



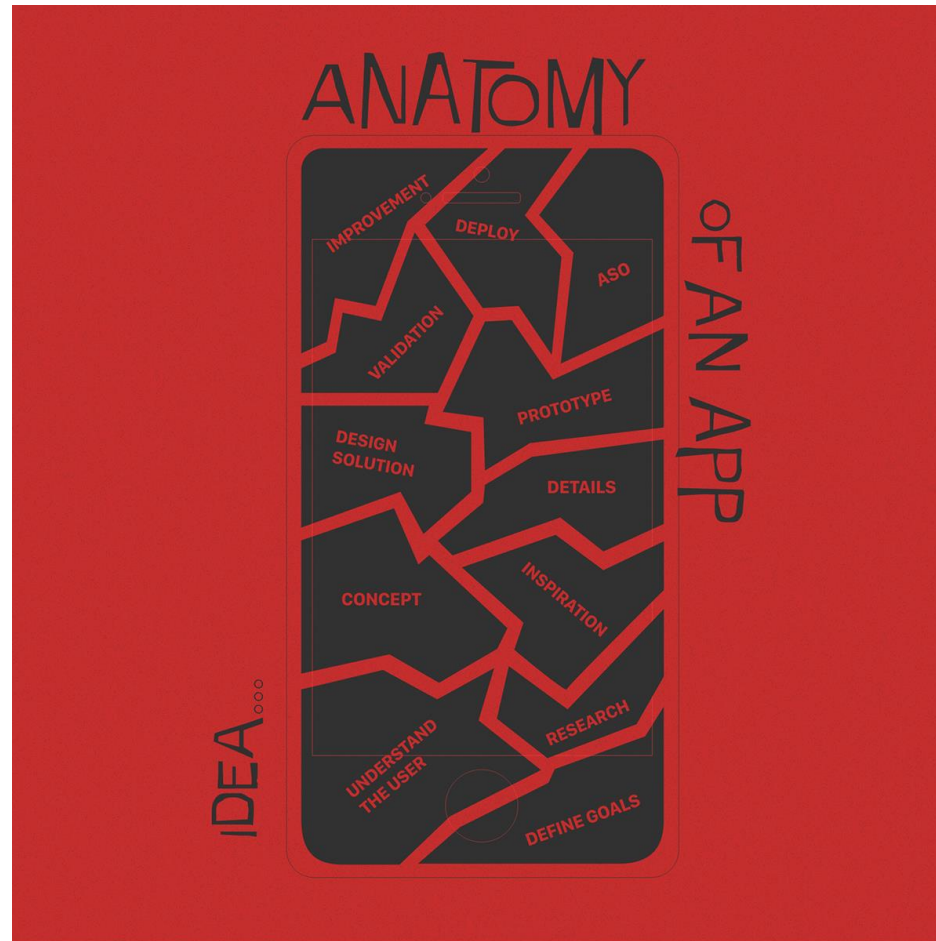
Bảo mật web và ứng dụng

Nội dung

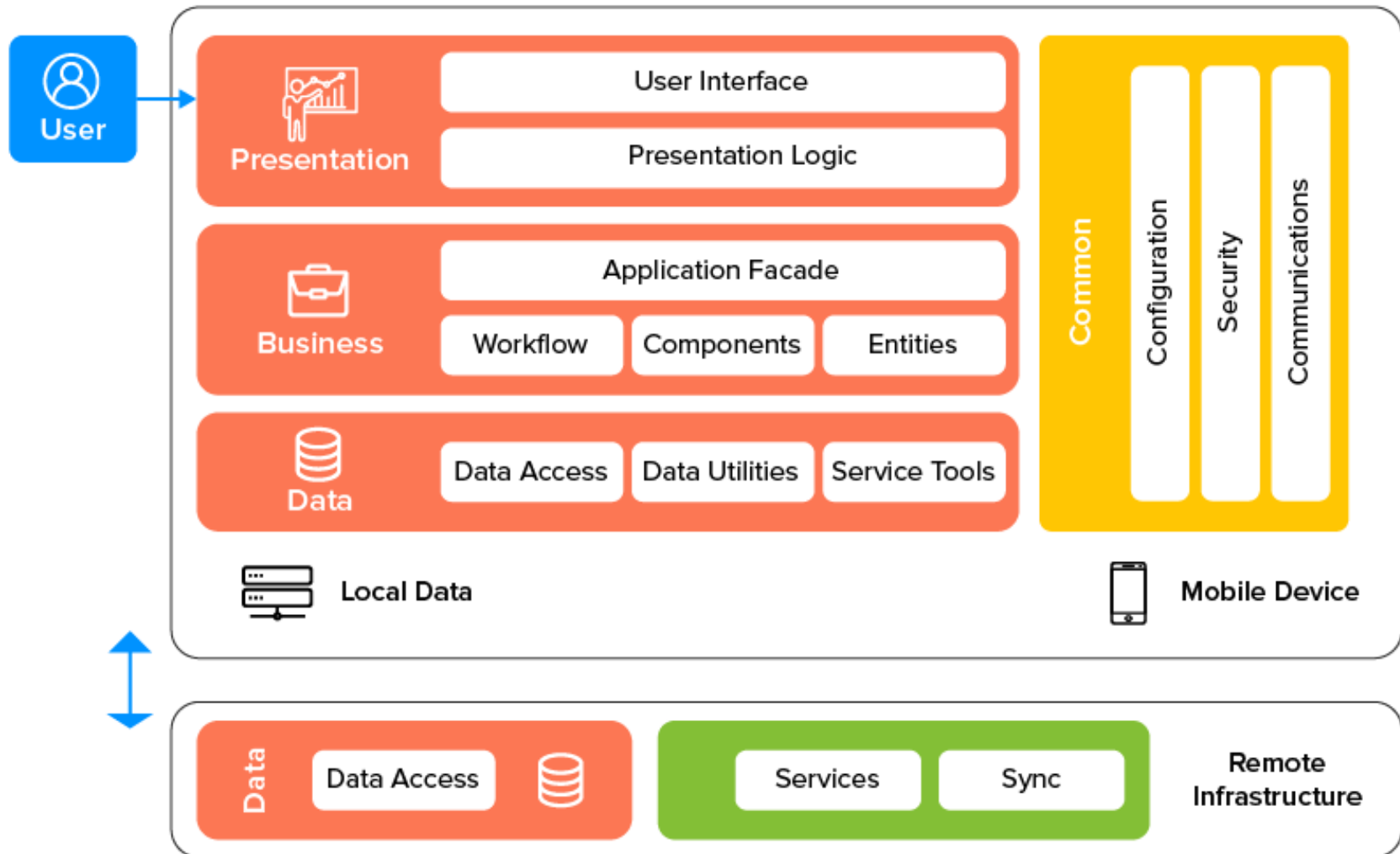
- Ứng dụng **Android**
 - Tổng quan
 - Lập trình ứng dụng Android **với Java**
- Vấn đề an toàn ứng dụng di động Android



Cấu trúc ứng dụng Android



Nguyên tắc hoạt động – kiến trúc



Nguyên tắc hoạt động – kiến trúc

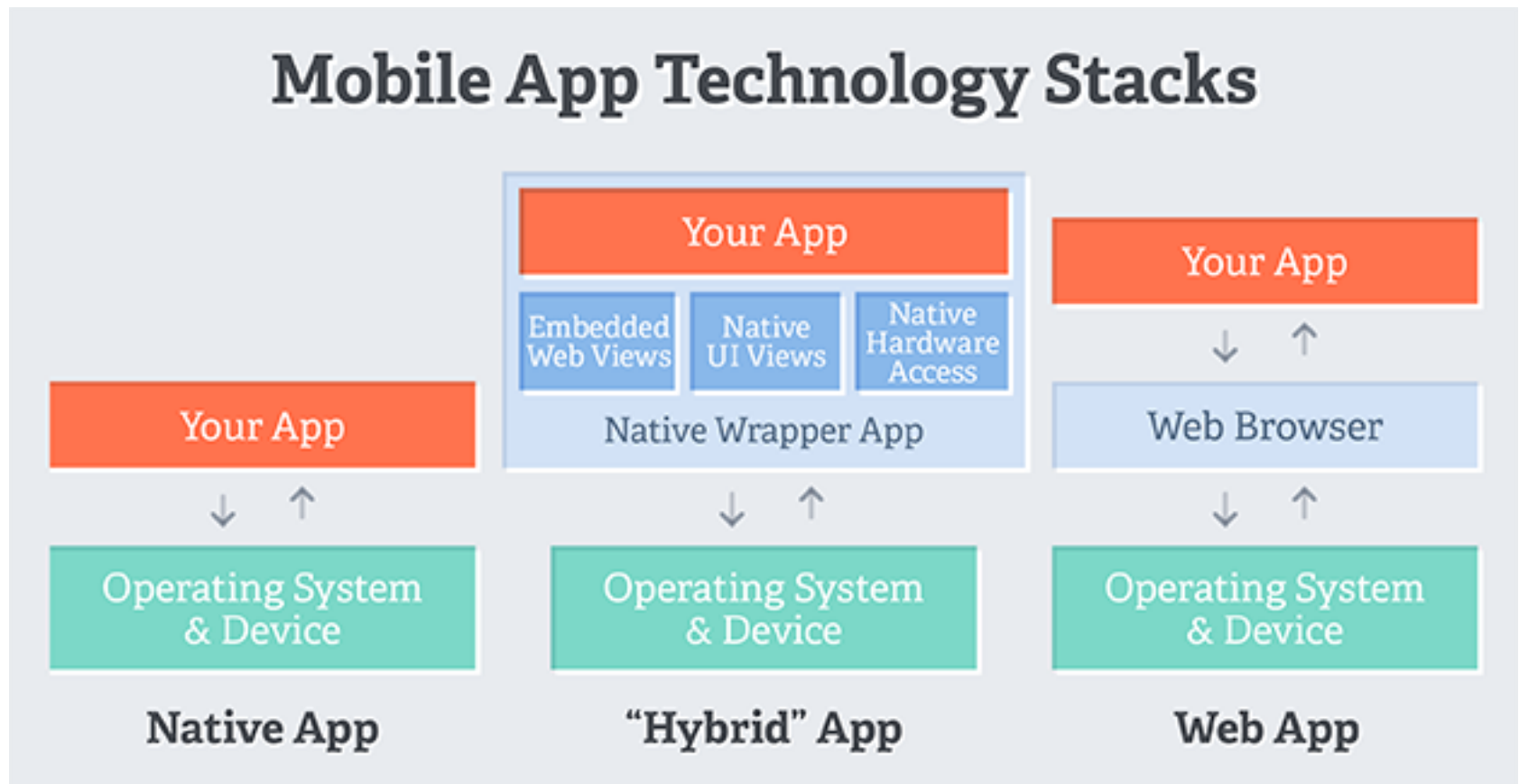


- Kiến trúc ứng dụng di động bao gồm nhiều tầng, bao gồm 3 tầng cơ bản sau:
 - Tầng Hiển thị: giao diện người dùng UI
 - Tầng Xử lý: luồng hoạt động, các thành phần, thực thể xử lý logic dữ liệu
 - Tầng dữ liệu: dữ liệu, truy cập dữ liệu

Chọn lựa nền tảng ứng dụng



- Sự khác biệt giữa các loại ứng dụng



Chọn lựa nền tảng ứng dụng



Native app	Hybrid app	Web app
<ul style="list-style-type: none">- Được phát triển cho platform cụ thể- Hỗ trợ đầy đủ từ app store- Chạy ổn định hơn- Có thể sử dụng tất cả tính năng của hệ điều hành- Cần phải được phê duyệt nên bảo mật hơn, đảm bảo chất lượng và tương thích với thiết bị	<ul style="list-style-type: none">- Được cài đặt trên thiết bị như native app, về kỹ thuật là một web app.- Không cần trình duyệt web- Có thể sử dụng các API và phần cứng trong thiết bị	<ul style="list-style-type: none">- Sử dụng thông qua trình duyệt web, hỗ trợ nhiều nền tảng di động- Không cần tải về từ app store- Dễ phát triển, bảo trì vì sử dụng mã nguồn chung cho các nền tảng- Không tốn bộ nhớ để lưu trữ trên thiết bị- Không cần thông qua phê duyệt của app store- Cập nhật chủ động
<ul style="list-style-type: none">- Người phát triển cần có kinh nghiệm với ngôn ngữ lập trình khó- Cần cài đặt trên thiết bị- Tốn chi phí	<ul style="list-style-type: none">- Chậm hơn native app- Phụ thuộc vào platform của bên thứ 3 để triển khai wrapper	<ul style="list-style-type: none">- Cần kết nối mạng và trình duyệt- Không có đầy đủ tính năng như native app.- Sử dụng các tính năng của thiết bị và phần cứng- Không được liệt kê trong app store

Lập trình ứng dụng di động



Trước khi lập trình ứng dụng di động cần xem xét **các yếu tố** sau:

- Xác định **kiểu thiết bị** mà ứng dụng sẽ chạy:
 - Độ phân giải màn hình
 - Kích thước màn hình
 - Tính năng CPU
 - Không gian lưu trữ
 - Bộ nhớ
 - Tính sẵn sàng của các framework phát triển
- Bandwidth situation (băng thông/ điện năng tiêu thụ)
 - Chọn giao thức phù hợp trên từng loại thiết bị
 - Caching, state management, data access

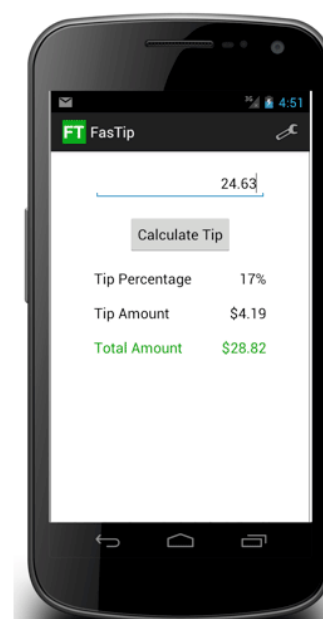
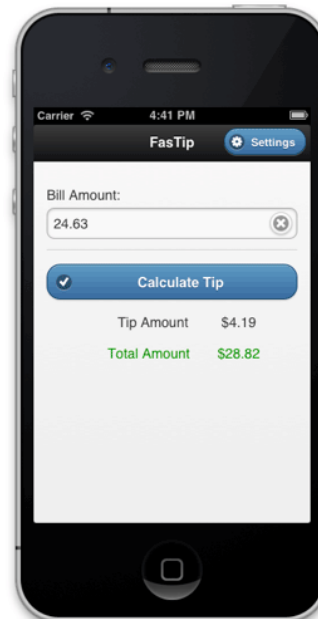
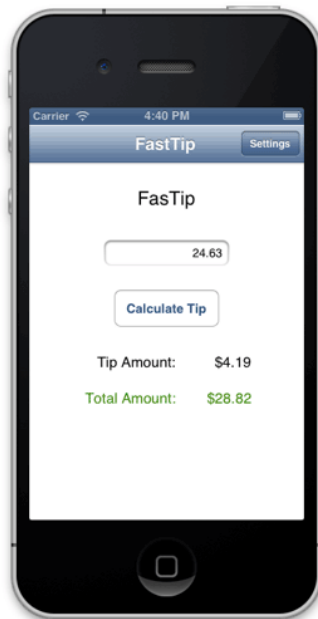


Lập trình ứng dụng di động



Trước khi lập trình ứng dụng di động cần xem xét **các yếu tố** sau:

- Thiết lập giao diện: đơn giản, dễ sử dụng
- Điều hướng phù hợp



Android



- **Android** là một **software stack** cho các thiết bị di động, bao gồm hệ điều hành, middleware và các ứng dụng cốt lõi
- Android và các ứng dụng của nó được phát triển bằng ngôn ngữ **JAVA/Kotlin**
- **Dalvik VM**, được tùy chỉnh và tối ưu để chạy trên các thiết bị di động
- **Android SDK** cung cấp các công cụ cho phát triển ứng dụng Android và nhiều API hữu ích



APPLICATIONS

Home

Contacts

Phone

Browser

...

APPLICATION FRAMEWORK

Activity Manager

Window
Manager

Content
Providers

View
System

Notification
Manager

Package Manager

Telephony
Manager

Resource
Manager

Location
Manager

GTalk Service

LIBRARIES

Surface Manager

Media
Framework

SQLite

OpenGL | ES

FreeType

WebKit

SSL

SSL

libc

ANDROID RUNTIME

Core Libraries

Dalvik Virtual
Machine

LINUX KERNEL

Display
Driver

Camera Driver

Bluetooth
Driver

Flash Memory
Driver

Binder (IPC)
Driver

USB Driver

Keypad Driver

WiFi Driver

Audio
Drivers

Power
Management

Linux Kernel

- Lưu ý rằng: Android dựa trên kernel Linux không phải là một hệ điều hành Linux.
- Cung cấp cơ chế Security, quản lý bộ nhớ (Memory management), quản lý tiến trình (Process management), Network stack và Driver model
- Hoạt động như một lớp trừu tượng giữa phần cứng và các phần khác của software stack.



Libraries

- Chạy ngầm trong hệ thống
- Viết bằng ngôn ngữ C/C++
- 4 loại Libraries
 - Bionic Libc, system C libraries
 - Function Libraries, supporting multimedia, web browser, SQLite...
 - Native Servers
 - Hardware Abstraction Libraries



Android Runtime



- **Cốt lõi của Android platform**
- Dalvik Virtual Machine (DVM)
 - Register-based
 - Thực thi các files định dạng Dalvik Executable (.dex)
- Các thư viện Java lõi
 - Hỗ trợ hầu hết các chức năng trong ngôn ngữ Java.



Android Runtime (tt)



- Chức năng của các thư viện Java cốt lõi tùy vào Dalvik VM và Linux kernel bên dưới.
- Có thể có nhiều Dalvik VMs chạy cùng lúc.
- **Mỗi ứng dụng Android** chạy trong tiến trình riêng của nó, với **một instance máy ảo Dalvik** riêng.
- Công cụ "**dx**" trong Android SDK: chuyển **class JAVA** đã biên dịch sang **định dạng .dex**

Dalvik Virtual Machine (DVM)



- Máy ảo tùy chỉnh cho Android
 - Application portability và runtime consistency
 - Chạy các file định dạng .dex và Dalvik bytecode
 - Các file Java .class/.jar được chuyển sang .dex khi build
- Được thiết kế cho môi trường nhúng
 - Hỗ trợ nhiều tiến trình máy ảo trên 1 thiết bị
 - Trình thông dịch bytecode được tối ưu hóa cho CPU
 - Sử dụng hiệu quả bộ nhớ trong thời gian chạy
- Core Libraries
 - APIs cốt lõi cho Java, cung cấp platform phát triển mạnh mẽ nhưng cũng đơn giản và quen thuộc

DVM vs. JVM



- DVM
 - Phát triển bởi Google
 - Chạy các file thực thi Dalvik (.dex)
 - Chỉ hỗ trợ một phần các thư viện Java chuẩn
- JVM
 - Phát triển bởi Sun
 - Chạy Java bytecode
- Vấn đề: có thể có nhiều cộng đồng về JAVA với những chuẩn JAVA khác nhau

Application Framework

- Đơn giản hóa việc sử dụng lại các components
 - Các ứng dụng có thể công bố các tính năng để các ứng dụng khác có thể sử dụng
- Tập hợp các services và systems
 - Views system,
 - Content providers
 - Resources manager
 - ...



Application Framework (tt)



- **Activity Manager:** quản lý vòng đời của các ứng dụng và cung cấp các điều hướng
- **Notification Manager:** cho phép các ứng dụng hiển thị các thông báo trên khung trạng thái
- **Resource Manager:** cung cấp khả năng truy cập vào các tài nguyên không phải mã nguồn như chuỗi, hình ảnh, các file layout...
- **Content Providers:** truy cập đến dữ liệu của các ứng dụng khác hoặc chia sẻ dữ liệu của chính ứng dụng
- **Views:** dùng để xây dựng ứng dụng, gồm danh sách, các khung nhập liệu, nút, hoặc thậm chí trình duyệt nhúng

Applications



- Tập các ứng dụng cốt lõi được cung cấp với Android platform
 - Email client, SMS, calendar, maps, browser, contacts...
- Lập trình bằng Java
- Các ứng dụng Android được phát triển sau sẽ được đặt cùng mức với những ứng dụng này



A large, stylized green graphic resembling a thick, continuous line that forms a complex, abstract shape. It has several loops and curves, with some parts being solid green and others having a lighter green gradient. The graphic is positioned on the left side of the slide, partially overlapping the text area.

Android software development

Lập trình ứng dụng **Android**

Ngôn ngữ lập trình Java



Java

- Phát triển bởi **Sun Microsystems** (hiện thuộc sở hữu của Oracle).
- Java kết hợp nhiều tính năng mạnh mẽ của nhiều ngôn ngữ.

Một số tính năng cốt lõi quan trọng của Java

- Nó được thiết kế để có nền tảng độc lập và an toàn, sử dụng cho các máy ảo.
- Là ngôn ngữ **lập trình hướng đối tượng**
- Android dựa chủ yếu vào các nguyên tắc cơ bản Java.
- **Android SDK** bao gồm **nhiều thư viện Java chuẩn** (thư viện cấu trúc dữ liệu, thư viện toán học, thư viện đồ họa, thư viện mạng...)



Lập trình ứng dụng Android



- JDK/Android JDK + Eclipse
- **Android Studio** (khuyến nghị)



Android Studio



- IDE dành cho phát triển ứng dụng Android
- Đã bao gồm Android SDK



Android Studio provides the fastest tools for building apps on every type of Android device.

DOWNLOAD ANDROID STUDIO

4.1 for Windows 64-bit (896 MB)

<https://developer.android.com/>

Android Emulator cho app testing



GENYMOTION^{oo}
By Genymobile

<https://www.genymotion.com/>



“Hello World” on Android

Ứng dụng Android đầu tiên

Tạo một **Android Project** mới



- Chọn **File** → **New** → **Project**
- Chọn **Android/Android Application project**, và điền thông tin:
 - Tên project
 - Tên ứng dụng
 - Tên package
 - SDK mục tiêu
 - Create Activity

New Android Application

Creates a new Android Application

Application Name: HelloWorld

Project Name: HelloWorld

Package Name: com.demo.helloworld

Minimum Required SDK: API 8: Android 2.2 (Froyo)

Target SDK: API 16: Android 4.1 (Jelly Bean)

Compile With: API 16: Android 4.1 (Jelly Bean)

Theme: Holo Light with Dark Action Bar

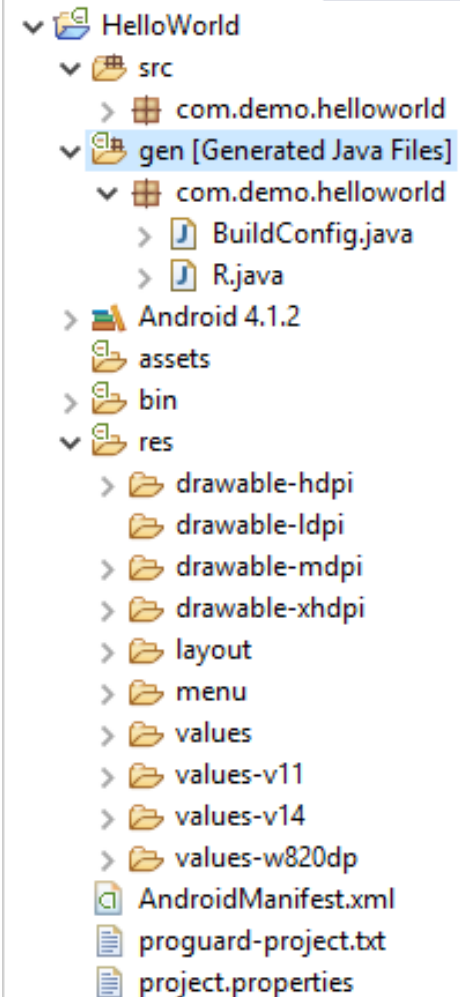
Choose the base theme to use for the application

< Back Next > Finish Cancel

Hello World Project



- **src**: thư mục mã nguồn
- **gen**: file SDK được tạo
- **android 4.1.2**: reference lib
- **assets**: tài nguyên
- **res**: resource files và các file mô tả
- **AndroidManifest.xml**: file mô tả ứng dụng
- **project.properties**: file thuộc tính của project



Say Hello World



- Điều chỉnh file **HelloWorld.java**

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

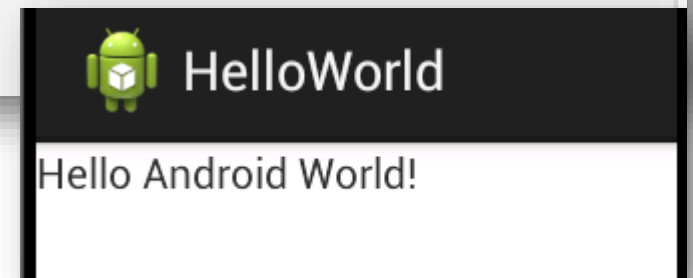
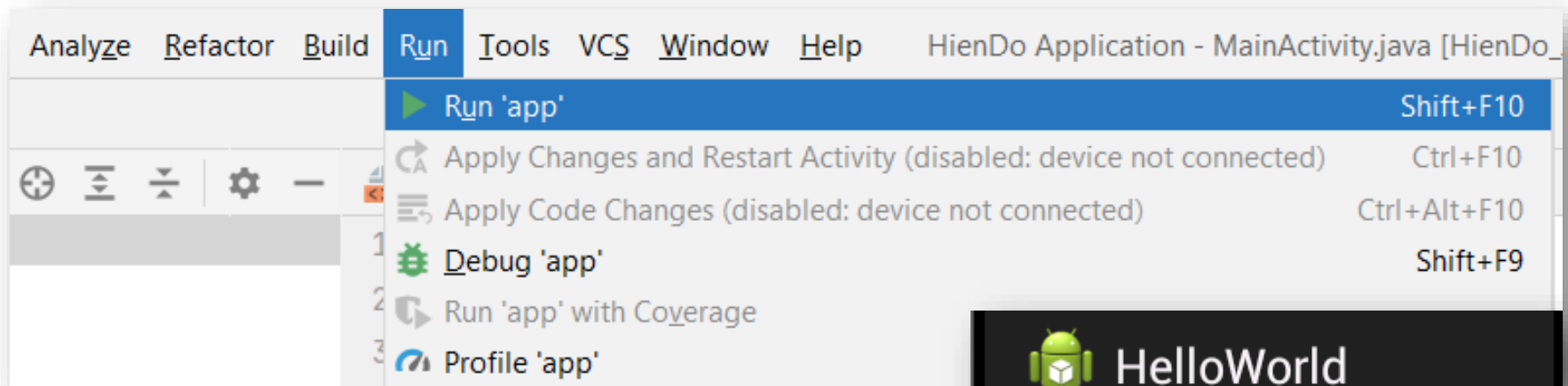
```
public void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    TextView text = new TextView(this);  
    text.setText("Hello Android World!");  
    setContentView(text);  
}
```



Chạy ứng dụng **Hello World**



- Chọn *HelloWorld* Project, sau đó chọn **Run** → **Run 'app'** → **Android Application**
- ADT sẽ chạy một AVD phù hợp và chạy ứng dụng HelloWorld trên đó



Vọc ứng dụng HelloWorld



- **res/layout** , chứa các định nghĩa thiết kế bố cục trong định dạng XML, giao diện của ứng dụng được built dựa trên các file layout

main.xml

Linear Layout

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context="com.demo.helloworld.HelloWorld" >
    <TextView android:id="@+id/textView"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/hello_world" />
</LinearLayout>
```

TextView, hiển thị
các chuỗi

Tham chiếu đến
resource string
'hello world'

Vọc ứng dụng HelloWorld



- **res/values**, chứa các định nghĩa chuỗi hoặc các giá trị khác (ví dụ: màu sắc) trong ứng dụng
 - **string.xml**, chứa các resources dạng string (chuỗi)

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="app_name">Hello World</string>
    <string name="hello_world">Hello world!</string>
    <string name="action_settings">Settings</string>
</resources>
```

Được tham chiếu
trong
res/layout/main.xml

Được tham chiếu
trong
AndroidManifest.xml

Vọc ứng dụng HelloWorld



- **res/drawable**, chứa các resources hình ảnh
 - Thư mục có thể có nhiều suffixes, ứng dụng sẽ lựa chọn cái phù hợp nhất
 - 3 thư mục: drawable-ldpi, drawable-hdpi, drawable-mdpi, mỗi thư mục chứa file icon.png
 - Ứng dụng sẽ chọn file icon phù hợp dựa trên DPI của thiết bị
 - Tên tham chiếu: @drawable/icon
- Các thư mục khác có thể được dùng đến khi cần
 - menu, anim (animation), xml (preference and searchable)



Vọc ứng dụng HelloWorld



- **AndroidManifest.xml** mô tả ứng dụng

- Định nghĩa tên app, phiên bản, icon, permission, etc...
- Định nghĩa các thành phần ứng dụng: activity, service, receiver or provider

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
package="com.demo.helloworld" android:versionCode="1" android:versionName="1.0" >
    <uses-sdk android:minSdkVersion="16" android:targetSdkVersion="16" />
    <application android:allowBackup="true" android:icon="@drawable/ic_launcher"
android:label="@string/app_name" android:theme="@style/AppTheme" >
        <activity android:name=".HelloWorld" android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

Cấu hình tập tin Manifest



- Mọi ứng dụng đều cần có một file **AndroidManifest.xml** ở thư mục gốc của project.
- File mô tả này để cung cấp các thông tin cần thiết về ứng dụng cho hệ thống Android.
- File mô tả Manifest này cần theo cấu trúc sau:
 - Chứa **tên package** dùng cho ứng dụng (tên package là **duy nhất** nhằm xác định đến package).
 - Mô tả **các thành phần** của ứng dụng: các Activity, Service, Broadcast Receiver, Content Provider.
 - Xác định **các tiến trình** chạy trong ứng dụng.
 - Khai báo **các quyền truy cập** hệ thống.

Cấu hình tập tin Manifest: Ví dụ 1



```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
package="sample.hello" android:versionCode="1" android:versionName="1.0">

<application android:icon="@drawable/icon"
android:label="@string/app_name">
    <activity android:name=".HelloWorld" android:label="@string/app_name">
        <intent-filter>
            <action android:name="android.intent.action.MAIN" />
            <category android:name="android.intent.category.LAUNCHER"/>
        </intent-filter>
    </activity>
</application>

<uses-sdk android:minSdkVersion="8" />
</manifest>
```


Thành phần cốt lõi - **Activity**



- Về cơ bản, một *Activity* hiển thị một ***giao diện người dùng***
- Một ứng dụng có thể có 1 hoặc nhiều Activity, mỗi Activity kế thừa từ ***android.app.Activity*** và cần được định nghĩa trong **AndroidManifest.xml**
- Mỗi activity được cung cấp 1 window để hiển thị, window này có thể toàn màn hình hoặc nhỏ hơn và nằm bên trên window khác
- Giao diện của window được dựng bằng một hệ phân cấp các object từ View class
- Phương thức **Activity.setContentView()** thiết lập một phân cấp các đối tượng view

Thành phần cốt lõi - Activity

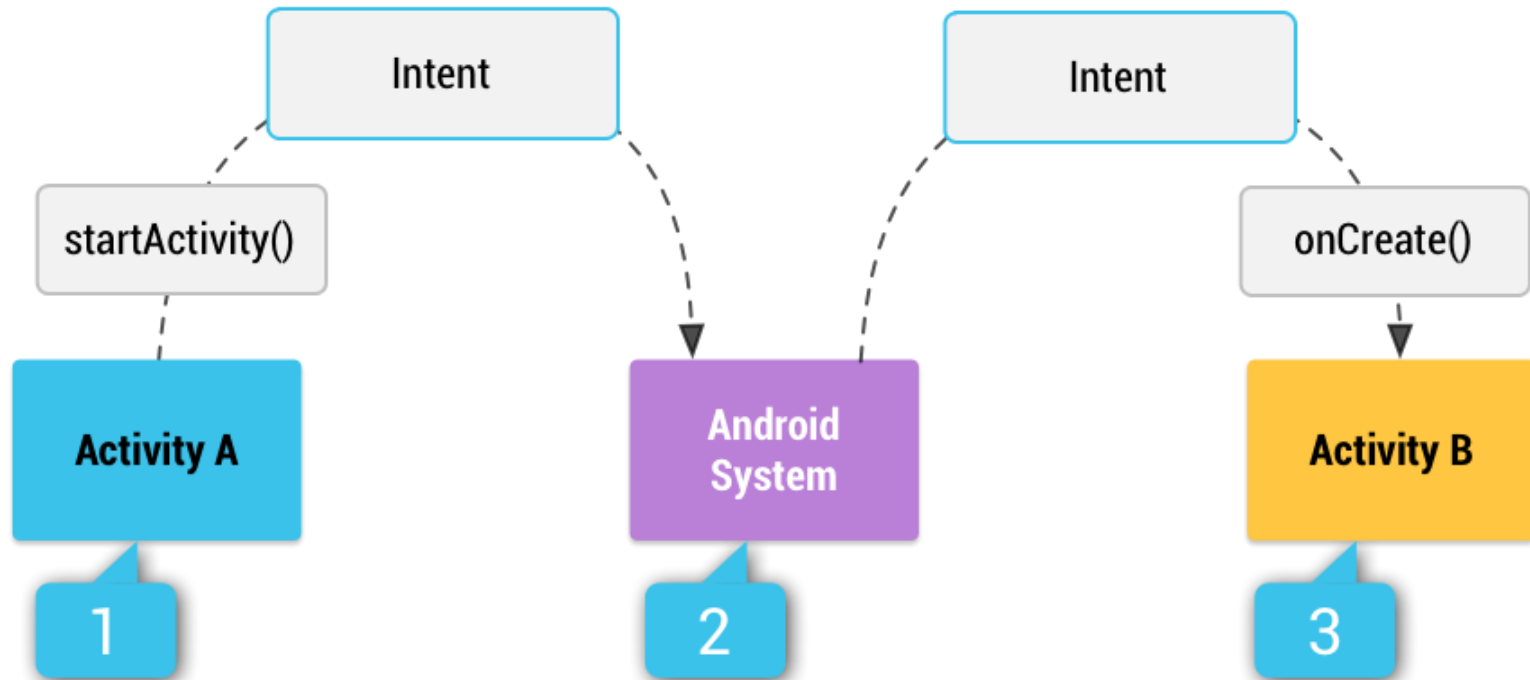


- **Activity** được kích hoạt bằng các message bắt đồng bộ được gọi là *intents*
 - Một Intent object chứa nội dung của message
- Label **<intent-filter>** trong **AndroidManifest.xml** xác định Intent có thể khởi chạy Activity

```
<intent-filter>  
    <action android:name="android.intent.action.MAIN" />  
    <category android:name="android.intent.category.LAUNCHER"/>  
</intent-filter>
```

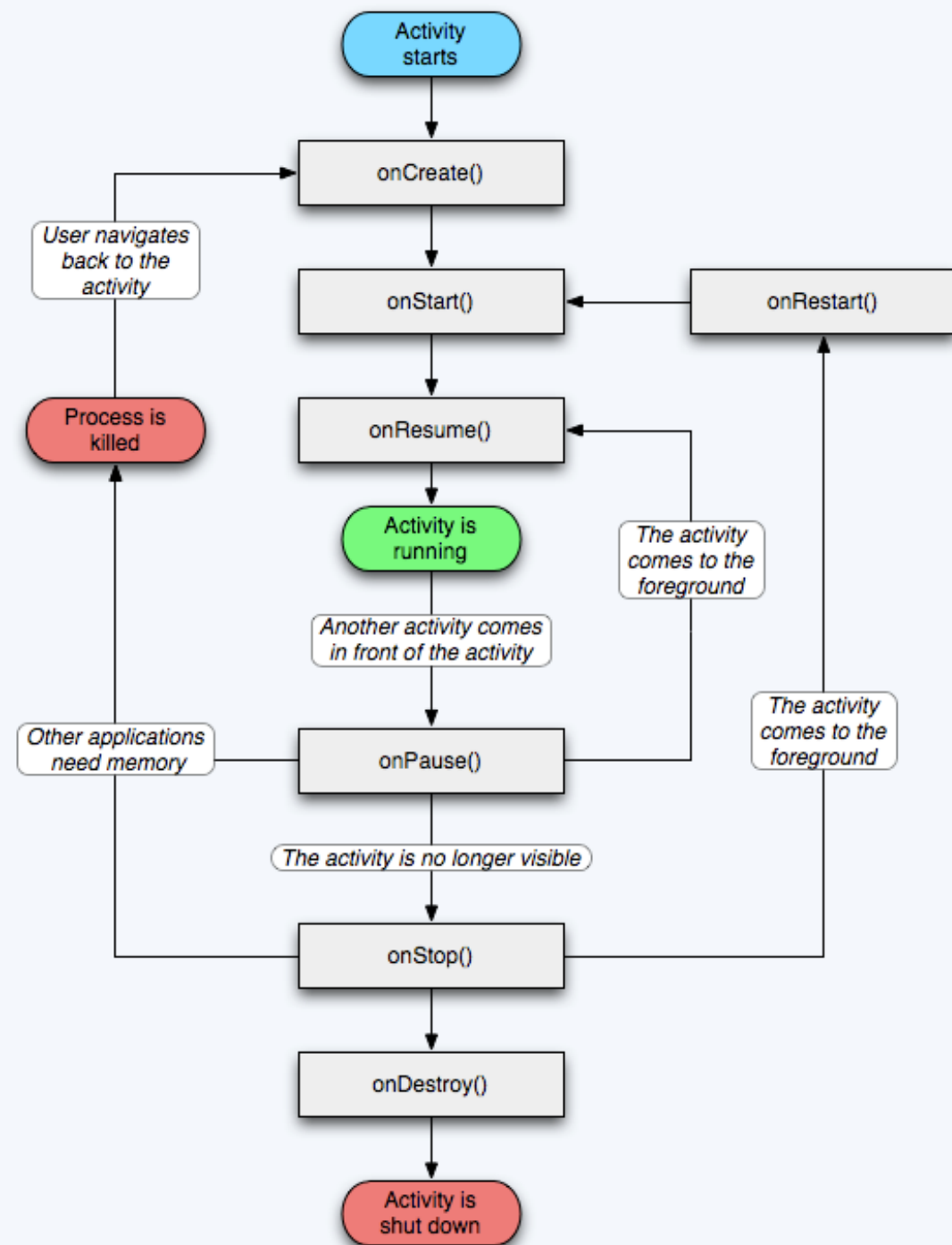
- Định nghĩa activity chính, được chạy tự động khi ứng dụng khởi chạy
- Một activity có thể được thực thi bằng cách truyền một đối tượng Intent cho **Context.startActivity()** hoặc **Activity.startActivityForResult()**

Thành phần cốt lõi - Activity



- **Explicit intent:** intent nêu rõ tên thành phần (ví dụ activity) sẽ chạy → Android chạy thành phần này ngay lập tức
- **Implicit intent:** intent chỉ định nghĩa hành động (action) cần chạy, Android sẽ tìm thành phần thỏa mãn trong file Manifest bằng intent-filters. Nếu nhiều hơn 1 kết quả, hệ thống sẽ hỏi người dùng

Vòng đời của Activity



Các thành phần cốt lõi khác



- Service
 - Một *service* không có giao diện người dùng, chạy ngầm trong một khoảng thời gian nào đó
- Broadcast receivers
 - Thành phần nhận và phản ứng với những thông báo được broadcast
- Content providers
 - Một *content provider* cho phép cung cấp một số dữ liệu của ứng dụng cho các ứng dụng khác.
 - Dữ liệu có thể được lưu trữ trong file system, trong SQLite database, hoặc bất kỳ cách nào khác

AndroidManifest.xml: Ví dụ 2



Mô tả một receiver

```
<receiver android:directBootAware=["true" | "false"] <!--Có nhận được Broadcast khi khóa máy hay không-->
    android:enabled=["true" | "false"] <!--Có kích hoạt hay không-->
    android:exported=["true" | "false"] <!--Có nhận được Broadcast từ thành phần bên ngoài ứng dụng hay không-->
    android:icon="drawable resource"
    android:label="string resource"
    android:name="string"
    android:permission="string"
    android:process="string" >

    . . .
</receiver>
```

Mô tả một intent-filter

```
<intent-filter android:icon="drawable resource"
    android:label="string resource"
    android:priority="integer" >

    . . .
</intent-filter>
```

Cơ sở dữ liệu SQLite trong ứng dụng Android



- Lưu dữ liệu cho từng ứng dụng, không bị truy cập bởi ứng dụng khác.
- Thư mục:
/data/data/<applicationId>/databases/
- Khai báo trong file AndroidManifest để thao tác CSDL:

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

Cơ sở dữ liệu SQLite trong ứng dụng Android



- Mở, tạo một SQLite DB
- Truy vấn
- Xóa

<https://developer.android.com/reference/android/database/sqlite/SQLiteDatabase>

Mở rộng Hello World (1)



- Dựng một ứng dụng cho phép nhập chuỗi chào mừng và sau đó hiển thị
 - Nhập chuỗi: EditText
 - Hiển thị: TextView
 - Một button (nút) để submit chuỗi
- Sửa **res/layout/main.xml** để thêm các thành phần
 - Mỗi thành phần đều có thuộc tính **android:id**, dùng để tham chiếu đến trong code

```
<EditText android:text="" android:id="@+id/editText"
    android:layout_width="fill_parent" android:layout_height="wrap_content"/>
<Button android:text="Show Greetings" android:id="@+id/showBtn"
    android:layout_width="wrap_content" android:layout_height="wrap_content"/>
<TextView android:id="@+id/textView" android:layout_width="fill_parent"
    android:layout_height="wrap_content" android:text="@string/hello_world" />
```

Mở rộng Hello World (2)



- Chỉnh sửa **HelloWorld.java**
 - Đầu tiên tham chiếu đến các thành phần trong **main.xml**

```
setContentView(R.layout.main);  
final EditText edit = (EditText) findViewById(R.id.editText);  
final TextView view = (TextView) findViewById(R.id.textView);  
final Button btn = (Button) findViewById(R.id.showBtn);
```

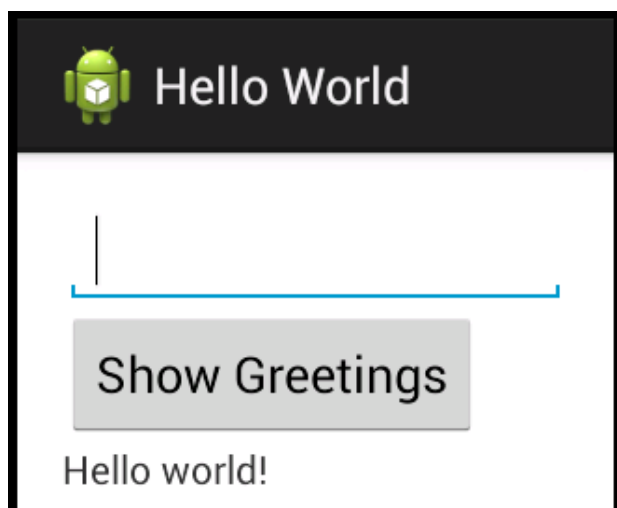
- Thêm xử lý sự kiện **click** cho button

```
btn.setOnClickListener(new OnClickListener() {  
    @Override  
    public void onClick(View arg0) {  
        view.setText(edit.getText()); // lấy thông tin nhập  
        trong khung gán cho TextView để hiển thị  
    }  
});
```

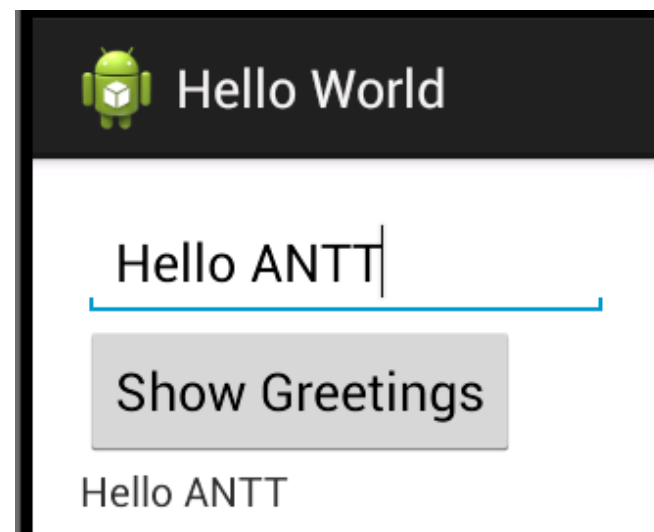
Mở rộng Hello World (3)



- Hoàn thành!
- Chạy ứng dụng: **Run → Run as → Android Application**



- Quite easy, isn't it?





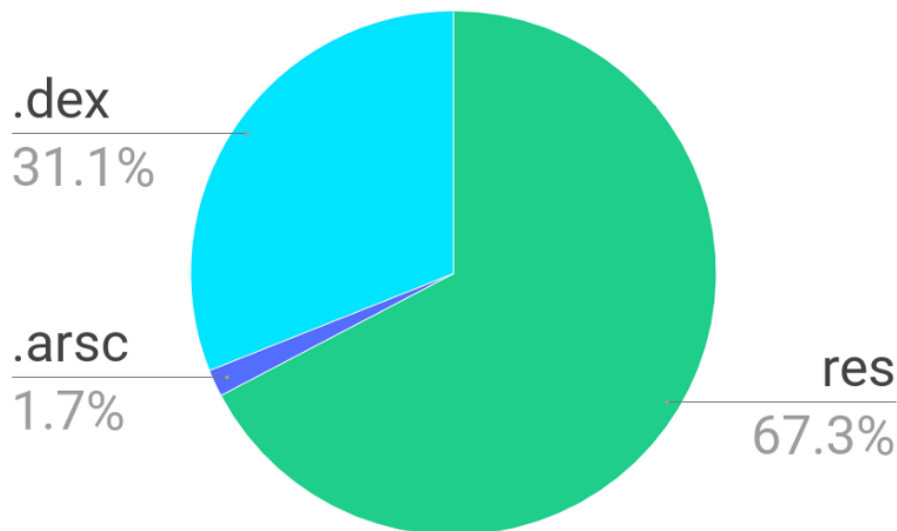
To Be Continued

Kho ứng dụng Android



Sử dụng ProGuard trước khi xuất bản ứng dụng trên App Store

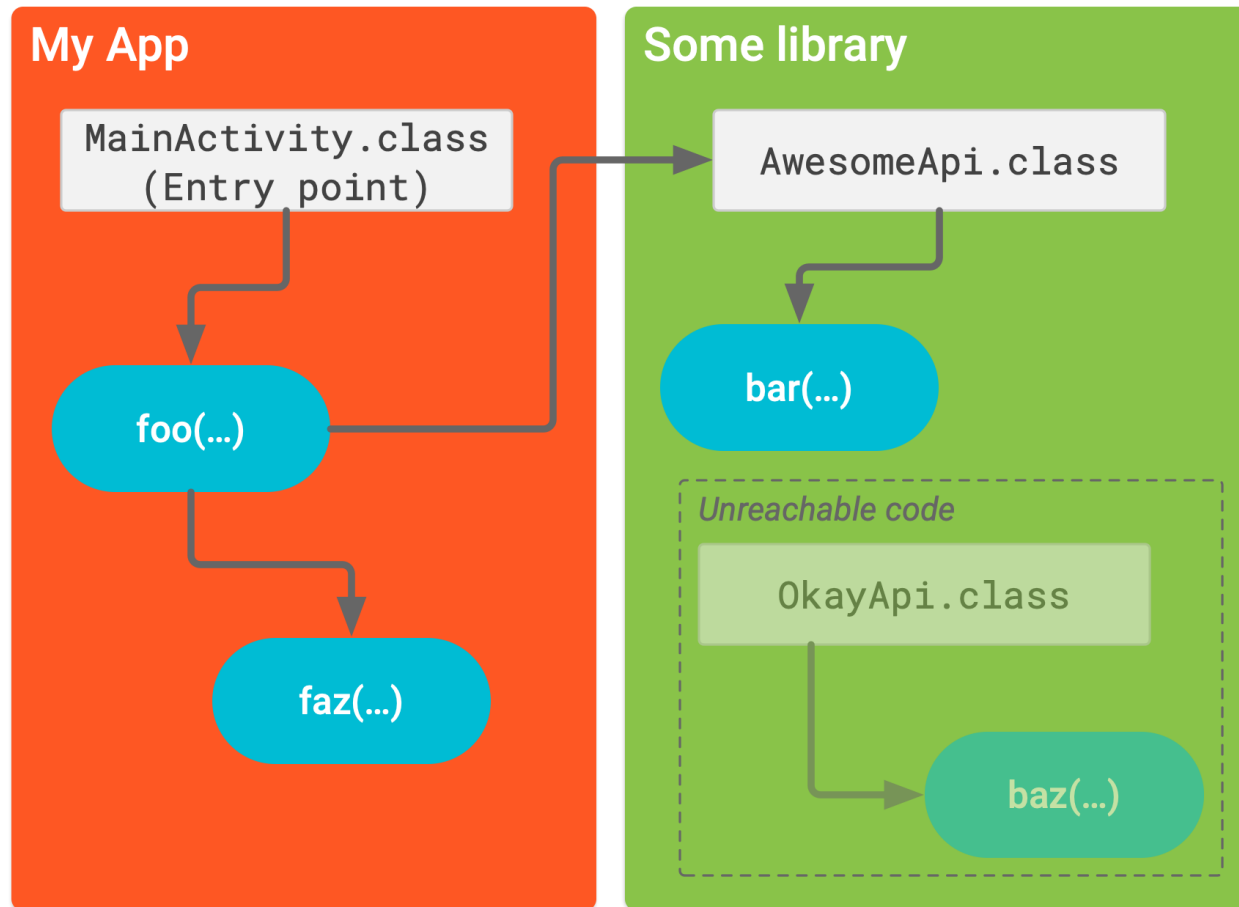
Kích thước ứng dụng Android



Kích thước ứng dụng ảnh hưởng tới:

- Sự duy trì và hài lòng của người dùng,
- Thời gian tải xuống và cài đặt,
- Tiếp cận người dùng trên các thiết bị cấp thấp

Kích thước ứng dụng Android



- Code không bao giờ được gọi
- Đối thủ phân tích ứng dụng, tái sử dụng code

ProGuard

trong phát triển Ứng dụng Android



- **ProGuard**
 - Công cụ tích hợp sẵn trong Android Studio
 - Tính năng:
 - Thu gọn mã nguồn ứng → dễ dễ phân phối
 - Làm rối → để chống dịch ngược, vì tên các hàm, biến ... bị đổi tên khó đọc
 - Tối ưu để ứng dụng chạy nhanh hơn.
- **ProGuard** là một công cụ rút gọn (shrink), tối ưu hoá (optimize) và làm mờ (obfuscate) code.
- Sử dụng **ProGuard** để bảo vệ và tối ưu ứng dụng Android

ProGuard

trong phát triển Ứng dụng Android



- Cấu hình ProGuard trong ứng dụng Android với Android Studio.
- Sử dụng **build.gradle** và thêm nội dung cấu hình:

```
android {  
    //..  
    buildTypes {  
        release { //Thêm một khối debug nếu muốn  
            minifyEnabled true //Thu gọn code, false nếu không dùng  
            useProguard true  
            proguardFiles getDefaultProguardFile('proguard-android.txt'),  
                'proguard-rules.pro'  
        }  
    }  
}
```

ProGuard

trong phát triển Ứng dụng Android



- Loại bỏ các tài nguyên (Resource) không dùng tới

```
android {  
    //..  
    buildTypes {  
        release {  
            shrinkResources true  
            minifyEnabled true  
            useProguard true  
            proguardFiles getDefaultProguardFile('proguard-android.txt'),  
                'proguard-rules.pro'  
        }  
    }  
}
```

ProGuard

trong phát triển Ứng dụng Android



- Thêm ProGuard cho Debug

```
android {  
    //..  
    buildTypes {  
        release { //Thêm một khối debug nếu muốn  
            minifyEnabled true //Thu gọn code, false nếu không dùng  
            useProguard true //Làm rối code  
            proguardFiles getDefaultProguardFile('proguard-android.txt'),  
                'proguard-rules.pro'  
        }  
  
        debug {  
            minifyEnabled true  
            useProguard false //Không làm rối code  
            proguardFiles getDefaultProguardFile('proguard-android.txt'),  
                'proguard-rules.pro'  
        }  
    }  
}
```

proguard-rules.pro



- **Proguard-rules.pro** là file cấu hình thêm về cách thức hoạt động cho ProGuard.
- Ví dụ: Khi sử dụng ProGuard làm rối code:
 - Đổi tên các class, có khi đổi các class tham khảo từ bên thứ 3 → có thể dẫn tới lỗi.
 - Trong trường hợp muốn bỏ chức năng làm rối một lớp nào đó → dùng cấu hình **keep**

Trong file proguard-rules.pro:

-keep class com.facebook. { *; }**

Với mã trên, sẽ giữ lại các lớp, phương thức `com.facebook.** {*}`

proguard-rules.pro: Ví dụ



```
-printmapping mapping.txt
-verbose
-dontoptimize
-dontpreverify
-dontshrink
-dontskipnonpubliclibraryclassmembers
-dontusemixedcaseclassnames
-keepparameternames
-renamesourcefileattribute SourceFile
-keepattributes *Annotation*
-keepattributes Exceptions,InnerClasses,Signature,Deprecated,SourceFile,LineNumberTable,*Annotation*,EnclosingMethod

-keep class * extends android.app.Activity
-assumenosideeffects class android.util.Log {
    public static *** d(...);
    public static *** v(...);
}

-keep class com.facebook.** { *; }
-keep class com.androidquery.** { *; }
-keep class com.google.** { *; }
-keep class org.acra.** { *; }
-keep class org.apache.** { *; }
-keep class com.mobileapptacker.** { *; }
-keep class com.nostra13.** { *; }
-keep class net.simonvt.** { *; }
-keep class android.support.** { *; }
-keep class com.nnacres.app.model.** { *; }
-keep class com.facebook.** { *; }
-keep class com.astuetz.** { *; }
-keep class twitter4j.** { *; }
-keep class com.actionbarsherlock.** { *; }
-keep class com.dg.libs.** { *; }
-keep class android.support.v4.** { *; }
-keep class com.bluetapestudio.templateproject.** { *; }
-keep class com.yourideatoreality.model.** { *; }
-keep interface com.yourideatoreality.model.** { *; }
-keep class com.bluetapestudio.** { *; }
-keep interface com.bluetapestudio.** { *; }
# Suppress warnings if you are NOT using IAP:
-dontwarn com.nnacres.app.**
```

proguard-rules.pro: Ví dụ (tt)



```
-dontwarn org.apache.**
-dontwarn com.mobileapptracker.**
-dontwarn com.nostra13.**
-dontwarn net.simonvt.**
-dontwarn android.support.**
-dontwarn com.facebook.**
-dontwarn twitter4j.**
-dontwarn com.astuetz.**
-dontwarn com.actionbarsherlock.**
-dontwarn com.dg.libs.**
-dontwarn com.bluetapestudio.templateproject.**

-keepattributes Signature

# For using GSON @Expose annotation
-keepattributes *Annotation*

# Gson specific classes
-keep class sun.misc.Unsafe { *; }
#-keep class com.google.gson.stream.** { *; }

# The official support library.
-keep class android.support.v4.app.** { *; }
-keep interface android.support.v4.app.** { *; }

# Library JARs.
#-keep class de.greenrobot.dao.** { *; }
#-keep interface de.greenrobot.dao.** { *; }
# Library projects.
-keep class com.actionbarsherlock.** { *; }
-keep interface com.actionbarsherlock.** { *; }
#Keep native
-keepclasseswithmembernames class * {
    native <methods>;
}

-dontwarn okio.**
-dontwarn javax.annotation.Nullable
-dontwarn javax.annotation.ParametersAreNonnullByDefault
-dontwarn com.squareup.okhttp.**
```

Tài liệu tham khảo



- Stanford - CS 193A: Android App Development, Winter 2019

<https://web.stanford.edu/class/cs193a/lectures.shtml>

<https://github.com/luong-komorebi/CS193A>

- Stanford – CS 193p: iOS App Development

https://github.com/rubenbaca/cs193p_iOS11

<https://github.com/Sencudra/CS193P>

Bài tập



SV lập trình ứng dụng di động đơn giản, với tính năng:

- ❖ Đăng ký/Login/logout tài khoản dựa vào thông tin lưu trữ CSDL SQLite trên thiết bị.
- ❖ Sử dụng ProGuard cho ứng dụng Android

Bảo mật web và ứng dụng



Trường ĐH CNTT TP. HCM