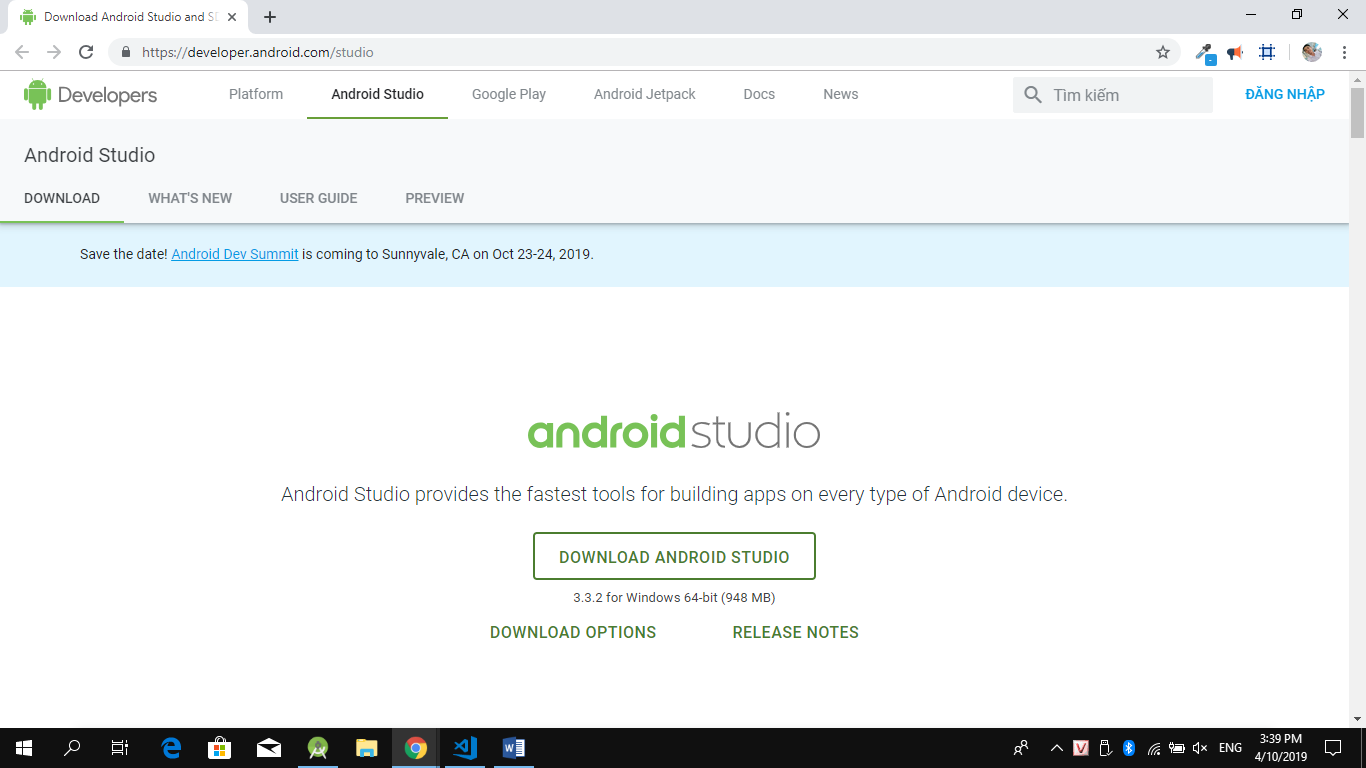
**Hướng dẫn cài đặt và tạo project trên android studio**

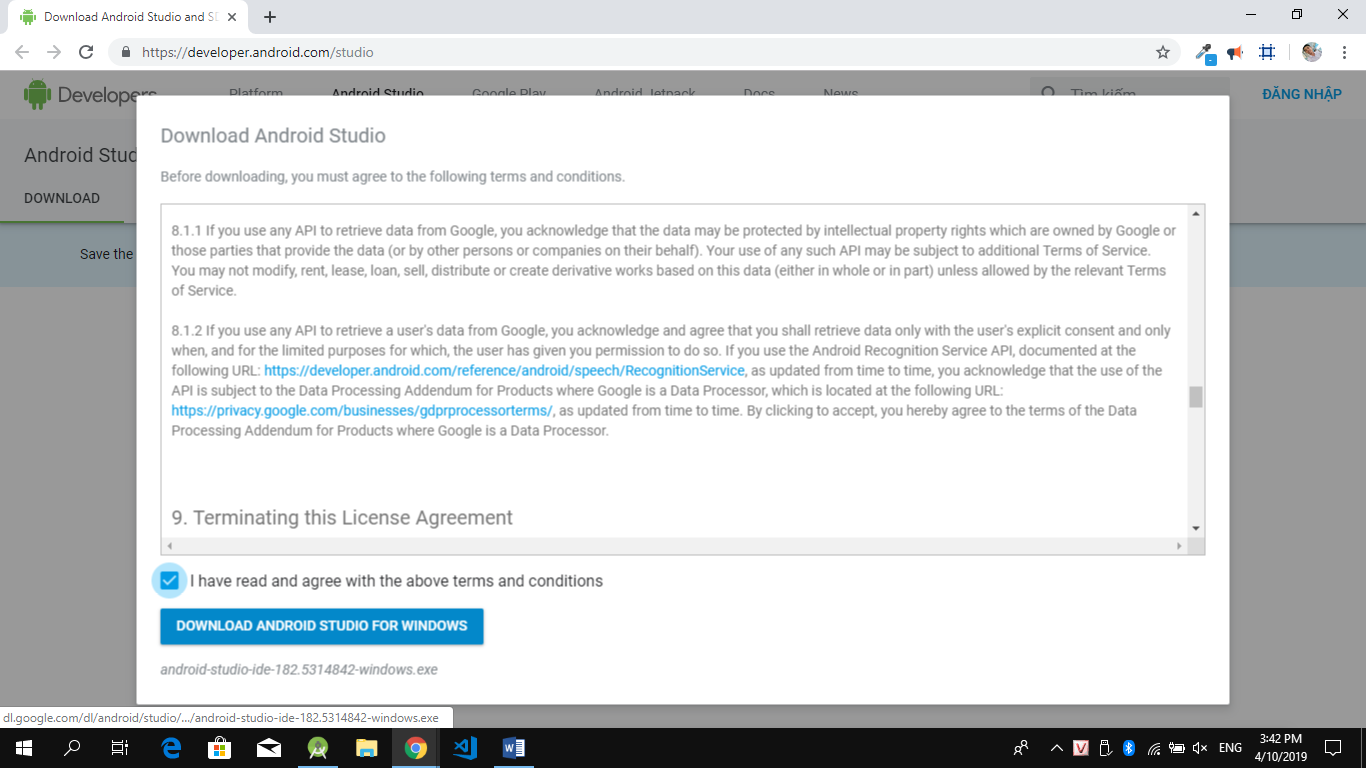
* + - 1. Hướng dẫn cài đặt
         1. Các bước thực hiện.

Download Android Studio.

Bước 1: Vào đường link <https://developer.android.com/studio>.



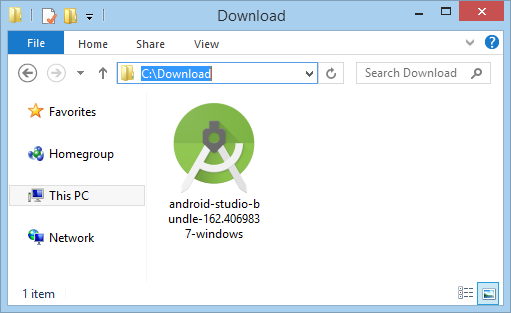
Click vào để tải file cài



Click vào để tải file cài

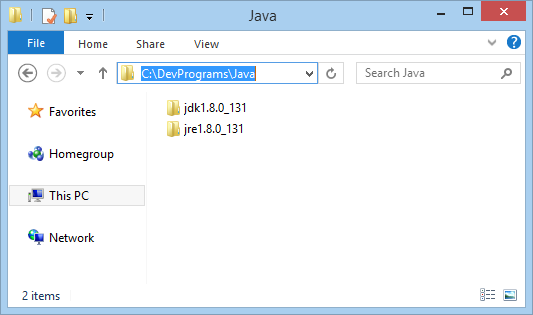
Chọn

Kết quả bạn download được:



Các cài đặt đòi hỏi

Đảm bảo rằng máy tính của bạn đã cài đặt Java phiên bản 7 trở lên. Ở đây tôi đã cài đặt sẵn Java phiên bản 8:



Nếu chưa cài Java bạn có thể xem hướng dẫn tại: <https://o7planning.org/vi/10377/huong-dan-cai-dat-va-cau-hinh-java>

Cài đặt Android Studio

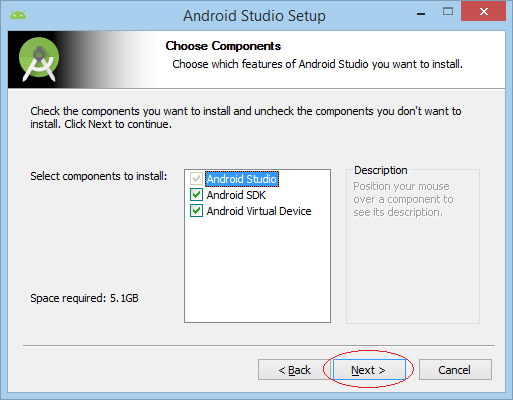


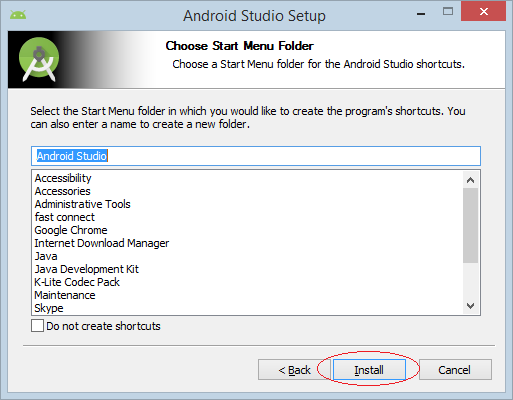
Lựa chọn tất cả các tùy chọn (options).

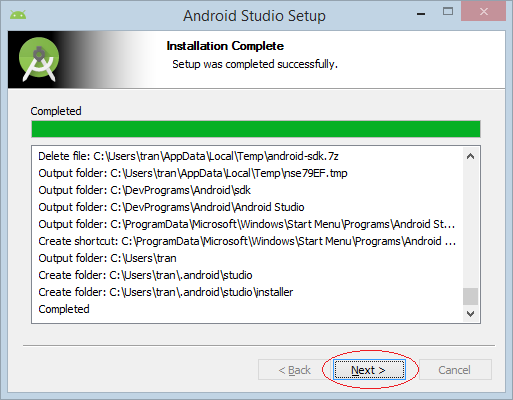
The Android SDK (software development kit) là một tập hợp các công cụ được sử dụng để phát triển ứng dụng cho Android. Android SDK bao gồm:

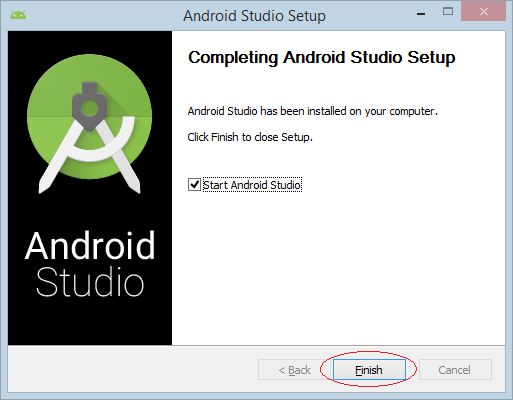
* Các thư viện đòi hỏi
* Bộ dò lỗi (Debugger)
* Thiết bị giả lập (emulator)
* Các tài liệu liên quan cho Android API.
* Các đoạn code mẫu.
* Các hướng dẫn cho hệ điều hành Android.

Android Virtual Device (AVD) là một thiết bị cấu hình, nó chạy với bộ giả lập Android (Android emulator). Nó làm việc với bộ giả lập để cung cấp một môi trường thiết bị ảo cụ thể, để cài đặt và chạy ứng dụng Android.





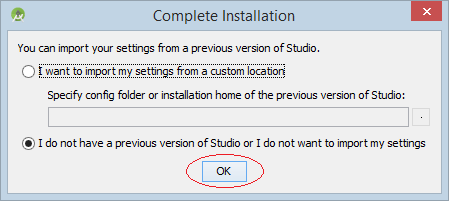


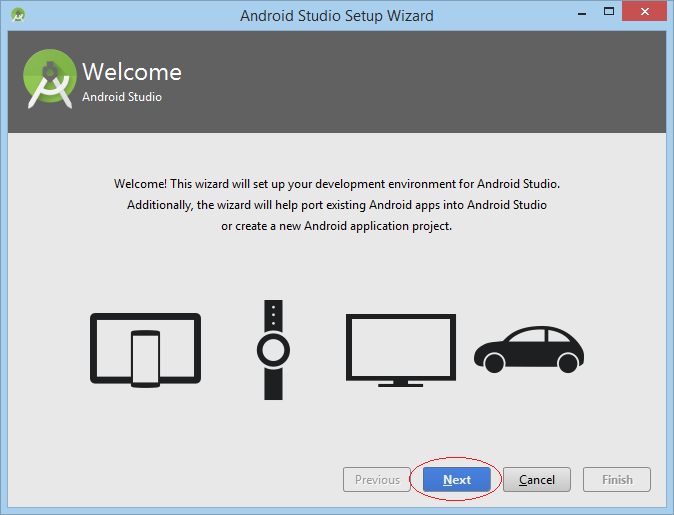


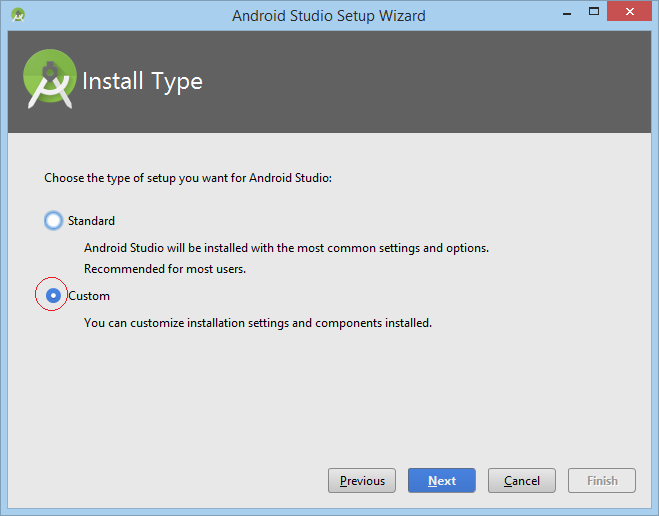
* + - 1. Hướng dẫn chạy Android Studio và tạo project android studio
         1. Các bước thực hiện.

Nếu bạn là người chạy lần đầu:

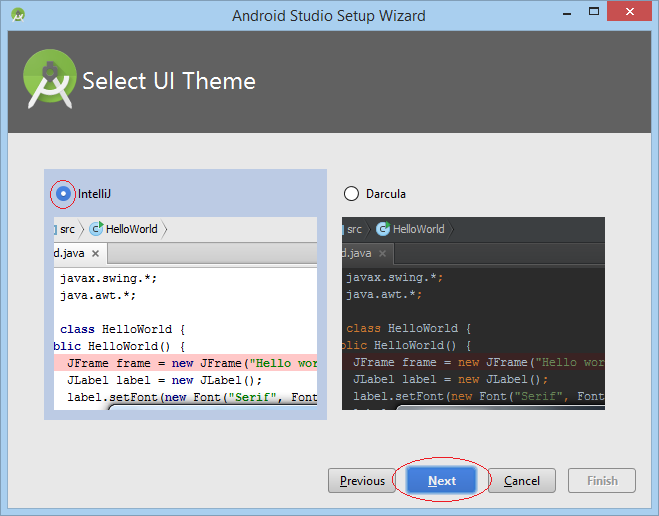
Trong lần chạy đầu tiên, Android Studio hỏi bạn có nhập khẩu các sét đặt từ phiên bản Android Studio mà bạn có thể đã cài đặt trước đó hay không. Bạn có thể chọn NO.







Lựa chọn một Theme mà bạn thích:



Setup Wizard mở ra một cửa sổ để bạn chọn các thành phần để bạn cập nhập, hoặc cài đặt thêm:

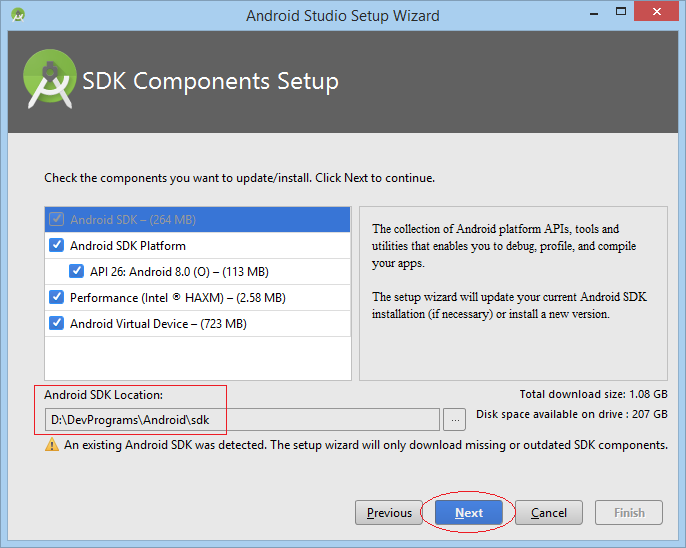
Performance (Intel® HAXM):

* Cho phép phần cứng hỗ trợ ảo hoá (hardware-assisted virtualization engine (hypervisor) ) để tăng tốc độ chạy ứng dụng Android trên máy tính phát triển ứng dụng của bạn. (Được đề nghị).

**Android Virtual Device**

* Thiết bị Android ảo được cấu hình sẵn và tối ưu hóa để bạn thử nghiệm ứng dụng trên trình giả lập (Emulator). (Được đề nghị).

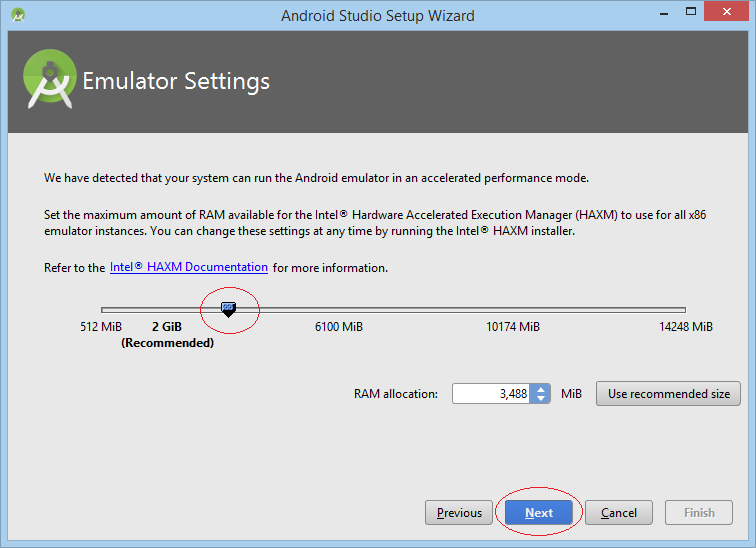
Lựa chọn thư mục SDK mà bạn đã cài đặt ở bước trước. Các thành phần SDK mới sẽ được cập nhập vào thư mục này.

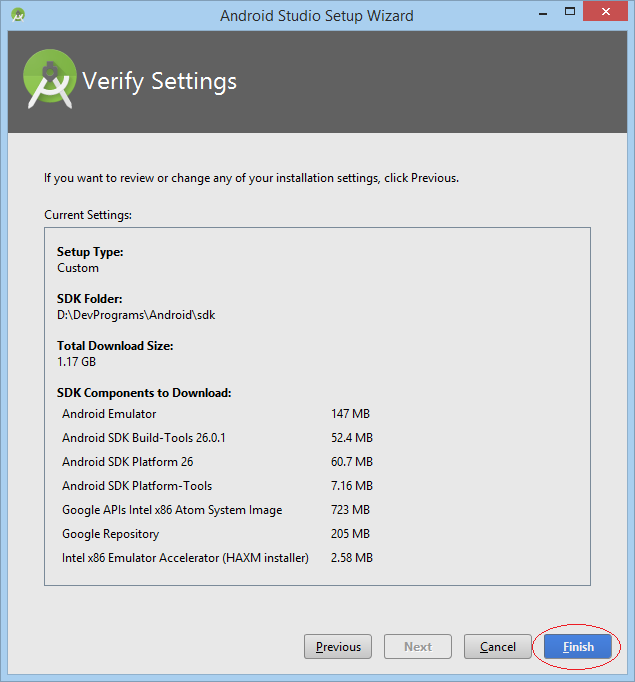


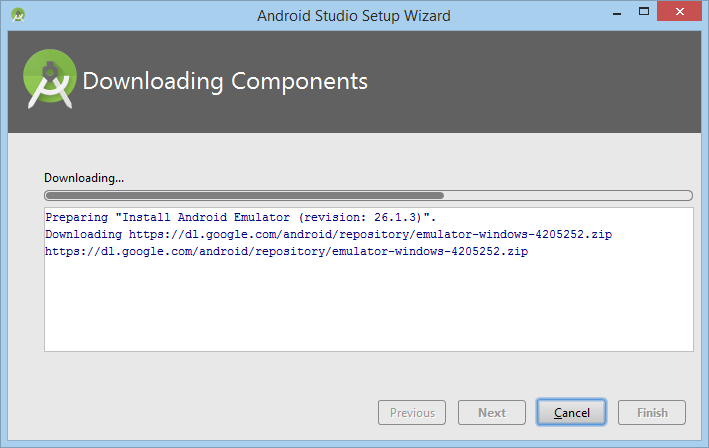
Nếu máy tính của bạn được trang bị phần cứng tốt, bộ giả lập Android ( Android Emulator) có thể chạy được trong chế độ tăng tốc (Accelerated performance mode).

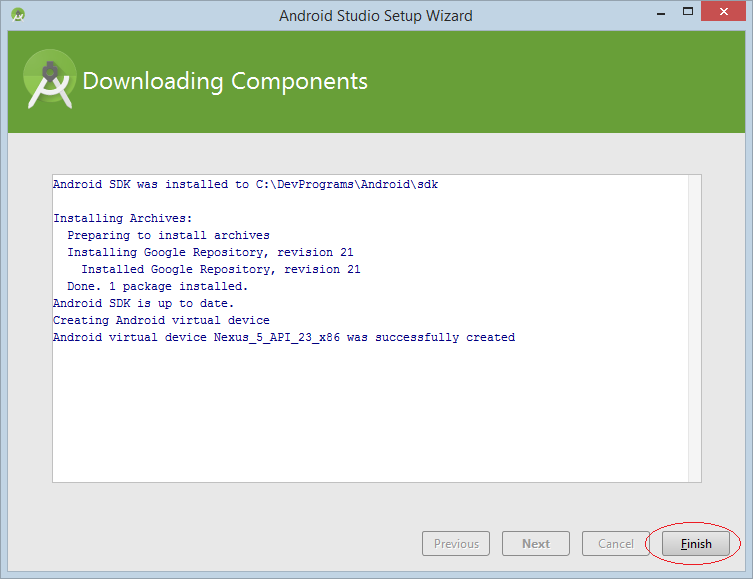
Bạn có thể cấu hình chỉ định số lượng RAM tối đa dành cho bộ quản lý tăng tốc phần cứng ( Intel Hardware Accelerated Manager - HAXM) . Khuyến nghị là 2GB.

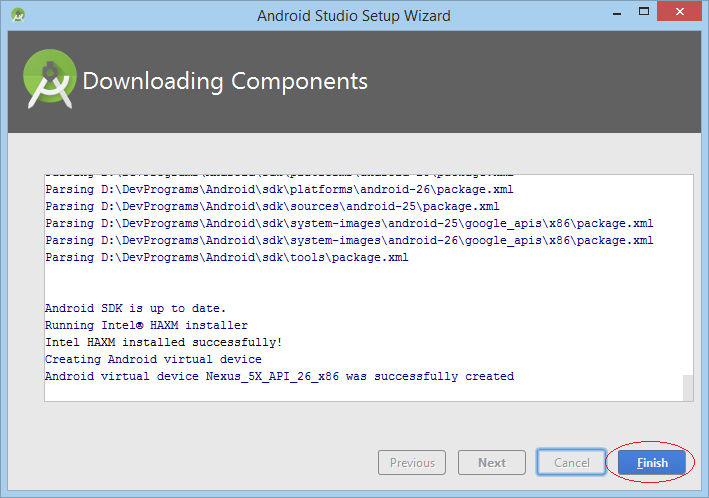
Chú ý: Khi bạn đặt chỗ một lượng RAM lớn, có thể là nguyên nhân làm các chương trình khác chạy chậm đi khi bạn đang sử dụng bộ giả lập Android x86 với HAXM.

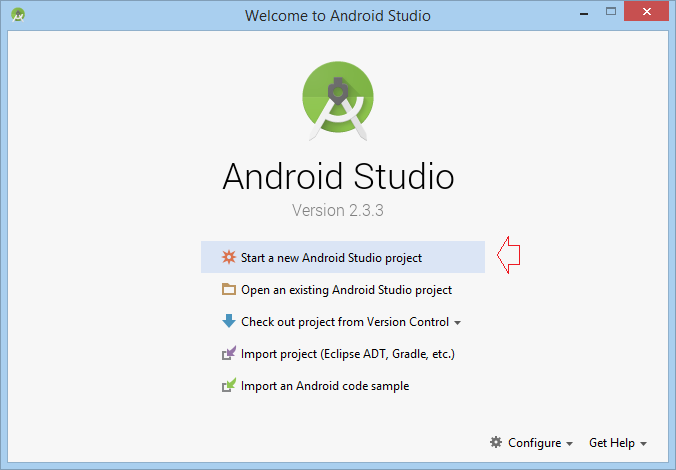




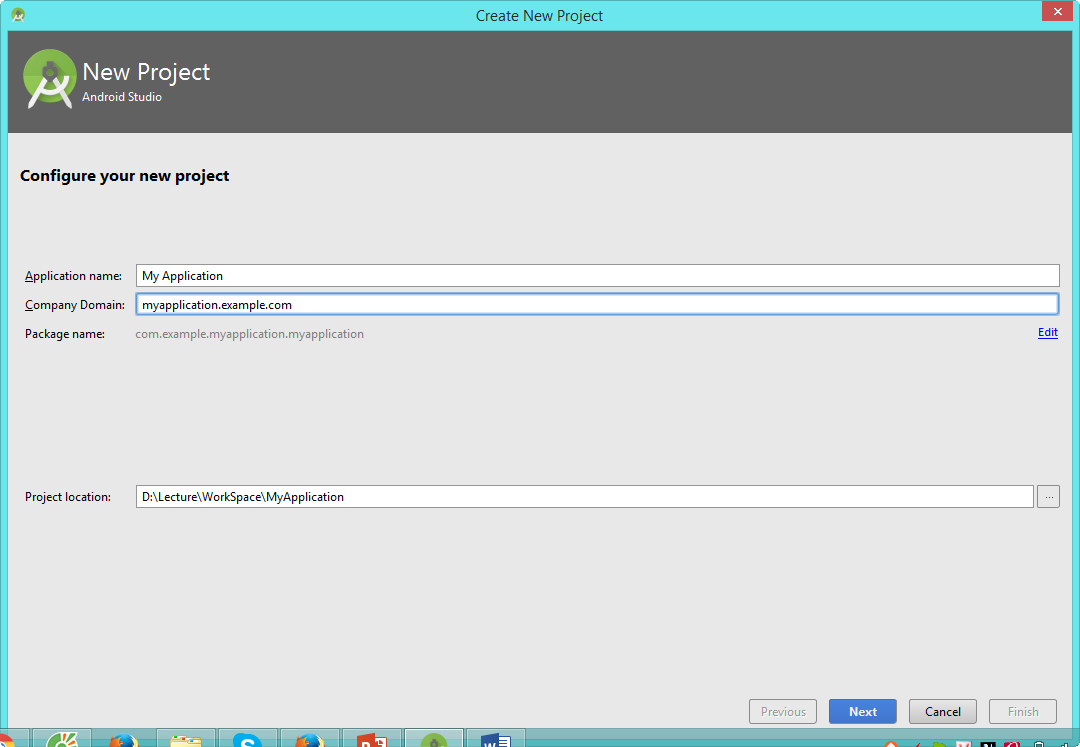




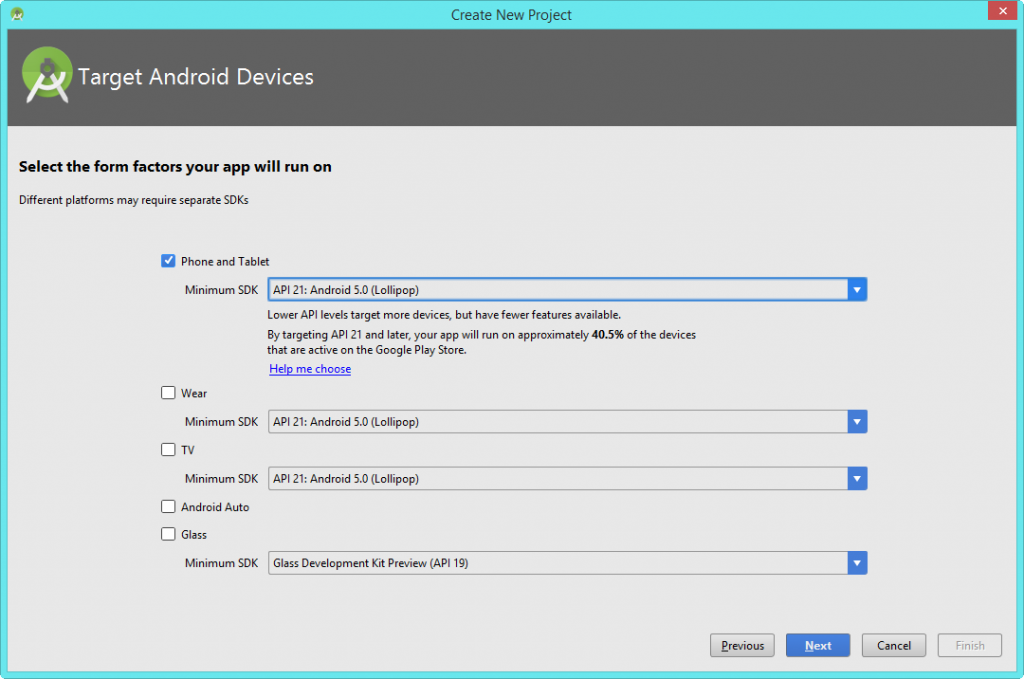




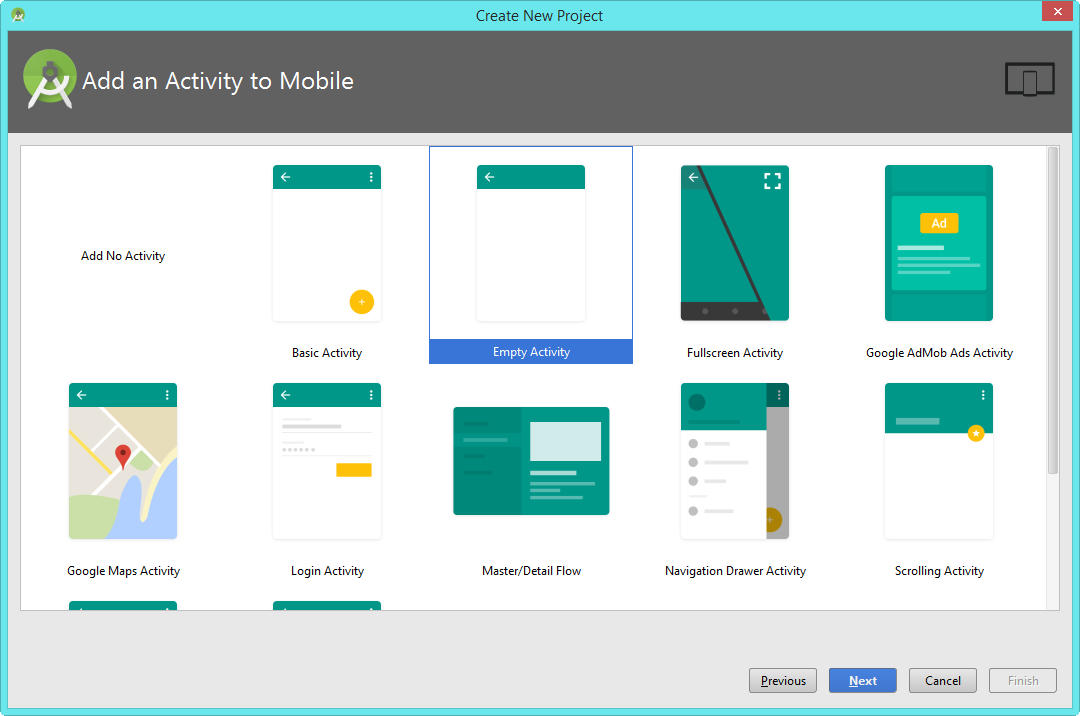
Bước 1: Khởi động Android Studio -> File -> chọn New -> chọn New Project -> Nhập tên ứng dụng (Application name), chỉ định thư mục chứa source code (Project location) -> chọn Next



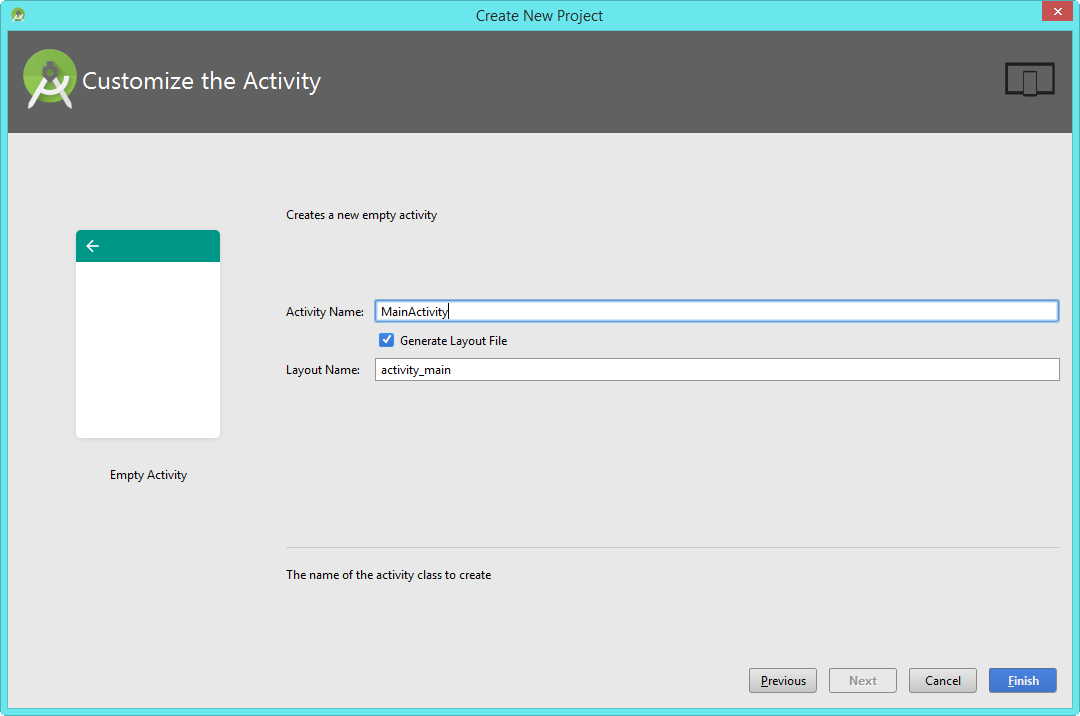
Bước 2: Chọn Phone and Tablet -> chọn phiên bản tối thiểu SDK (tương ứng với phiên bản hệ điều hành Android) -> chọn Next



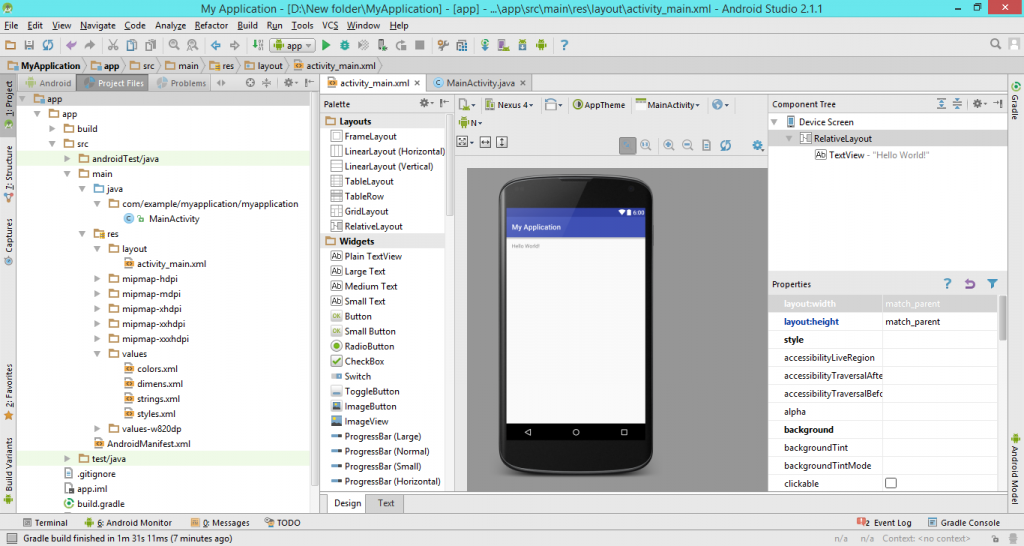
Bước 3: Chọn Activity, trong hình chúng tôi chọn Empty Activity (Việc lựa chọn Activity nào còn tùy thuộc vào mục đích của người phát triển ứng dụng) -> Next



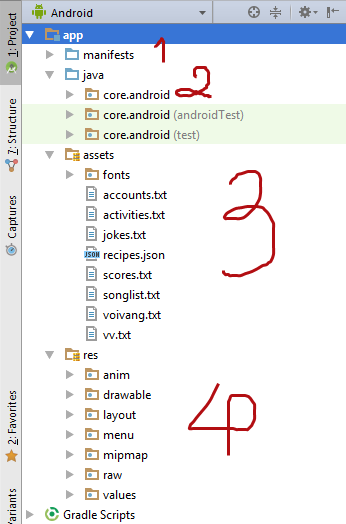
Bước 4: Nhập tên cho Activity tại Activity Name -> Finish



Sau khi tạo thành công project, chúng ta sẽ nhìn thấy như hình bên dưới



* 1. Cấu trúc project

****

**1/ manifests**: Chứa tập tin AndroidManifest.xml. Đây là tập tin khai báo tất cả các Activity được sử dụng trong project. Ngoài ra, tập tin này còn có chức năng cấp quyền cho ứng dụng, chẳng hạn cấp quyền truy cập internet, …

**2/ java**: Chứa tất cả các tập tin java. Mỗi một Activity được tạo ra sẽ tương ứng cho một tập tin java và tập tin này sẽ được chứa ở đây. Đây cũng là nơi lập trình viên viết code cho ứng dụng.

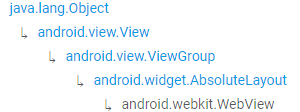
**3/ assets**: Mặc định thư mục này chưa tồn tại. Lập trình viên phải tạo thư mục này. Trong thư mục này có thể chừa hình, txt, …

**4/ res:** Chứa layout, menu, animation, …Khi một Activity được tạo, đồng thời Android sẽ tạo 2 tập tin. Một là tập tin java, chứa code. Một là tập tin xml, chứa layout. Ngoài ra, khi chúng ta tạo menu hoặc tạo animation (anim), cũng sẽ được thực hiện ở đây.

**GIỚI THIỆU VÀ HƯỚNG DẪN SỬ DỤNG WEBVIEW**

GIỚI THIỆU WEBVIEW

Trong Android, Đối tượng **WebView**dùng để hiện thị trang trong một ứng dụng. Trang web có thể được tải từ cùng một ứng dụng hoặc URL. **WebView**được sử dụng để hiển thị nội dung trực tuyến trong **Activity**của**Android**. Chúng ta cũng có thể hiện thị HTML trong ứng dụng Android, bằng cách sử dụng **WebView**

Android WebView sử dụng bộ máy webkit để hiển thị trang web

 android.webkit.WebView là lớp con của lớp AbsoluteLayout.

HƯỚNG DẪN SỬ DỤNG WEBVIEW

Để thêm Web View trong ứng dụng, Trong tập tin chúng ta thêm phần tử**<WebView>,**Cũng có thể thêm trong**Java Class**

**WebView code trong XML**

<WebView

android:id="@+id/simpleWebView"

android:layout\_width="fill\_parent"

android:layout\_height="fill\_parent" />

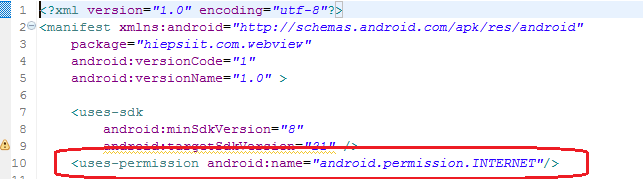
**Phân Quyền truy cập Internet cho WebView**

Để Activity có thể truy cập được internet và tải trang web trong webview. Chúng ta cần phải phân quyền truy cập internet cho ứng dụng trong tập tin Manifest **(Manifest.xml).**

**Code phân quyền:**

<!--Add this before application tag in AndroidManifest.xml-->

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



CÁC PHƯƠNG THỨC THƯỜNG DỤNG CỦA WEBVIEW

Dưới đây là một số phương thức thường dùng để cấu hình **WebView** trong ứng dụng của chúng ta

* **loadUrl() – Tải một trang web vào ứng dụng**

loadUrl(String url)

Phương thức này dùng để tải trang web vào trong ứng dụng. Trong phương thức này chúng ta phải chỉ ra url của trang web mà chúng ta muốn tải trong **WebView**

\*Add in Oncreate() funtion after setContentView()\*/

// initiate a web view

WebView simpleWebView=(WebView) findViewById(R.id.simpleWebView);

// specify the url of the web page in loadUrl function

simpleWebView.loadUrl("http://www.w3schools.com ");

* **loadData() – Tải**[**Html**](http://hiepsiit.com/khoa-hoc/html/ngonnguhtml/51/0)**tĩnh vào WebView**

loadData(String data, String mimeType, String encoding)

Phương thức này sử dụng để tải trang [HTML](http://hiepsiit.com/khoa-hoc/html/ngonnguhtml/51/0) tĩnh vào trong **WebView**. Phương thức loadData có 3 tham số:

**String Data:** Nội dung văn bản trong thẻ HTML

**String mimeType:**cho phép ấn định định dạng mở tập tin cho trình duyệt

**String encoding:** Bảng mã cho văn bản

/\*Add in Oncreate() funtion after setContentView()\*/

// initiate a web view

WebView webView = (WebView) findViewById(R.id.simpleWebView);

// static html string data

String customHtml = "<html><body><h1>Hello, AbhiAndroid</h1>" +

"<h1>Heading 1</h1><h2>Heading 2</h2><h3>Heading 3</h3>" +

"<p>This is a sample paragraph of static HTML In Web view</p>" +

"</body></html>";

// load static html data on a web view

webView.loadData(customHtml, "text/html", "UTF-8");

* **shouldOverrideUrlLoading ()** **- Tải trang web từ xa vào WebView sử dụng WebViewClient:**

WebViewClient giúp chúng tôi giám sát sự kiện trong WebView. Chúng ta phải nạp chồng (Override) lại phương thức shouldOverrideUrlLoading (). Phương thức này cho phép chúng ta thực hiện hành động khi một url được chọn. Chúng ta có thể thiết lập WebViewClient trong WebView bằng phương thức **setWebViewClient ()**.Vi dụ chúng ta tải một url bằng cách sử dụng ứng dụng web view client trong **WebView**.

WebView simpleWebView=(WebView) findViewById(R.id.simpleWebView);

/\*Add in Oncreate() funtion after setContentView()\*/

// set web view client

simpleWebView.setWebViewClient(new MyWebViewClient());

// string url which you have to load into a web view

String url = "http://hiepsiit.com/";

simpleWebView.getSettings().setJavaScriptEnabled(true);

simpleWebView.loadUrl(url); // load the url on the web view

}

// custom web view client class who extends WebViewClient

private class MyWebViewClient extends WebViewClient {

@Override

public boolean shouldOverrideUrlLoading(WebView view, String url) {

view.loadUrl(url); // load the url

return true;

* **canGoBack() – Di chuyển trang một trang web về trước nếu trang web đó tồn tại trong history.**

Phương thức này được sử dụng để xem một WebView có một trang web tồn tại trong mục lịch sử (history) để trở lại hay không. Phương thức này trả về giá trị Boolean đúng hoặc sai. Nếu nó trả về true thì phương thức canGoBack () được sử dụng để di chuyển một trang trước đó. Ví dụ kiểm tra xem trang web có history để trở về trang trước đó?

// initiate a web view

WebView simpleWebView=(WebView)findViewById(R.id.simpleWebView);

// checks whether a web view has a back history item or not

Boolean canGoBack=simpleWebView.canGoBack();

* **canGoForward() –Di chuyển đến trang kế tiếp nếu trang kế tiếp tồn tại trong history.**

Phương thức này được sử dụng để xem một WebView có một trang web tồn tại trong mục lịch sử (history) để di chuyển đến trang kế tiếp hay không. Phương thức này trả về giá trị Boolean đúng hoặc sai. Nếu nó trả về true thì phương thức **canGoForward()** được sử dụng để di chuyển một trang kế tiếp. Ví dụ kiểm tra xem trang web có history để di chuyển đến kế tiếp?

// initiate a web view

WebView simpleWebView=(WebView)findViewById(R.id.simpleWebView);

// checks whether a web view has a forward history item or not

Boolean canGoForword=simpleWebView.canGoForward() ;

* **clearHistory() – Xóa tất cả lịch sử  trong WebView**

Phương thức này dùng để xóa tất cả lịch sử của WebView

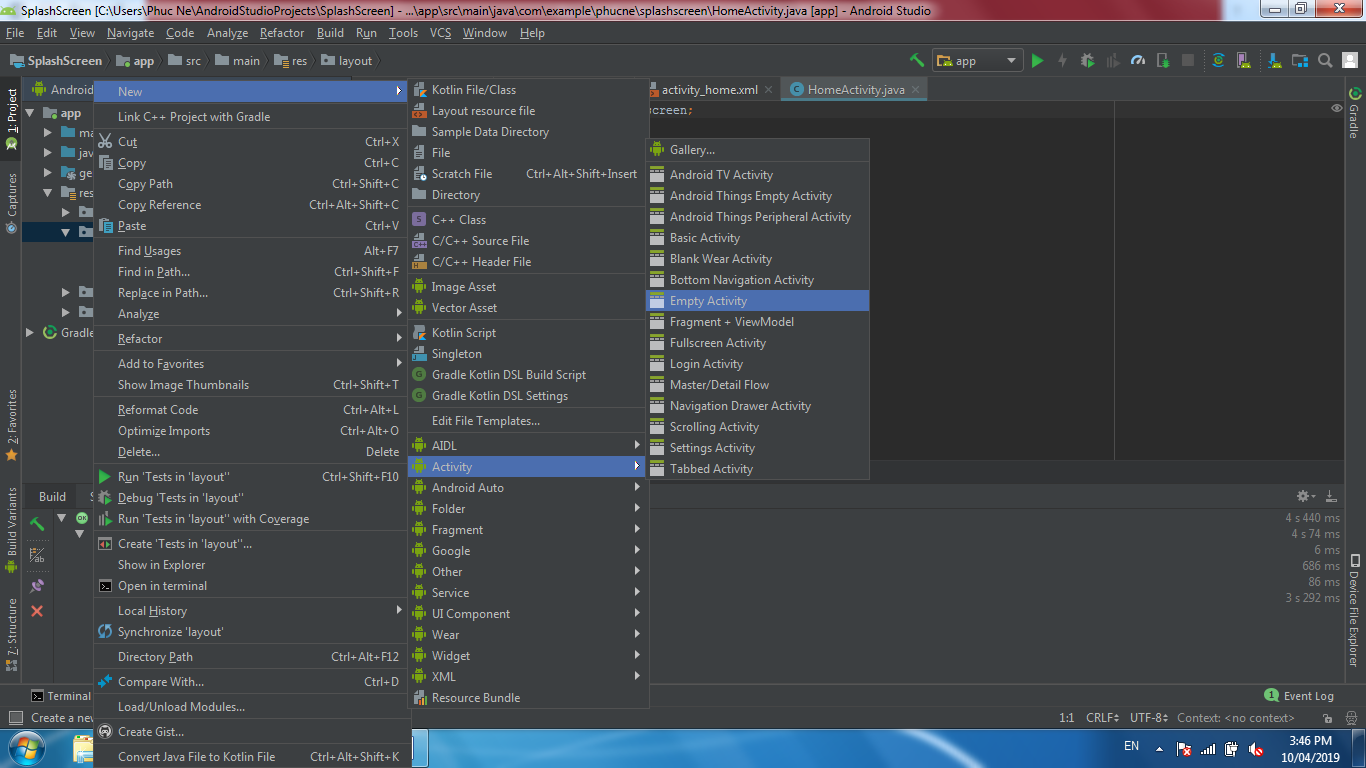
WebView simpleWebView=(WebView)findViewById(R.id.simpleWebView); // initiate a web view

simpleWebView.clearHistory(); // clear the forward and backward history

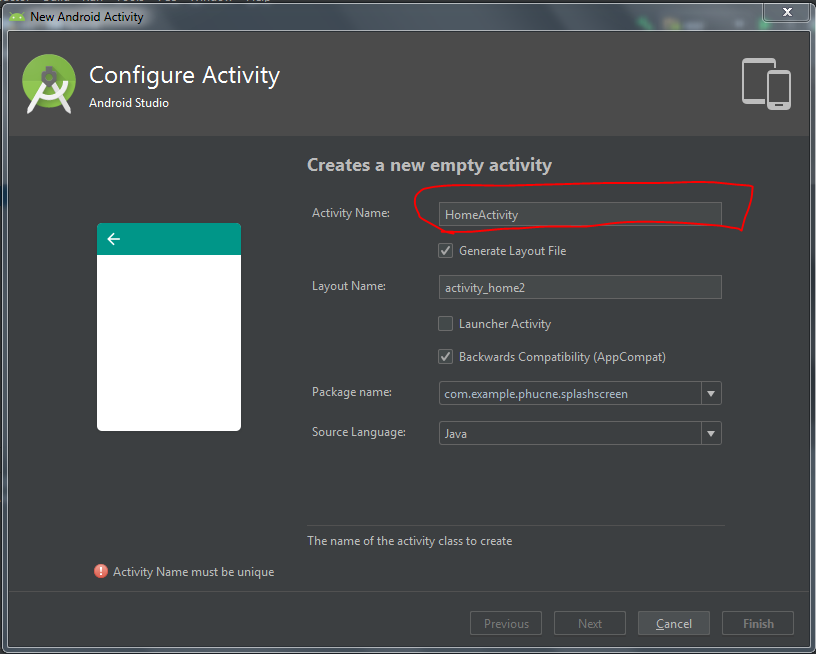
**Hướng dẫn tạo màn hình chờ khởi động app và chuyển đến trang Home**

Sau khi tạo xong 1 project mới

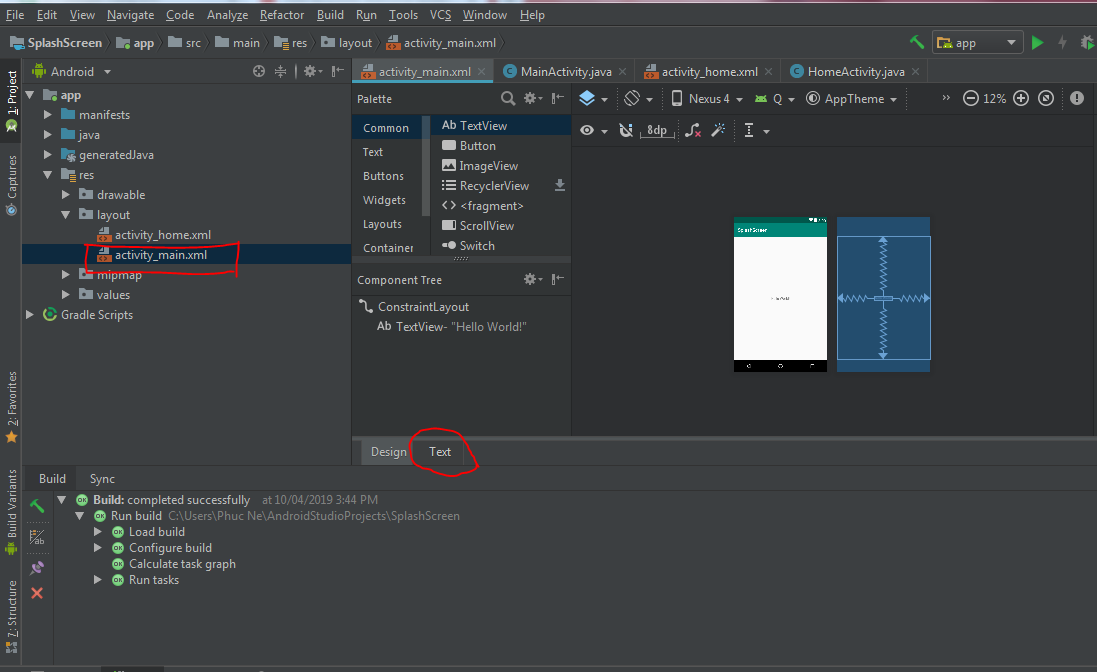
Vào thư mục res-> layout chuột phải chọn new Activiry -> Empty Activiry



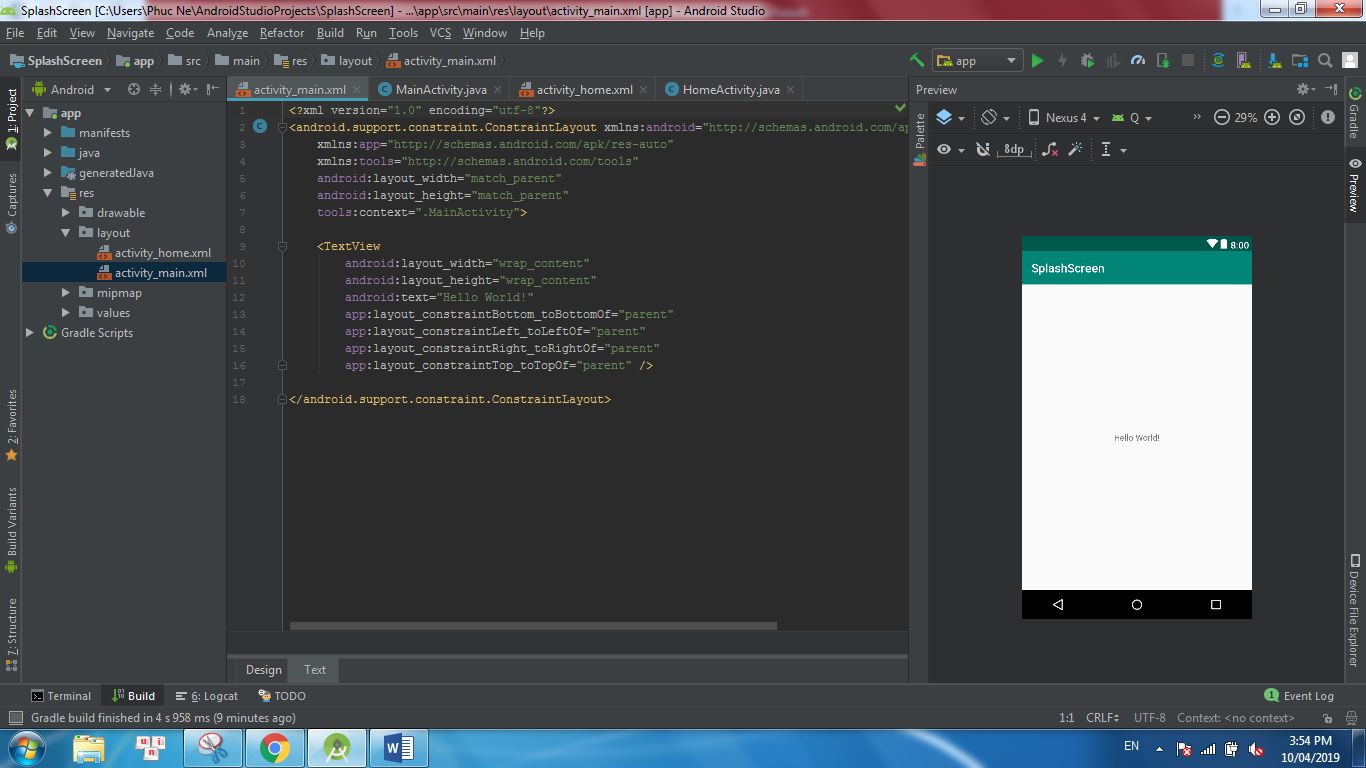
Đặt tên cho Activity



Tiếp theo chọn layout -> activity\_main và chọn text :



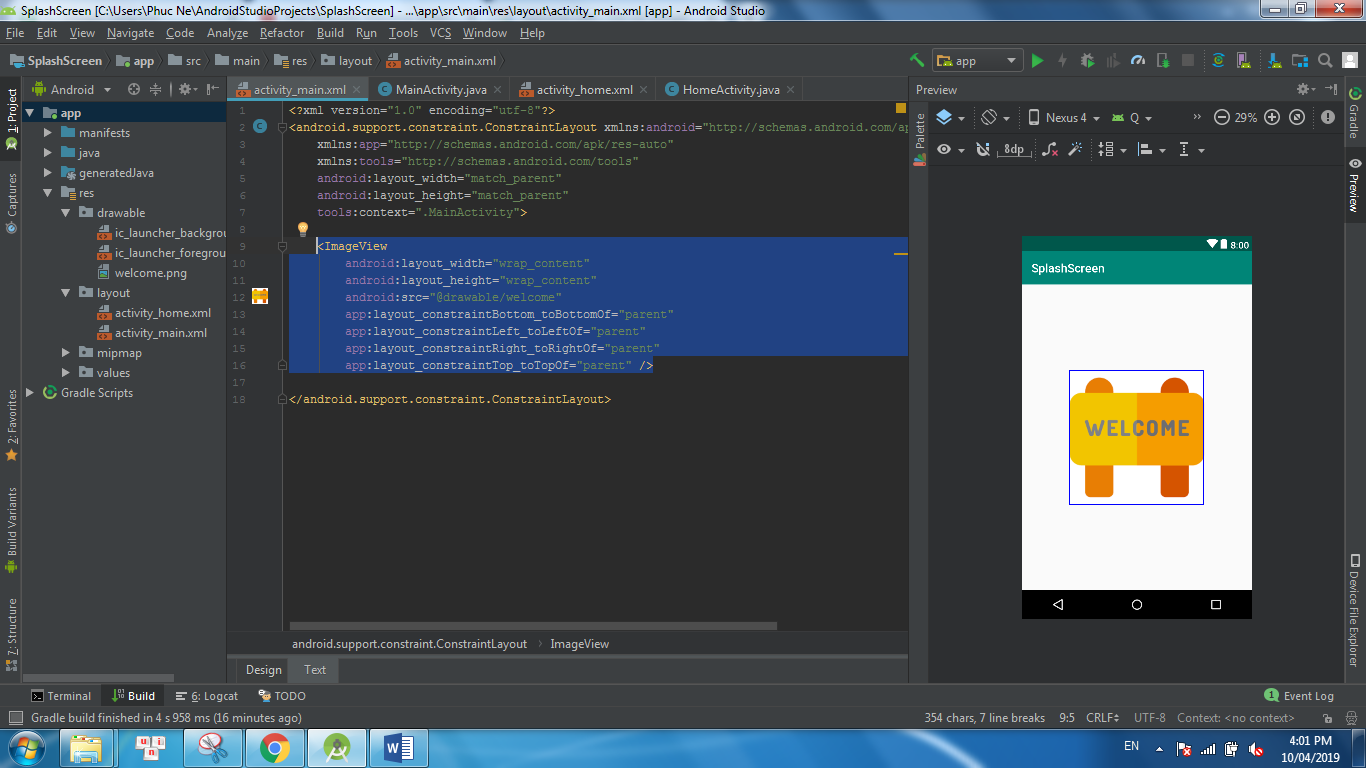
Sẽ giao diện như hình



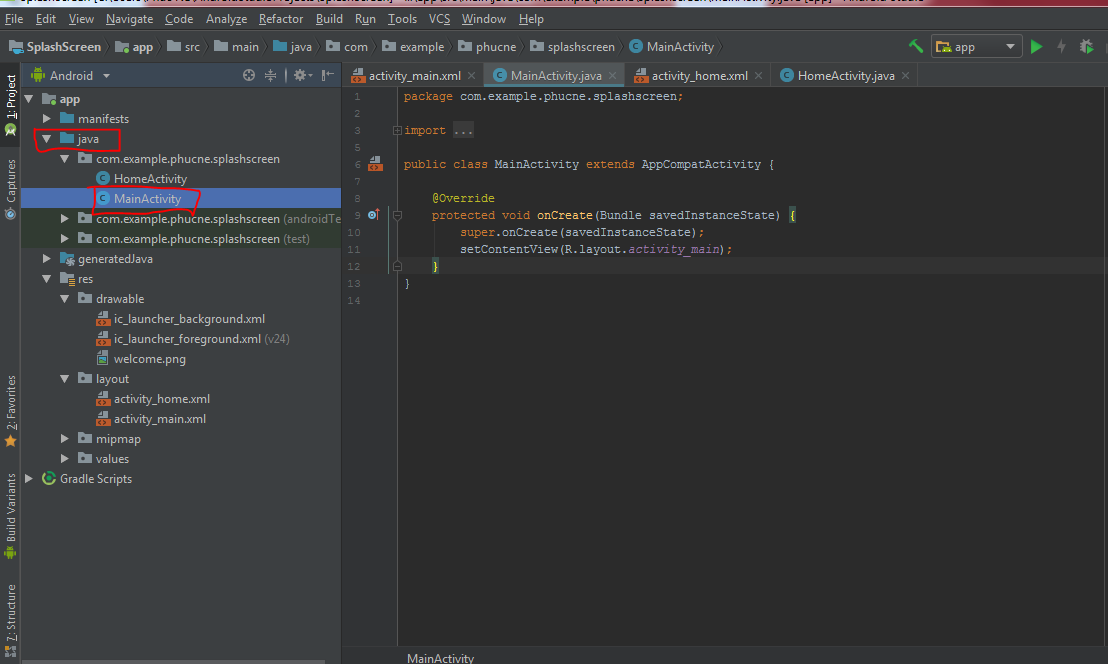
Tiếp theo thêm đoạn code vào :

<ImageView  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:src="@drawable/welcome"  
 app:layout\_constraintBottom\_toBottomOf="parent"  
 app:layout\_constraintLeft\_toLeftOf="parent"  
 app:layout\_constraintRight\_toRightOf="parent"  
 app:layout\_constraintTop\_toTopOf="parent" />

Kết quả như hình dưới (Hình ảnh sẽ được tải và bỏ vào thư mục drawable)



Tiếp theo vào thư mục java -> MainActivity

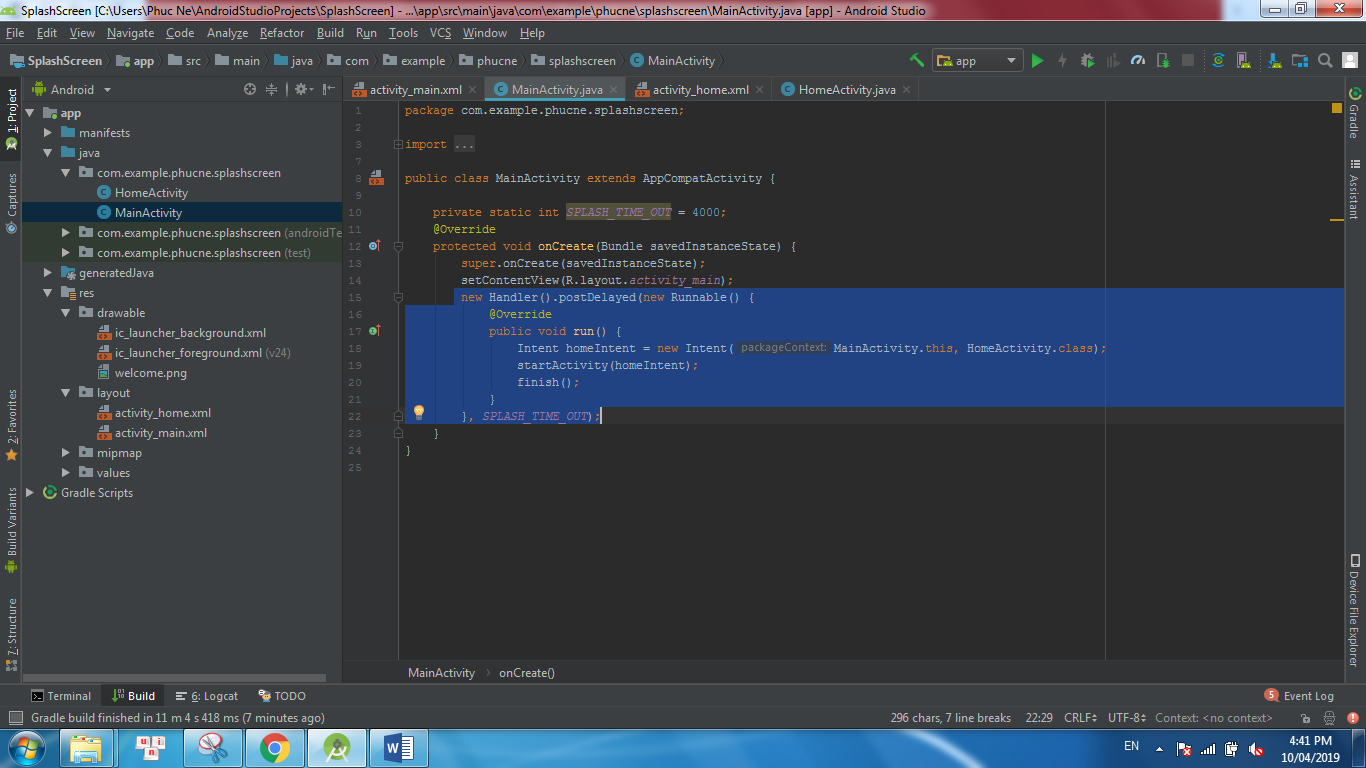


Thêm đoạn code vào MainActivity

private static int *SPLASH\_TIME\_OUT* = 4000;

4000 là thời gian màn hình chờ xuất hiện với đơn vị milisecon

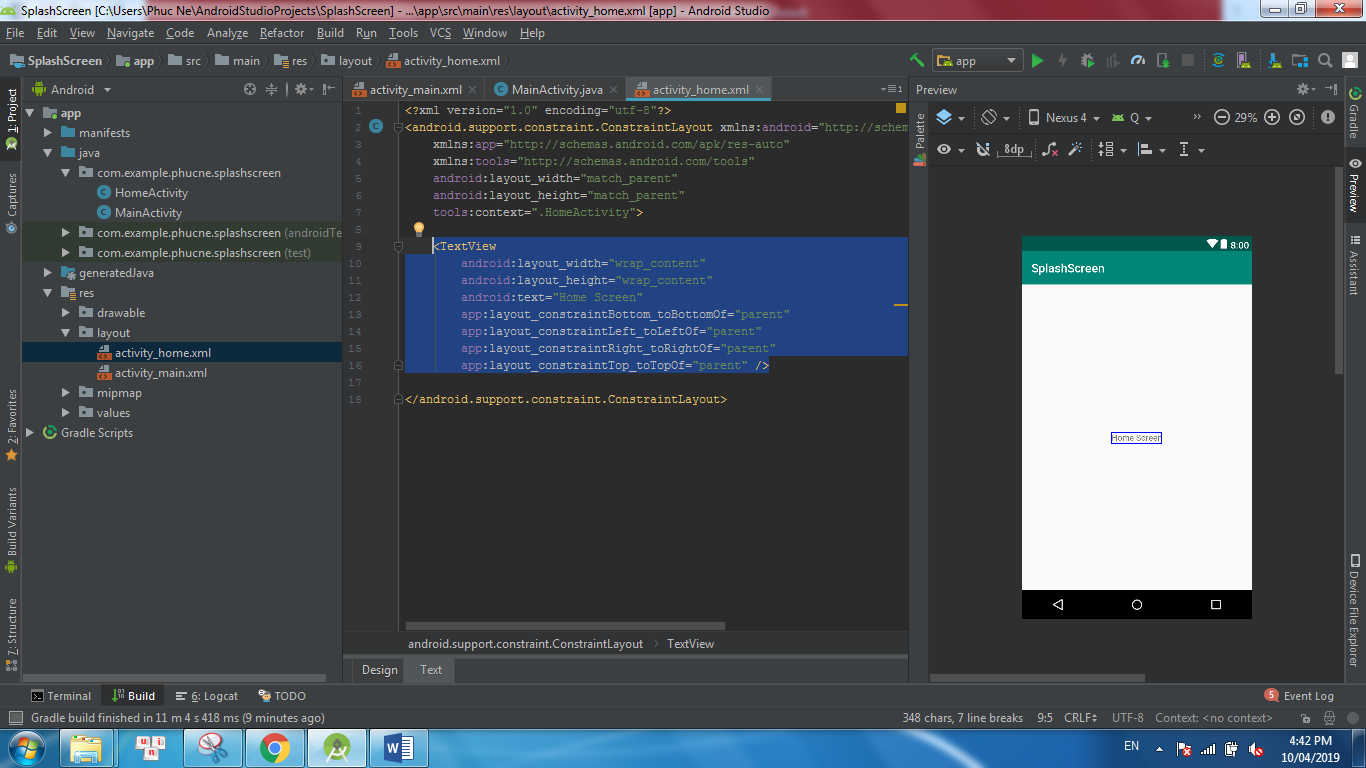
new Handler().postDelayed(new Runnable() {  
 @Override  
 public void run() {  
 Intent homeIntent = new Intent(MainActivity.this, HomeActivity.class);  
 startActivity(homeIntent);  
 finish();  
 }  
}, *SPLASH\_TIME\_OUT*);



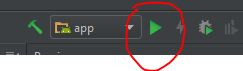
Tiếp theo vào layout -> activity\_home

Thêm đoạn code vào :

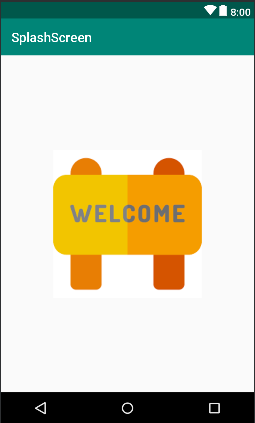
<TextView  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="Home Screen"  
 app:layout\_constraintBottom\_toBottomOf="parent"  
 app:layout\_constraintLeft\_toLeftOf="parent"  
 app:layout\_constraintRight\_toRightOf="parent"  
 app:layout\_constraintTop\_toTopOf="parent" />



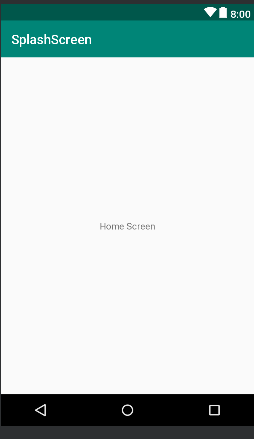
Cuối cùng bấm run



Kết quả demo



Màn hình chờ 4s



Màn hình home