolder(RecyclerView.ViewHolder holder, int position) {  
    switch (getItemViewType(position)) {  
        case TEXT:  
            TextViewHolder textViewHolder = (TextViewHolder) holder;  
            textViewHolder.tvText.setText(mObjects.get(position).toString());  
            break;  
        case IMAGE:  
            ImageViewHolder imageViewHolder = (ImageViewHolder) holder;  
            imageViewHolder.imvImage.setImageResource((int) mObjects.get(position));  
            break;  
        case USER:  
            UserModel user = (UserModel) mObjects.get(position);  
            UserViewHolder userViewHolder = (UserViewHolder) holder;  
            userViewHolder.tvName.setText(user.getName());  
            userViewHolder.tvAddress.setText(user.getAddress());  
            break;  
    }  
  
    @Override  
    public int getItemCount() {  
        return mObjects.size();  
    }  
}
```

- Sau khi đã tạo xong CustomAdapter thì thêm RecyclerView vào **activity_user.xml** và cài đặt cho RecyclerView trong **UserActivity.java** như sau:

```
<androidx.recyclerview.widget.RecyclerView  
    android:id="@+id/rv_multipe_view_type"  
    android:layout_width="match_parent"  
    android:layout_height="match_parent"  
    android:layout_marginStart="16dp"  
    android:layout_marginTop="16dp"  
    android:layout_marginEnd="16dp"  
    android:layout_marginBottom="16dp"  
    app:layout_constraintBottom_toBottomOf="parent"  
    app:layout_constraintEnd_toEndOf="parent"  
    app:layout_constraintStart_toStartOf="parent"  
    app:layout_constraintTop_toTopOf="parent" />
```

- Sau khi đã tạo xong CustomAdapter thì thêm RecyclerView vào **activity_user.xml** và cài đặt cho RecyclerView trong **UserActivity.java** như sau:
- Mỗi phần tử trong danh sách đều có kiểu Object từ đó có thể thêm bất cứ kiểu dữ liệu vào danh sách này (tất cả đối tượng trong Java đều kế thừa từ Object).

```
public class UserActivity extends AppCompatActivity {  
    private static final String TAG = "MainActivity";  
    private RecyclerView rvMultipleViewType;  
    private List<Object> mData;  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_user);  
        rvMultipleViewType = (RecyclerView) findViewById(R.id.rv_multiple_view_type);  
        mData = new ArrayList<>();  
        mData.add(new UserModel( name: "Nguyen Van Nghia", address: "Quan 11"));  
        mData.add(R.drawable.avatar_1);  
        mData.add("Text 0");  
        mData.add("Text 1");  
        mData.add(new UserModel( name: "Nguyen Hoang Minh", address: "Quan 3"));  
        mData.add("Text 2");  
        mData.add(R.drawable.avatar_2);  
        mData.add(R.drawable.avatar_3);  
        mData.add(new UserModel( name: "Pham Nguyen Tam Phu", address: "Quan 10"));  
        mData.add("Text 3");  
        mData.add("Text 4");  
        mData.add(new UserModel( name: "Tran Van Phuc", address: "Quan 1"));  
        mData.add(R.drawable.avatar_4);  
        CustomAdapter adapter = new CustomAdapter( context: this, mData);  
        rvMultipleViewType.setAdapter(adapter);  
        rvMultipleViewType.setLayoutManager(new LinearLayoutManager( context: this));  
    }  
}
```



- ItemAnimator là thành phần hỗ trợ animation khi thêm vào hay xóa 1 item ra khỏi RecyclerView có các lớp chính sau:
 - ItemAnimator: là lớp đại diện, khung sườn của animation trong RecyclerView.
 - SimpleItemAnimator: lớp wrapper lại ItemAnimator.
 - DefaultItemAnimator: lớp xử lý animation mặc định, sử dụng trong RecyclerView.
- Viết lại lớp xử lý animation bằng cách kế thừa lại lớp SimpleItemAnimator.
- Để đặt ItemAnimator cho ReyclerView, sử dụng phương thức **setItemAnimator(ItemAnimator)**.
- Tuy nhiên, phải tạo custom adapter và gọi thông báo đúng trường hợp, ứng với các thao tác thêm và xóa item.

- Thông báo Adapter xử lý animation: Không giống như ListView khi có thay đổi về mặt dữ liệu thì phải gọi notifyDataSetChanged để hiển thị lại ListView, RecyclerView có 1 số điểm khác như sau:
 - Gọi notify tại vị trí thay đổi bằng các phương thức:

notifyItemChanged(int)	Notify item tại vị trí position nếu có thay đổi, phương thức này cũng được sử dụng nếu muốn thực hiện animation khi item changed.
notifyItemInserted(int)	Notify nếu insert 1 item tại vị trí position, phương thức này cũng được sử dụng nếu muốn thực hiện animation khi add 1 item vào RecyclerView.
notifyItemRemoved(int)	Notify khi remove 1 item tại vị trí position, phương thức này cũng được sử dụng nếu muốn thực hiện animation khi remove 1 item ra khỏi RecyclerView.
notifyItemRangeChanged(int, int)	Notify những item thay đổi trên 1 miền từ fromPosition, phương thức này cũng được sử dụng nếu muốn xử lý animation những item changed.
notifyItemRangeInserted(int, int)	Notify các item được insert vào từ fromPosition, phương thức này cũng được sử dụng nếu muốn xử lý animation những item được insert.
notifyItemRangeRemoved(int, int)	Notify các item remove ra khỏi RecyclerView từ fromPosition, phương thức này cũng được sử dụng nếu muốn xử lý animation những item được removed.

- ❑ Ngoài ra RecyclerView cũng có phương thức notifyDataSetChanged() giống như ListView, tuy nhiên phương thức này sẽ không xảy ra animation.

Ví dụ khi thêm 1 item vào Adapter:

```
public void addItem(String item) {  
    mDatas.add(item);  
    // Notify tại vị trí add item.  
    notifyItemInserted(mDatas.size() - 1);  
}
```

Hoặc:

```
public void addItem(int position, String item) {  
    mDatas.add(position, item);  
    notifyItemInserted(position);  
}
```

Xóa 1 item:

```
public void removeItem(int position) {  
    mDatas.remove(position);  
    notifyItemRemoved(position);  
}
```

Hoặc:

```
public void removeItem(String item) {  
    int index = mDatas.indexOf(item);  
    if (index < 0)  
        return;  
    mDatas.remove(index);  
    notifyItemRemoved(index);  
}
```

Đổi 1 item:

```
public void replaceItem(int position, String item) {  
    mDatas.remove(position);  
    mDatas.add(position, item);  
    notifyItemChanged(position);  
}
```

Việc sử dụng ItemAnimator:

- Đặt ItemAnimator cho RecyclerView thông qua phương thức setItemAnimator()
- Viết custom adapter sử dụng những phương thức thông báo đã giới thiệu ở trên.

Ứng dụng demo

Tổng quan về ứng dụng như sau:

- 1 button dùng để thực hiện thêm item vào adapter và có animation.
- Khi nhấp chuột vào item thì thực hiện thay đổi animation trên item đó.
- Khi nhấp giữ item thì thực hiện xóa animation trên item đó.

Ứng dụng demo

Tổng quan về ứng dụng như sau:

- 1 button dùng để thực hiện thêm item vào adapter và có animation.
- Khi nhấp chuột vào item thì thực hiện thay đổi animation trên item đó.
- Khi nhấp giữ item thì thực hiện xóa animation trên item đó.

```
<Button  
    android:id="@+id/btn_add_item"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:layout_marginTop="112dp"  
    android:text="Button"  
    app:layout_constraintEnd_toEndOf="parent"  
    app:layout_constraintHorizontal_bias="0.498"  
    app:layout_constraintStart_toStartOf="parent"  
    app:layout_constraintTop_toTopOf="parent" />
```

```
<androidx.recyclerview.widget.RecyclerView  
    android:id="@+id/rv_items"  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    android:layout_marginTop="64dp"  
    app:layout_constraintEnd_toEndOf="parent"  
    app:layout_constraintHorizontal_bias="0.0"  
    app:layout_constraintStart_toStartOf="parent"  
    app:layout_constraintTop_toBottomOf="@+id/btn_add_item" />
```

activity_animation_main.xml

row_animation.xml

```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="3dp"
        android:background="#ecf0f1">
    <TextView
        android:id="@+id/tv_item"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:padding="8dp"
        android:textColor="#2c3e50"
        android:textSize="16dp" />
</FrameLayout>
```

CustomAnimationAdapter.java

```
public class CustomAnimationAdapter extends RecyclerView.Adapter<CustomAnimationAdapter.ViewHolder> {
    private List<String> mDatas;
    public CustomAnimationAdapter(List<String> data) {
        mDatas = data;
    }
    @Override
    public ViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
        LayoutInflator li = LayoutInflator.from(parent.getContext());
        View itemView = li.inflate(R.layout.row_animation, parent, attachToRoot: false);
        return new ViewHolder(itemView);
    }
    @Override
    public void onBindViewHolder(ViewHolder holder, int position) {
        String item = mDatas.get(position);
        holder.tvItem.setText(item);
    }
    @Override
    public int getItemCount() {
        return mDatas.size();
    }
    public void addItem(String item) {
        mDatas.add(item);
        notifyDataSetChanged( position: mDatas.size() - 1 );
    }
    public void addItem(int position, String item) {
        mDatas.add(position, item);
        notifyDataSetChanged(position);
    }
}
```

CustomAnimationAdapter.java

```
public void removeItem(int position) {  
    mDatas.remove(position);  
    notifyItemRemoved(position);  
}  
  
public void removeItem(String item) {  
    int index = mDatas.indexOf(item);  
    if (index < 0)  
        return;  
    mDatas.remove(index);  
    notifyItemRemoved(index);  
}  
  
public void replaceItem(int postion, String item) {  
    mDatas.remove(postion);  
    mDatas.add(postion, item);  
    notifyItemChanged(postion);  
}
```

CustomAnimationAdapter.java

```
public class ViewHolder extends RecyclerView.ViewHolder {  
    private TextView tvItem;  
    public ViewHolder(final View itemView) {  
        super(itemView);  
        tvItem = (TextView) itemView.findViewById(R.id.tv_item);  
        itemView.setOnLongClickListener(new View.OnLongClickListener() {  
            @Override  
            public boolean onLongClick(View view) {  
                removeItem(getAdapterPosition());  
                Toast.makeText(itemView.getContext(), text: "Removed Animation", Toast.LENGTH_SHORT).show();  
                return false;  
            }  
        });  
        itemView.setOnClickListener(new View.OnClickListener() {  
            @Override  
            public void onClick(View view) {  
                replaceItem(getAdapterPosition(), item: "item changed");  
                Toast.makeText(itemView.getContext(), text: "Changed Animation", Toast.LENGTH_SHORT).show();  
            }  
        });  
    }  
}
```

MainAnimationActivity.java

```
public class MainAnimationActivity extends AppCompatActivity {
    private Button btnAddItem;
    private RecyclerView rvItems;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main_animation);
        btnAddItem = (Button) findViewById(R.id.btn_add_item);
        rvItems = (RecyclerView) findViewById(R.id.rv_items);
        List<String> data = new ArrayList<>();
        for (int i = 0; i < 5; i++) {
            data.add("item " + i);
        }
        final CustomAnimationAdapter adapter = new CustomAnimationAdapter(data);
        rvItems.setAdapter(adapter);
        rvItems.setLayoutManager(new LinearLayoutManager(context: this));
        //set ItemAnimator for RecyclerView
        rvItems.setItemAnimator(new DefaultItemAnimator());
        btnAddItem.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                adapter.addItem("new item");
                rvItems.scrollToPosition(adapter.getItemCount() - 1);
            }
        });
    }
}
```

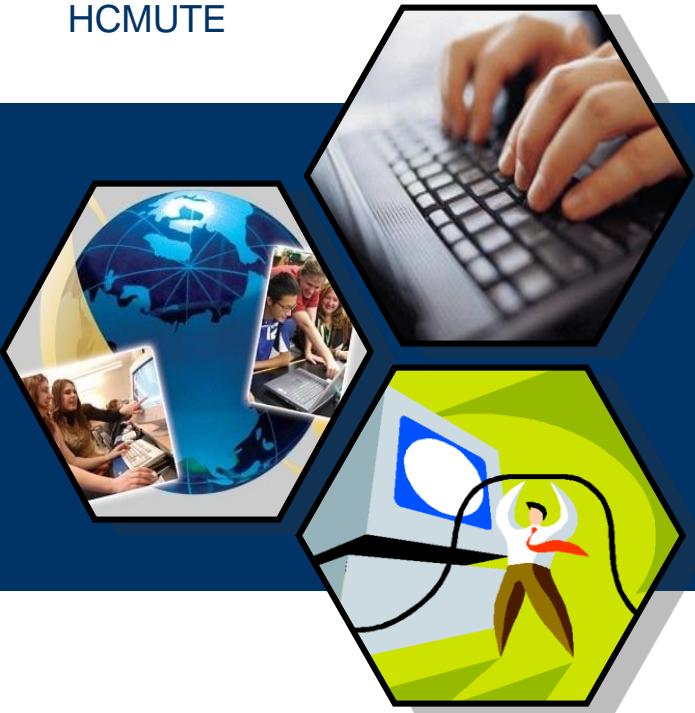
Vì sử dụng DefaultAnimator nên animation sẽ như sau:

- Animation Added: thực hiện thay đổi giá trị alpha của itemView từ 0 đến 1.
- Animation Removed: thực hiện thay đổi giá trị alpha của itemView từ 1 về 0.
- Animation Changed: thực hiện animation từ 1 về 0, sau đó đặt animation từ 0 đến 1 để thay đổi item.

Đó là cách mà DefaultItemAnimator xử lý, nếu muốn animation theo mong muốn thì phải viết lại animation bằng cách kế thừa từ SimpleItemAnimator.



KHOA CÔNG NGHỆ THÔNG TIN



DATABINDING

Khoa Công nghệ Thông tin
Đại học Sư phạm Kỹ thuật TP.HCM

ThS. Nguyễn Hữu Trung



- Data Binding là một thư viện được tích hợp trong Android Jetpack. Nó cho phép liên kết giữa dữ liệu logic với các UI Element(ví dụ như : TextView, EditText, ImageView...).
- Điều này giúp các nhà phát triển lược bớt rất nhiều đoạn code liên kết kiểu như: findViewById() không cần thiết nữa. Điều này sẽ giúp cho dự án của bạn sẽ dễ Unit Test hơn.
- Nhờ những ưu điểm của Data Binding mà Data Binding được sử dụng rất nhiều trong các kiến trúc ứng dụng MVP, MVVM...

□ Cấu hình build.gradle

- Thêm thư viện support trong mục dependencies.

- Bật Data Binding trong build.gradle.

- dataBinding {
 - enabled = true
 - }

Hoặc:

```
buildFeatures {  
    databinding true  
}
```

```
buildTypes {  
    release {  
        minifyEnabled false  
        proguardFiles getDefaultProguardFile("proguard-android.txt"),  
                    "proguard-rules.pro"  
    }  
}  
buildFeatures{  
    dataBinding true  
}  
...
```

Ví dụ: **Preview** pane sẽ hiển thị giá trị default_value của **TextView** được khai báo như bên dưới.

<TextView

```
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    android:text="@{person.country, default=default_value}"/>
```

- Đầu tiên các bạn thêm class User để lưu thông tin User gồm 2 thuộc tính: lastName và firstName.

```
public class User {  
    // firstName và lastName.  
    private String firstName;  
    private String lastName;  
  
    // tạo constructor, getter và setter  
  
}
```

- ☐ Bây giờ chúng ta sẽ làm việc với file layout với **activity_main.xml**. Để sử dụng Databinding trong layout, phần tử gốc nên bắt đầu bằng thẻ **<layout>**. Cùng với nó, các thẻ **<data>** và **<variable>** được sử dụng.

```
<layout ...>
    <data>
        <variable
            name="..."
            type="..." />
    </data>
    <LinearLayout ...>
        <!-- YOUR LAYOUT HERE -->
    </LinearLayout>
</layout>
```

- Tất cả các biến và phương thức được đặt trong thẻ data. Thuộc tính name sẽ là tên và type phải thuộc class object User.
- Và giờ chúng ta thiết kế giao diện trong file **activity_main.xml**

```
<?xml version="1.0" encoding="utf-8"?>
<layout 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">
    <data>
        <variable
            name="user"
            type="vn.iotstar.databinding.UserModel" />
    </data>
    <androidx.constraintlayout.widget.ConstraintLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        tools:context=".MainActivity">
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginTop="136dp"
            android:text="@{user.lastName}"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toTopOf="parent" />
    </androidx.constraintlayout.widget.ConstraintLayout>
</layout>
```

- Vì trong **activity_main.xml** bạn đã khai báo layout này sẽ sử dụng DataBinding để hiển thị dữ liệu thì Android Studio sẽ tự động sinh ra cho bạn class **ActivityMainBinding**.
- Tiếp theo mở **MainActivity.java** lên

```
public class MainActivity extends AppCompatActivity {  
    private UserModel userModel;  
    private ActivityMainBinding binding;  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        binding = DataBindingUtil.setContentView( activity: this,R.layout.activity_main);  
        userModel = new UserModel("Huu", "Trung");  
        binding.setUser(userModel);  
    }  
}
```

- Khi dữ liệu có sự thay đổi thì cần làm gì để chúng ta có thể update được sự thay đổi đó lên view *thì bạn* thêm annotation `@Bindable` vào hàm *Getter* và thêm `notifyPropertyChanged(BR.firstName)`; và `notifyPropertyChanged(BR.lastName)` tại hàm *setter* để thông báo với hệ thống rằng chúng ta cần update lại dữ liệu của 2 thuộc tính này. (thường thì chúng ta sẽ đặt 2 dòng này ở nơi mà dữ liệu được thay đổi).

```
binding.setUser(userModel);
//thêm để tự cập nhật
userModel.setFirstName("Vinh");
userModel.setLastName("Hoàng");
```

```
public class UserModel extends BaseObservable implements Serializable {
    private String firstName;
    private String lastName;
}

public UserModel(String firstName, String lastName) {
    this.firstName = firstName;
    this.lastName = lastName;
}

@Bindable
public String getFirstName() { return firstName; }
public void setFirstName(String firstName) {
    this.firstName = firstName;
    notifyPropertyChanged(BR.firstName);
}

@Bindable
public String getLastName() { return lastName; }

public void setLastName(String lastName) {
    this.lastName = lastName;
    notifyPropertyChanged(BR.lastName);
}
```

□ Bước 1: Tạo Model

```
public class User implements Serializable {  
    private String firstName;  
    private String lastName;  
  
    public String getFirstName() {  
        return firstName;  
    }  
  
    public void setFirstName(String firstName) {  
        this.firstName = firstName;  
    }  
  
    public String getLastName() {  
        return lastName;  
    }  
  
    public void setLastName(String lastName) {  
        this.lastName = lastName;  
    }  
}
```

□ Bước 2: Tạo layout activity_home.xml

```
<?xml version="1.0" encoding="utf-8"?>
<layout 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">
    <data>
        <variable
            name="home"
            type="vn.iotstar.databinding.HomeActivity" />
    </data>
    <androidx.constraintlayout.widget.ConstraintLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        tools:context=".HomeActivity">
        <TextView
            android:id="@+id/tv_Title"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginTop="72dp"
            android:text="@{home.title}"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintHorizontal_bias="0.498"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toTopOf="parent" />
        <androidx.recyclerview.widget.RecyclerView
            android:id="@+id/rcView"
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:layout_marginTop="132dp"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintHorizontal_bias="0.0"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toBottomOf="@+id/tv_Title" />
    </layout>
</androidx.constraintlayout.widget.ConstraintLayout>
```

□ Bước 3: Tạo layout item_list_user.xml

```
<?xml version="1.0" encoding="utf-8"?>
<layout xmlns:android="http://schemas.android.com/apk/res/android">
    <data>
        <variable
            name="viewHolder"
            type="vn.iotstar.databinding.ListUserAdapter.MyViewHolder" />
    </data>
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="50dp"
        android:orientation="horizontal"
        android:weightSum="5">
        <TextView
            android:layout_weight="1"
            android:layout_width="0dp"
            android:layout_height="match_parent"
            android:text="@{viewHolder.stt}" />
        <TextView
            android:layout_weight="2"
            android:layout_width="0dp"
            android:layout_height="match_parent"
            android:text="@{viewHolder.firstName}" />
        <TextView
            android:layout_weight="2"
            android:layout_width="0dp"
            android:layout_height="match_parent"
            android:text="@{viewHolder.lastName}" />
    </LinearLayout>
</layout>
```

□ Bước 4: Tạo Adapter: ListUserAdapter.java

```
public class ListUserAdapter extends RecyclerView.Adapter<ListUserAdapter.MyViewHolder> {
    private List<User> userList;

    public ListUserAdapter(List<User> userList) {
        this.userList = userList;
    }

    @NonNull
    @Override
    public MyViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
        ItemListUserBinding itemListUserBinding = DataBindingUtil.inflate(LayoutInflater.from(parent.getContext())
                ,R.layout.item_list_user,parent, attachToParent: false);
        return new MyViewHolder(itemListUserBinding);
    }

    @Override
    public void onBindViewHolder(@NonNull ListUserAdapter.MyViewHolder holder, int position) {
        holder.setBinding(userList.get(position),position);
    }

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

□ Bước 4: Tạo Adapter: ListUserAdapter.java

```
//tạo class MyViewHolder
public class MyViewHolder extends RecyclerView.ViewHolder{
    public ObservableField<String> stt = new ObservableField<>();
    public ObservableField<String> firstName = new ObservableField<>();
    public ObservableField<String> lastName = new ObservableField<>();
    private ItemListUserBinding itemListUserBinding;

    public MyViewHolder(ItemListUserBinding itemView) {
        super(itemView.getRoot());
        this.itemListUserBinding = itemView;
    }

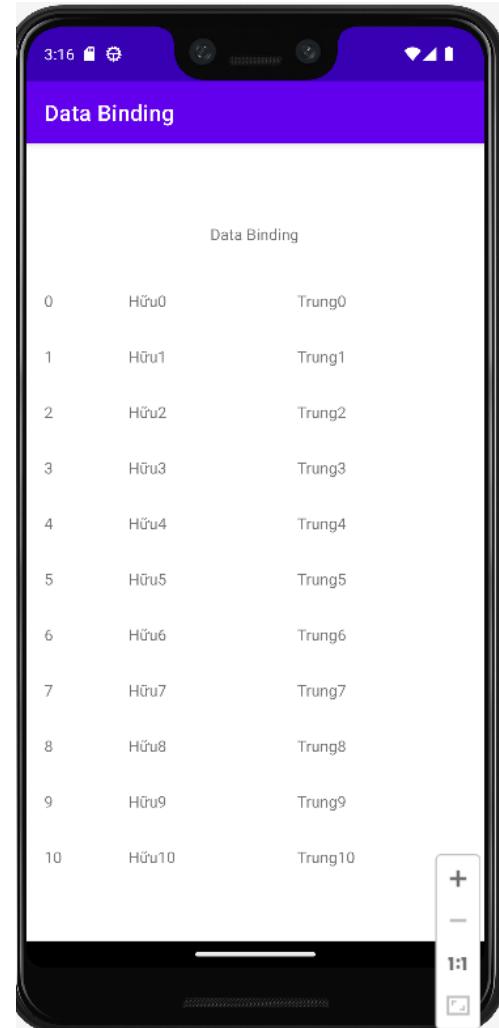
    public void setBinding(User user, int position){
        if(itemListUserBinding.getViewHolder() == null){
            itemListUserBinding.setViewHolder(this);
        }
        stt.set(String.valueOf(position));
        firstName.set(user.getFirstName());
        lastName.set(user.getLastName());
    }
}
```

□ Bước 5: Xét adapter cho RecyclerView

```
public class HomeActivity extends AppCompatActivity {
    public ObservableField<String> title = new ObservableField<>();
    private ListUserAdapter listUserAdapter;
    private ActivityHomeBinding binding;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        binding = DataBindingUtil.setContentView( activity: this,R.layout.activity_home);
        title.set("Ví dụ về DataBinding cho RecyclerView");
        binding.setHome(this);
        setData();
        binding.rcView.setLayoutManager(new LinearLayoutManager( context: this));
        binding.rcView.setAdapter(listUserAdapter);

    }
    private void setData() {
        List<User> userList = new ArrayList<>();
        for (int i =0;i<userList.size();i++){
            User user = new User();
            user.setFirstName("Hữu" + i);
            user.setLastName("Trung" + i);
            userList.add(user);
        }
        listUserAdapter = new ListUserAdapter(userList);
    }
}
```



□ Bước 6: Bắt sự kiện trên RecyclerView

RecyclerView thì mặc định không có Callback trả về sự kiện onItemClickListener. Các bạn muốn bắt sự kiện này thì cần tự viết thêm Callback. Và chúng ta sẽ chỉnh sửa trong file Adapter để có thể bắt sự kiện này ngoài HomeActivity.

```
public class ListUserAdapter extends RecyclerView.Adapter<ListUserAdapter.MyViewHolder> {
    private List<User> userList;
    private OnItemClickListener onItemClickListener;//gọi interface

    public ListUserAdapter(List<User> userList) { this.userList = userList; }

    @NotNull
    @Override
    public MyViewHolder onCreateViewHolder(@NotNull ViewGroup parent, int viewType) {
        ItemListUserBinding itemListUserBinding = DataBindingUtil.inflate(LayoutInflater.from(parent.getContext())
                ,R.layout.item_list_user,parent, attachToParent: false);
        return new MyViewHolder(itemListUserBinding, onItemClickListener);
    }

    @Override
    public void onBindViewHolder(@NotNull ListUserAdapter.MyViewHolder holder, int position) {
        holder.setBinding(userList.get(position),position);
    }

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

□ Bước 6: Bắt sự kiện trên RecyclerView

```
public class MyViewHolder extends RecyclerView.ViewHolder implements View.OnClickListener {  
    public ObservableField<String> stt = new ObservableField<>();  
    public ObservableField<String> firstName = new ObservableField<>();  
    public ObservableField<String> lastName = new ObservableField<>();  
    public ItemListUserBinding itemListUserBinding;  
    private OnItemClickListener onItemClickListener;  
    private User user;  
  
    1 related problem  
    public MyViewHolder(ItemListUserBinding itemView, OnItemClickListener onItemClickListener) {  
        super(itemView.getRoot());  
        this.itemListUserBinding = itemView;  
        this.onItemClickListener = onItemClickListener;  
        itemView.getRoot().setOnClickListener(this);  
    }  
  
    public void setBinding(User user, int position){  
        if(itemListUserBinding.getViewHolder() == null){  
            itemListUserBinding.setViewHolder(this);  
        }  
        this.user=user;  
        stt.setValue(position);  
        firstName.set(user.getFirstName());  
        lastName.set(user.getLastName());  
    }  
    @Override  
    public void onClick(View v) { this.onItemClickListener.itemClick(user); }  
}
```

□ Bước 6: Bắt sự kiện trên RecyclerView

```
public interface OnItemClickListener{  
    void itemClick(User user);  
}  
  
public void setOnItemClickListener(OnItemClickListener onItemClickListener){  
    this.onItemClickListener = onItemClickListener;  
}  
}
```

- Bước 7: Bổ sung vào HomeActivity

Thêm vào hàm onCreate

```
listUserAdapter.setOnItemClickListener(this);
```

```
//Thêm vào class HomeActivity
@Override
public void itemClick(User user) {
    Toast.makeText(this, "Bạn vừa click", Toast.LENGTH_SHORT).show();
}
```

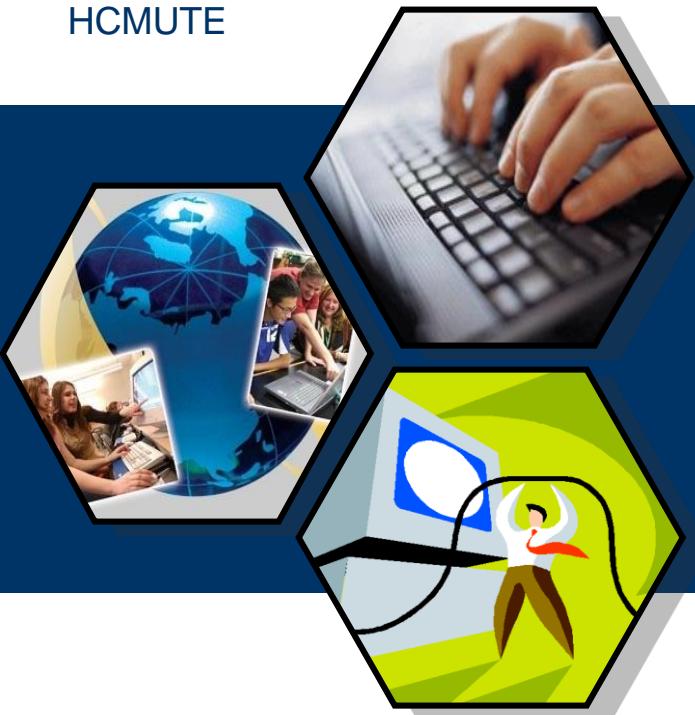


HCMUTE

KHOA CÔNG NGHỆ THÔNG TIN

UTEx

LƯU TRỮ DỮ LIỆU TRONG ANDROID



Khoa Công nghệ Thông tin
Đại học Sư phạm Kỹ thuật TP.HCM

ThS. Nguyễn Hữu Trung



Giới thiệu

243

- Tuỳ thuộc vào các đặc điểm và yêu cầu của từng ứng dụng thì việc duy trì thông tin về trạng thái, cấu hình cho mỗi lần ứng dụng chạy bằng cách **lưu trữ dữ liệu cục bộ trên thiết bị di động** là cần thiết.
- Lúc này bạn sẽ phải xem xét ứng dụng của bạn cần phải lưu trữ và khai thác dữ liệu có cấu trúc hay đơn giản và nên lưu trữ trong hoặc ngoài ứng dụng.
- **Android** cung cấp một số tùy chọn hỗ trợ cho việc lưu trữ dữ liệu trong một ứng dụng như sau:

- **Shared Preferences:** Lưu trữ dữ liệu thuộc kiểu Primitive theo các cặp KEY-VALUE. Cách này rất hữu ích khi lưu những dữ liệu đơn giản.
- **Internal/External Storage:** Cách này bạn có thể sử dụng File API để thực hiện các thao tác đọc, ghi dữ liệu. Nếu muốn bảo mật dữ liệu thì bạn nên lưu trữ dữ liệu trên Internal Storage nhưng với kích thước có giới hạn, ngược lại trong các trường hợp cần lưu dữ liệu lớn, cần chia sẻ dữ liệu cho các ứng dụng khác hoặc đơn giản là không có nhu cầu bảo mật dữ liệu thì sẽ lưu trên External Storage (hay gọi là thẻ nhớ của thiết bị).
- **SQLite Database:** Lưu trữ dữ liệu có cấu trúc với mục đích bảo mật dữ liệu. Android hỗ trợ toàn diện các tính năng của SQLite, TinyDB, Realm thông qua các API có sẵn.
- **Content Provider:** Lưu trữ dữ liệu và chia sẻ dữ liệu bảo mật từ ứng dụng của bạn cho các ứng dụng khác thông qua các Uri.

- **Shared Preferences** là đối tượng Android cung cấp cho việc lưu trữ và truy xuất các dữ liệu có kiểu cơ bản như **Boolean, string, float, long, and integer** trong lập trình các ứng dụng đơn giản. Ví dụ như ứng dụng của bạn cho phép người dùng cài đặt các tùy chọn liên quan đến màu chữ hay màu nền của các màn hình. Trong trường hợp này, bạn có thể sử dụng đối tượng **Shared Preferences** để lưu lại các thông tin tùy chọn người dùng thiết lập theo từng cặp key/value và truy xuất sử dụng cho mỗi lần mở lại ứng dụng.
- **Shared Preference** sẽ được lưu lại thông qua việc sử dụng chỉ định khóa cho từng giá trị dữ liệu (**theo từng cặp key/value**), những giá trị key.value này sẽ được tự động ghi vào một tập tin XML được chứa bên trong thư mục của ứng dụng. Những cặp giá trị này sẽ tồn tại và sẵn sàng cho việc sử dụng (đọc và cập nhật) trong suốt phiên làm việc của ứng dụng và được chia sẻ bên trong các thành phần của ứng dụng (như activity, service,...). Các giá trị shared preference này không thể truy xuất và sử dụng từ những ứng dụng khác.
- Dữ liệu của ứng dụng được lưu trữ trong thư mục data/data/ Tên package của ứng dụng. Ví dụ: data/data/com.hiepsiit.com. Do đó, trong một Context tại phiên làm việc hiện tại của ứng dụng, bạn có thể truy xuất đối tượng lớp **Shared Preferences** thông qua hàm **getSharedPreferences(String name, int mode)**.

- Có 2 cách lưu trữ dữ liệu trong Shared Preference

- **Cách 1: Sử dụng Activity Preferences:**

- Chúng ta phải gọi getPreferences (int mode) có trong lớp Activity.
 - Chỉ sử dụng khi preference file là cần thiết trong Activity
 - Nó không yêu cầu tên vì nó sẽ là tập tin ưu tiên duy nhất cho Activity.
 - Ít sử dụng tính năng này ngay cả khi họ chỉ cần một tệp tùy chọn trong Activity. Họ thích sử dụng phương thức tùy biến getSharedPreferences (tên String, chế độ int).

- **Cách 2: Sử dụng Custom Preferences:**

- Sử dụng phương thức **getSharedPreferences(String name,int mode)** cho **Custom Preferences**
 - Được sử dụng trong các trường hợp khi yêu cầu nhiều tập tin tùy **Preferences** trong **Activity**
 - Tên của tập tin **preference** được truyền cho tham số đầu tiên

- **Mode và kiểu trong Shared Preference:** Để tạo Activity dựa vào preferences hoặc custom preferences phải truyền mode để cho hệ thống biết về quyền riêng tư của tập tin preference.
- **Context là gì:** Context là một lớp trừu tượng được sử dụng để lấy thông tin toàn cục trong các tài nguyên trong ứng dụng Android strings, values, drawable, assets v.v
- **Có 3 loại Mode trong Shared Preference:**
 - **1. MODE_PRIVATE** – Đây là chế độ mặc định. MODE_PRIVATE được sử dụng để thiết lập chế độ riêng tư, không chấp nhận ứng dụng từ bên ngoài.
 - **2. MODE_WORLD_READABLE** – mode điều khiển cho phép đọc tập tin Preferences từ ứng dụng khác.
 - **3. MODE_WORLD_WRITEABLE** – mode điều khiển cho phép chỉnh sửa tập tin Preferences từ ứng dụng khác.

□ Các bước lưu dữ liệu trong Shared Preference

- **Bước 1:** Để lưu dữ liệu trong SharedPreferences chúng ta cần gọi phương thức của lớp SharedPreferences. Lớp SharedPreferences trả về một đối tượng **Editor**.
- **Bước 2:** Lớp **Editor** và sử dụng các phương thức put<type> để thêm mới (nếu chưa có) hoặc cập nhật giá trị cho các từ khóa (key) được chỉ định như đoạn code bên dưới.

```
SharedPreferences.Editor editor = mySharedPreferences.edit();
// Store new primitive types in the shared preferences object.
editor.putBoolean("isTrue", true);
editor.putFloat("floatValue", 1f);
editor.putInt("intValue", 2);
editor.putLong("longValue", 3l);
editor.putString("stringValue", "Not Empty");
```

Bước 3: Sử dụng phương thức **Commit()** để gọi đồng bộ và trả về kết quả lưu thành công hay không hoặc sử dụng phương thức **Apply()** nếu chúng ta không quan tâm đến kết quả trả về

□ Các bước đọc và lấy dữ liệu trong Shared Preference

- **Bước 1:** Để đọc dữ liệu reference từ đối tượng SharedPreferences chúng ta sử dụng phương thức **getPreferences (int mode)** hoặc **getSharedPreferences (String name,int mode)**.
- **Bước 2:** Chúng ta sẽ truy xuất lại các giá trị đã lưu bằng cách dùng các phương thức an toàn (gọi là type-safe) **SharedPreferences.get<type>** với 2 thông số: thông số đầu là key để lấy dữ liệu, và thông số thứ hai là giá trị mặc định (trong trường hợp key đó chưa được lưu trước đó).

```
// Retrieve the saved values.  
boolean isTrue = mySharedPreferences.getBoolean("isTrue", false);  
float lastFloat = mySharedPreferences.getFloat("floatValue", 0f);  
int wholeNumber = mySharedPreferences.getInt("intValue", 1);  
long aNumber = mySharedPreferences.getLong("longValue", 0);  
String stringPreference =  
mySharedPreferences.getString("stringValue", "");
```

- **Xóa dữ liệu hoặc xóa tất cả dữ liệu:** Sử dụng phương thức **remove(String key)** trong `SharedPreferences.Editor` để xóa. Để xóa tất cả dữ liệu chúng ta gọi phương thức `clear()`. Sau đó gọi **commit()** để lưu dữ liệu.

```
public static final String PREFS_GAME = "vn.iotstar.Games";
public static final String GAME_SCORE = "GameScore";
//===== Code to save data =====
SharedPreferences sp = getSharedPreferences(PREFS_GAME ,
Context.MODE_PRIVATE);
sp.edit().putInt(GAME_SCORE, 100).commit();
//===== Code to get saved/ retrieve data =====
SharedPreferences sp = getSharedPreferences(PREFS_GAME
, Context.MODE_PRIVATE);
int sc = sp.getInt(GAME_SCORE, 0);
Log.d("iotstar", "Số điểm hiện tại" + sc);
```

Use Name

password

Remember me

LOGIN

```
<CheckBox
    android:id="@+id/cbmemberme"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Remember me" />

<Button
    android:id="@+id/buttonTxt"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Login" />
```

```
<EditText
    android:id="@+id/usernameTxt"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:ems="10"
    android:hint="Use Name"
    android:inputType="textPersonName" />
```

```
<EditText
    android:id="@+id/passwordTxt"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:ems="10"
    android:hint="password"
    android:inputType="textPersonName" />
```

Use Name

password

Remember me

LOGIN

```
private void AnhXa(){
    buttonTxt = (Button) findViewById(R.id.buttonTxt);
    usernameTxt = (EditText) findViewById(R.id.usernameTxt);
    passwordTxt = (EditText) findViewById(R.id.passwordTxt);
    cbRememberMe = (CheckBox) findViewById(R.id.cbmemberme);
}
```

```
public class HomeActivity extends AppCompatActivity {
    //Khai báo biến toàn cục
    Button buttonTxt;
    EditText usernameTxt, passwordTxt;
    CheckBox cbRememberMe;
    SharedPreferences sharedPreferences;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_home);
        AnhXa();
```

Use Name

password

Remember me

LOGIN

```
//khởi tạo sharePreference
sharedPreferences = getSharedPreferences( name: "dataLogin", MODE_PRIVATE);
//lấy giá trị sharedpreferences
usernameTxt.setText(sharedPreferences.getString( key: "taikhoan", defValue: ""));
passwordTxt.setText(sharedPreferences.getString( key: "matkhau", defValue: ""));
cbRememberMe.setChecked(sharedPreferences.getBoolean( key: "trangthai", defValue: false));
```

```
buttonTxt.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        String username = usernameTxt.getText().toString().trim();
        String password = passwordTxt.getText().toString().trim();
        if(username.equals("admin")&& password.equals("admin")){
            Toast.makeText( context: HomeActivity.this, text: "Đăng nhập thành công", Toast.LENGTH_SHORT).show();
            //nếu có check
            if(cbRememberMe.isChecked()){
                //Chỉnh sửa nội dung file lưu trữ
                SharedPreferences.Editor editor = sharedPreferences.edit();
                editor.putString("taikhoan",username);
                editor.putString("matkhau", password);
                editor.putBoolean("trangthai",true);
                editor.commit(); //xác nhận việc lưu
            }else{
                SharedPreferences.Editor editor = sharedPreferences.edit();
                editor.remove("taikhoan");
                editor.remove("matkhau");
                editor.remove("trangthai");
                editor.commit();
            }
        }else{
            Toast.makeText( context: HomeActivity.this, text: "Đăng nhập thất bại", Toast.LENGTH_SHORT).show();
        }
    }
});
```

- **Ví dụ 2:** Trong ví dụ này chúng ta sẽ làm ứng dụng để lưu trữ Email và Password trên thiết bị khi chọn Remember Me. Có kiểm tra cảnh báo đăng nhập.

Share Preferences

trungnh@hcmute.edu.vn

.....

Remember Me

ĐĂNG NHẬP

vn.iotstar.sharepreferences	drwx-----	2023-02-18 22:11	4 KB
cache	drwxrws--x	2023-02-18 22:11	4 KB
code_cache	drwxrws--x	2023-02-18 22:11	4 KB
shared_prefs	drwxrwx--x	2023-02-18 22:11	4 KB
LoginDetails.xml	-rw-rw----	2023-02-18 23:03	170 B

```
<?xml version='1.0' encoding='utf-8' standalone='yes' ?>
<map>
    <string name="Email">trungnh@hcmute.edu.vn</string>
    <string name="Password">123456</string>
</map>
```

Nơi lưu trữ: **data/data/<tên
project>/shared_prefs**

Ví dụ 2

255

□ **Bước 1:** Thiết kế giao diện activity_main.xml.

Share Preferences

trungnh@hcmute.edu.vn

.....

Remember Me

ĐĂNG NHẬP

```
<EditText  
    android:id="@+id/email"  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    android:hint="Nhập email"  
    android:inputType="textEmailAddress"  
    android:maxLines="1"  
    android:singleLine="true" />  
  
<EditText  
    android:id="@+id/password"  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    android:hint="Nhập mật khẩu"  
    android:imeActionId="@+id/login"  
    android:imeOptions="actionUnspecified"  
    android:inputType="textPassword"  
    android:maxLines="1"  
    android:singleLine="true"  
    tools:ignore="InvalidImeActionId" />
```

□ **Bước 1:** Thiết kế giao diện activity_main.xml.

Share Preferences

trungnh@hcmute.edu.vn

.....

Remember Me

ĐĂNG NHẬP

```
<CheckBox  
    android:id="@+id/checkBoxRememberMe"  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    android:text="Remember Me" />
```

```
<Button  
    android:id="@+id/email_sign_in_button"  
    style="?android:textAppearanceSmall"  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    android:layout_marginTop="16dp"  
    android:text="Đăng nhập"  
    android:textColor="#ffffffff"  
    android:textStyle="bold" />
```

□ Bước 2: Tạo Class PreManager.java để lưu dữ liệu.

```
1 package vn.iotstar.sharepreferences;
2 import android.content.Context;
3 import android.content.SharedPreferences;
4 public class PrefManager {
5     Context context;
6     PrefManager(Context context) {
7         this.context = context;
8     }
9     public void saveLoginDetails(String email, String password) {
10        SharedPreferences sharedpreferences = context.getSharedPreferences( name: "LoginDetails", Context.MODE_PRIVATE);
11        SharedPreferences.Editor editor = sharedpreferences.edit();
12        editor.putString("Email", email);
13        editor.putString("Password", password);
14        editor.commit();
15    }
16    public String getEmail() {
17        SharedPreferences sharedpreferences = context.getSharedPreferences( name: "LoginDetails", Context.MODE_PRIVATE);
18        return sharedpreferences.getString( key: "Email", defaultValue: "" );
19    }
20    public boolean isUserLogedOut() {
21        SharedPreferences sharedpreferences = context.getSharedPreferences( name: "LoginDetails", Context.MODE_PRIVATE);
22        boolean isEmpty = sharedpreferences.getString( key: "Email", defaultValue: "").isEmpty();
23        boolean isPassEmpty = sharedpreferences.getString( key: "Password", defaultValue: "").isEmpty();
24        return isEmpty || isPassEmpty;
25    }
26 }
```

□ Bước 3: Viết code MainActivity.java.

```
16 public class MainActivity extends AppCompatActivity {
17     // Khai báo biến toàn cục.
18     private EditText mEmailView;
19     private EditText mPasswordView;
20     private CheckBox checkBoxRememberMe;
21
22     @Override
23     protected void onCreate(Bundle savedInstanceState) {
24         super.onCreate(savedInstanceState);
25         setContentView(R.layout.activity_main);
26         mEmailView = (EditText) findViewById(R.id.email);
27         mPasswordView = (EditText) findViewById(R.id.password);
28         mPasswordView.setOnEditorActionListener(new TextView.OnEditorActionListener() {
29             @Override
30             public boolean onEditorAction(TextView textView, int id, KeyEvent keyEvent) {
31                 if (id == R.id.login || id == EditorInfo.IME_NULL) {
32                     attemptLogin();
33                     return true;
34                 }
35                 return false;
36             }
37         });
38     }
39 }
```

Bước 3: Viết code MainActivity.java.

```
39         Button mEmailSignInButton = (Button) findViewById(R.id.email_sign_in_button);
40         mEmailSignInButton.setOnClickListener(new View.OnClickListener() {
41             @Override
42             ↑             public void onClick(View view) {
43                 attemptLogin();
44             }
45         });
46
47         checkBoxRememberMe = (CheckBox) findViewById(R.id.checkBoxRememberMe);
48         //Here we will validate saved preferences
49         if (!new PrefManager( context: this).isUserLogedOut()) {
50             //user's email and password both are saved in preferences
51             startHomeActivity();
52         }
53     }
54 }
```

□ Bước 3: Viết code MainActivity.java.

```
60     private void attemptLogin() {
61         // Reset errors.
62         mEmailView.setError(null);           mPasswordView.setError(null);
63         // Store values at the time of the login attempt.
64         String email = mEmailView.getText().toString(); String password = mPasswordView.getText().toString();
65         boolean cancel = false;
66         View focusView = null;
67         // Check for a valid password, if the user entered one.
68         if (!TextUtils.isEmpty(password) && !isPasswordValid(password)) {
69             mPasswordView.setError(getString(R.string.error_invalid_password));
70             focusView = mPasswordView;
71             cancel = true;
72         }
73         // Check for a valid email address.
74         if (TextUtils.isEmpty(email)) {
75             mEmailView.setError(getString(R.string.error_field_required));
76             focusView = mEmailView;
77             cancel = true;
78         } else if (!isValidEmail(email)) {
79             mEmailView.setError(getString(R.string.error_invalid_email));
80             focusView = mEmailView;
81             cancel = true;
82         }
83         if (cancel) {
84             focusView.requestFocus();
85         } else {           // save data in local shared preferences
86             if (checkBoxRememberMe.isChecked())
87                 saveLoginDetails(email, password);
88             startHomeActivity();
89         }
}
```

□ **Bước 3:** Viết code MainActivity.java.

```
91     private void startHomeActivity() {
92         Intent intent = new Intent( packageContext: this, HomeActivity.class);
93         startActivity(intent);
94         finish();
95     }
96
97     private void saveLoginDetails(String email, String password) {
98         new PrefManager( context: this).saveLoginDetails(email, password);
99     }
100
101
102    @
103    private boolean isEmailValid(String email) {
104        //TODO: Replace this with your own logic
105        return email.contains("@");
106    }
107
108    @
109    private boolean isPasswordValid(String password) {
110        //TODO: Replace this with your own logic
111        return password.length() > 4;
112    }
113 }
```

□ **Bước 4:** Thêm dữ liệu vào string.xml.

```
<resources>
    <string name="app_name">Share Preferences</string>
    <string name="error_field_required">Email không được rỗng</string>
    <string name="error_invalid_email">Email không đúng định dạng</string>
    <string name="error_invalid_password">Mật khẩu phải lớn hơn 4 ký tự</string>
</resources>
```

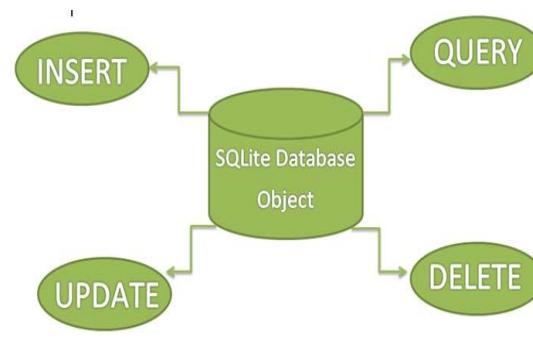
□ **Bước 5:** Tạo Empty Activity: HomeActivity.java và activity_home.xml

SQLite là một cơ sở dữ liệu SQL mã nguồn mở, nó lưu trữ dữ liệu vào một tập tin văn bản trên một thiết bị. Nó mặc định đã được tích hợp trên thiết bị Android. Để truy cập dữ liệu này, bạn không cần phải thiết lập bất kỳ loại kết nối nào cho nó như JDBC, ODBC, ... SQLite được Richard Hipp viết dưới dạng thư viện bằng ngôn ngữ lập trình C.



SQLite có các ưu điểm sau:

- Tin cậy: các hoạt động transaction (chuyển giao) nội trong cơ sở dữ liệu được thực hiện trọn vẹn, không gây lỗi khi xảy ra sự cố phần cứng.
- Tuân theo chuẩn SQL92 (chỉ có một vài đặc điểm không hỗ trợ)
- Không cần cài đặt cấu hình
- Kích thước chương trình gọn nhẹ, với cấu hình đầy đủ chỉ không đầy 300 kB
- Thực hiện các thao tác đơn giản nhanh hơn các hệ thống cơ sở dữ liệu khách/chủ khác
- Không cần phần mềm phụ trợ
- Phần mềm tự do với mã nguồn mở, được chú thích rõ ràng

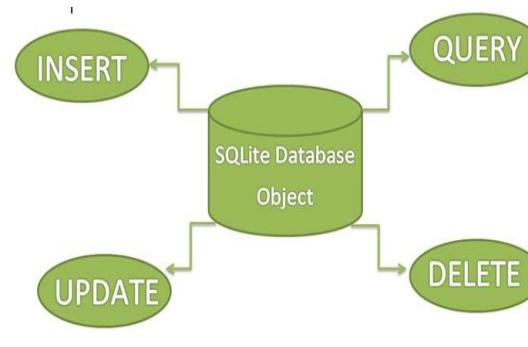


Cơ sở dữ liệu được tạo và lưu trong thư mục **data/data/APP_Name/databases/DATABASE_NAME**.



Cách sử dụng SQLite trong lập trình Android

Tương tự như các hệ quản trị cơ sở dữ liệu khác thì khi thao tác với Database thì bạn sẽ có những hành động cổ điển đó là **CRUD: Create, Read, Update, Delete**



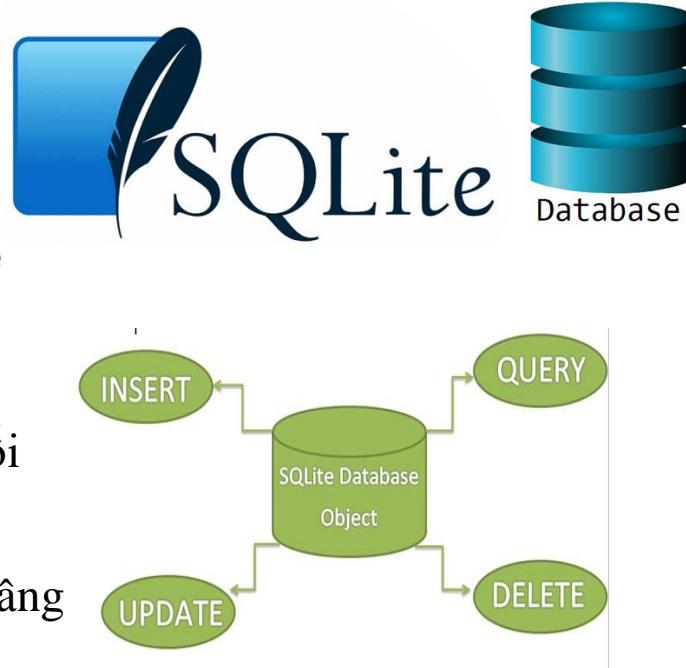
Cơ sở dữ liệu được tạo và lưu trong thư mục **data/data/APP_Name/databases/DATABASE_NAME**.

Cách sử dụng SQLite trong lập trình Android

Tương tự như các hệ quản trị cơ sở dữ liệu khác thì khi thao tác với Database thì bạn sẽ có những hành động cổ điển đó là **CRUD: Create, Read, Update, Delete**

Đầu tiên, để thao tác với SQLite, ta phải dùng 2 đối tượng

- **SQLiteOpenHelper**: đối tượng dùng để tạo, nâng cấp, đóng mở kết nối CSDL
- **SQLiteDatabase**: đối tượng dùng để thực thi các câu lệnh SQL trên một CSDL



1. SQLiteOpenHelper

SQLiteOpenHelper(Context context, String name, SQLiteDatabase.CursorFactory factory, int version)

- Tham số 1: **Context context**: Context là một lớp trừu tượng của hệ thống, chứa thông tin môi trường ứng dụng, cung cấp các phương thức để có thể tương tác với hệ điều hành, giúp chúng ta dễ dàng truy cập và tương tác tới các tài nguyên của hệ thống...
- Tham số 2: **String name**: Tên database
- Tham số 3: **CursorFactory factory**: thường để null
- Tham số 4: **Int version**: version của database

Khi khởi tạo một đối tượng của lớp này, ta phải ghi đè 2 phương thức

1. onCreate(): phương thức này được gọi bởi framework, nếu có yêu cầu truy cập database mà lại chưa khởi tạo database, ở đây ta phải viết code khởi tạo database, cụ thể là khởi tạo bảng (chú ý: khi khởi tạo bảng, ta phải đặt tên khóa chính là **_id**)

2. onUpgrade(): phương thức này được dùng khi ứng dụng của bạn có nhiều phiên bản database được thêm vào. Nó sẽ cập nhật database hiện có hoặc khởi tạo lại thông qua onCreate().

Lớp này có 2 phương thức **getReadableDatabase()** (chỉ đọc) và **getWritableDatabase()** (cho phép ghi đọc). Thông qua 2 phương thức này, ta có thể tạo ra một đối tượng **SQLiteDatabase**.

Ví dụ: Chúng ta xây dựng một lớp **DatabaseHandler** kế thừa từ lớp **SQLiteOpenHelper**

```
public class DatabaseHandler extends SQLiteOpenHelper {  
    private static final String DATABASE_NAME = "schoolManager";  
    private static final int DATABASE_VERSION = 1;  
    private static final String TABLE_NAME = "students";  
  
    private static final String KEY_ID = "id";  
    private static final String KEY_NAME = "name";  
    private static final String KEY_ADDRESS = "address";  
    private static final String KEY_PHONE_NUMBER = "phone_number";  
  
    public DatabaseHandler(Context context) {  
        super(context, DATABASE_NAME, null, DATABASE_VERSION);  
    }  
  
    @Override  
    public void onCreate(SQLiteDatabase db) {  
        String create_students_table = String.format("CREATE TABLE %s(%s INTEGER PRIMARY KEY, %s TEXT, %s TEXT, %s TEXT)"  
            , TABLE_NAME, KEY_ID, KEY_NAME, KEY_ADDRESS, KEY_PHONE_NUMBER);  
        db.execSQL(create_students_table);  
    }  
    @Override  
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {  
        String drop_students_table = String.format("DROP TABLE IF EXISTS %s", TABLE_NAME);  
        db.execSQL(drop_students_table);  
        onCreate(db);  
    }  
}
```

2. SQLiteDatabase

Để tạo ra một cơ sở dữ liệu, bạn chỉ cần gọi phương thức *openOrCreateDatabase* này với tên cơ sở dữ liệu của bạn và chế độ như một tham số

```
SQLiteDatabase mydatabase = openOrCreateDatabase("your database name", MODE_PRIVATE, null);
```

Một số hàm có sẵn trong gói cơ sở dữ liệu:

- **openDatabase(String path, SQLiteDatabase.CursorFactory factory, int flags, DatabaseErrorHandler errorHandler):** Phương thức này chỉ mở thêm cơ sở dữ liệu hiện có với các chế độ cài đặt hợp. Các chế độ cài đặt có thể là **OPEN_READWRITE**, **OPEN_READONLY**.
- **openDatabase(String path, SQLiteDatabase.CursorFactory factory, int flags):** Phương thức này tương tự như phương thức trên vì nó cũng mở ra các cơ sở dữ liệu hiện có, nhưng nó không định nghĩa bất kỳ xử lý để xử lý các lỗi cơ sở dữ liệu.
- **openOrCreateDatabase(String path, SQLiteDatabase.CursorFactory factory):** Phương thức này không chỉ mở ra nhưng lại tạo ra cơ sở dữ liệu nếu nó không tồn tại. Phương thức này là tương đương với phương thức OpenDatabase
- **openOrCreateDatabase(File file, SQLiteDatabase.CursorFactory factory):** Phương thức này cũng tương tự như phương thức trên nhưng phải mất File đối tượng như là một con đường mà không phải là một chuỗi. Nó tương đương với file.getPath()

2. SQLiteDatabase

2.1. insert(): Sử dụng phương thức này để insert một bản ghi vào CSDL

Ví dụ. ta có một đối tượng **database** thuộc kiểu **SQLiteDatabase**

Dùng đối tượng **ContentValues** để đưa dữ liệu vào bảng. Đối tượng này có các phương thức **put** (tên cột , dữ liệu) . Sau đó gọi phương thức **insert** để đưa đối tượng vào bảng.

```
ContentValues ct = new ContentValues();  
  
ct.put("full_name", "Nguyễn Hữu Trung");  
  
ct.put("student_id", "20110271");  
  
ct.put("gender", 1);  
  
ct.put("year", 20);
```

Sau đó sử dụng phương thức insert

```
database.insert(DATABASE_NAME, null, ct);
```

2. SQLiteDatabase

2.2. public int update(String table, ContentValues values, String whereClause, String[] whereArgs): Phương thức `update` của `SQLiteDatabase` để cập nhật dữ liệu trong bảng theo một điều kiện bất kỳ nào đó. Phương thức này trả về số dòng bị ảnh hưởng. Ví dụ nếu có 3 dòng bị thay đổi thì nó trả về 3. Nếu không có dòng nào bị ảnh hưởng thì nó trả về 0.

- Tham số 1: tên bảng
- Tham số 2: đối tượng muốn chỉnh sửa (với giá trị mới)
- Tham số 3: tập các điều kiện lọc (dùng dấu ? để tạo điều kiện lọc)
- Tham số 4: tập các giá trị của điều kiện lọc (lấy theo đúng thứ tự)

```
public void updateStudent(Student student) {  
    SQLiteDatabase db = this.getWritableDatabase();  
    ContentValues values = new ContentValues();  
    values.put(KEY_NAME, student.getName());  
    values.put(KEY_ADDRESS, student.getAddress());  
    values.put(KEY_PHONE_NUMBER, student.getPhone_number());  
    db.update(TABLE_NAME, values, KEY_ID + " = ?", new String[] {  
        String.valueOf(student.getId())});  
    db.close();  
}
```

2. SQLiteDatabase

2.3. public int delete(String table, String whereClause, String[] whereArgs): Chúng ta sử dụng phương thức **delete** của **SQLiteDatabase** để xóa dữ liệu của một hoặc một số record trong bảng theo một điều kiện bất kỳ nào đó. Phương thức này trả về số dòng bị ảnh hưởng. Muốn xóa toàn bộ dữ liệu trong bảng thì ta truyền null vào 2 đối số cuối

- Tham số 1: tên bảng
- Tham số 2: tập điều kiện lọc
- Tham số 3: tập các giá trị của điều kiện lọc

```
public void deleteStudent(int studentId) {  
  
    SQLiteDatabase db = this.getWritableDatabase();  
  
    db.delete(TABLE_NAME, KEY_ID + " = ?", new String[] {  
        String.valueOf(studentId) });  
  
    db.close();  
}
```

2. SQLiteDatabase

2.4. execSQL(String sql): Phương thức này dùng để thực thi một câu lệnh SQL trực tiếp. Phương pháp **exeSQL** không chỉ thêm dữ liệu, nhưng cũng được sử dụng để cập nhật hoặc sửa đổi dữ liệu đã tồn tại trong cơ sở dữ liệu bằng cách sử dụng tham số ràng buộc

```
public void doCreateSinhvienTable()
{
    String sql="CREATE TABLE tblsinhvien ("+
        "masv TEXT PRIMARY KEY ,"+
        "tensv TEXT,"+
        "malop TEXT NOT NULL CONSTRAINT malop "+
        " REFERENCES tblllop(malop) ON DELETE CASCADE)";
    database.execSQL(sql);
}
```

2. SQLiteDatabase

2.5. public Cursor query(String table, String[] columns, String selection, String[] selectionArgs, String groupBy, String having, String orderBy): Phương thức này sử dụng để truy vấn dữ liệu trong bảng. Là thao tác phức tạp nhất trong truy suất SQLite

```
Cursor cursor = null;

cursor = sqlDB.query(TABLE_NAME, null, "student_id = " + studentID,
null, null, null, null);

cursor.moveToFirst();

Student sv = new Student(cursor.getInt(0), cursor.getString(1),
cursor.getString(2), cursor.getInt(3), 22);
```

2. SQLiteDatabase

2.5. public Cursor query(String table, String[] columns, String selection, String[] selectionArgs, String groupBy, String having, String orderBy): Phương thứ này sử dụng để truy vấn dữ liệu trong bảng. Là thao tác phức tạp nhất trong truy suất SQLite

Bảng mô tả tham số của phương thức query

Tham số	Giải thích
table	Tên bảng cần truy vấn
columns	Danh sách các cột cần truy vấn, nếu tham số null là lấy tất cả các cột
selection	Lọc điều kiện, giống như mệnh đề WHERE của SQL. Nếu truyền giá trị là null sẽ lấy tất cả dữ liệu. các truyền tên cột = ? ví dụ: Id=?.
selectionArgs	Giá trị cần lọc ở tham số selection.
groupBy	Nhóm các dòng giống nhau, giống mệnh đề GROUP BY của SQL . Truyền giá trị null sẽ không nhóm.
having	Cho phép lọc nhóm kết quả nào sẽ xuất hiện trong kết quả cuối cùng của bảng ghi
orderBy	Sắp xếp các dòng dữ liệu, giống mệnh đề ORDER BY trong SQL. Nếu truyền giá trị null sẽ sắp xếp mặc định, có thể là không sắp xếp.

3. Đối tượng Cursor

Đối tượng cursor hiểu đơn giản là một con trỏ, trỏ đến kết quả trả về của câu truy vấn. con trỏ này trỏ đến cái bảng trả về của câu truy vấn

```
Cursor cursor = database.query(TABLE_NAME, null, null, null,  
null, null, null);
```

3.1. cursor.getCount(): Phương thức trả về số dòng của bảng kết quả.

3.2. cursor.moveToFirst(): Phương thức này di chuyển con trỏ này lên đầu bảng .

3.3. cursor.moveToNext(): Phương thức này để di chuyển sang dòng tiếp theo.

3.4. cursor.isAfterLast(): Kiểm tra xem cursor ở cuối bảng?, không trỏ vào dòng nào, phương thức này sẽ trả về giá trị true.

3.5.cursor.getString(), cursor.getInt(): Hai phương này để lấy ra thông tin cột theo tên cột hoặc index (chỉ mục).

4. Đối tượng ContentValues

Các đối tượng ContentValues cho phép xác định khóa / giá trị. Các key đại diện nhận dạng cột bảng và value đại diện cho nội dung cho các bảng ghi trong cột này. ContentValues Có thể được sử dụng để chèn và cập nhật các mục cơ sở dữ liệu

```
ContentValues ct = new ContentValues();  
  
ct.put("full_name", "Nguyễn Hữu Trung");  
  
ct.put("student_id", "20110271");  
  
ct.put("gender", 1);  
  
ct.put("year", 20);
```

Sau đó sử dụng phương thức insert

```
database.insert(DATABASE_NAME, null, ct);
```

□ Bước 1: Tạo class DatabaseHandler

```
public class DatabaseHandler extends SQLiteOpenHelper {  
  
    public DatabaseHandler(@Nullable Context context, @Nullable String name  
        , @Nullable SQLiteDatabase.CursorFactory factory, int version) {  
        super(context, name, factory, version);  
    }  
    //truy vấn không trả kết quả Create, Insert, update, delete,..  
    public void QueryData(String sql){  
        SQLiteDatabase database = getWritableDatabase();  
        database.execSQL(sql);  
    }  
  
    //truy vấn có trả kết quả Select  
    public Cursor GetData(String sql){  
        SQLiteDatabase database = getReadableDatabase();  
        return database.rawQuery(sql, selectionArgs: null);  
    }  
  
    @Override  
    public void onCreate(SQLiteDatabase sqLiteDatabase) {  
    }  
  
    @Override  
    public void onUpgrade(SQLiteDatabase sqLiteDatabase, int i, int i1) {  
    }  
}
```

□ Bước 2: Tạo database, tạo bảng và nhập dữ liệu

```
public class MainActivity extends AppCompatActivity {
    //khai báo biến toàn cục
    DatabaseHandler databaseHandler;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        //gọi hàm databaseSQLite
        InitDatabaseSQLite();
        //createDatabaseSQLite();
        databaseSQLite();
    }
    private void createDatabaseSQLite() {
        //thêm dữ liệu vào bảng
        databaseHandler.QueryData( sql: "INSERT INTO Notes VALUES(null, ' Ví dụ SQLite 1')");
        databaseHandler.QueryData( sql: "INSERT INTO Notes VALUES(null, ' Ví dụ SQLite 2')");
    }
    private void InitDatabaseSQLite() {
        //khởi tạo database
        databaseHandler = new DatabaseHandler( context: this, name: "notes.sqlite", factory: null, version: 1);
        //tạo bảng Notes
        databaseHandler.QueryData( sql: "CREATE TABLE IF NOT EXISTS Notes(Id INTEGER PRIMARY KEY AUTOINCREMENT, NameNotes VARCHAR(200))");
    }
    private void databaseSQLite(){
        //Lấy dữ liệu
        Cursor cursor = databaseHandler.GetData( sql: "SELECT * FROM Notes");
        while (cursor.moveToNext()){
            String name = cursor.getString( columnIndex: 1);
            Toast.makeText( context: this, name, Toast.LENGTH_SHORT).show();
        }
    }
}
```

□ Bước 3: Tạo Model

```
public class NotesModel implements Serializable {  
    private int IdNote;  
    private String NameNote;  
  
    public NotesModel(int idNote, String nameNote) {  
        IdNote = idNote;  
        NameNote = nameNote;  
    }  
  
    public int getIdNote() {  
        return IdNote;  
    }  
  
    public void setIdNote(int idNote) {  
        IdNote = idNote;  
    }  
  
    public String getNameNote() {  
        return NameNote;  
    }  
  
    public void setNameNote(String nameNote) {  
        NameNote = nameNote;  
    }  
}
```

□ Bước 4: Tạo Giao diện

```
<ScrollView  
    android:layout_width="match_parent"  
    android:layout_height="match_parent"  
    app:layout_constraintBottom_toBottomOf="parent"  
    app:layout_constraintEnd_toEndOf="parent"  
    app:layout_constraintStart_toStartOf="parent"  
    app:layout_constraintTop_toTopOf="parent">  
    <LinearLayout  
        android:layout_width="match_parent"  
        android:layout_height="0dp"  
        android:layout_margin="10dp"  
        android:orientation="vertical">  
        <TextView  
            android:id="@+id/textView"  
            android:layout_width="match_parent"  
            android:layout_height="wrap_content"  
            android:layout_margin="20dp"  
            android:gravity="center_horizontal"  
            android:text="DANH SÁCH CÔNG VIỆC"  
            android:textSize="30sp"  
            android:textStyle="bold" />  
        <ListView  
            android:id="@+id/listView1"  
            android:layout_width="match_parent"  
            android:layout_height="545dp"  
            android:layout_margin="10dp" />  
    </LinearLayout>  
</ScrollView>
```

□ Bước 4: Tạo Adapter

```
public class NotesAdapter extends BaseAdapter {  
    //Khai báo biến toàn cục  
    private Context context;  
    private int layout;  
    private List<NotesModel> noteList;  
    //tạo constructor  
    public NotesAdapter(Context context, int layout, List<NotesModel> noteList) {  
        this.context = context;  
        this.layout = layout;  
        this.noteList = noteList;  
    }  
    @Override  
    public int getCount() {  
        return noteList.size();  
    }
```

□ Bước 4: Tạo Adapter

```
//tạo viewHolder
private class ViewHolder{
    TextView textViewNote ;
    ImageView imageViewEdit ;
    ImageView imageViewDelete;
}

@Override
public View getView(int position, View convertView, ViewGroup parent) {
    //gọi viewHolder
    ViewHolder viewHolder;
    if(convertView == null){
        viewHolder = new ViewHolder();
        LayoutInflator inflater = (LayoutInflator) context.getSystemService(Context.LAYOUT_INFLATER_SERVICE);
        convertView = inflater.inflate(layout, root: null);
        viewHolder.textViewNote = (TextView) convertView.findViewById(R.id.textViewNameNote);
        viewHolder.imageViewDelete=(ImageView) convertView.findViewById(R.id.imageViewDelete);
        viewHolder.imageViewEdit = (ImageView) convertView.findViewById(R.id.imageViewEdit);
        convertView.setTag(viewHolder);
    }else{
        viewHolder =(ViewHolder) convertView.getTag();
    }
    //lấy giá trị
    NotesModel notes = noteList.get(position);
    viewHolder.textViewNote.setText(notes.getNameNote());

    return convertView;
}
```

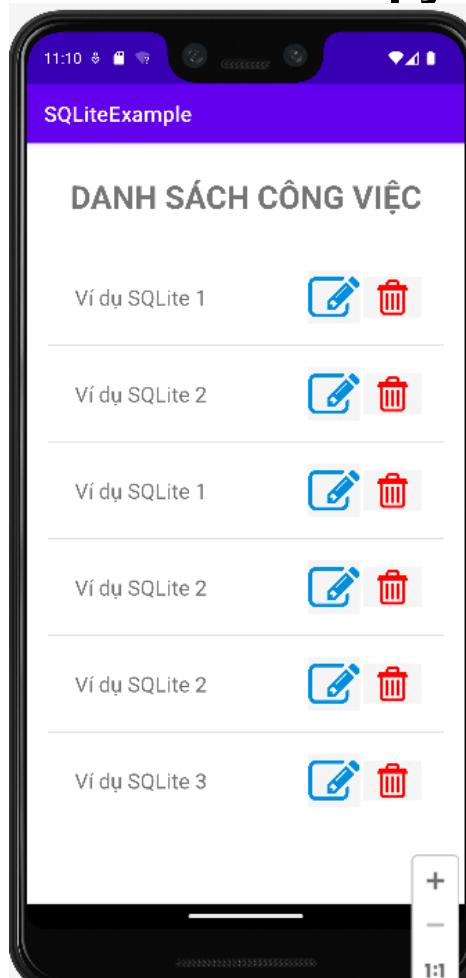
□ Bước 6: Chỉnh sửa và load dữ liệu ra giao diện trong file MainActivity.java

```
//khai báo biến toàn cục  
DatabaseHandler databaseHandler;  
ListView listView;  
ArrayList<NotesModel> arrayList;  
NotesAdapter adapter;
```

```
private void databaseSQLite(){  
    //Lấy dữ liệu  
    Cursor cursor = databaseHandler.GetData( sql: "SELECT * FROM Notes");  
    while (cursor.moveToNext()){  
        //them dữ liệu vào arraylist  
        String name = cursor.getString( columnIndex: 1);  
        int id = cursor.getInt( columnIndex: 0);  
        arrayList.add(new NotesModel(id,name));  
        //Toast.makeText(this, name, Toast.LENGTH_SHORT).show();  
    }  
    adapter.notifyDataSetChanged();  
}
```

```
//ánh xạ listview và gọi adapter  
listView = (ListView) findViewById(R.id.listView1);  
arrayList = new ArrayList<>();  
adapter = new NotesAdapter( context: this,R.layout.row_notes,arrayList);  
listView.setAdapter(adapter);
```

□ Bước 7: Chạy ví dụ ra kết quả



□ Bước 8: Mở rộng chức năng: thêm Notes

- Xây dựng menu để thêm Notes, tạo file menu.xml trong thư mục res\menu

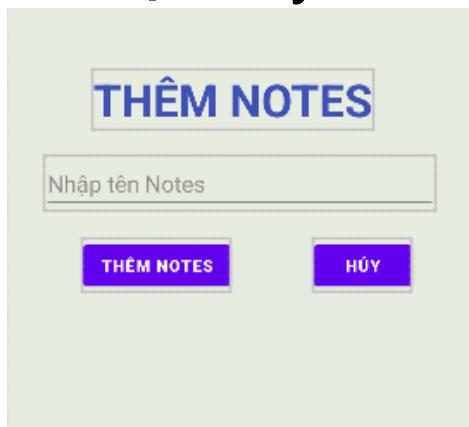


```
<?xml version="1.0" encoding="utf-8"?>
<menu
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto">
    <item android:title="Thêm Notes"
        android:icon="@android:drawable/ic_menu_add"
        app:showAsAction="always"
        android:id="@+id/menuAddNotes"/>
</menu>
```

- Bước 8: Mở rộng chức năng: thêm Notes
- Vào MainActivity.java gọi menu:

```
//gọi menu  
@Override  
public boolean onCreateOptionsMenu(Menu menu) {  
    getMenuInflater().inflate(R.menu.menu,menu);  
    return super.onCreateOptionsMenu(menu);  
}
```

- Tạo layout cho Dialog: tạo file layout resources file



```
//ảnh xạ trong dialog  
EditText editText = dialog.findViewById(R.id.editTextName);  
Button buttonAdd = dialog.findViewById(R.id.buttonEdit);  
Button buttonHuy = dialog.findViewById(R.id.buttonHuy);
```

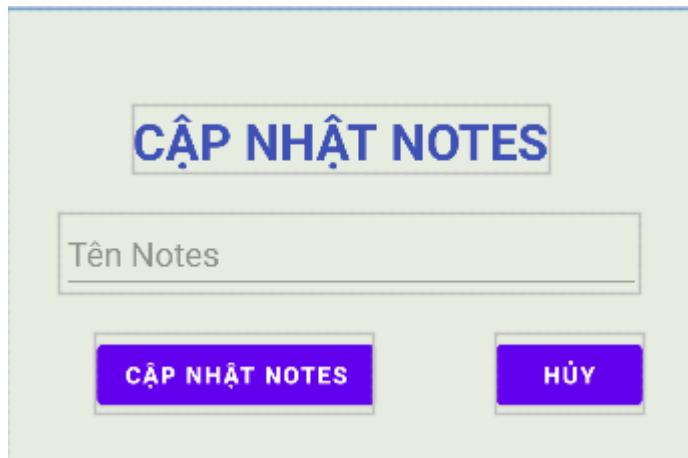
- Bước 8: Mở rộng chức năng: thêm Notes
- Vào MainActivity.java bắt sự kiện cho menu thêm notes:

```
//bắt sự kiện cho menu
@Override
public boolean onOptionsItemSelected(@NonNull MenuItem item) {
    if(item.getItemId() == R.id.menuAddNotes){
        DialogThem();
    }
    return super.onOptionsItemSelected(item);
}
```

- Bước 8: Mở rộng chức năng: thêm Notes
- Vào MainActivity.java định nghĩa hàm DialogThem():

```
private void DialogThem(){  
    Dialog dialog = new Dialog(context: this);  
    dialog.requestWindowFeature(Window.FEATURE_NO_TITLE);  
    dialog.setContentView(R.layout.dialog_add_notes);  
    //ảnh xạ trong dialog  
    EditText editText = dialog.findViewById(R.id.editTextName);  
    Button buttonAdd = dialog.findViewById(R.id.buttonThem);  
    Button buttonHuy = dialog.findViewById(R.id.buttonHuy);  
    //bắt sự kiện cho nút thêm và huy  
    buttonAdd.setOnClickListener(new View.OnClickListener() {  
        @Override  
        public void onClick(View v) {  
            String name = editText.getText().toString().trim();  
            if (name.equals("")){  
                Toast.makeText(context: MainActivity.this, text: "Vui lòng nhập tên Notes", Toast.LENGTH_SHORT).show();  
            }else{  
                databaseHandler.QueryData(sql: "INSERT INTO Notes VALUES(null, '"+ name +"')");  
                Toast.makeText(context: MainActivity.this, text: "Đã thêm Notes", Toast.LENGTH_SHORT).show();  
                dialog.dismiss();  
                databaseSQLite(); //gọi hàm load lại dữ liệu  
            }  
        }  
    });  
    buttonHuy.setOnClickListener(new View.OnClickListener() {  
        @Override  
        public void onClick(View v) {  
            dialog.dismiss();  
        }  
    });  
    dialog.show();  
}
```

- Bước 9: Mở rộng chức năng: cập nhật
- Tạo dialog cho chức năng cập nhật:



```
//ảnh xạ
EditText editText = dialog.findViewById(R.id.editTextName);
Button buttonEdit = dialog.findViewById(R.id.buttonEdit);
Button buttonHuy = dialog.findViewById(R.id.buttonHuy);
```

- Bước 9: Mở rộng chức năng: cập nhật
 - Viết hàm cập nhật trong MainActivity.java:

```
//hàm dialog cập nhật Notes
public void DialogCapNhatNotes(String name, int id){
    Dialog dialog = new Dialog(context: this);
    dialog.requestWindowFeature(Window.FEATURE_NO_TITLE);
    dialog.setContentView(R.layout.dialog_edit_notes);
    //ánh xạ
    EditText editText = dialog.findViewById(R.id.editTextName);
    Button buttonEdit = dialog.findViewById(R.id.buttonEdit);
    Button buttonHuy = dialog.findViewById(R.id.buttonHuy);
    editText.setText(name);
    //bắt sự kiện
    buttonEdit.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            String name = editText.getText().toString().trim();
            databaseHandler.QueryData(sql: "UPDATE Notes SET NameNotes ='"+ name +"' WHERE Id = '"+ id +"')");
            Toast.makeText(context: MainActivity.this, text: "Đã cập nhật Notes thành công", Toast.LENGTH_SHORT).show();
            dialog.dismiss();
            databaseSQLite(); //gọi hàm load lại dữ liệu
        }
    });
    buttonHuy.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            dialog.dismiss();
        }
    });
    dialog.show();
}
```

- Bước 9: Mở rộng chức năng: cập nhật
- Gọi hàm DialogCapNhatNotes() trong NotesAdapter.java:

```
private MainActivity context; //điều chỉnh biến context
```

```
//lấy giá trị
final NotesModel notes = noteList.get(position);
viewHolder.textViewNote.setText(notes.getNameNote());
//bắt sự kiện nút cập nhật
viewHolder.imageViewEdit.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        Toast.makeText(context, text: "Cập nhật " + notes.getNameNote(), Toast.LENGTH_SHORT).show();
        //gọi Dialog trong MainActivity.java
        context.DialogCapNhatNotes(notes.getNameNote(), notes.getIdNote());
    }
});

return convertView;
```

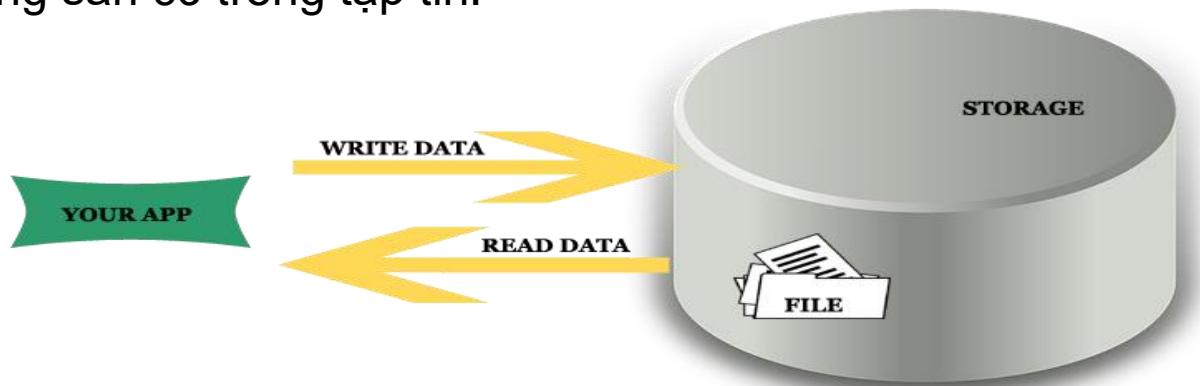
- Bước 10: Mở rộng chức năng: xóa
- Vào NotesAdapter.java bắt sự kiện cho nút xóa Notes:

```
//bắt sự kiện xóa notes
viewHolder.imageViewDelete.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        context.DialogDelete(notes.getNameNote(), notes.getIdNote());
    }
});
```

- Bước 10: Mở rộng chức năng: xóa
- Vào MainActivity.java viết hàm DialogDelete() để xóa Notes:

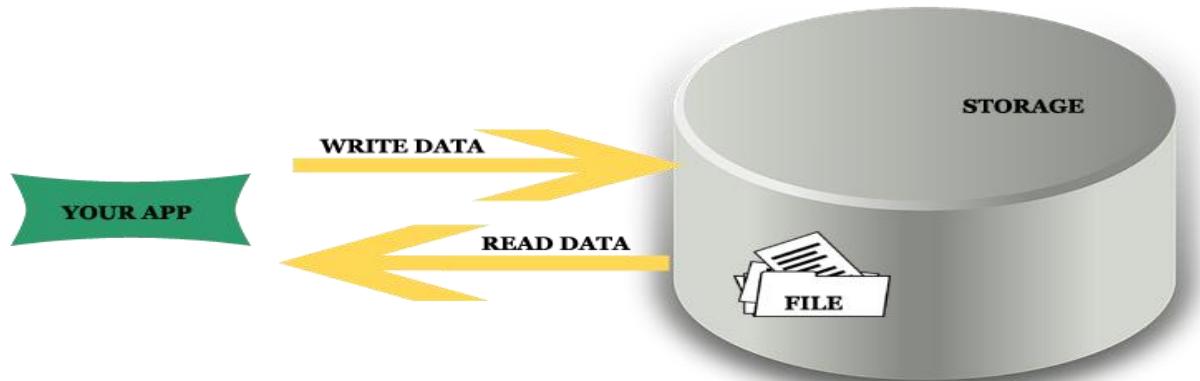
```
//hàm dialog xóa
public void DialogDelete(String name, final int id){
    AlertDialog.Builder builder = new AlertDialog.Builder( context: this);
    builder.setMessage("Bạn có muốn xóa Notes " + name + " này không ?");
    builder.setPositiveButton( text: "Có", new DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialog, int which) {
            databaseHandler.QueryData( sq: "DELETE FROM Notes WHERE Id = '"+ id +"");
            Toast.makeText( context: MainActivity.this, text: "Đã xóa Notes " +name+" thành công", Toast.LENGTH_SHORT).show();
            databaseSQLite(); //gọi hàm load lại dữ liệu
        }
    });
    builder.setNegativeButton( text: "Không", new DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialog, int which) {
        }
    });
    builder.show();
}
```

- Để đơn giản hóa việc đọc và ghi luồng dữ liệu tập tin được lưu ở bộ nhớ trong (hay gọi là thư mục dữ liệu bên trong ứng dụng trong hệ thống) Android cung cấp các phương thức ***openFileInput*** / ***openFileOutput*** tương ứng cho 2 luồng đọc và ghi từ context của phiên làm việc hiện tại của ứng dụng. Nếu tập tin có tên là giá trị được truyền làm đối số của hàm ***FileOutputStream()*** không tồn tại (tập tin chưa được tạo ra và lưu lại), hệ thống sẽ tự động tạo tập tin và ghi xuống bộ nhớ. Nếu đã có sẵn thì hệ thống mặc định sẽ chép đè nội dung mới lên nội dung cũ của tập tin. Trừ trường hợp nếu mode được gán là ***Context.MODE_APPEND*** thì nội dung mới sẽ được thêm vào nối tiếp nội dung sẵn có trong tập tin.



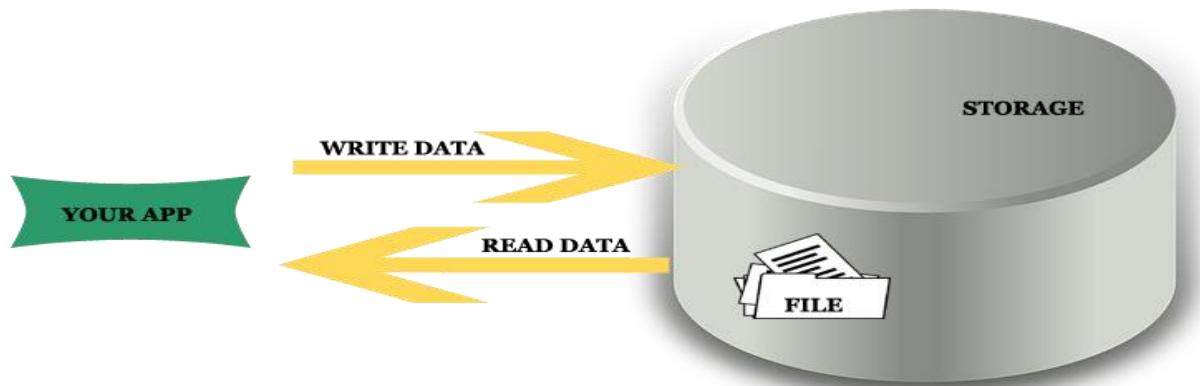
□ Một số điểm quan trọng trong việc lưu trữ trong bộ nhớ trong

- Dữ liệu trong bộ nhớ được phép đọc và ghi vào tập tin.
- Các tập tin này chỉ được truy cập bởi chính ứng dụng đó, không cho phép truy cập bởi ứng dụng khác.
- Các tập tin này vẫn tồn tại khi ứng dụng vẫn tồn tại, các tập tin sẽ bị xóa tự động khi ứng dụng uninstall.
- Các tập tin này lưu trữ trong thư mục data/data theo sau là tên package của ứng dụng.
- Có thể phân quyền cho các ứng dụng khác truy cập các tập tin này.
- Để phân quyền dữ liệu chúng ta có thể dùng chế độ MODE_PRIVATE



□ Các Mode lưu trữ ở bộ nhớ trong

- **MODE_PRIVATE** – Đây là chế độ mặc định. MODE_PRIVATE được sử dụng để thiết lập chế độ riêng tư, không chấp nhận ứng dụng từ bên ngoài.
- **MODE_APPEND** – Dữ liệu được thêm vào tập tin đã tồn tại



Lưu tập tin ở bộ nhớ trong

□ **Ghi dữ liệu vào tập tin**

- Định nghĩa tên tập tin và cũng định nghĩa dữ liệu bạn muốn ghi vào tập tin.
- Sử dụng phương thức FileOutputStream .
- Đổi dữ liệu thành byte stream.

```
String File_Name= "Demo.txt"; //gives file name

String Data="Hello!!"; //define data

FileOutputStream fileobj = openFileOutput( File_Name,
Context.MODE_PRIVATE);

byte[] ByteArray = Data.getBytes(); //Converts into bytes
stream

fileobj.write(ByteArray); //writing to file

fileobj.close(); //File closed
```

□ Ví dụ: Ghi dữ liệu vào tập tin

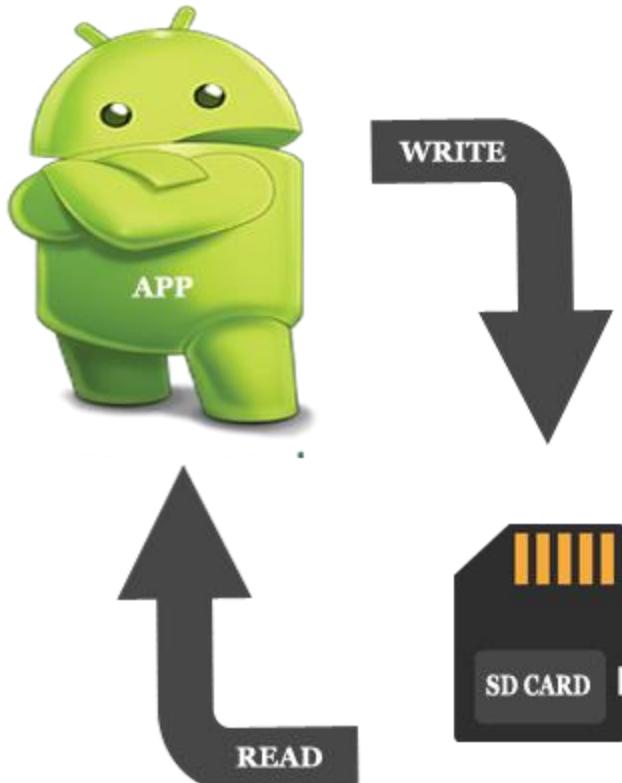
```
public void saveMe(View view) // SAVE
{
    File file= null;
    String name = editname.getText().toString();
    String password = editpass.getText().toString();

    FileOutputStream fileOutputStream = null;
    try {
        name = name + " ";
        file = getFilesDir();
        fileOutputStream = openFileOutput("Code.txt", Context.MODE_PRIVATE); //MODE PRIVATE
        fileOutputStream.write(name.getBytes());
        fileOutputStream.write(password.getBytes());
        Toast.makeText(this, "Saved \n" + "Path --" + file + "\tCode.txt", Toast.LENGTH_SHORT).show();
        editname.setText("");
        editpass.setText("");
        return;
    } catch (Exception ex) {
        ex.printStackTrace();
    } finally {
        try {
            fileOutputStream.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

□ Load dữ liệu

```
public void loadMe(View view)
{
    try {
        FileInputStream fileInputStream = openFileInput("Code.txt");
        int read = -1;
        StringBuffer buffer = new StringBuffer();
        while((read =fileInputStream.read()) != -1){
            buffer.append((char)read);
        }
        Log.d("Code", buffer.toString());
        String name = buffer.substring(0,buffer.indexOf(" "));
        String pass = buffer.substring(buffer.indexOf(" ")+1);
        getname.setText(name);
        getpass.setText(pass);
    } catch (Exception e) {
        e.printStackTrace();
    }
    Toast.makeText(this,"Loaded", Toast.LENGTH_SHORT).show();
}
```

- Cũng giống như Lưu tập tin ở bộ nhớ trong (internal storage), chúng ta có thể lưu và đọc dữ liệu từ bộ nhớ ngoài (external storage) của thiết bị như sdcard. Hai lớp FileInputStream and FileOutputStream được sử dụng để đọc và ghi dữ liệu vào tập tin.



Chú ý: Để có thể đọc/ghi trong tập tin lưu trong bộ nhớ ngoài, chúng ta cần phân quyền trong tập tin `AndroidManifest.xml`

```
<uses-permission  
    android:name="android.permission.WRITE_  
    EXTERNAL_STORAGE" />  
  
<uses-permission  
    android:name="android.permission.READ_  
    EXTERNAL_STORAGE" />
```

□ Các phương thức để lưu trữ dữ liệu

- **getExternalStorageState()**: Nên luôn kiểm tra tính sẵn sàng cho phép truy xuất của bộ nhớ ngoài thông qua hàm **getExternalStorageState()** – trả về trạng thái hiện tại của bộ nhớ ngoài. Vì bộ nhớ ngoài (thẻ nhớ, SD Card) có thể sẽ không cho phép các ứng dụng trên thiết bị truy xuất đến nó nếu đang ở tình trạng tháo ra khỏi thiết bị hoặc đang được “mount” với một thiết bị khác như PC, laptop chẳng hạn.

```
String state = Environment.getExternalStorageState();
if (Environment.MEDIA_MOUNTED.equals(state)) {
    // External storage is available for read and write
}
if (Environment.MEDIA_MOUNTED.equals(state) ||
Environment.MEDIA_MOUNTED_READ_ONLY.equals(state)) {
    External storage is available to at least read
    // TODO:
}
```

□ Các phương thức để lưu trữ dữ liệu

- **getExternalFileDir(String type)**: Đối tượng File có đường dẫn thư mục nội bộ bên trong thư mục dữ liệu của ứng dụng đặt trong bộ nhớ ngoài. Nếu bạn thông số truyền vào là Null, thì kết quả trả về là thư mục gốc của thư mục dữ liệu của ứng dụng trong bộ nhớ ngoài.

```
// Truy xuất thư mục pictures trong thư mục ứng dụng trong bộ nhớ ngoài
File file = new File(context.getExternalFilesDir(
    Environment.DIRECTORY_PICTURES),
"UserPrivatePicturesDirectory");
if (!file.mkdirs()) {
    Log.e("ITLog", "Directory not created");
}
```

□ Các phương thức để lưu trữ dữ liệu

- **Environment.getExternalStoragePublicDirectory():** Đối tượng File có đường dẫn thư mục trong bộ nhớ ngoài dùng chung, vì mục đích chia sẻ, các tập tin này sẽ không bị xóa đi khi gỡ ứng dụng ra khỏi thiết bị. Ví dụ như, ứng dụng bạn dùng camera để chụp ảnh, và những file ảnh này nên được giữ lại cho người dùng cho dù ứng dụng của bạn đã được gỡ ra khỏi thiết bị rồi.

```
// Truy xuất thư mục pictures trong bộ nhớ ngoài dùng chung.  
File file = new File(Environment.getExternalStoragePublicDirectory(  
    Environment.DIRECTORY_PICTURES),  
    "UserPublicPicturesDirectory");  
if (!file.mkdirs()) {  
    Log.e("ITLog", "Directory not created");  
}
```

- Phương thức trên nhận đối số là chuỗi giá trị đường dẫn của thư mục con chứa các tập tin theo từng loại (âm thanh, hình ảnh, tập tin dạng text, ...) bên trong thư mục dữ liệu của ứng dụng. Những giá trị đường dẫn này là các hằng giá trị được định nghĩa trong lớp Environment:
 - **DIRECTORY_ALARMS** — Các file âm thanh được dùng làm báo thức
 - **DIRECTORY_DCIM** — Hình ảnh và video được chụp hoặc quay bằng thiết bị
 - **DIRECTORY_DOWNLOADS** — chứa các file được tải về
 - **DIRECTORY_MOVIES** — chứa các file video
 - **DIRECTORY_MUSIC** — Chứa các file âm thanh cho chương trình nghe nhạc
 - **DIRECTORY_NOTIFICATIONS** — Các file âm thanh cho các thông báo tin nhắn
 - **DIRECTORY_PICTURES** — Chứa các file hình ảnh
 - **DIRECTORY_RINGTONES** — Các file âm thanh được dùng làm nhạc chuông

Ví dụ:

```
public void savePublic(View view) {  
    //Permission to access external storage  
    ActivityCompat.requestPermissions(this, new  
String[]{Manifest.permission.READ_EXTERNAL_STORAGE},  
STORAGE_PERMISSION_CODE);  
    String info = editText.getText().toString();  
    File folder =  
Environment.getExternalStoragePublicDirectory(Environment.DIRECTO  
RY_ALARMS); // Folder Name  
    File myFile = new File(folder, "myData1.txt"); //  
    Filename  
    writeData(myFile, info);  
    editText.setText("");  
}
```

Ví dụ:

```
public void savePrivate(View view) {  
    String info = editText.getText().toString();  
    File folder = getExternalFilesDir("Demo");// Folder  
Name  
    File myFile = new File(folder, "myData2.txt");//  
Filename  
    writeData(myFile, info);  
    editText.setText("");  
}
```

Ví dụ:

```
private void writeData(File myFile, String data) {
    FileOutputStream fileOutputStream = null;
    try {
        fileOutputStream = new FileOutputStream(myFile);
        fileOutputStream.write(data.getBytes());
        Toast.makeText(this, "Done" + myFile.getAbsolutePath(),
Toast.LENGTH_SHORT).show();
    } catch (Exception e) {
        e.printStackTrace();
    } finally {
        if (fileOutputStream != null) {
            try {
                fileOutputStream.close();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }
}
```

Ví dụ:

```
public void getPublic(View view) {  
    File folder =  
Environment.getExternalStoragePublicDirectory(Environment.DIRECTORY_  
ALARMS); // Folder Name  
    File myFile = new File(folder, "myData1.txt"); //  
Filename  
    String text = getdata(myFile);  
    if (text != null) {  
        showText.setText(text);  
    } else {  
        showText.setText("No Data");  
    }  
}
```

Ví dụ:

```
public void getPrivate(View view) {  
    File folder = getExternalFilesDir("Demo"); // Folder  
    Name  
    File myFile = new File(folder, "myData2.txt"); //  
    Filename  
    String text = getdata(myFile);  
    if (text != null) {  
        showText.setText(text);  
    } else {  
        showText.setText("No Data");  
    }  
}
```

□ Ví dụ:

```
private String getdata(File myfile) {
    FileInputStream fileInputStream = null;
    try {
        fileInputStream = new FileInputStream(myfile);
        int i = -1;
        StringBuffer buffer = new StringBuffer();
        while ((i = fileInputStream.read()) != -1) {
            buffer.append((char) i);
        }
        return buffer.toString();
    } catch (Exception e) {
        e.printStackTrace();
    } finally {
        if (fileInputStream != null) {
            try {
                fileInputStream.close();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }
    return null;
}
```

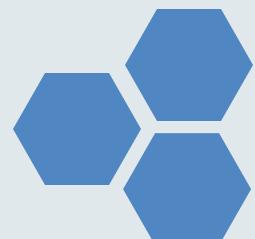
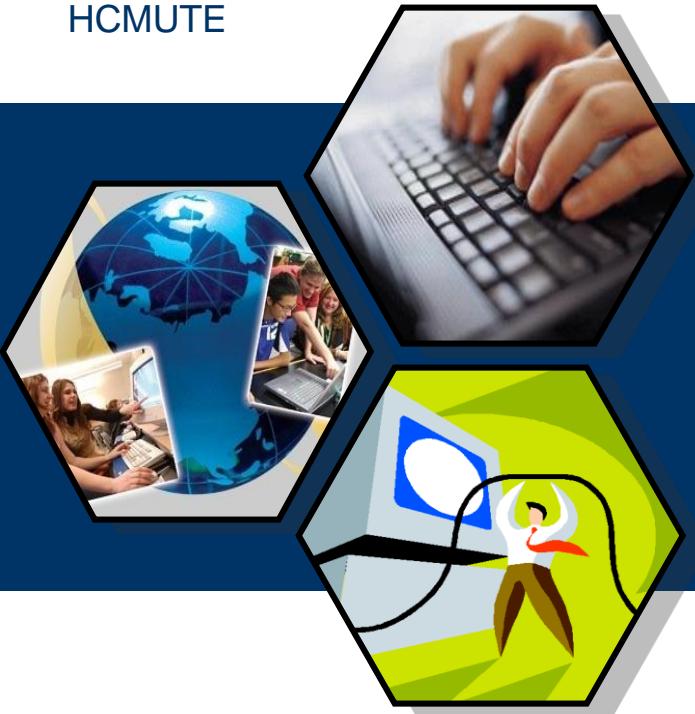


HCMUTE

KHOA CÔNG NGHỆ THÔNG TIN

UTEx

Thread, Handler và AsyncTask trong Android



Process và Thread

313

- Khi một ứng dụng Android được khởi chạy, thì hệ thống Android sẽ start một New Linux Process cho ứng dụng đó cùng với một Thread duy nhất để thực thi. Vậy là có cả Process và Thread đều được sinh ra phải không nào. Ta đã nghe về Process nhưng lại thường không nhớ nó là gì, có nhiệm vụ gì, và với Thread thì giữa chúng có mối quan hệ gì, chúng giống và khác nhau như thế nào.
 - Mỗi **Process** cung cấp tài nguyên cần thiết để thực thi chương trình. Mỗi Process có một không gian địa chỉ ảo, có các mã thực thi, có các lệnh xử lý các đối tượng hệ thống, một ngũ cảnh bảo mật, kích thước làm việc tối thiểu và tối đa, ... và phải có ít nhất một **Thread** thực thi.
 - Một **Thread** là một thực thể trong một **Process**, là đối tượng được lên kế hoạch để thực thi. Trong **Process** thường sẽ có nhiều **Thread** và tất cả các **Thread** này sẽ chia sẻ không gian địa chỉ ảo và tài nguyên hệ thống trong **Process**.
 - Ngoài ra mỗi **Thread** lại có công việc riêng của nó đó là: duy trì xử lý ngoại lệ, ưu tiên lập lịch trình, lưu trữ cục bộ luồng,... Mỗi trường Thread có thể bao gồm: phần đăng ký Thread với máy chủ, nhân stack, ... quan trọng là có một môi trường (Thread cũng có ngũ cảnh riêng của nó).
- Tóm lại **Process = Program + State of all Threads executing in Program**. Hay là Process bao gồm chương trình cùng với tất cả trạng thái thực thi của các Thread trong chương trình đó.

- Thread được định nghĩa là một luồng dùng để thực thi một chương trình.
- Java Virtual Machine cho phép một chương trình có thể có nhiều Thread thực thi đồng thời. Mỗi Thread đều có độ ưu tiên của nó. Rồi mỗi Thread có thể được đánh dấu là *daemon*.
- Daemon Thread là một loại Thread có độ ưu tiên thấp, cung cấp dịch vụ cho người dùng, là Thread duy trì hoạt động cho đến khi tất cả các Threads khác hoàn thành công việc hay chết đi thì nó cũng mới chết theo. Ví dụ cụ thể là trình dọn rác trong Java là một Daemon Thread. Để tạo mới Thread ta có hai cách. Cách thứ nhất là kế thừa (extends) từ class Thread và Cách thứ 2 là thực thi (implements) interface Runnable:

```
private class MyThread extends Thread{  
    @Override  
    public void run() {  
        //TODO  
    }  
    new MyThread().start();
```

```
private class MyRunnable implements  
Runnable{  
    @Override  
    public void run() { //TODO }  
}  
new Thread(new MyRunnable()).start();
```

- Multiple Thread là đa luồng, hay nhiều Thread. Tức là trong một chương trình, có thể có đồng thời nhiều Thread được thực thi.
- Khi một chương trình nhiều công việc mà chỉ có một Thread thực thi thì sẽ không hiệu quả, như vậy nhiều Thread làm việc thì sẽ hiệu quả hơn, và chúng chạy đồng thời.
- Các Thread này chia sẻ cùng một không gian tài nguyên, số lượng Thread càng nhiều thì độ phức tạp sẽ càng tăng, nên phải sử dụng cho hợp lý không thì chương trình sẽ xung đột, quá trình thực thi sẽ sai, thậm chí có thể chết chương trình.

```
private class MyThread extends Thread{  
    @Override  
    public void run() {  
        //TODO  
    }  
    new MyThread().start();
```

```
private class MyRunnable implements Runnable{  
    @Override  
    public void run() { //TODO }  
}  
new Thread(new MyRunnable()).start();
```

- Trong Android, khi chương trình được khởi chạy, hệ thống sẽ start một Thread ban đầu cùng với một Process. Thì Thread đó chính là **Main Thread**. Vậy vì sao Main Thread lại thường được gọi là **UI Thread** thì có 2 lý do sau đây.
 - ▣ Thread này có nhiệm vụ gửi các sự kiện đến widget, tức là đến các view ở giao diện điện thoại, thậm chí cả các sự kiện vẽ.
 - ▣ Ngoài ra Thread này cũng phải tương tác với bộ công cụ Android UI (Android UI toolkit) gồm hai gói thư viện là *android.widget* và *android.view*.
- Có khi nào Main Thread lại không được gọi là UI Thread không? Đó là khi một chương trình có nhiều hơn một Thread phụ trách việc xử lý giao diện. Còn một trường hợp nữa là **Worker Thread**, chính là Thread mà bạn tạo thêm cho chương trình để nó thực thi một công việc nào đó không liên quan đến giao diện, Thread này cũng được gọi là Background Thread.

- Khi có nhiều thứ thực thi trên UI Thread, và có một công việc gì đó cần phải thực hiện lâu như kết nối mạng hay truy vấn cơ sở dữ liệu, khi đó UI sẽ bị block. Người dùng cảm thấy như ứng dụng đang bị treo, nhưng thực ra nó đang thực thi công việc của mình trên UI Thread. Nếu UI bị block hơn vài giây (trung bình là 5 giây) thì hệ thống Android sẽ xuất hiện hộp thoại như trên, cho phép người dùng có thể đóng chương trình hoặc chờ đợi. Nếu như ứng dụng thường xuyên có những hiện tượng như vậy thì sẽ bất tiện cho người dùng. Chính vì vậy, để không xảy ra hiện tượng trên thì Android là đề ra 2 rules sau đây yêu cầu lập trình viên phải tuân theo:
 - Không được block UI Thread.
 - Không được kết nối tới bộ công cụ Android UI (Android UI toolkit) từ một Thread không phải là UI Thread.

```
public void onClick(View view) { new Thread(new Runnable() {  
    @Override public void run() { Bitmap b =  
        loadImageFromNetwork("http://example.com/img.png");  
        mImageView.setImageBitmap(b); } }).start(); }
```

- Ví dụ bên dưới ta thấy việc kết nối internet để load hình ảnh được thực hiện một Thread khác (Worker Thread hay Background Thread) nên sẽ không block UI, thỏa mãn rule thứ nhất. Nhưng ở trong Thread này nó đã trực tiếp tác động đến UI bằng câu lệnh ***mImageView.setImageBitmap(b)*** và không thỏa mãn rule thứ 2 nên code sẽ không chạy được. Từ Worker Thread ta không thể update UI.

```
public void onClick(View view) {  
    new Thread(new Runnable() {  
        @Override  
        public void run() {  
            Bitmap b = loadImageFromNetwork("http://example.com/img.png");  
            mImageView.setImageBitmap(b);  
        }  
    }).start(); }
```

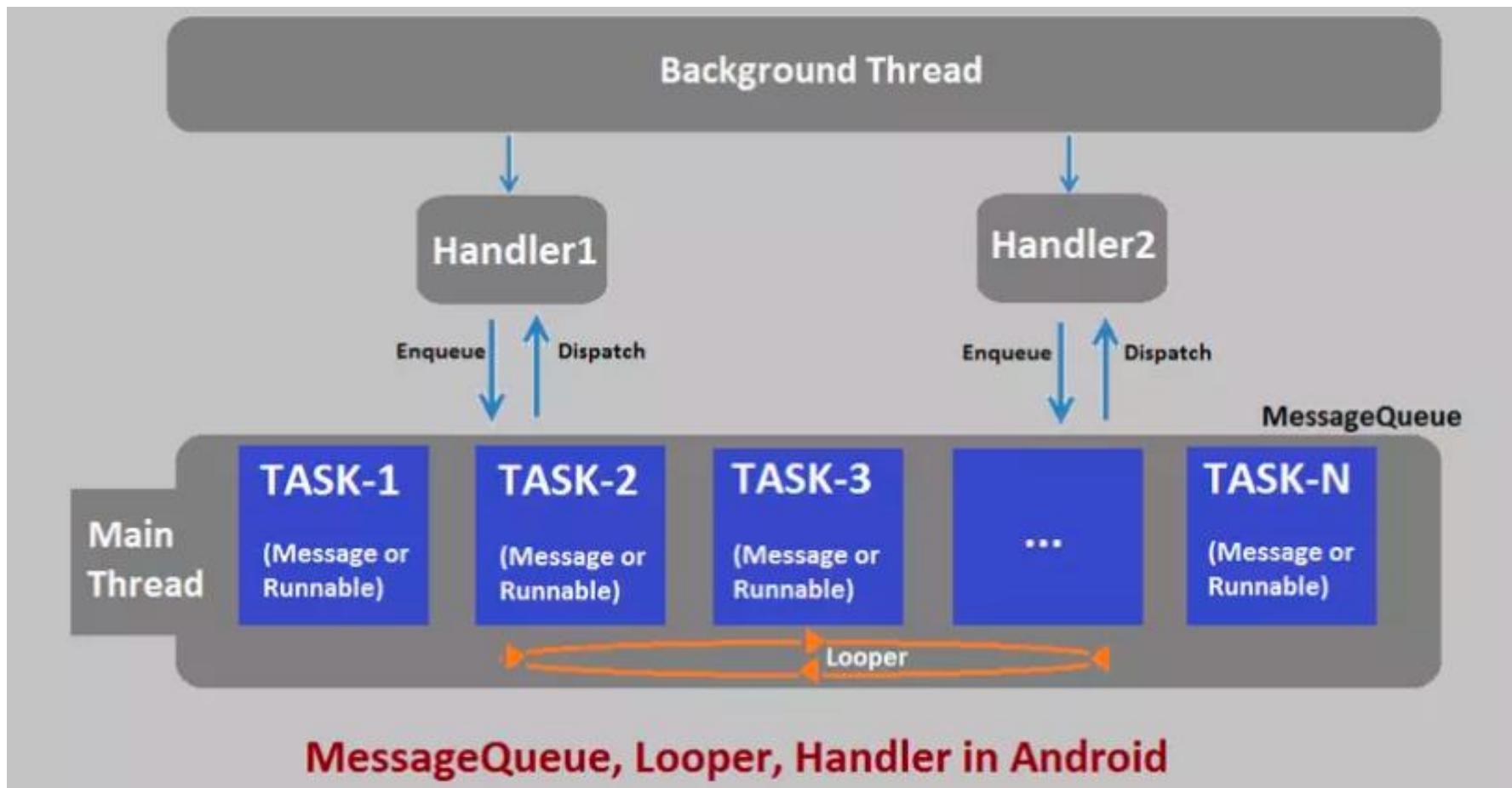
- Từ đó Android đã cung cấp cho Worker Thread một số phương thức sau để làm điều đó. Ví dụ trên sẽ được sửa lại như sau:
 - 1.*Activity.runOnUiThread(Runnable)*
 - 2.*View.post(Runnable)*
 - 3. *View.postDelayed(Runnable, long)*

```
public void onClick(View view) {  
    new Thread(new Runnable() {  
        @Override  
        public void run() {  
            final Bitmap b = loadImageFromNetwork("http://example.com/img.png");  
            mImageView.post(new Runnable() {  
                @Override  
                public void run() {  
                    mImageView.setImageBitmap(b);  
                }  
            });  
        }  
    }).start();  
}
```

- Handler là một đối Android cung cấp dùng để liên kết, trao đổi giữa các Thread với nhau, là trao đổi giữa Thread sinh ra Handler và các Thread khác.
- Thường là Main Thread (UI Thread) với các Worker Thread (Background Thread).
- Handler có nhiệm vụ gửi và thực thi các Message hoặc Runnable tới Message Queue của Thread sinh ra nó (Handler).
- Handler luôn được gắn kết với một Thread (Thread sinh ra nó) cũng với Message Queue (của Thread đó).
- Các Message và Runnable sẽ được thực thi khi đi ra khỏi Message Queue. Có 2 nhiệm vụ mà Handler thường làm đó là:
 - ▣ Lên lịch thực thi các Message và Runnable ở các thời điểm trong tương lai.
 - ▣ Sắp xếp một hành động được thực hiện trong một Thread khác.

Handler

321



- Mesage Queue có thể xem là một hàng đợi, nó nắm giữ một list các Message được gửi tới bởi đối tượng Looper.
- Các Message không được add ngay lập tức vào Message Queue này mà phải thông qua Handler kết hợp với Looper.
- Looper là một đối tượng dùng để chạy vòng lặp trong Message trong Thread.
- Các bạn có thể hiểu Message Queue là một đường hầm, còn tập hợp các Task (Message or Runnable) là một con tàu, mỗi Task là một toa tàu. Còn Looper là người lái tàu, và người lái tàu này lại lái tàu qua khỏi hầm rồi lại vòng lại đi vào hầm tiếp, nó duy trì công việc liên tục trong Main Thread.
- Nếu Handler sinh ra ở Main Thread thì đã có mặc định một Looper sinh ra, nhưng nếu nó sinh ra một Thread không phải là Main Thread thì phải tạo Handler trong cặp lệnh *Looper.prepare()* và *Looper.loop()*.

- Khi dùng Handler để giao tiếp giữa Worker Thread với UI thì có một số bất tiện là phải code nhiều, dài dòng thì AsyncTask là một giải pháp thích hợp hơn.

```
class LooperThread extends Thread {  
    public Handler mHandler;  
    @Override  
    public void run() {  
        Looper.prepare();  
        mHandler = new Handler() {  
            @Override  
            public void handleMessage(Message msg) {  
                // process incoming messages here  
            }  
        }; Looper.loop();  
    } }
```

```
//tự động chuyển activity
(Thread) run() {
    int n = 0;
    try {
        do {
            if (n >= 2000) {
                IntroActivity.this.finish();
                Intent intent = new Intent(IntroActivity.this.getApplicationContext(), (Class) LoginActivity.class);
                IntroActivity.this.startActivity(intent);
                return;
            }
            Thread.sleep((long) 100);
            n += 100;
        } while (true);
    }catch (InterruptedException interruptedException) {
        IntroActivity.this.finish();
        Intent intent = new Intent(IntroActivity.this.getApplicationContext(), (Class) LoginActivity.class);
        IntroActivity.this.startActivity(intent);
        return;
    }
    catch (Throwable throwable) {
        IntroActivity.this.finish();
        Intent intent = new Intent(IntroActivity.this.getApplicationContext(), (Class) LoginActivity.class);
        IntroActivity.this.startActivity(intent);
        throw throwable;
    }
}.start();
```

Thêm code sau vào hàm onCreate của Activity

- **Bước 1: Thiết kế giao diện activity_main.xml**

```
<androidx.recyclerview.widget.RecyclerView  
    android:id="@+id/recyclerView"  
    android:layout_width="match_parent"  
    android:layout_height="match_parent"  
    app:layout_constraintBottom_toBottomOf="parent"  
    app:layout_constraintLeft_toLeftOf="parent"  
    app:layout_constraintRight_toRightOf="parent"  
    app:layout_constraintTop_toTopOf="parent" />
```

- **Bước 2: Thiết kế giao diện item_row.xml**

```
<androidx.cardview.widget.CardView  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    app:cardElevation="8dp"  
    app:cardUseCompatPadding="true"  
    app:layout_constraintEnd_toEndOf="parent"  
    app:layout_constraintStart_toStartOf="parent"  
    app:layout_constraintTop_toTopOf="parent">  
    <TextView  
        android:id="@+id/tvItem"  
        android:layout_width="match_parent"  
        android:layout_height="wrap_content"  
        android:padding="16dp"  
        android:text="Item X" />  
</androidx.cardview.widget.CardView>
```

- **Bước 2: Thiết kế giao diện item_progressbar.xml**

```
<ProgressBar  
    android:id="@+id/progressBar"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:layout_gravity="center_horizontal"  
    android:indeterminate="true"  
    android:paddingLeft="8dp"  
    android:paddingRight="8dp" />
```

□ Bước 4: Tạo class Adapter

```
public class RecyclerViewAdapter extends RecyclerView.Adapter<RecyclerView.ViewHolder> {
    private final int VIEW_TYPE_ITEM = 0;
    private final int VIEW_TYPE_LOADING = 1;
    public List<String> mItemList;
    public RecyclerViewAdapter(List<String> itemList) {
        mItemList = itemList;
    }
    @NonNull
    @Override
    public RecyclerView.ViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
        if (viewType == VIEW_TYPE_ITEM) {
            View view = LayoutInflater.from(parent.getContext()).inflate(R.layout.item_row, parent, attachToRoot: false);
            return new ItemViewHolder(view);
        } else {
            View view = LayoutInflater.from(parent.getContext()).inflate(R.layout.item_loading, parent, attachToRoot: false);
            return new LoadingViewHolder(view);
        }
    }
    @Override
    public void onBindViewHolder(@NonNull RecyclerView.ViewHolder viewHolder, int position) {
        if (viewHolder instanceof ItemViewHolder) {
            populateItemRows((ItemViewHolder) viewHolder, position);
        } else if (viewHolder instanceof LoadingViewHolder) {
            showLoadingView((LoadingViewHolder) viewHolder, position);
        }
    }
}
```

□ Bước 4: Tạo class Adapter

```
@Override  
public int getItemCount() {  
    return mItemList == null ? 0 : mItemList.size();  
}  
  
@Override  
public int getItemViewType(int position) {  
    return mItemList.get(position) == null ? VIEW_TYPE_LOADING : VIEW_TYPE_ITEM;  
}  
  
private class ItemViewHolder extends RecyclerView.ViewHolder {  
    TextView tvItem;  
    public ItemViewHolder(@NonNull View itemView) {  
        super(itemView);  
        tvItem = itemView.findViewById(R.id.tvItem);  
    }  
}  
  
private class LoadingViewHolder extends RecyclerView.ViewHolder {  
    ProgressBar progressBar;  
    public LoadingViewHolder(@NonNull View itemView) {  
        super(itemView);  
        progressBar = itemView.findViewById(R.id.progressBar);  
    }  
}  
  
private void showLoadingView(LoadingViewHolder viewHolder, int position) {  
    //ProgressBar would be displayed  
}  
  
private void populateItemRows(ItemViewHolder viewHolder, int position) {  
    String item = mItemList.get(position);  
    viewHolder.tvItem.setText(item);  
}
```

□ Bước 5: Viết code MainActivity.java

```
public class MainActivity extends AppCompatActivity {
    RecyclerView recyclerView;
    RecyclerViewAdapter recyclerViewAdapter;
    ArrayList<String> rowsArrayList = new ArrayList<>();
    boolean isLoading = false;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        recyclerView = findViewById(R.id.recyclerView);
        populateData();
        initAdapter();
        initScrollListener();
    }
    private void populateData() {
        int i = 0;
        while (i < 10) {
            rowsArrayList.add("Item " + i);
            i++;
        }
    }
    private void initAdapter() {
        recyclerViewAdapter = new RecyclerViewAdapter(rowsArrayList);
        recyclerView.setHasFixedSize(true);
        recyclerView.setLayoutManager(new GridLayoutManager(getApplicationContext(), spanCount: 2));
        recyclerView.setAdapter(recyclerViewAdapter);
    }
}
```

□ Bước 5: Viết code MainActivity.java

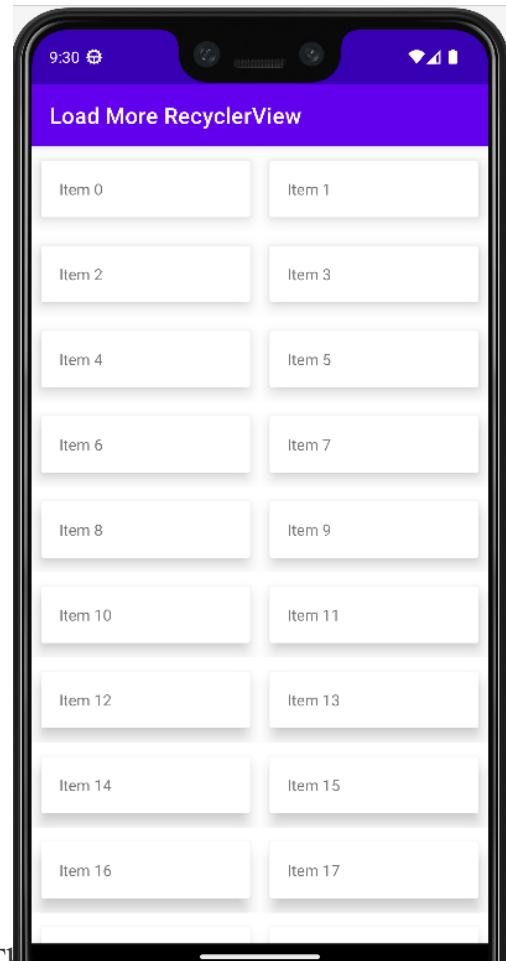
```
private void initScrollListener() {
    recyclerView.addOnScrollListener(new RecyclerView.OnScrollListener() {
        @Override
        public void onScrollStateChanged(@NonNull RecyclerView recyclerView, int newState) {
            super.onScrollStateChanged(recyclerView, newState);
        }
        @Override
        public void onScrolled(@NonNull RecyclerView recyclerView, int dx, int dy) {
            super.onScrolled(recyclerView, dx, dy);

            LinearLayoutManager linearLayoutManager = (LinearLayoutManager) recyclerView.getLayoutManager();

            if (!isLoading) {
                if (linearLayoutManager != null
                    && linearLayoutManager.findLastCompletelyVisibleItemPosition() == rowsArrayList.size() - 1) {
                    //bottom of list!
                    loadMore();
                    isLoading = true;
                }
            }
        }
    });
}
```

□ Bước 5: Viết code MainActivity.java

```
private void loadMore() {  
    rowsArrayList.add(null);  
    recyclerViewAdapter.notifyItemInserted( position: rowsArrayList.size() - 1);  
    Handler handler = new Handler();  
    handler.postDelayed(new Runnable() {  
        @Override  
        public void run() {  
            rowsArrayList.remove( index: rowsArrayList.size() - 1);  
            int scrollPosition = rowsArrayList.size();  
            recyclerViewAdapter.notifyItemRemoved(scrollPosition);  
            int currentSize = scrollPosition;  
            int nextLimit = currentSize + 10;  
  
            while (currentSize - 1 < nextLimit) {  
                rowsArrayList.add("Item " + currentSize);  
                currentSize++;  
            }  
            recyclerViewAdapter.notifyDataSetChanged();  
            isLoading = false;  
        }  
    }, delayMillis: 2000);  
}
```



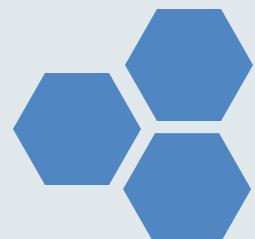
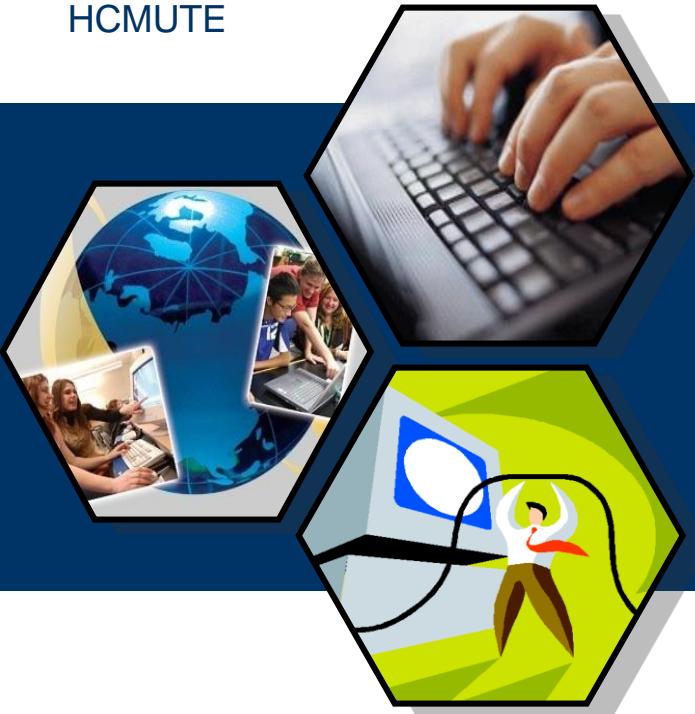


HCMUTE

KHOA CÔNG NGHỆ THÔNG TIN

UTEx

Thread, Handler và AsyncTask trong Android



Process và Thread

332

- Khi một ứng dụng Android được khởi chạy, thì hệ thống Android sẽ start một New Linux Process cho ứng dụng đó cùng với một Thread duy nhất để thực thi. Vậy là có cả Process và Thread đều được sinh ra phải không nào. Ta đã nghe về Process nhưng lại thường không nhớ nó là gì, có nhiệm vụ gì, và với Thread thì giữa chúng có mối quan hệ gì, chúng giống và khác nhau như thế nào.
 - Mỗi **Process** cung cấp tài nguyên cần thiết để thực thi chương trình. Mỗi Process có một không gian địa chỉ ảo, có các mã thực thi, có các lệnh xử lý các đối tượng hệ thống, một ngũ cảnh bảo mật, kích thước làm việc tối thiểu và tối đa, ... và phải có ít nhất một **Thread** thực thi.
 - Một **Thread** là một thực thể trong một **Process**, là đối tượng được lên kế hoạch để thực thi. Trong **Process** thường sẽ có nhiều **Thread** và tất cả các **Thread** này sẽ chia sẻ không gian địa chỉ ảo và tài nguyên hệ thống trong **Process**.
 - Ngoài ra mỗi **Thread** lại có công việc riêng của nó đó là: duy trì xử lý ngoại lệ, ưu tiên lập lịch trình, lưu trữ cục bộ luồng,... Mỗi trường Thread có thể bao gồm: phần đăng ký Thread với máy chủ, nhân stack, ... quan trọng là có một môi trường (Thread cũng có ngũ cảnh riêng của nó).
- Tóm lại **Process = Program + State of all Threads executing in Program**. Hay là Process bao gồm chương trình cùng với tất cả trạng thái thực thi của các Thread trong chương trình đó.

- Thread được định nghĩa là một luồng dùng để thực thi một chương trình.
- Java Virtual Machine cho phép một chương trình có thể có nhiều Thread thực thi đồng thời. Mỗi Thread đều có độ ưu tiên của nó. Rồi mỗi Thread có thể được đánh dấu là *daemon*.
- Daemon Thread là một loại Thread có độ ưu tiên thấp, cung cấp dịch vụ cho người dùng, là Thread duy trì hoạt động cho đến khi tất cả các Threads khác hoàn thành công việc hay chết đi thì nó cũng mới chết theo. Ví dụ cụ thể là trình dọn rác trong Java là một Daemon Thread. Để tạo mới Thread ta có hai cách. Cách thứ nhất là kế thừa (extends) từ class Thread và Cách thứ 2 là thực thi (implements) interface Runnable:

```
private class MyThread extends Thread{  
    @Override  
    public void run() {  
        //TODO  
    }  
    new MyThread().start();
```

```
private class MyRunnable implements  
Runnable{  
    @Override  
    public void run() { //TODO }  
}  
new Thread(new MyRunnable()).start();
```

- Multiple Thread là đa luồng, hay nhiều Thread. Tức là trong một chương trình, có thể có đồng thời nhiều Thread được thực thi.
- Khi một chương trình nhiều công việc mà chỉ có một Thread thực thi thì sẽ không hiệu quả, như vậy nhiều Thread làm việc thì sẽ hiệu quả hơn, và chúng chạy đồng thời.
- Các Thread này chia sẻ cùng một không gian tài nguyên, số lượng Thread càng nhiều thì độ phức tạp sẽ càng tăng, nên phải sử dụng cho hợp lý không thì chương trình sẽ xung đột, quá trình thực thi sẽ sai, thậm chí có thể chết chương trình.

```
private class MyThread extends Thread{  
    @Override  
    public void run() {  
        //TODO  
    }  
    new MyThread().start();
```

```
private class MyRunnable implements Runnable{  
    @Override  
    public void run() { //TODO }  
}  
new Thread(new MyRunnable()).start();
```

- Trong Android, khi chương trình được khởi chạy, hệ thống sẽ start một Thread ban đầu cùng với một Process. Thì Thread đó chính là **Main Thread**. Vậy vì sao Main Thread lại thường được gọi là **UI Thread** thì có 2 lý do sau đây.
 - ▣ Thread này có nhiệm vụ gửi các sự kiện đến widget, tức là đến các view ở giao diện điện thoại, thậm chí cả các sự kiện vẽ.
 - ▣ Ngoài ra Thread này cũng phải tương tác với bộ công cụ Android UI (Android UI toolkit) gồm hai gói thư viện là *android.widget* và *android.view*.
- Có khi nào Main Thread lại không được gọi là UI Thread không? Đó là khi một chương trình có nhiều hơn một Thread phụ trách việc xử lý giao diện. Còn một trường hợp nữa là **Worker Thread**, chính là Thread mà bạn tạo thêm cho chương trình để nó thực thi một công việc nào đó không liên quan đến giao diện, Thread này cũng được gọi là Background Thread.

- Khi có nhiều thứ thực thi trên UI Thread, và có một công việc gì đó cần phải thực hiện lâu như kết nối mạng hay truy vấn cơ sở dữ liệu, khi đó UI sẽ bị block. Người dùng cảm thấy như ứng dụng đang bị treo, nhưng thực ra nó đang thực thi công việc của mình trên UI Thread. Nếu UI bị block hơn vài giây (trung bình là 5 giây) thì hệ thống Android sẽ xuất hiện hộp thoại như trên, cho phép người dùng có thể đóng chương trình hoặc chờ đợi. Nếu như ứng dụng thường xuyên có những hiện tượng như vậy thì sẽ bất tiện cho người dùng. Chính vì vậy, để không xảy ra hiện tượng trên thì Android là đề ra 2 rules sau đây yêu cầu lập trình viên phải tuân theo:
 - Không được block UI Thread.
 - Không được kết nối tới bộ công cụ Android UI (Android UI toolkit) từ một Thread không phải là UI Thread.

```
public void onClick(View view) { new Thread(new Runnable() {  
    @Override public void run() { Bitmap b =  
        loadImageFromNetwork("http://example.com/img.png");  
        mImageView.setImageBitmap(b); } }).start(); }
```

- Ví dụ bên dưới ta thấy việc kết nối internet để load hình ảnh được thực hiện một Thread khác (Worker Thread hay Background Thread) nên sẽ không block UI, thỏa mãn rule thứ nhất. Nhưng ở trong Thread này nó đã trực tiếp tác động đến UI bằng câu lệnh ***mImageView.setImageBitmap(b)*** và không thỏa mãn rule thứ 2 nên code sẽ không chạy được. Từ Worker Thread ta không thể update UI.

```
public void onClick(View view) {  
    new Thread(new Runnable() {  
        @Override  
        public void run() {  
            Bitmap b = loadImageFromNetwork("http://example.com/img.png");  
            mImageView.setImageBitmap(b);  
        }  
    }).start(); }
```

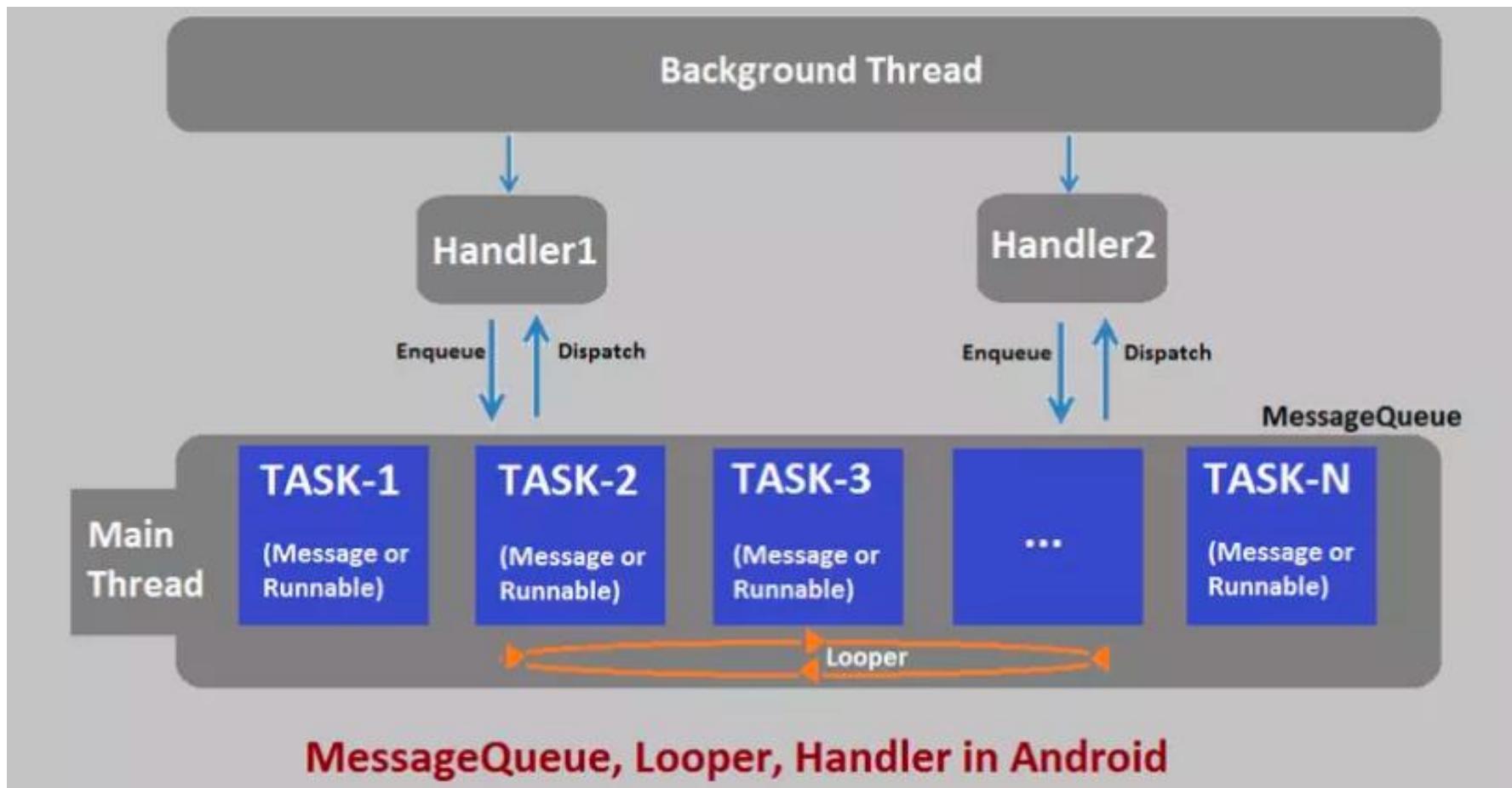
- Từ đó Android đã cung cấp cho Worker Thread một số phương thức sau để làm điều đó. Ví dụ trên sẽ được sửa lại như sau:
 - 1.*Activity.runOnUiThread(Runnable)*
 - 2.*View.post(Runnable)*
 - 3. *View.postDelayed(Runnable, long)*

```
public void onClick(View view) {  
    new Thread(new Runnable() {  
        @Override  
        public void run() {  
            final Bitmap b = loadImageFromNetwork("http://example.com/img.png");  
            mImageView.post(new Runnable() {  
                @Override  
                public void run() {  
                    mImageView.setImageBitmap(b);  
                }  
            });  
        }  
    }).start();  
}
```

- Handler là một đối Android cung cấp dùng để liên kết, trao đổi giữa các Thread với nhau, là trao đổi giữa Thread sinh ra Handler và các Thread khác.
- Thường là Main Thread (UI Thread) với các Worker Thread (Background Thread).
- Handler có nhiệm vụ gửi và thực thi các Message hoặc Runnable tới Message Queue của Thread sinh ra nó (Handler).
- Handler luôn được gắn kết với một Thread (Thread sinh ra nó) cũng với Message Queue (của Thread đó).
- Các Message và Runnable sẽ được thực thi khi đi ra khỏi Message Queue. Có 2 nhiệm vụ mà Handler thường làm đó là:
 - ▣ Lên lịch thực thi các Message và Runnable ở các thời điểm trong tương lai.
 - ▣ Sắp xếp một hành động được thực hiện trong một Thread khác.

Handler

340



- Mesage Queue có thể xem là một hàng đợi, nó nắm giữ một list các Message được gửi tới bởi đối tượng Looper.
- Các Message không được add ngay lập tức vào Message Queue này mà phải thông qua Handler kết hợp với Looper.
- Looper là một đối tượng dùng để chạy vòng lặp trong Message trong Thread.
- Các bạn có thể hiểu Message Queue là một đường hầm, còn tập hợp các Task (Message or Runnable) là một con tàu, mỗi Task là một toa tàu. Còn Looper là người lái tàu, và người lái tàu này lại lái tàu qua khỏi hầm rồi lại vòng lại đi vào hầm tiếp, nó duy trì công việc liên tục trong Main Thread.
- Nếu Handler sinh ra ở Main Thread thì đã có mặc định một Looper sinh ra, nhưng nếu nó sinh ra một Thread không phải là Main Thread thì phải tạo Handler trong cặp lệnh *Looper.prepare()* và *Looper.loop()*.

- Khi dùng Handler để giao tiếp giữa Worker Thread với UI thì có một số bất tiện là phải code nhiều, dài dòng thì AsyncTask là một giải pháp thích hợp hơn.

```
class LooperThread extends Thread {  
    public Handler mHandler;  
    @Override  
    public void run() {  
        Looper.prepare();  
        mHandler = new Handler() {  
            @Override  
            public void handleMessage(Message msg) {  
                // process incoming messages here  
            }  
        }; Looper.loop();  
    } }
```

```
//tự động chuyển activity
(Thread) run() {
    int n = 0;
    try {
        do {
            if (n >= 2000) {
                IntroActivity.this.finish();
                Intent intent = new Intent(IntroActivity.this.getApplicationContext(), (Class) LoginActivity.class);
                IntroActivity.this.startActivity(intent);
                return;
            }
            Thread.sleep((long) 100);
            n += 100;
        } while (true);
    }catch (InterruptedException interruptedException) {
        IntroActivity.this.finish();
        Intent intent = new Intent(IntroActivity.this.getApplicationContext(), (Class) LoginActivity.class);
        IntroActivity.this.startActivity(intent);
        return;
    }
    catch (Throwable throwable) {
        IntroActivity.this.finish();
        Intent intent = new Intent(IntroActivity.this.getApplicationContext(), (Class) LoginActivity.class);
        IntroActivity.this.startActivity(intent);
        throw throwable;
    }
}.start();
```

Thêm code sau vào hàm onCreate của Activity

- Bước 1: Thiết kế giao diện activity_main.xml

```
<androidx.cardview.widget.CardView  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    app:cardElevation="8dp"  
    app:cardUseCompatPadding="true"  
    app:layout_constraintEnd_toEndOf="parent"  
    app:layout_constraintStart_toStartOf="parent"  
    app:layout_constraintTop_toTopOf="parent">|  
    <TextView  
        android:id="@+id/tvItem"  
        android:layout_width="match_parent"  
        android:layout_height="wrap_content"  
        android:padding="16dp"  
        android:text="Item X" />  
  
</androidx.cardview.widget.CardView>
```

- Bước 2: Thiết kế giao diện item_row.xml

```
<androidx.recyclerview.widget.RecyclerView  
    android:id="@+id/recyclerView"  
    android:layout_width="match_parent"  
    android:layout_height="match_parent"  
    app:layout_constraintBottom_toBottomOf="parent"  
    app:layout_constraintLeft_toLeftOf="parent"  
    app:layout_constraintRight_toRightOf="parent"  
    app:layout_constraintTop_toTopOf="parent" />
```

- Bước 2: Thiết kế giao diện item_progressbar.xml

```
<ProgressBar  
    android:id="@+id/progressBar"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:layout_gravity="center_horizontal"  
    android:indeterminate="true"  
    android:paddingLeft="8dp"  
    android:paddingRight="8dp"  
    />
```

□ Bước 4: Tạo class Adapter

```
public class RecyclerViewAdapter extends RecyclerView.Adapter<RecyclerView.ViewHolder> {
    private final int VIEW_TYPE_ITEM = 0;
    private final int VIEW_TYPE_LOADING = 1;
    public List<String> mItemList;
    public RecyclerViewAdapter(List<String> itemList) {
        mItemList = itemList;
    }
    @NonNull
    @Override
    public RecyclerView.ViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
        if (viewType == VIEW_TYPE_ITEM) {
            View view = LayoutInflater.from(parent.getContext()).inflate(R.layout.item_row, parent, attachToRoot: false);
            return new ItemViewHolder(view);
        } else {
            View view = LayoutInflater.from(parent.getContext()).inflate(R.layout.item_loading, parent, attachToRoot: false);
            return new LoadingViewHolder(view);
        }
    }
    @Override
    public void onBindViewHolder(@NonNull RecyclerView.ViewHolder viewHolder, int position) {
        if (viewHolder instanceof ItemViewHolder) {
            populateItemRows((ItemViewHolder) viewHolder, position);
        } else if (viewHolder instanceof LoadingViewHolder) {
            showLoadingView((LoadingViewHolder) viewHolder, position);
        }
    }
}
```

□ Bước 4: Tạo class Adapter

```
@Override  
public int getItemCount() {  
    return mItemList == null ? 0 : mItemList.size();  
}  
  
@Override  
public int getItemViewType(int position) {  
    return mItemList.get(position) == null ? VIEW_TYPE_LOADING : VIEW_TYPE_ITEM;  
}  
  
private class ItemViewHolder extends RecyclerView.ViewHolder {  
    TextView tvItem;  
    public ItemViewHolder(@NonNull View itemView) {  
        super(itemView);  
        tvItem = itemView.findViewById(R.id.tvItem);  
    }  
}  
  
private class LoadingViewHolder extends RecyclerView.ViewHolder {  
    ProgressBar progressBar;  
    public LoadingViewHolder(@NonNull View itemView) {  
        super(itemView);  
        progressBar = itemView.findViewById(R.id.progressBar);  
    }  
}  
  
private void showLoadingView(LoadingViewHolder viewHolder, int position) {  
    //ProgressBar would be displayed  
}  
  
private void populateItemRows(ItemViewHolder viewHolder, int position) {  
    String item = mItemList.get(position);  
    viewHolder.tvItem.setText(item);  
}
```

□ Bước 5: Viết code MainActivity.java

```
public class MainActivity extends AppCompatActivity {
    RecyclerView recyclerView;
    RecyclerViewAdapter recyclerViewAdapter;
    ArrayList<String> rowsArrayList = new ArrayList<>();
    boolean isLoading = false;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        recyclerView = findViewById(R.id.recyclerView);
        populateData();
        initAdapter();
        initScrollListener();
    }
    private void populateData() {
        int i = 0;
        while (i < 10) {
            rowsArrayList.add("Item " + i);
            i++;
        }
    }
    private void initAdapter() {
        recyclerViewAdapter = new RecyclerViewAdapter(rowsArrayList);
        recyclerView.setHasFixedSize(true);
        recyclerView.setLayoutManager(new GridLayoutManager(getApplicationContext(), spanCount: 2));
        recyclerView.setAdapter(recyclerViewAdapter);
    }
}
```

□ Bước 5: Viết code MainActivity.java

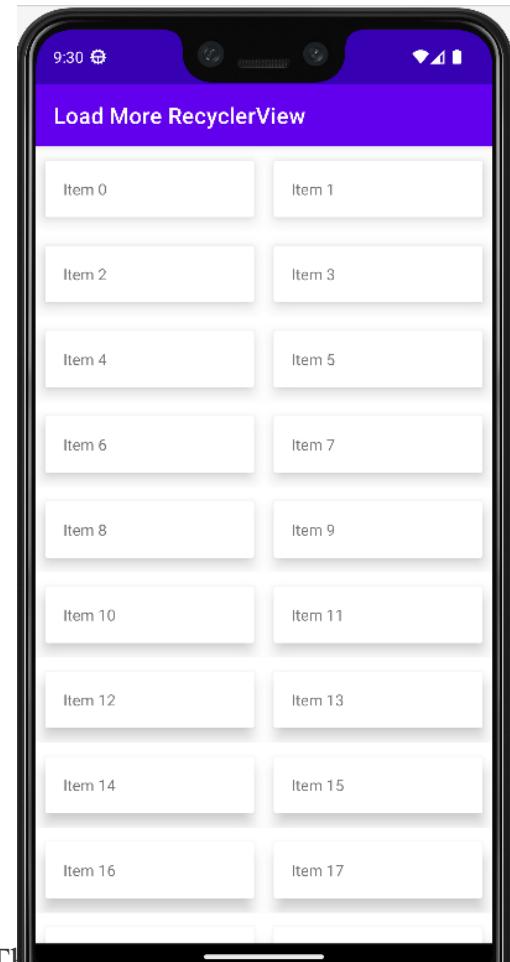
```
private void initScrollListener() {
    recyclerView.addOnScrollListener(new RecyclerView.OnScrollListener() {
        @Override
        public void onScrollStateChanged(@NonNull RecyclerView recyclerView, int newState) {
            super.onScrollStateChanged(recyclerView, newState);
        }
        @Override
        public void onScrolled(@NonNull RecyclerView recyclerView, int dx, int dy) {
            super.onScrolled(recyclerView, dx, dy);

            LinearLayoutManager linearLayoutManager = (LinearLayoutManager) recyclerView.getLayoutManager();

            if (!isLoading) {
                if (linearLayoutManager != null
                    && linearLayoutManager.findLastCompletelyVisibleItemPosition() == rowsArrayList.size() - 1) {
                    //bottom of list!
                    loadMore();
                    isLoading = true;
                }
            }
        }
    });
}
```

□ Bước 5: Viết code MainActivity.java

```
private void loadMore() {  
    rowsArrayList.add(null);  
    recyclerViewAdapter.notifyItemInserted( position: rowsArrayList.size() - 1);  
    Handler handler = new Handler();  
    handler.postDelayed(new Runnable() {  
        @Override  
        public void run() {  
            rowsArrayList.remove( index: rowsArrayList.size() - 1);  
            int scrollPosition = rowsArrayList.size();  
            recyclerViewAdapter.notifyItemRemoved(scrollPosition);  
            int currentSize = scrollPosition;  
            int nextLimit = currentSize + 10;  
  
            while (currentSize - 1 < nextLimit) {  
                rowsArrayList.add("Item " + currentSize);  
                currentSize++;  
            }  
            recyclerViewAdapter.notifyDataSetChanged();  
            isLoading = false;  
        }  
    }, delayMillis: 2000);  
}
```

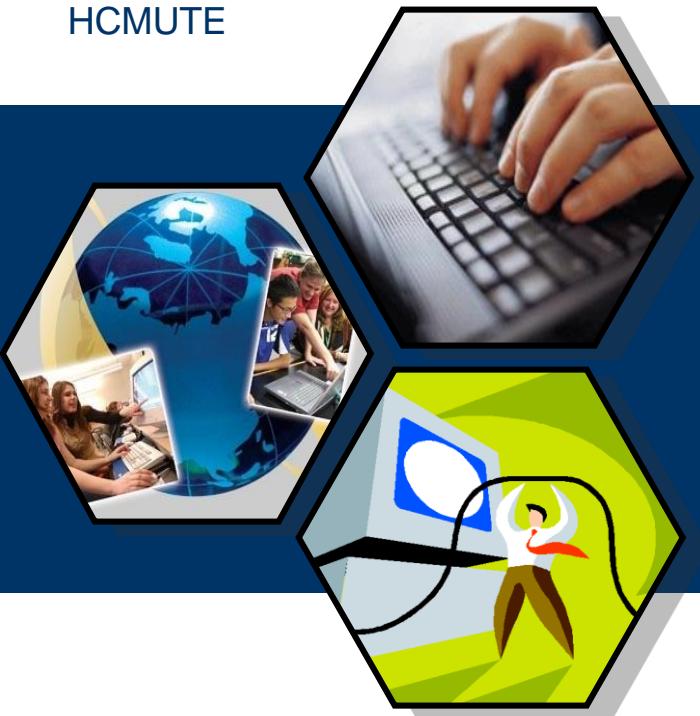




HCMUTE

KHOA CÔNG NGHỆ THÔNG TIN

UTEx

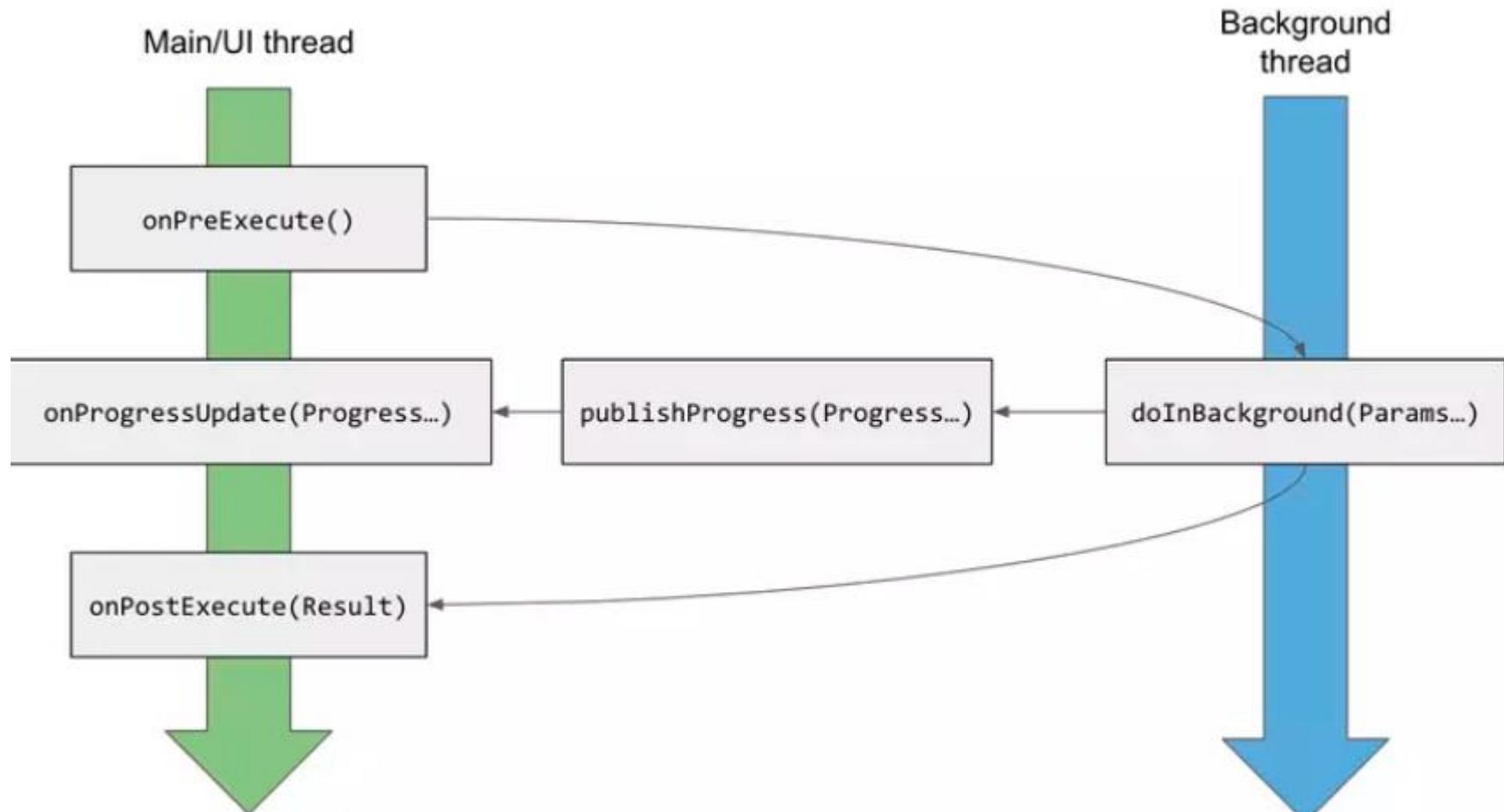


Xử Lý Đa Tiến Trình Trong Android Bằng AsyncTask

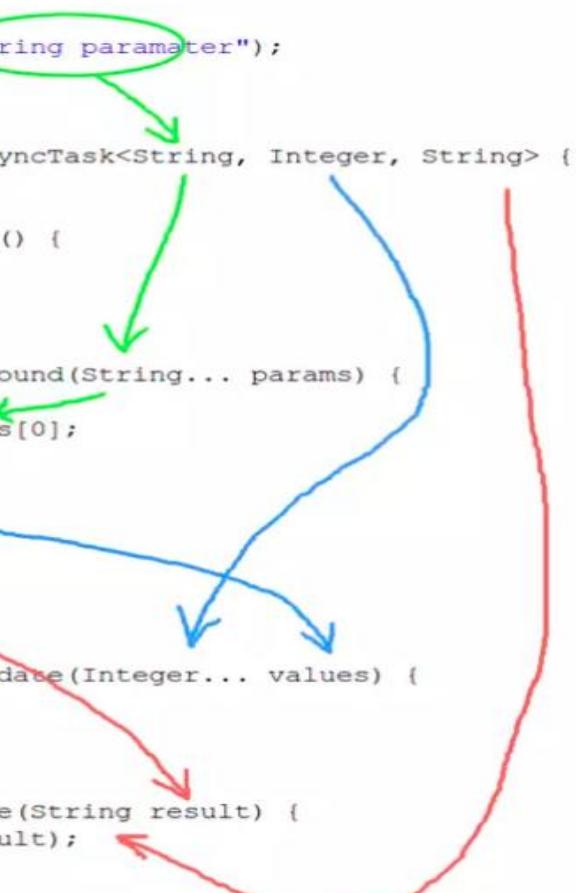


- *AsyncTask* là phương tiện khác để xử lý công việc sử dụng *background thread* và giao tiếp với *UI thread* mà không dùng *Thread* hay *Handler*
- Trong **AsyncTask<Params, Progress, Result>** có 3 đối số là các Generic Type:
 - **Params:** Là giá trị ((biến) được truyền vào khi gọi thực thi tiến trình và nó sẽ được truyền vào **doInBackground**
 - **Progress:** Là giá trị (biến) dùng để update giao diện điện thoại lúc tiến trình thực thi, biến này sẽ được truyền vào hàm **onProgressUpdate**.
 - **Result:** Là biến dùng để lưu trữ kết quả trả về sau khi tiến trình thực hiện xong.
- Những đối số nào không sử dụng trong quá trình thực thi tiến trình thì ta thay bằng Void.

- Thông thường trong 1 *AsyncTask* sẽ chứa 4 hàm, đó là :
 - **onPreExecute()** : Tự động được gọi đầu tiên khi tiến trình được kích hoạt.
 - **doInBackground()**: Được thực thi trong quá trình tiến trình chạy nền, thông qua hàm này để ta gọi hàm *onProgressUpdate* để cập nhật giao diện (gọi lệnh *publishProgress*). Ta không thể cập nhật giao diện trong hàm *doInBackground()*.
 - **onProgressUpdate ()**: Dùng để cập nhật giao diện lúc runtime
 - **onPostExecute()**: Sau khi tiến trình kết thúc thì hàm này sẽ tự động xảy ra. Ta có thể lấy được kết quả trả về sau khi thực hiện tiến trình kết thúc ở đây.
- Trong 4 hàm trên thì hàm **doInBackground()** bắt buộc phải tồn tại, còn các hàm khác có thể khuyết, nhưng các bạn nên sử dụng đầy đủ 4 hàm đã nêu.



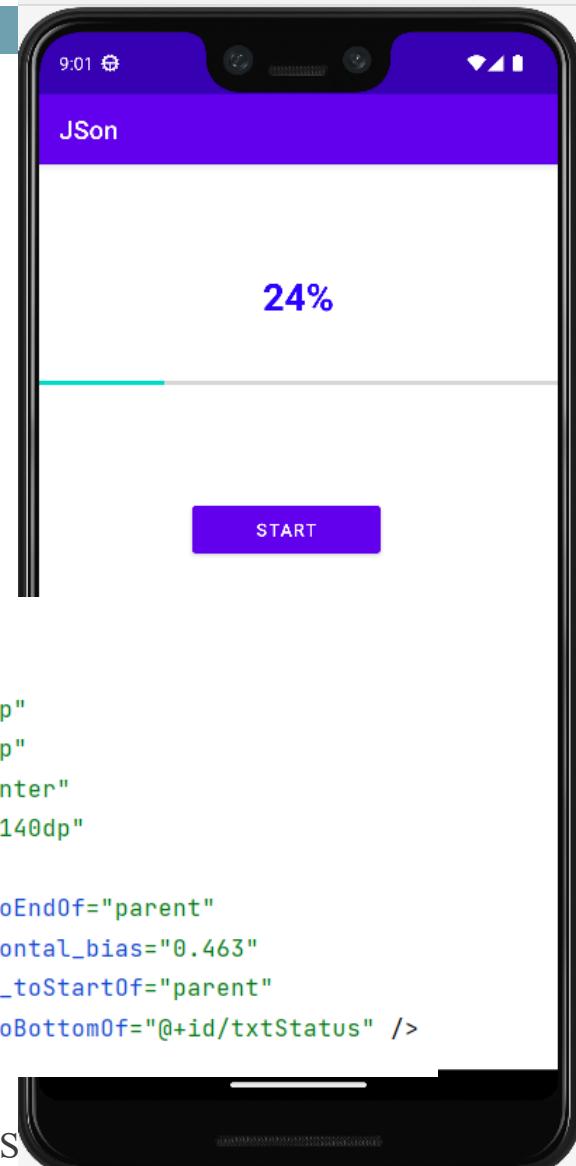
```
public class AsyncTaskTestActivity extends Activity {  
  
    @Override  
    public void onCreate(Bundle savedInstanceState) {  
        ...  
  
        new MyTask().execute("my string parameter");  
    }  
  
    private class MyTask extends AsyncTask<String, Integer, String> {  
  
        @Override  
        protected void onPreExecute() {  
        }  
  
        @Override  
        protected String doInBackground(String... params) {  
            String myString = params[0];  
  
            int i = 0;  
            publishProgress(i);  
  
            return "some string";  
        }  
  
        @Override  
        protected void onProgressUpdate(Integer... values) {  
        }  
  
        @Override  
        protected void onPostExecute(String result) {  
            super.onPostExecute(result);  
        }  
    }  
}
```



- Khi ấn nút start mình sẽ sử dụng `AsyncTask` để load progressbar và cập nhật `textview` ở phía trên.
- Bước 1: Thiết kế giao diện

```
<TextView  
    android:id="@+id/txtStatus"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:layout_gravity="center"  
    android:layout_marginTop="84dp"  
    android:text="0%"  
    android:textColor="#2c02ff"  
    android:textSize="30sp"  
    android:textStyle="bold"  
    app:layout_constraintEnd_toEndOf="parent"  
    app:layout_constraintStart_toStartOf="parent"  
    app:layout_constraintTop_toTopOf="parent" />  
  
<ProgressBar  
    android:id="@+id/prbDemo"  
    style="@style/Base.Widget.AppCompat.ProgressBar.Horizontal"  
    android:layout_width="match_parent"  
    android:layout_height="10dp"  
    android:layout_marginTop="44dp"  
    app:layout_constraintEnd_toEndOf="parent"  
    app:layout_constraintStart_toStartOf="parent"  
    app:layout_constraintTop_toBottomOf="@+id/txtStatus" />
```

```
<Button  
    android:id="@+id/btnStart"  
    android:layout_width="150dp"  
    android:layout_height="50dp"  
    android:layout_gravity="center"  
    android:layout_marginTop="140dp"  
    android:text="Start"  
    app:layout_constraintEnd_toEndOf="parent"  
    app:layout_constraintHorizontal_bias="0.463"  
    app:layout_constraintStart_toStartOf="parent"  
    app:layout_constraintTop_toBottomOf="@+id/txtStatus" />
```



□ Bước 2: Viết Class MyAsyncTask.java

```
public class MyAsyncTask extends AsyncTask<Void, Integer, Void> {
    Activity contextParent;
    //tạo constructor
    public MyAsyncTask(Activity contextParent) {
        this.contextParent = contextParent;
    }
    //Implement Method
    @Override
    protected void onPreExecute() {
        super.onPreExecute();
        //Hàm này sẽ chạy đầu tiên
        //Ở đây mình sẽ thông báo
        Toast.makeText(contextParent, "Đang làm việc...", Toast.LENGTH_SHORT).show();
    }
    @Override
    protected Void doInBackground(Void aVoid) {
        //Hàm được thực hiện tiếp
        //Hàm này thực hiện các tác vụ
        //Tuyệt đối không giao tiếp với UI
        for (int i = 0; i <= 100; i++) {
            SystemClock.sleep(10);
            //khi gọi hàm này thì
            publishProgress(i);
        }
        return null;
    }
    @Override
    protected void onProgressUpdate(Integer... values) {
        //Hàm thực hiện update giao diện khi có dữ liệu từ hàm doInBackground gửi xuống
        super.onProgressUpdate(values);
        //Qua contextParent để lấy được control trong MainActivity
        ProgressBar progressBar = (ProgressBar) contextParent.findViewById(R.id.prbDemo);
        //vì publishProgress chỉ truyền 1 đối số
        //nên mang values chỉ có 1 phần tử
        int number = values[0];
        //tăng giá trị của Progressbar lên
        progressBar.setProgress(number);
        //đóng thời gian hiển thị giá trị là % lên TextView
        TextView textView = (TextView) contextParent.findViewById(R.id.txtStatus);
        textView.setText(number + "%");
    }
    @Override
    protected void onPostExecute(Void aVoid) {
        super.onPostExecute(aVoid);
        //Hàm này được thực hiện khi tiến trình kết thúc
        //Ở đây mình thông báo là đã "Finished" để người dùng biết
        Toast.makeText(contextParent, "Đã hoàn thành", Toast.LENGTH_SHORT).show();
    }
}
```

□ Bước 2: Viết Class MyAsyncTask.java

```
@Override  
protected void onProgressUpdate(Integer... values) {  
    //Hàm thực hiện update giao diện khi có dữ liệu từ hàm doInBackground gửi xuống  
    super.onProgressUpdate(values);  
    //Thông qua contextCha để lấy được control trong MainActivity  
    ProgressBar progressBar = (ProgressBar) contextParent.findViewById(R.id.prbDemo);  
    //vì publishProgress chỉ truyền 1 đối số  
    //nên mang values chỉ có 1 phần tử  
    int number = values[0];  
    //tăng giá trị của Progressbar lên  
    progressBar.setProgress(number);  
    //đồng thời hiển thị giá trị là % lên TextView  
    TextView textView = (TextView) contextParent.findViewById(R.id.txtStatus);  
    textView.setText(number + "%");  
}  
  
@Override  
protected void onPostExecute(Void aVoid) {  
    super.onPostExecute(aVoid);  
    //Hàm này được thực hiện khi tiến trình kết thúc  
    //Ở đây mình thông báo là đã "Finshed" để người dùng biết  
    Toast.makeText(contextParent, text: "Đã hoàn thành, Finished", Toast.LENGTH_SHORT).show();  
}
```

Demo *AsyncTask*

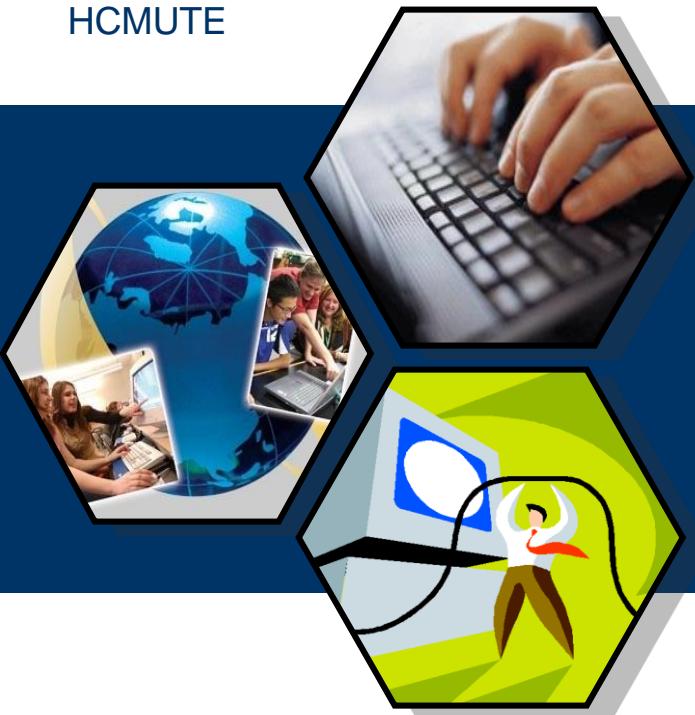
□ *Bước 3: Bắt sự kiện cho nút Start*

```
public class AsyncTaskActivity extends AppCompatActivity {
    Button btnStart;
    MyAsyncTask myAsyncTask;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_asynctask);
        btnStart = (Button) findViewById(R.id.btnStart);
        btnStart.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                //Khởi tạo tiến trình của bạn
                //Truyền Activity chính là MainActivity sang bên tiến trình của mình
                myAsyncTask = new MyAsyncTask(contextParent: AsyncTaskActivity.this);
                //Gọi hàm execute để kích hoạt tiến trình
                myAsyncTask.execute();
            }
        });
    }
}
```



HCMUTE

KHOA CÔNG NGHỆ THÔNG TIN



JSON và kết nối API

**Khoa Công nghệ Thông tin
Đại học Sư phạm Kỹ thuật TP.HCM**

ThS. Nguyễn Hữu Trung



JSON là gì?

360

- JSON (JavaScript Object Notation) là một kiểu định dạng dữ liệu tuân theo một quy luật nhất định mà hầu hết các ngôn ngữ lập trình hiện nay đều có thể đọc được. JSON là một tiêu chuẩn mở để trao đổi dữ liệu trên web.

XML

```
<empinfo>
  <employees>
    <employee>
      <name>James Kirk</name>
      <age>40</age>
    </employee>
    <employee>
      <name>Jean-Luc Picard</name>
      <age>45</age>
    </employee>
    <employee>
      <name>Wesley Crusher</name>
      <age>27</age>
    </employee>
  </employees>
</empinfo>
```

JSON

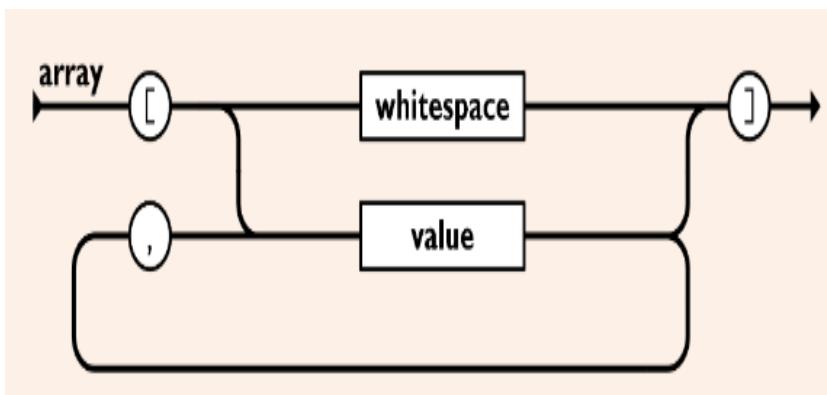
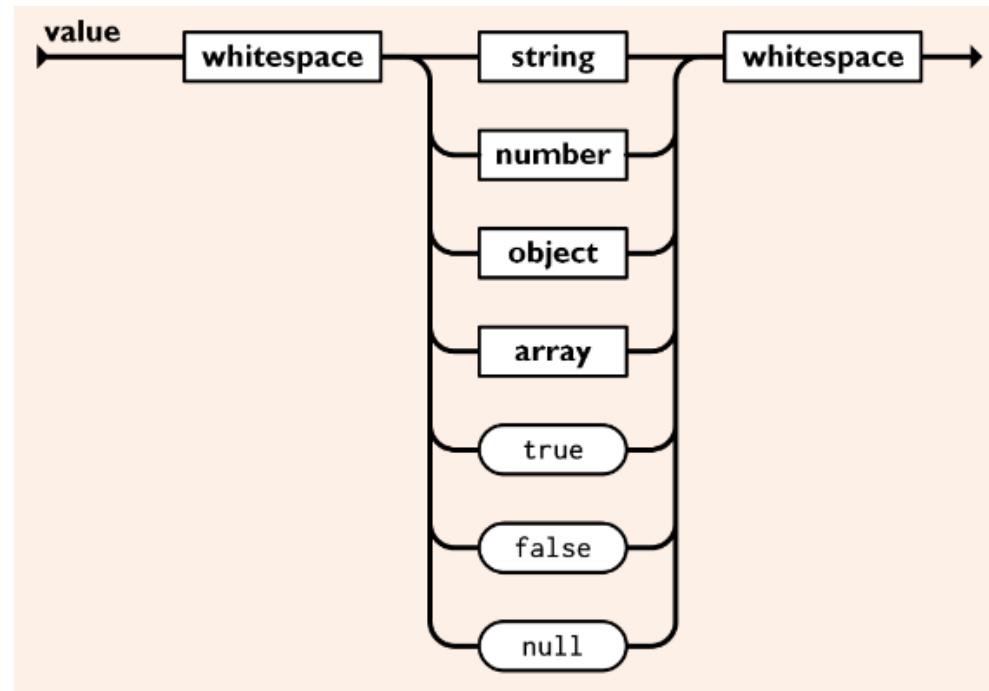
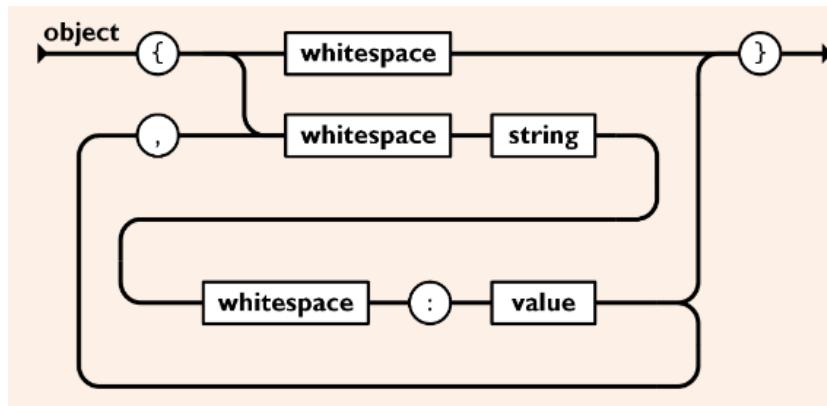
```
{ "empinfo" :
  {
    "employees" : [
      {
        "name" : "James Kirk",
        "age" : 40,
      },
      {
        "name" : "Jean-Luc Picard",
        "age" : 45,
      },
      {
        "name" : "Wesley Crusher",
        "age" : 27,
      }
    ]
  }
}
```

- JSON sử dụng các cặp *key – value* để định dạng. Nó hỗ trợ các cấu trúc dữ liệu như đối tượng và mảng. Ví dụ một tập tin có tên `iotstar.json` với nội dung như ở dưới đây sử dụng format kiểu JSON để lưu trữ thông tin:

```
{ "name" : "iotstar",
  "title" : "Nguyễn Hữu Trung"
}
```

- Chuỗi JSON được bao lại bởi dấu ngoặc nhọn {}
- Các key, value của JSON bắt buộc phải đặt trong dấu nháy kép "{}".
- Nếu có nhiều dữ liệu (nhiều cặp key => value) thì ta dùng dấu phẩy (,) để ngăn cách
- Hai thư viện Json phổ biến: Gson và JackSon

- Truy cập trang json.org xem cấu trúc JSON (Javascript Object Notation)



- Tìm json formatter & Validator

JSON Data/URL

```
{"name": "Java",  
 "desc": "Java 1"}
```

Formatted JSON Data

```
{  
    "name": "Java",  
    "desc": "Java 1"  
}
```

□ Tải file data.json về bỏ vào web

← → ⌛ Not secure | 192.168.31.214:8090/data.json

 Android - Login Scr...  Kotlin Tutorial  Setting Up Swagger...

```
{  
    "name": "Java",  
    "desc": "Java 1"  
}
```

Network Connection Details

Network Connection Details:

Property	Value
Connection-specific DNS S...	
Description	Realtek RTL8188EU Wireless LAN 802.11n L
Physical Address	50-3E-AA-A3-2C-61
DHCP Enabled	Yes
IPv4 Address	192.168.31.214
IPv4 Subnet Mask	255.255.255.0
Lease Obtained	Friday, January 13, 2023 6:45:45 PM
Lease Expires	Saturday, January 14, 2023 6:45:39 AM
IPv4 Default Gateway	192.168.31.1
IPv4 DHCP Server	192.168.31.1
IPv4 DNS Server	192.168.31.1
IPv4 WINS Server	
NetBIOS over Tcpip Enabl...	Yes
Link-local IPv6 Address	fe80::5b2d:3aef:36cc:2856%41
IPv6 Default Gateway	
IPv6 DNS Server	

< >

Close

CN

- Khởi tạo dự án Empty Android. Tạo Class ReadJSON

//Tạo Class đọc Json

```
private class ReadJSONObject extends AsyncTask<String,Void,String>{  
    @Override  
    protected String doInBackground(String... strings) {  
        StringBuilder content = new StringBuilder();  
        try {  
            URL url = new URL(strings[0]);  
            InputStreamReader inputStreamReader = new  
InputStreamReader(url.openConnection().getInputStream());  
            BufferedReader bufferedReader = new BufferedReader(inputStreamReader);  
            String line = "";  
            while ((line = bufferedReader.readLine()) != null){  
                content.append(line);  
            }  
            bufferedReader.close();  
        } catch (MalformedURLException e) {  
            e.printStackTrace();  
        } catch (IOException e) {  
            e.printStackTrace();  
        }  
        return content.toString();  
    }  
}
```

@Override

```
protected void onPostExecute(String s) {  
    super.onPostExecute(s);  
    //phân tích JSON  
    try {  
        JSONObject object = new JSONObject(s);  
        String name = object.getString("name");  
        String desc = object.getString("desc");  
        String pic = object.getString("pic");  
  
        String kq = name + "\n" + desc + "\n" + pic;  
        Toast.makeText(MainActivity.this,kq,Toast.LENGTH_SHORT).show();  
  
    } catch (JSONException e) {  
        e.printStackTrace();  
    }  
}
```

JSON Data/URL

```
{  
    "monhoc": [  
        {  
            "name": "Java", "desc": "Java 1", "pic": "pic1.png"},  
        {  
            "name": "C#", "desc": "C# 1", "pic": "pic2.png"},  
        {  
            "name": "Kotlin", "desc": "Kotlin 1", "pic": "pic3.png"},  
        {  
            "name": "PHP", "desc": "PHP 1", "pic": "pic4.png"},  
        {  
            "name": "Dart", "desc": "Dart 1", "pic": "pic5.png"}  
    ]  
}
```

```
new ReadJSONObject().execute("http://app.iotstar.vn/json/data3.json");
```

@Override

```
protected void onPostExecute(String s) {
    super.onPostExecute(s);
    //phân tích JSON
    try {
        JSONObject object = new JSONObject(s);
        //xử lý mảng
        JSONArray array = object.getJSONArray("monhoc");
        //đọc các phần tử trong mảng
        for(int i=0;i<array.length();i++){
            JSONObject object1 = array.getJSONObject(i);
            String name = object1.getString("name");
            String desc = object1.getString("desc");
            String pic = object1.getString("pic");
            String kq = name + "\n" + desc + "\n" + pic;
            Toast.makeText(MainActivity.this,kq,Toast.LENGTH_SHORT).show();
        }
    } catch (JSONException e) {
        e.printStackTrace();
    }
}
```

@Override

```
protected void onPostExecute(String s) {
    super.onPostExecute(s);
    //phân tích JSON
    try {
        JSONObject object = new JSONObject(s);
        JSONObject objectLang = object.getJSONObject("language");
        JSONObject objectVN = objectLang.getJSONObject("vn");
        String ten = objectVN.getString("ten");

        Toast.makeText(MainActivity.this,ten,Toast.LENGTH_SHORT).show();
    }
} catch (JSONException e) {
    e.printStackTrace(); }
```

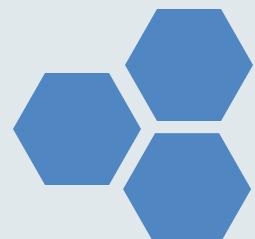
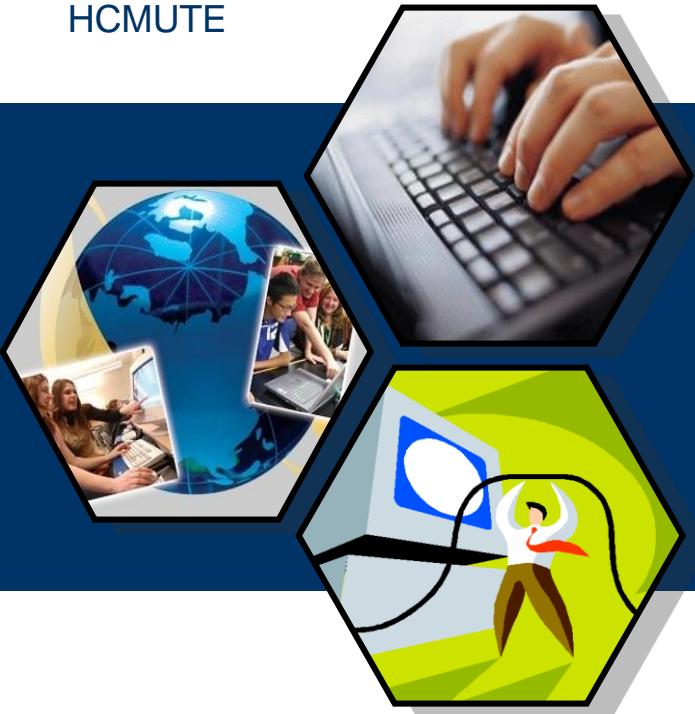


HCMUTE

KHOA CÔNG NGHỆ THÔNG TIN

UTEx

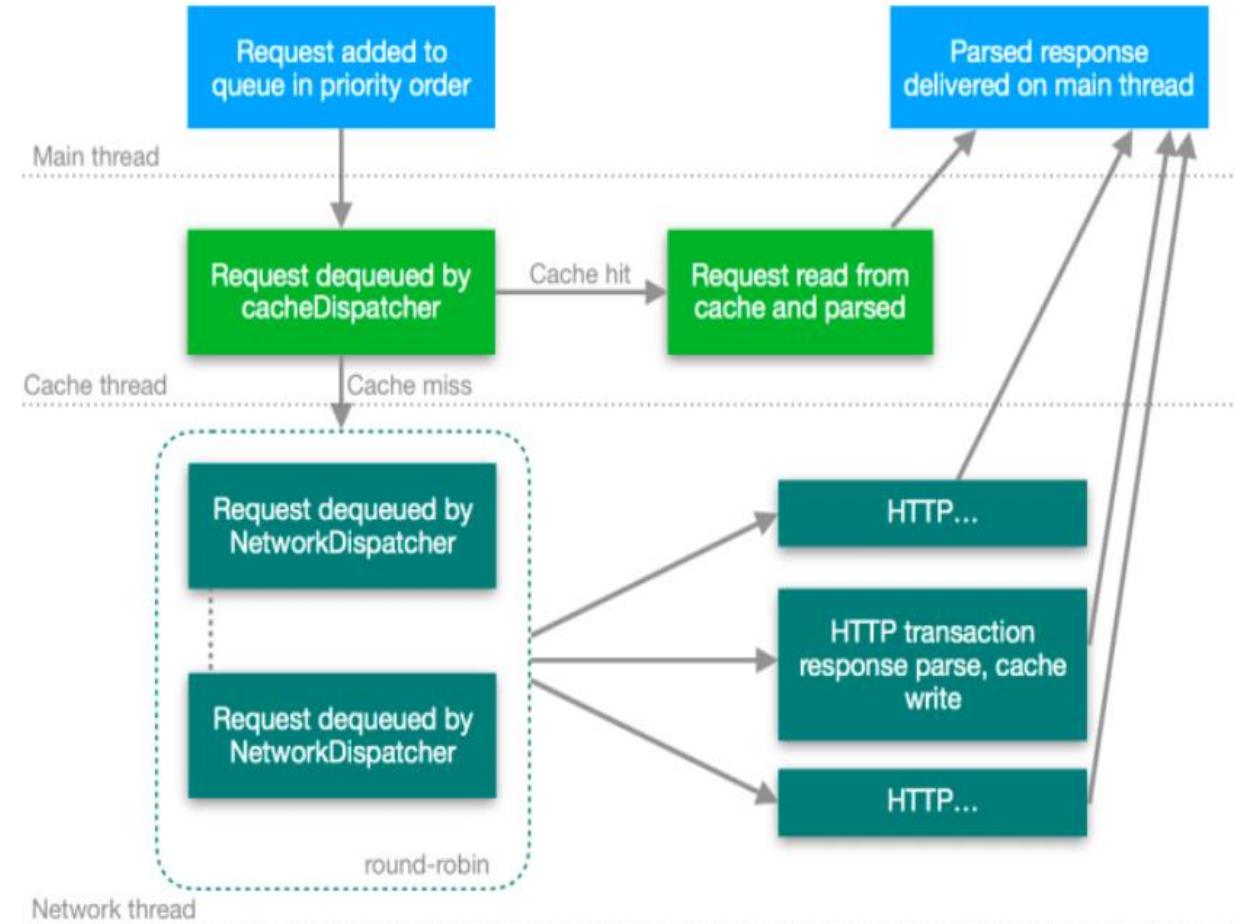
Kết nối API với thư viện Volley



- Volley library Android là thư viện networking cho các dự án Android. Volley được dùng để quản lý các network request, giúp cho developer đơn giản hóa việc thực hiện kết nối và xử lý kết quả trả về từ server.
- Volley hỗ trợ đầy đủ các HTTP request như GET, POST, PUT, DELETE. Ngoài ra, các thư viện network kiểu như Volley có tính năng network cache rất hữu ích.
- Tất cả request được cache nhằm đảm bảo khi reload, sẽ có kết quả nhanh hơn. Ngoài ra, response được lưu trong memory nên với các response có dữ liệu như JSON, Image, String... thì rất hiệu quả.

□ Một số tính năng nổi bật của Volley:

- Hỗ trợ lên schedule để tạo request
- Cho phép cùng lúc thực hiện nhiều request trên các thread khác nhau theo độ ưu tiên.
- Cache trên Disk hay RAM
- Cho phép hủy một request.
- Dễ dàng tùy chỉnh cho phù hợp với yêu cầu của ứng dụng. Như thiết lập retry, back off.
- Dễ debug.



□ Các class sử dụng trong Volley:

▫ **RequestQueue**: Là hằng đợi giữ các Request.

`RequestQueue queue = Volley.newRequestQueue(this);`

▫ **Request**: là lớp cơ sở của các Request trong Volley, chưa thông tin về request HTTP.

▫ **StringRequest**: Kết thừa từ Request, là class đại diện cho request trả về String.

▫ **JSONObjectRequest**: Là HTTP request có kết quả trả về là JSONObject.

▫ **JSONArrayRequest**: Là HTTP request có kết quả trả về là JSONArray.

▫ **ImageRequest**: Là HTTP request có kết quả trả về là Bitmap.

Forming a Request

374

- Hầu hết những Request trong Volley đều sử dụng Constructor giống như bên dưới. Các tham số trong Constructor:
 - **RequestMethod**: GET, POST, DELETE...
 - **JSONObject** **: Một Object tùy chọn sẽ được gửi cùng request của bạn.
 - **ResponseListener** : Nơi bạn nhận dữ liệu trả về từ server khi request hoàn thành.
 - **ErrorListener** **: Nơi bạn nhận lại những problem bên trong request của bạn từ server.

```
JsonObjectRequest request  
JsonObjectRequest(Requestmethod,  
url, null, newResponseListener(), new ErrorListener());
```

```
private class ResponseListener implements Response.Listener{
    @Override
    public void onResponse(JSONObject response){
    }
}
```

```
private class ErrorListener implements Response.ErrorListener{
    @Override
    public void onErrorResponse(VolleyError error){
    }
}
```

- ☐ Nếu bạn chọn add nhiều Request tới RequestQueue tại một thời điểm, bạn cũng có thể xét độ ưu tiên cho mỗi request để bạn có thể nhận được các thông tin quan trọng trước. Thật không may Volley không có phương thức setPriority(). Tuy nhiên bạn có thể Custom mở rộng request class và override hai phương thức getPriority() và setPriority() như ví dụ bên dưới:

```
public class CustomStringRequest extends StringRequest {  
    private Priority priority = Priority.LOW;  
    CustomStringRequest(String url, Listener<String> listener, ErrorListener  
errorListener){  
        super(url, listener, errorListener);  
    }  
    @Override  
    public Priority getPriority(){  
        return priority;  
    }  
    @Override  
    public void setPriority(Priority priority){  
        this.priority= priority;  
    }  
}
```

- Lớp trên mặc định Priority là LOW . Tuy nhiên trong Activity bạn có thể xét lại nó như bên dưới:

```
Request.setPriority(Priority.IMMEDIATE);
```

Utilizing the Response Cache: Cache là thực hiện tự động trong các request như RequestJSon hoặc các request khác mở rộng từ Volley Request class . Tuy nhiên bạn có thể tắt cache bằng dòng lệnh như bên dưới trước khi bạn add request tới RequestQueue :

```
request.setShouldCache(false);
```

Accessing the cache: Sau khi request được hoàn thành và dữ liệu được lấy về. Bạn có thể truy cập vào cache bởi 1 trong 2 cách :

```
request.getCacheEntry().data
```

```
queue.getCache().get(url).data
```

```
queue.getCache().remove(url);
queue.getCache().clear();
queue.getCache().invalidate(url, true);
queue.getCache().get(url).serverDate
```

- Ví dụ:

```
if(queue.getCache().get(url)!=null){  
    //response exists  
    String cachedResponse = new String(queue.getCache().get(url).data);  
    results.setText("From Cache: " + cachedResponse);  
}else{  
    //no response  
    queue.add(stringRequest);  
}
```

□ Sử dụng thư viện Volley library Android

Thêm dependencies vào **build.gradle (Module App)**

```
//thư viện load dữ liệu API  
implementation 'com.android.volley:volley:1.2.1'
```

Thêm permission INTERNET vào manifest

```
<uses-permission android:name="android.permission.INTERNET" />  
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
```

Trong thẻ Application của AndroidManifest.xml thêm lệnh sau:

```
android:usesCleartextTraffic="true"
```

- Bước 0: Thêm thư viện
- Bước 1: Thiết kế giao diện Login, Profile
- Bước 2: Tạo class Model
- Bước 3: Tạo Class Contants để lưu đường dẫn API
- Bước 4: Tạo Class VolleySingle để khởi tạo Request
- Bước 5: Tạo class SharedPrefManager để lưu dữ liệu User
- Bước 6: Viết LoginActivity
- Bước 7: Viết ProfileActivity

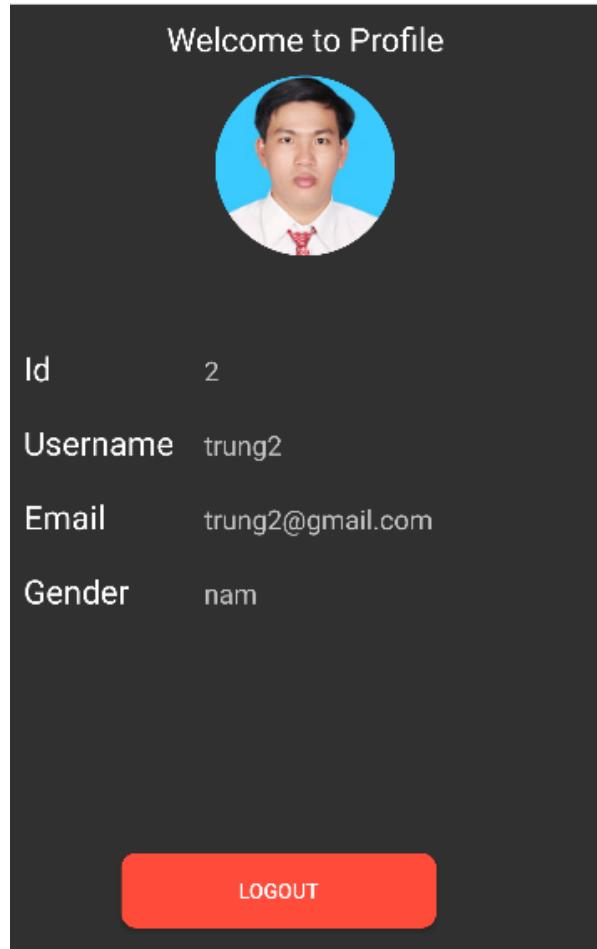
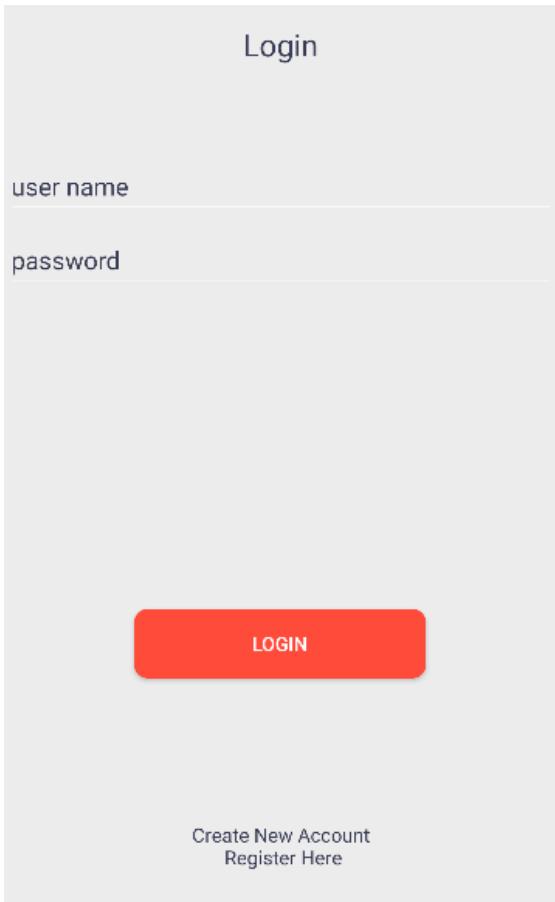
□ Bước 0: Thêm thư viện

```
//thu viện load image
implementation 'com.github.bumptech.glide:glide:4.14.2'
annotationProcessor 'com.github.bumptech.glide:compiler:4.14.2'
//thu viện load dữ liệu API
implementation 'com.android.volley:volley:1.2.1'
//thư viện circle images
//bo goc tron cho ImageView
implementation 'de.hdodenhof:circleimageview:3.1.0'
```

Thêm permission INTERNET vào manifest

```
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
```

□ Bước 1: Thiết kế giao diện Login, Profile



- Login: các em dùng ConstraintLayout để thiết kế hoặc sử dụng lại giao diện bài tập tuần 4.
- Profile: các em dùng ConstraintLayout, LinearLayout và TableLayout để thiết kế

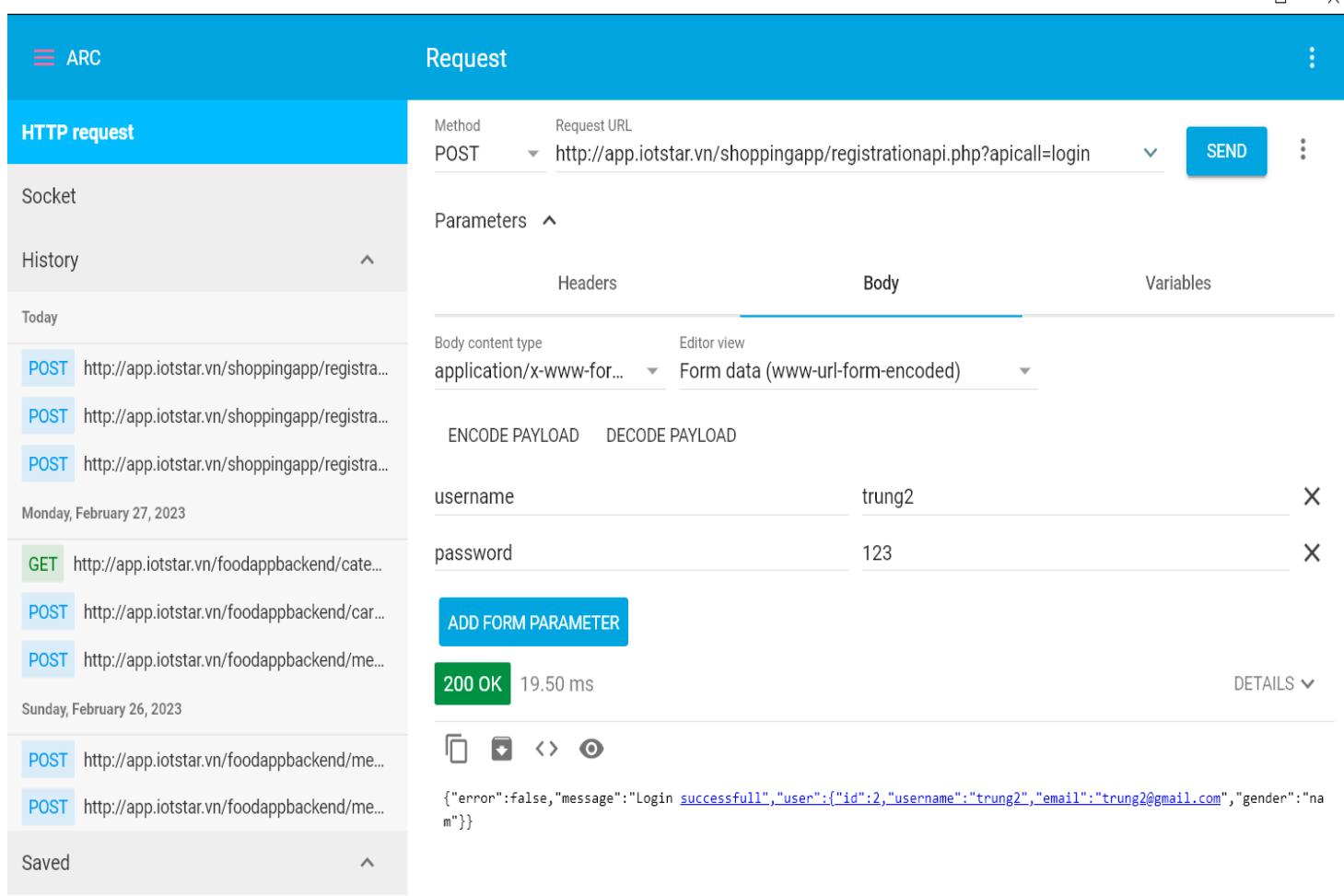
- Bước 2: Tạo class Model, nhớ tạo constructor và getter, setter

```
public class User implements Serializable {  
    private int id;  
    private String name, email, gender, images;
```

- Bước 3: Tạo Class Contants để lưu đường dẫn API

```
public class contants {  
    public static String localhost = "app.iotstar.vn";  
    //login và register API  
    private static final String ROOT_URL = "http://" + localhost +  
    "/shoppingapp/registrationapi.php?apicall=";  
    public static final String URL_REGISTER = ROOT_URL + "signup";  
    public static final String URL_LOGIN= ROOT_URL + "login";  
}
```

❑ Bước 3: Tạo Class Contants để lưu đường dẫn API



The screenshot shows the ARC (Advanced REST Client) interface. On the left, there's a sidebar with 'HTTP request' selected, showing a list of previous requests. The main area is titled 'Request' and contains the following details:

- Method:** POST
- Request URL:** <http://app.iotstar.vn/shoppingapp/registrationapi.php?apicall=login>
- Parameters:** Headers, Body, Variables
- Body content type:** application/x-www-form-urlencoded
- Editor view:** Form data (www-url-form-encoded)
- Payload:** ENCODE PAYLOAD, DECODE PAYLOAD
- Form Parameters:** username: trung2, password: 123
- Buttons:** ADD FORM PARAMETER, DETAILS
- Status:** 200 OK, 19.50 ms
- Response:** {"error":false,"message":"Login [successful](#)","user":{"id":2,"username":"trung2","email":"trung2@gmail.com","gender":"na m"}}

□ Bước 4: Tạo Class VolleySingle để khởi tạo Request

```
public class VolleySingle {  
    private static VolleySingle mInstance;  
    private RequestQueue mRequestQueue;  
    private static Context mCtx;  
    //Khởi tạo constructor  
    private VolleySingle(Context context) {  
        mCtx = context;  
        mRequestQueue = getRequestQueue();  
    }  
    //hàm lấy context  
    public static synchronized VolleySingle getInstance(Context context) {  
        if (mInstance == null) {  
            mInstance = new VolleySingle(context);  
        }  
        return mInstance;  
    }  
    //Hàm RequestQueue  
    public RequestQueue getRequestQueue() {  
        if (mRequestQueue == null) {  
            // getApplicationContext() is key, it keeps you from leaking the  
            // Activity or BroadcastReceiver if someone passes one in.  
            mRequestQueue = Volley.newRequestQueue(mCtx.getApplicationContext());  
        }  
        return mRequestQueue;  
    }  
    //Hàm addtoRequest|  
    public <T> void addToRequestQueue(Request<T> req) {  
        getRequestQueue().add(req);  
    }  
}
```

□ Bước 5: Tạo class SharedPrefManager để lưu dữ liệu User

```
public class SharedPrefManager {  
    //khởi tạo các hằng key  
    private static final String SHARED_PREF_NAME = "volleyregisterlogin";  
    private static final String KEY_USERNAME = "keyusername";  
    private static final String KEY_EMAIL = "keyemail";  
    private static final String KEY_GENDER = "keygender";  
    private static final String KEY_ID = "keyid";  
    private static final String KEY_IMAGES = "keyimages";  
    private static SharedPrefManager mInstance;  
    private static Context ctx;  
    //khởi tạo constructor  
    private SharedPrefManager(Context context) {  
        ctx = context;  
    }  
    public static synchronized SharedPrefManager getInstance(Context context) {  
        if (mInstance == null) {  
            mInstance = new SharedPrefManager(context);  
        }  
        return mInstance;  
    }  
}
```

□ Bước 5: Tạo class SharedPrefManager để lưu dữ liệu User

```
//this method will store the user data in shared preferences
public void userLogin(User user) {
    SharedPreferences sharedPreferences = ctx.getSharedPreferences(SHARED_PREF_NAME, Context.MODE_PRIVATE);
    SharedPreferences.Editor editor = sharedPreferences.edit();
    editor.putInt(KEY_ID, user.getId());
    editor.putString(KEY_USERNAME, user.getName());
    editor.putString(KEY_EMAIL, user.getEmail());
    editor.putString(KEY_GENDER, user.getGender());
    editor.putString(KEY_IMAGES, user.getImages());
    editor.apply();
}

//this method will checker whether user is already logged in or not
public boolean isLoggedIn() {
    SharedPreferences sharedPreferences = ctx.getSharedPreferences(SHARED_PREF_NAME, Context.MODE_PRIVATE);
    return sharedPreferences.getString(KEY_USERNAME, defaultValue: null) != null;
}

//this method will give the logged in user
public User getUser() {
    SharedPreferences sharedPreferences = ctx.getSharedPreferences(SHARED_PREF_NAME, Context.MODE_PRIVATE);
    return new User(
        sharedPreferences.getInt(KEY_ID, defaultValue: -1),
        sharedPreferences.getString(KEY_USERNAME, defaultValue: null),
        sharedPreferences.getString(KEY_EMAIL, defaultValue: null),
        sharedPreferences.getString(KEY_GENDER, defaultValue: null),
        sharedPreferences.getString(KEY_IMAGES, defaultValue: null)
    );
}
```

}

- Bước 5: Tạo class SharedPrefManager để lưu dữ liệu User

```
//this method will logout the user
public void logout() {
    SharedPreferences sharedpreferences = ctx.getSharedPreferences(SHARED_PREF_NAME, Context.MODE_PRIVATE);
    SharedPreferences.Editor editor = sharedpreferences.edit();
    editor.clear();
    editor.apply();
    ctx.startActivity(new Intent(ctx, LoginActivity.class));
}
```

□ Bước 6: Viết LoginActivity

```
public class LoginActivity extends AppCompatActivity {  
    EditText etName, etPassword;  
    ProgressBar progressBar;  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_login);  
        if (SharedPrefManager.getInstance(this).isLoggedIn()) {  
            finish();  
            startActivity(new Intent(packageContext: this, MainActivity.class));  
        }  
        progressBar = findViewById(R.id.progressBar);  
        etName = findViewById(R.id.etUserName);  
        etPassword = findViewById(R.id.etUserPassword);  
        //calling the method userLogin() for login the user  
        findViewById(R.id.btnLogin).setOnClickListener(new View.OnClickListener() {  
            @Override  
            public void onClick(View view) { userLogin(); }  
        });  
        //if user presses on textView not register calling RegisterActivity  
        findViewById(R.id.tvRegister).setOnClickListener(new View.OnClickListener() {  
            @Override  
            public void onClick(View view) {  
                finish();  
                startActivity(new Intent(getApplicationContext(), RegisterActivity.class));  
            }  
        });  
    }  
}
```

□ Bước 6: Viết LoginActivity

```
private void userLogin() {  
    //first getting the values  
    final String username = etName.getText().toString();  
    final String password = etPassword.getText().toString();  
    //validating inputs  
    if (TextUtils.isEmpty(username)) {  
        etName.setError("Please enter your username");  
        etName.requestFocus();  
        return;  
    }  
    if (TextUtils.isEmpty(password)) {  
        etPassword.setError("Please enter your password");  
        etPassword.requestFocus();  
        return;  
    }  
    //if everything is fine  
    StringRequest stringRequest = new StringRequest(Request.Method.POST, contants.URL_LOGIN,  
        new Response.Listener<String>() {  
            @Override  
            public void onResponse(String response) {  
                progressBar.setVisibility(View.GONE);  
                try {  
                    //converting response to json object  
                    JSONObject obj = new JSONObject(response);  
                    //if no error in response  
                    if (!obj.getBoolean( name: "error")) {  
                        Toast.makeText(getApplicationContext(), obj.getString( name: "message"), Toast.LENGTH_SHORT).show();  
                        //getting the user from the response  
                        JSONObject userJson = obj.getJSONObject("user");  
                }  
            }  
        }  
    );  
    RequestQueue requestQueue = Volley.newRequestQueue(this);  
    requestQueue.add(stringRequest);  
}
```

□ Bước 6: Viết LoginActivity

```
//creating a new user object
User user = new User(
    userJson.getInt( name: "id"),
    userJson.getString( name: "username"),
    userJson.getString( name: "email"),
    userJson.getString( name: "gender"),
    userJson.getString( name: "images")
);
//storing the user in shared preferences
SharedPrefManager.getInstance(getApplicationContext()).userLogin(user);
//starting the profile activity
finish();
Intent intent = new Intent( packageContext: LoginActivity.this, MainActivity.class);
startActivity(intent);
} else {
    Toast.makeText(getApplicationContext(), obj.getString( name: "message"), Toast.LENGTH_SHORT).show();
}
} catch (JSONException e) {
    e.printStackTrace();
}
},
new Response.ErrorListener() {
    @Override
    public void onErrorResponse(VolleyError error) {
        Toast.makeText(getApplicationContext(), error.getMessage(), Toast.LENGTH_SHORT).show();
    }
}
```

□ Bước 6: Viết LoginActivity

```
    @Override
    protected Map<String, String> getParams() throws AuthFailureError {
        Map<String, String> params = new HashMap<>();
        params.put("username", username);
        params.put("password", password);
        return params;
    }
}
VolleySingle.getInstance(this).addToRequestQueue(stringRequest);
}
```

□ Bước 7: Viết ProfileActivity

```
public class ProfilesActivity extends AppCompatActivity implements View.OnClickListener {
    TextView id,userName,userEmail,gender;
    Button btnLogout;
    ImageView imageViewpprofile;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_profiles);
        if(SharedPrefManager.getInstance(this).isLoggedIn()){
            id = findViewById(R.id.textViewId);
            userName = findViewById(R.id.textViewUsername);
            userEmail = findViewById(R.id.textViewEmail);
            gender = findViewById(R.id.textViewGender);
            btnLogout = findViewById(R.id.buttonLogout);
            imageViewpprofile = findViewById(R.id.imageViewProfile);
            User user = SharedPrefManager.getInstance(this).getUser();
            id.setText(String.valueOf(user.getId()));
            userEmail.setText(user.getEmail());
            gender.setText(user.getGender());
            userName.setText(user.getName());
            Glide.with(getApplicationContext()).RequestManager
                .load(user.getImages()).RequestBuilder<Drawable>
                .into(imageViewpprofile);
            btnLogout.setOnClickListener(this);
        }
        else{
            Intent intent = new Intent(packageContext: ProfilesActivity.this,LoginActivity.class);
            startActivity(intent);
            finish();
        }
    }
    @Override
    public void onClick(View v) {
        switch (v.getId()){
            case R.id.buttonLogout:
                SharedPrefManager.getInstance(this).logOut();
                Intent intent = new Intent(packageContext: ProfilesActivity.this,LoginActivity.class);
                startActivity(intent);
                finish();
                break;
        }
    }
}
```

□ Bước 7: Viết ProfileActivity

```
public void onClick(View view){  
    if(view.equals(btnLogout)){  
        SharedPrefManager.getInstance(getApplicationContext()).logout();  
    }  
}  
}
```

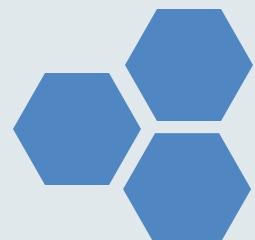
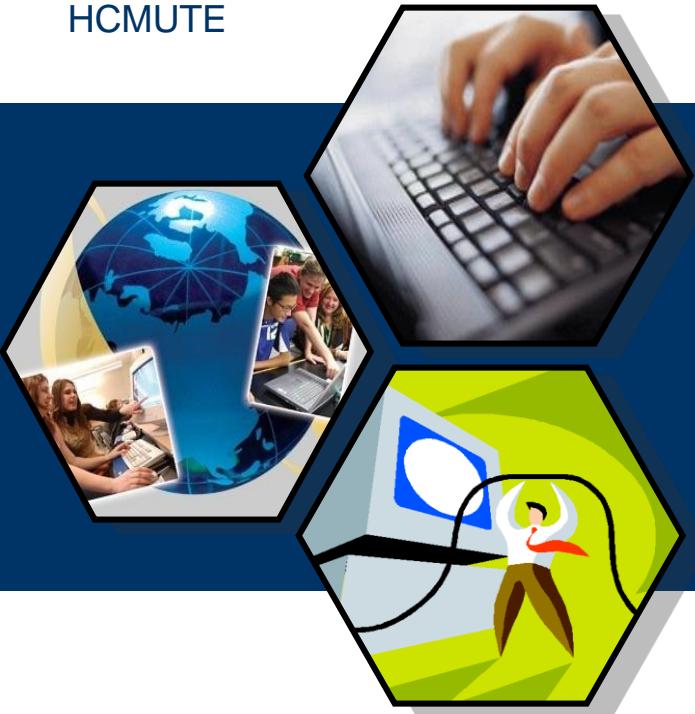


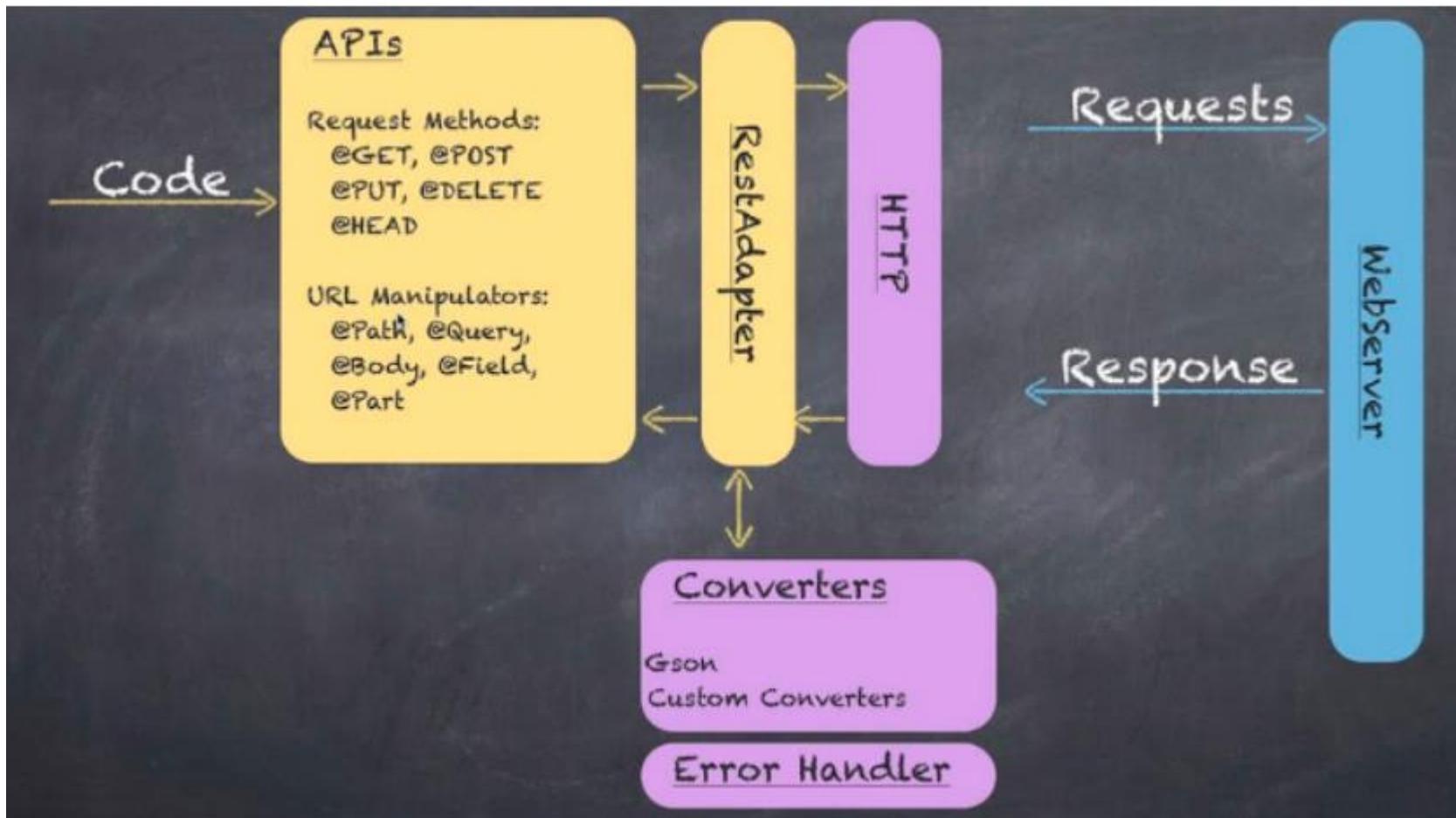
HCMUTE

KHOA CÔNG NGHỆ THÔNG TIN

UTEx

Kết nối API với thư viện Retrofit2





- Retrofit là một Rest Client theo chuẩn RESTFul cho Android và Java và được tạo ra bởi Square. Việc nhận và tải lên JSON (hoặc dữ liệu khác) với Retrofit một cách khá dễ dàng tới một WebService dựa trên mô hình REST.
- Các gói trang bị thêm cho phép sử dụng các bộ chuyển đổi sau đây:
 - **Gson: com.squareup.retrofit:converter-gson**
 - **Jackson: com.squareup.retrofit:converter-jackson**
 - **Moshi: com.squareup.retrofit:converter-moshi**
 - **Protobuf: com.squareup.retrofit:converter-protobuf**
 - **Wire: com.squareup.retrofit:converter-wire**
 - **Simple XML: com.squareup.retrofit:converter-simplexml**

- Để làm việc với **Retrofit** bạn cần triển khai cơ bản 3 lớp:
 - ❑ Model class to map JSON Data
 - ❑ Interfaces để định nghĩa các API cho Webservice
 - ❑ Retrofit.Builder Lớp để định nghĩa URL Endpoint cho các hoạt động liên quan tới Http

	One Discussion	Dashboard (7 requests)	25 Discussions
AsyncTask	941 ms	4,539 ms	13,957 ms
Volley	560 ms	2,202 ms	4,275 ms
Retrofit	312 ms	889 ms	1,059 ms

- Sử dụng Annotations để mô tả yêu cầu HTTP:
 - Hỗ trợ tham số URL và tham số truy vấn
 - Chuyển đổi đối tượng để yêu cầu nội dung
 - Multipart request body và file upload

- Api endpoint là API của server mà bạn cần sử dụng để yêu cầu dữ liệu từ server.
- Mỗi phương thức phải có Annotation HTTP cung cấp request method và URL. Có năm Annotation được tích hợp sẵn: [@GET](#), [@POST](#), [@PUT](#), [@DELETE](#) và [@HEAD](#) URL tương đối của tài nguyên được chỉ định trong Annotation.

[@GET\("users/list"\)](#)

- Bạn cũng có thể chỉ định tham số truy vấn trong URL.

[@GET\("users/list?sort=desc"\)](#)

```
public interface RetrofitClient {  
    @GET("/users/{user}/repos")  
    Call<List<Users>> getUser( @Path("user") String user );  
}
```

- URL request có thể được cập nhật tự động bằng cách sử dụng các khối thay thế và tham số trên phương thức.
- Chúng ta có thể sử dụng URL 1 cách động dựa vào biến truyền vào, bằng cách sử dụng annotation [@Path](#)

```
@GET("group/{id}/users")
Call<List<User>> groupList(@Path("id") int groupId);
```

- Chúng ta có thể nối thêm paramater vào sau URL bằng cách sử dụng [@Query](#)

```
@GET("group/{id}/users")
Call<List<User>> groupList(@Path("id") int groupId, @Query("sort") String sort);
```

- Đối với các kết hợp tham số truy vấn phức tạp, có thể sử dụng [@QueryMap](#).

```
@GET("group/{id}/users")
Call<List<User>> groupList(@Path("id") int groupId, @QueryMap Map<String,
String> options);
```

- Một đối tượng có thể được chỉ định để sử dụng làm phần thân yêu cầu HTTP với Annotation [@Body](#).

```
@POST("users/new")
Call<User> createUser(@Body User user);
```

- Đối tượng cũng sẽ được chuyển đổi bằng cách sử dụng **Converter** được chỉ định trên instance của Retrofit. Nếu không có Converter nào được thêm vào, chỉ có thể sử dụng **RequestBody**.

- Các phương thức cũng có thể được khai báo để gửi dữ liệu được mã hóa và dữ liệu multipart(nhiều phần). Dữ liệu được mã hóa theo form được gửi khi `@FormUrlEncoded` được chỉ định trên phương thức. Mỗi cặp key-value được chú thích bằng `@Field` chứa tên và đối tượng cung cấp giá trị.

```
@FormUrlEncoded
```

```
@POST("user/edit")
```

```
Call<User> updateUser(@Field("first_name") String first, @Field("last_name") String last);
```

- Các yêu cầu multipart được sử dụng khi `@Multipart` xuất hiện trên phương thức. Các phần được khai báo bằng cách sử dụng `@Part`

```
@Multipart
```

```
@PUT("user/photo")
```

```
Call<User> updateUser(@Part("photo") RequestBody photo, @Part("description")  
RequestBody description);
```

Các phần của multiparts sử dụng một trong các bộ chuyển đổi của Retrofit hoặc chúng có thể implement RequestBody để xử lý serialization của riêng chúng.

- Mặc định Retrofit chỉ có thể deserialize phần thân bản tin HTTP thành kiểu OkHttp's ResponseBody
Và nó chỉ chấp nhận kiểu RequestBody cho Annotation [@Body](#).
- Converter có thể được thêm vào để hỗ trợ các loại khác:
 - **JSON**: com.squareup.retrofit2:converter-gson
 - Gson là một thư viện Java có thể được sử dụng để chuyển đổi các đối tượng Java thành biểu diễn JSON của chúng. Nó cũng có thể được sử dụng để chuyển đổi một chuỗi JSON thành một đối tượng Java tương đương.

```
public class Question {  
    @SerializedName("title")  
    @Expose  
    private String mTitle;  
  
    public String toString() { return mTitle; } }
```

[@SerializedName](#) cần thiết cho Gson để ánh xạ các khoá JSON với các trường dữ liệu. Để phù hợp với quy ước đặt tên kiểu camelCase của Java cho các thuộc tính thành viên của lớp, bạn không nên sử dụng dấu gạch dưới để tách các từ ngữ trong một biến.

[@Expose](#) chỉ ra rằng trường này nên được định nghĩa với JSON serialization hoặc deserialization.

Các phương thức

405

- Luôn request bao gồm: Đồng bộ và không đồng bộ
 - ▣ Với Retrofit 2, tất cả các request đều được bọc trong đối tượng **Call**. Đây là Interface được sử dụng cho cả quá trình request đồng bộ và không đồng bộ trong Retrofit 2.
 - ▣ Sử dụng phương thức **.execute()** trên đối tượng **call** nên request này sẽ được thực hiện một cách đồng bộ. Dữ liệu được trả về qua method **.body()**

```
Call<List<Task>> call = APIService.getTasks();
List<Task>> tasks = call.execute().body();
```

- ▣ Retrofit 2 thực hiện request bằng phương thức **.enqueue()** để truyền không đồng bộ như ví dụ ở bước 4 của slide trước.

- **onResponse():** được gọi khi nhận được một phản hồi HTTP. Phương thức này được gọi khi có một phản hồi mà có thể được xử lý một cách chính xác ngay cả khi máy chủ trả về một thông báo lỗi. Vì vậy nếu bạn nhận được một code trạng thái là 404 hoặc 500, phương thức này sẽ vẫn được gọi. Để có được code trạng thái để bạn xử lý các tình huống dựa trên chúng, bạn có thể sử dụng phương thức response.code(). Bạn cũng có thể sử dụng phương thức isSuccessful() để tìm ra code trạng thái trong khoảng 200-300, xác định một yêu cầu thành công.
- **onFailure():** được gọi khi một ngoại lệ kết nối mạng xảy ra trong quá trình giao tiếp đến máy chủ, hoặc khi một ngoại lệ bất ngờ xảy ra trong quá trình xử lý yêu cầu hoặc xử lý phản hồi.

- Bước 1: Thêm thư viện vào build.gradle
- Bước 2: Tạo các phương thức truy vấn
- Bước 3: Tạo Retrofit Client
- Bước 4: Sử dụng Retrofit

□ Bước 1: Thêm thư viện Retrofit2 vào build.gradle

```
// Network & Retrofit
implementation 'com.squareup.retrofit2:retrofit:2.9.0'
implementation 'com.squareup.retrofit2:converter-gson:2.9.0'
implementation 'com.squareup.okhttp3:logging-interceptor:3.12.0'
```

□ Bước 2: Tạo các phương thức truy vấn

```
public interface APIService {
    @GET("categories.php")
    Call<List<Category>> getCategoriesAll();

    @GET("category.php")
    Call<CategoryModel> getCategory();

    @POST("/v1/user")
    @FormUrlEncoded
    Call<User> login(@Field("username") String username,
                      @Field("password") String password);
}
```

Tạo ra 1 **interface** định nghĩa các phương thức sẽ được sử dụng trong project

Trong thẻ Application của AndroidManifest.xml thêm lệnh sau:

`android:usesCleartextTraffic="true"`

- Bước 2: Tạo Retrofit Client, có nhiều cách tạo Retrofit Client. Đây là 01 trong số đó.

Sử dụng **HttpLoggingInterceptor** để kiểm tra truy vấn mà Retrofit tạo dựa vào Mô tả API Endpoint bên trên. Các câu lệnh truy vấn này sẽ được hiển thị trong mục debug của Logcat. OkHttpClient sẽ giúp chúng ta tạo request to server. Còn GsonConverterFactory sẽ chuyển đổi dữ liệu sang dạng object. Với BaseClient class bên dưới chúng ta có thể tạo Retrofit cho các baseUrl khác nhau bằng cách truyền vào trong hàm `createService()`.

Các bước thực hiện

410

- Bước 2: Tạo Retrofit Client: gồm 01 class chung gọi là BaseClient và các class riêng cụ thể từng URL.

```
import okhttp3.OkHttpClient;
import okhttp3.logging.HttpLoggingInterceptor;
import retrofit2.Retrofit;
import retrofit2.converter.gson.GsonConverterFactory;
|
public class BaseClient {
    private static HttpLoggingInterceptor sLogging =
        new HttpLoggingInterceptor()
            .setLevel(HttpLoggingInterceptor.Level.BODY);
    private static OkHttpClient.Builder sHttpClient =
        new OkHttpClient.Builder();
    static <S> S createService(Class<S> serviceClass, String baseUrl) {
        Retrofit.Builder builder = new Retrofit.Builder()
            .baseUrl(baseUrl)
            .addConverterFactory(GsonConverterFactory.create());
        Retrofit retrofit = builder.build();
        if (!sHttpClient.interceptors().contains(sLogging)) {
            sHttpClient.addInterceptor(sLogging);
            builder.client(sHttpClient.build());
            retrofit = builder.build();
        }
        return retrofit.create(serviceClass);
    }
}
```

- Bước 2: Tạo Retrofit Client: gồm 01 class chung gọi là BaseClient và các class riêng cụ thể từng URL và APIService.

```
public class RetrofitClient1 extends BaseClient {  
    private static final String BASE_URL = "http://app.iotstar.vn/appfoods/";  
    private static APIService apiService;  
    public static APIService getInstance() {  
        if (apiService == null) {  
            return createService(APIService.class, BASE_URL);  
        }  
        return apiService;  
    }  
}
```

- Bước 4: Sử dụng Retrofit: gọi hàm getInstance() trong Retrofit Client ở bước 3 và thực hiện luôn Request

```
private void GetCategory() {
    //gọi Instance và thực hiện luôn Request
    apiService = RetrofitClient1.getInstance();
    apiService.getCategoriesAll().enqueue(new Callback<List<Category>>() {
        @Override
        public void onResponse(Call<List<Category>> call, Response<List<Category>> response) {
            if(response.isSuccessful()) {
                categoryList = response.body(); //nhận dữ liệu bảng
                //khởi tạo Adapter
                categoryAdapter = new CategoryAdapter( context: RetrofitActivity.this,categoryList);
                rcCate.setHasFixedSize(true);
                RecyclerView.LayoutManager layoutManager = new LinearLayoutManager(getApplicationContext()
                    ,LinearLayoutManager.HORIZONTAL, reverseLayout: false);
                rcCate.setLayoutManager(layoutManager);
                rcCate.setAdapter(categoryAdapter);
                categoryAdapter.notifyDataSetChanged();
            }else {
                int statusCode = response.code();
                // handle request errors depending on status code
            }
        }
        @Override
        public void onFailure(Call<List<Category>> call, Throwable t) {
            Log.d( tag: "logg",t.getMessage());
        }
    });
}
```

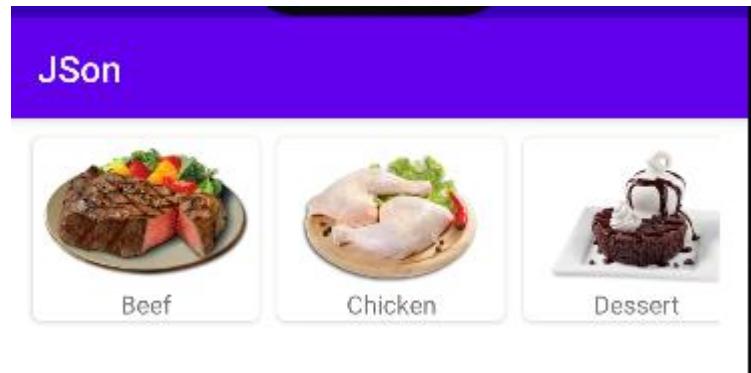
Demo ví dụ:

413

- 0. Thiết kế giao diện
- 1. Cài đặt thư viện retrofit
- 2. Thêm cài đặt quyền truy cập internet
- 3. Chuẩn bị API
- 4. Chuẩn bị Model
- 5. Định nghĩa Retrofit Client
- 6. Định nghĩa API Interface
- 7. Tạo Adapter cho RecyclerView
- 8. Thực hiện request

□ 0. Thiết kế giao diện: activity_main.xml

```
<androidx.core.widget.NestedScrollView  
    android:layout_width="match_parent"  
    android:layout_height="match_parent"  
    app:layout_constraintBottom_toBottomOf="parent"  
    app:layout_constraintEnd_toEndOf="parent"  
    app:layout_constraintStart_toStartOf="parent"  
    app:layout_constraintTop_toTopOf="parent">  
  
<androidx.recyclerview.widget.RecyclerView  
    android:id="@+id/rc_category"  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    android:layout_marginStart="8dp"  
    android:layout_marginTop="5dp"  
    android:layout_marginEnd="16dp" />  
  
</androidx.core.widget.NestedScrollView>
```



□ 0. Thiết kế giao diện: item_category.xml

```
<androidx.cardview.widget.CardView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_margin="5dp"
    android:layout_marginRight="8dp"
    android:elevation="8dp"
    android:padding="5dp"
    app:cardCornerRadius="5dp"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent">
<androidx.constraintlayout.widget.ConstraintLayout
    android:layout_width="wrap_content"
    android:layout_height="wrap_content">
<ImageView
    android:id="@+id/image_cate"
    android:layout_width="120dp"
    android:layout_height="80dp"
    android:layout_marginStart="3dp"
    android:layout_marginTop="3dp"
    android:layout_marginEnd="3dp"
    android:src="@drawable/ic_launcher_background"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent" />
<TextView
    android:id="@+id/tvNameCategory"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Name cate"
    app:layout_constraintEnd_toEndOf="@+id/image_cate"
    app:layout_constraintStart_toStartOf="@+id/image_cate"
    app:layout_constraintTop_toBottomOf="@+id/image_cate" />
</androidx.constraintlayout.widget.ConstraintLayout>
</androidx.cardview.widget.CardView>
```

Demo ví dụ:

416

- 1. Cài đặt thư viện retrofit trong build.gradle (Module: app)

// Network & Retrofit

```
implementation 'com.squareup.retrofit2:retrofit:2.9.0'  
implementation 'com.squareup.retrofit2:converter-gson:2.9.0'  
implementation 'com.squareup.okhttp3:logging-interceptor:3.12.0'
```

- 2. Thêm cài đặt quyền truy cập internet trong file AndroidManifest.xml

```
<uses-permission android:name="android.permission.INTERNET" />  
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
```

- 2. Thêm thư viện Load ảnh Glide (xem chi tiết)

//load ảnh với Glide

```
implementation 'com.github.bumptech.glide:glide:4.14.2'  
annotationProcessor 'com.github.bumptech.glide:compiler:4.14.2'
```

□ 3. Chuẩn bị API: ví dụ ta có 01 API liệt kê tất cả Category như sau:

<http://app.iotstar.vn/appfoods/categories.php>

Method Request URL

GET http://app.iotstar.vn/appfoods/categories.php

▼ SEND :

Parameters

200 OK 148.10 ms

DETAILS ▾

[{"id": "1", "name": "Beef", "images": "https://www.themealdb.com/images/category/beef.png", "description": "Beef is the culinary name for meat from cattle, particularly skeletal muscle. Humans have been eating beef since prehistoric times.[1] Beef is a source of high-quality protein and essential nutrients.[2]"}, {"id": "2", "name": "Chicken", "images": "https://www.themealdb.com/images/category/chicken.png", "description": "Chicken is a type of domesticated fowl, a subspecies of the red junglefowl. It is one of the most common and widespread domestic animals, with a total population of more than 19 billion as of 2011.[1] Humans commonly keep chickens as a source of food (consuming both their meat and eggs) and, more rarely, as pets."}, {"id": "3", "name": "Dessert", "images": "https://www.themealdb.com/images/category/dessert.png", "description": "Dessert is a course that concludes a meal. The course usually consists of sweet foods, such as confections, dishes or fruit, and possibly a beverage such as dessert wine or liqueur; however, in the United States it may include coffee, cheeses, nuts, or other savory items regarded as a separate course elsewhere. In some parts of the world, such as much of central and western Africa, and most parts of China, there is no tradition of a dessert course to conclude a meal.\r\n\r\nThe term dessert can apply to many confections, such as biscuits, cakes, cookies, custards, gelatin, ice creams, pastries, pies, puddings, and sweet soups, and tarts. Fruit is also commonly found in dessert courses because of its naturally occurring sweetness. Some cultures sweeten foods that are more commonly savory to create desserts."}, {"id": "4", "name": "Lamb", "images": "https://www.themealdb.com/images/category/lamb.png", "description": "Lamb, hogget, and mutton are the meat of domestic sheep (species Ovis aries) at different ages.\r\n\r\nA sheep in its first year is called a lamb, and its meat is also called lamb. The meat of a juvenile sheep older than one year is hogget; outside the USA this is also a term for the living animal. The meat of an adult sheep is mutton, a term only used for the meat, not the living animal. The term mutton is almost always used to refer to goat meat in the Indian subcontinent.\r\n\r\n"}, {"id": "5", "name": "Miscellaneous", "images": "https://www.themealdb.com/images/category/miscellaneous.png", "description": "General foods that don't fit into another category."}, {"id": "6", "name": "Pasta", "images": "https://www.themealdb.com/images/category/pasta.png", "description": "Pasta is a staple food of traditional Italian cuisine, with the first reference dating to 1154 in Sicily.\r\n\r\nAlso commonly used to refer to the variety of pasta dishes, pasta is typically a noodle made from an unleavened dough of a durum wheat flour mixed with water or eggs and formed into sheets or various shapes, then cooked by boiling or baking. As an alternative for those wanting a different taste, or who need to avoid products containing gluten, some pastas can be made using rice flour in place of wheat.[3][4] Pastas may be divided into two broad categories, dried (pasta secca) and fresh (pasta fresca)."}, {"id": "7", "name": "Pork", "images": "https://www.themealdb.com/images/category/pork.png", "description": "Pork is the culinary name for meat from a domestic pig (Sus scrofa domesticus). It is the most commonly consumed meat worldwide,[1] with evidence of pig husbandry dating back to 5000 BC. Pork is eaten both freshly cooked and preserved. Curing extends the shelf life of the pork products. Ham, smoked pork, gammon, bacon and sausages are examples of preserved pork. Charcuterie is the branch of cooking devoted to prepared meat products, many from pork.\r\n\r\nPork is the most popular meat in Eastern and Southeastern Asia, and is also very common in the Western world, especially in Central Europe. It is highly prized in Asian cuisines for its fat content and pleasant texture. Consumption of pork is forbidden by Jewish and Muslim dietary law, a taboo that is deeply rooted in tradition, with several suggested possible causes. The sale of pork is limited in Israel and illegal in certain Muslim countries."}, {"id": "9", "name": "Side", "images": "https://www.themealdb.com/images/category/side.png", "description": "Side dishes are complementary foods served alongside the main dish, such as vegetables, bread, or potatoes. They are often served in smaller portions than the main dish and are not the focus of the meal."}]

```
[  
  {  
    "id": "1",  
    "name": "Beef",  
    "images": "https://www.themealdb.com/images/category/beef.png",  
    "description": "Beef is the culinary name for meat from cattle, particularly skeletal muscle. Humans have been eating beef since prehistoric times.[1] Beef is a source of high-quality protein and essential nutrients.[2]"},  
  {  
    "id": "2",  
    "name": "Chicken",  
    "images": "https://www.themealdb.com/images/category/chicken.png",  
    "description": "Chicken is a type of domesticated fowl, a subspecies of the red junglefowl. It is one of the most common and widespread domestic animals, with a total population of more than 19 billion as of 2011.[1] Humans commonly keep chickens as a source of food (consuming both their meat and eggs) and, more rarely, as pets."},  
  {  
    "id": "3",  
    "name": "Dessert",  
    "images": "https://www.themealdb.com/images/category/dessert.png",  
    "description": "Dessert is a course that concludes a meal. The course usually consists of sweet foods, such as confections, dishes or fruit, and possibly a beverage such as dessert wine or liqueur; however, in the United States it may include coffee, cheeses, nuts, or other savory items regarded as a separate course elsewhere. In some parts of the world, such as much of central and western Africa, and most parts of China, there is no tradition of a dessert course to conclude a meal.\r\n\r\nThe term dessert can apply to many confections, such as biscuits, cakes, cookies, custards, gelatin, ice creams, pastries, pies, puddings, and sweet soups, and tarts. Fruit is also commonly found in dessert courses because of its naturally occurring sweetness. Some cultures sweeten foods that are more commonly savory to create desserts."},  
  {  
    "id": "4",  
    "name": "Lamb",  
    "images": "https://www.themealdb.com/images/category/lamb.png",  
    "description": "Lamb, hogget, and mutton are the meat of domestic sheep (species Ovis aries) at different ages.\r\n\r\nA sheep in its first year is called a lamb, and its meat is also called lamb. The meat of a juvenile sheep older than one year is hogget; outside the USA this is also a term for the living animal. The meat of an adult sheep is mutton, a term only used for the meat, not the living animal. The term mutton is almost always used to refer to goat meat in the Indian subcontinent.\r\n\r\n"},  
  {  
    "id": "5",  
    "name": "Miscellaneous",  
    "images": "https://www.themealdb.com/images/category/miscellaneous.png",  
    "description": "General foods that don't fit into another category."},  
  {  
    "id": "6",  
    "name": "Pasta",  
    "images": "https://www.themealdb.com/images/category/pasta.png",  
    "description": "Pasta is a staple food of traditional Italian cuisine, with the first reference dating to 1154 in Sicily.\r\n\r\nAlso commonly used to refer to the variety of pasta dishes, pasta is typically a noodle made from an unleavened dough of a durum wheat flour mixed with water or eggs and formed into sheets or various shapes, then cooked by boiling or baking. As an alternative for those wanting a different taste, or who need to avoid products containing gluten, some pastas can be made using rice flour in place of wheat.[3][4] Pastas may be divided into two broad categories, dried (pasta secca) and fresh (pasta fresca)."},  
  {  
    "id": "7",  
    "name": "Pork",  
    "images": "https://www.themealdb.com/images/category/pork.png",  
    "description": "Pork is the culinary name for meat from a domestic pig (Sus scrofa domesticus). It is the most commonly consumed meat worldwide,[1] with evidence of pig husbandry dating back to 5000 BC. Pork is eaten both freshly cooked and preserved. Curing extends the shelf life of the pork products. Ham, smoked pork, gammon, bacon and sausages are examples of preserved pork. Charcuterie is the branch of cooking devoted to prepared meat products, many from pork.\r\n\r\nPork is the most popular meat in Eastern and Southeastern Asia, and is also very common in the Western world, especially in Central Europe. It is highly prized in Asian cuisines for its fat content and pleasant texture. Consumption of pork is forbidden by Jewish and Muslim dietary law, a taboo that is deeply rooted in tradition, with several suggested possible causes. The sale of pork is limited in Israel and illegal in certain Muslim countries."},  
  {  
    "id": "9",  
    "name": "Side",  
    "images": "https://www.themealdb.com/images/category/side.png",  
    "description": "Side dishes are complementary foods served alongside the main dish, such as vegetables, bread, or potatoes. They are often served in smaller portions than the main dish and are not the focus of the meal."}]
```

Demo ví dụ:

418

- 4. Chuẩn bị Model: tạo Getters và Setters cho các trường, dùng @SerializedName nếu muốn tên trường database khác với tên trong Model

```
public class Category implements Serializable {  
    @SerializedName("id")  
    private int id;  
    @SerializedName("name")  
    private String name;  
    @SerializedName("images")  
    private String images;  
    @SerializedName("description")  
    private String description;  
  
    //tạo getter & setters  
}
```

□ 5. Định nghĩa Retrofit Client

```
import retrofit2.Retrofit;
import retrofit2.converter.gson.GsonConverterFactory;

public class RetrofitClient {
    private static Retrofit retrofit;

    public static Retrofit getRetrofit(){
        if (retrofit == null){
            retrofit = new Retrofit.Builder()
                //đường dẫn API
                .baseUrl("http://app.iotstar.vn/appfoods/")
                .addConverterFactory(GsonConverterFactory.create())
                .build();
        }
        return retrofit;
    }
}
```

□ 6. Định nghĩa API Interface

```
import java.util.List;
import retrofit2.Call;
import retrofit2.http.GET;
import vn.iotstar.json.model.Category;

public interface APIService {
    @GET("categories.php")
    Call<List<Category>> getCategoryAll();
}
```

Giải thích các thông tin trong đoạn code trên:

@GET: phương thức thực hiện request.

□ 7. Tạo Adapter cho RecyclerView

```
public class CategoryAdapter extends RecyclerView.Adapter<CategoryAdapter.MyViewHolder> {
    Context context;
    List<Category> array;
    public CategoryAdapter(Context context, List<Category> array) {
        this.context = context;
        this.array = array;
    }
    @NonNull
    @Override
    public MyViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
        View view = LayoutInflater.from(parent.getContext())
            .inflate(R.layout.item_category, root: null);
        MyViewHolder myViewHolder = new MyViewHolder(view);
        return myViewHolder;
    }
    public class MyViewHolder extends RecyclerView.ViewHolder{
        public ImageView images;
        public TextView tenSp;
        public MyViewHolder(@NonNull View itemView) {
            super(itemView);
            images = (ImageView) itemView.findViewById(R.id.image_cate);
            tenSp = (TextView) itemView.findViewById(R.id.tvNameCategory);
            //bắt sự kiện cho item holder trong MyViewHolder
            itemView.setOnClickListener(new View.OnClickListener() {
                @Override
                public void onClick(View v) {
                    //xử lý khi nhấp vào Item trên category
                    Toast.makeText(context, text: "Bạn đã chọn category" + tenSp.getText().toString(), Toast.LENGTH_SHORT).show();
                }
            });
        }
    }
}
```

□ 7. Tạo Adapter cho RecyclerView

```
@Override  
public void onBindViewHolder(@NonNull MyViewHolder holder, int position) {  
    Category category = array.get(position);  
    holder.tenSp.setText(category.getName());  
    //load ảnh với Glide  
    Glide.with(context).RequestManager  
        .load(category.getImages()).RequestBuilder<Drawable>  
        .into(holder.images);  
}  
@Override  
public int getItemCount() { return array == null ? 0 :array.size(); }  
}
```

Demo ví dụ:

423

□ 8. Thực hiện Request

```
public class RetrofitActivity extends AppCompatActivity {  
    RecyclerView rcCategory;  
    //Khai báo Adapter  
    CategoryAdapter categoryAdapter;  
    APIService apiService;  
    List<Category> categoryList;  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity.Retrofit);  
        AnhXa();  
        GetCategory(); //load dữ liệu cho category  
    }  
    private void AnhXa() {  
        //ánh xạ  
        rcCategory = (RecyclerView) findViewById(R.id.rc_category);  
    }  
}
```

□ 8. Thực hiện Request

```
private void GetCategory() {  
    //Gọi Interface trong APIService  
    apiService = RetrofitClient.getRetrofit().create(APIService.class);  
    apiService.getCategoryAll().enqueue(new Callback<List<Category>>() {  
        @Override  
        public void onResponse(Call<List<Category>> call, Response<List<Category>> response) {  
            if(response.isSuccessful()) {  
                categoryList = response.body(); //nhận mảng  
                //khởi tạo Adapter  
                categoryAdapter = new CategoryAdapter( context: RetrofitActivity.this,categoryList);  
                rcCate.setHasFixedSize(true);  
                RecyclerView.LayoutManager layoutManager = new LinearLayoutManager(getApplicationContext())  
                    ,LinearLayoutManager.HORIZONTAL, reverseLayout: false);  
                rcCate.setLayoutManager(layoutManager);  
                rcCate.setAdapter(categoryAdapter);  
                categoryAdapter.notifyDataSetChanged();  
            } else {  
                int statusCode = response.code();  
                // handle request errors depending on status code  
            }  
        }  
        @Override  
        public void onFailure(Call<List<Category>> call, Throwable t) {  
            Log.d( tag: "logg",t.getMessage());  
        }  
    });  
}
```

Load Images with Glide

425

- Glide là một thư viện open source hỗ trợ load ảnh trên Android. Dùng Glide sẽ đơn giản hóa các công việc mà bạn cần làm khi sử dụng một bức ảnh trong Android đi rất nhiều. Chúng ta không cần quan tâm đến việc decoding, memory and disk caching mà thay vào đó chỉ cần sử dụng interface rất đơn giản từ Glide.
- Glide hỗ trợ fetching, decoding và hiển thị cả ảnh tĩnh hoặc ảnh động đó là điểm nổi bật của nó so với các thư viện load ảnh khác cho android. Glide có thể được dễ dàng đưa vào bất kỳ mô hình mạng nào. Mặc định Glide sử dụng mô hình custom của HttpURLConnection tuy nhiên chúng ta có thể dễ dàng cho nó hoạt động với Volley hoặc OkHttp, Retrofit.

- Thêm thư viện Glide vào build.gradle (Module:app)

```
//load ảnh với Glide  
implementation 'com.github.bumptech.glide:glide:4.14.2'  
annotationProcessor 'com.github.bumptech.glide:compiler:4.14.2'
```

- Thêm thư viện Picasso vào build.gradle (Module:app)

```
implementation 'com.squareup.picasso:picasso:2.5.2'
```

- Thêm quyền truy cập trong AndroidManifest.xml

```
<uses-permission android:name="android.permission.INTERNET"/>  
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>  
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
```

□ Load ảnh

```
Glide.with(context)
```

```
    .load("đường dẫn hình")
```

```
    .into(tên đối tượng ImageView);
```

```
.override(200,300) //resize ảnh
```

```
.centerCrop()//cắt hình và canh giữa
```

```
.fitCenter() //diền hình tràn ở giữa
```

```
.placeholder(R.drawable.placeholder)
```

```
.error(R.drawable.error)
```

```
Picasso.with(context)
```

```
    .load("đường dẫn hình")
```

```
    .into(tên đối tượng ImageView);
```

```
@Override
```

```
public void onBindViewHolder(@NotNull MyViewHolder holder, int position) {
```

```
    Category category = array.get(position);
```

```
    holder.tenSp.setText(category.getName());
```

```
    //load ảnh với Glide
```

```
    Glide.with(context) RequestManager
```

```
        .load(category.getImages()) RequestBuilder<Drawable>
```

```
        .into(holder.images);
```

```
}
```

- Nguyễn Hữu Trung
- 0908617108
- trungh@hcmute.edu.vn
- utex.hcmute.edu.vn