**LỜI CẢM ƠN**

Lời đầu tiên tôi xin chân thành cảm ơn các thầy cô giáo Trường Học Viện Quản Lý Giáo Dục đã nhiệt tình dạy bảo và truyền đạt những kiến thức kinh nghiệm quý báu cho tôi trong suốt thời gian học tập tại trường.

Trong thời gian làm đồ án tôi xin cảm ơn Tiến sĩ Nguyễn Mạnh Hùng đã tận tình chỉ dẫn tôi. Tôi cũng xin chân thành cảm ơn tới các thầy cô trong khoa Công nghệ thông tin – đã trang bị cho tôi những kiến thức chuyên môn quý giá, giúp đỡ và tạo điều kiện để tôi có thể học tập và thực hành tại khoa.

Sau cùng tôi xin gửi lời cảm ơn đến gia đình, bạn bè – những người đã luôn ủng hộ, động viên, củng cố niềm tin và tạo điều kiện tốt nhất cho tôi trong suốt quá trình hoàn thành bài luận này.

Vì thời gian làm đồ án có hạn nên trong quá trình làm không tránh khỏi những thiếu sót, tôi rất mong sự đóng góp ý kiến từ thầy cô và các bạn.

Tôi xin chân thành cảm ơn!

Hà Nội, ngày 25 tháng 4 năm 2017

Sinh viên

Nguyễn Hữu Thọ

**CHƯƠNG 1 : GIỚI THIỆU CÔNG NGHỆ**

1. **Hệ điều hành Android**
   1. **Lịch sử**

[Android](https://vi.wiktionary.org/wiki/android) là một [hệ điều hành](https://vi.wikipedia.org/w/index.php?title=H%E1%BB%87_%C4%91i%E1%BB%81u_h%C3%A0nh_di_%C4%91%E1%BB%99ng&action=edit&redlink=1) dựa trên nền tảng [Linux](https://vi.wikipedia.org/wiki/Linux) được thiết kế dành cho các thiết bị di động có [màn hình cảm ứng](https://vi.wikipedia.org/wiki/M%C3%A0n_h%C3%ACnh_c%E1%BA%A3m_%E1%BB%A9ng) như [điện thoại thông minh](https://vi.wikipedia.org/wiki/%C4%90i%E1%BB%87n_tho%E1%BA%A1i_th%C3%B4ng_minh) và [máy tính bảng](https://vi.wikipedia.org/wiki/M%C3%A1y_t%C3%ADnh_b%E1%BA%A3ng). Ban đầu, Android được phát triển bởi Tổng công ty Android, với sự hỗ trợ tài chính từ [Google](https://vi.wikipedia.org/wiki/Google) và sau này được chính Google mua lại vào năm 2005. Android ra mắt vào năm 2007 cùng với tuyên bố thành lập [Liên minh thiết bị cầm tay mở](https://vi.wikipedia.org/wiki/Li%C3%AAn_minh_thi%E1%BA%BFt_b%E1%BB%8B_c%E1%BA%A7m_tay_m%E1%BB%9F): một hiệp hội gồm các công ty [phần cứng](https://vi.wikipedia.org/wiki/Ph%E1%BA%A7n_c%E1%BB%A9ng), [phần mềm](https://vi.wikipedia.org/wiki/Ph%E1%BA%A7n_m%E1%BB%81m), và [viễn thông](https://vi.wikipedia.org/wiki/Vi%E1%BB%85n_th%C3%B4ng) với mục tiêu đẩy mạnh các [tiêu chuẩn mở](https://vi.wikipedia.org/w/index.php?title=Ti%C3%AAu_chu%E1%BA%A9n_m%E1%BB%9F&action=edit&redlink=1) cho các thiết bị di động. Chiếc điện thoại đầu tiên chạy Android được bán vào tháng 10 năm 2008

Android có [mã nguồn mở](https://vi.wikipedia.org/wiki/Ph%E1%BA%A7n_m%E1%BB%81m_ngu%E1%BB%93n_m%E1%BB%9F) và Google phát hành mã nguồn theo [Giấy phép Apache](https://vi.wikipedia.org/wiki/Gi%E1%BA%A5y_ph%C3%A9p_Apache). Chính mã nguồn mở cùng với một giấy phép không có nhiều ràng buộc đã cho phép các nhà phát triển thiết bị, mạng di động và các lập trình viên nhiệt huyết được điều chỉnh và phân phối Android một cách tự do. Ngoài ra, Android còn có một cộng đồng lập trình viên đông đảo chuyên viết các ứng dụng để mở rộng chức năng của thiết bị, bằng một loại ngôn ngữ lập trình [Java](https://vi.wikipedia.org/wiki/Java_(ng%C3%B4n_ng%E1%BB%AF_l%E1%BA%ADp_tr%C3%ACnh)) có sửa đổi. Vào tháng 10 năm 2012, có khoảng 700.000 ứng dụng trên Android, và số lượt tải ứng dụng từ [Google Play](https://vi.wikipedia.org/wiki/Google_Play), cửa hàng ứng dụng chính của Android, ước tính khoảng 25 tỷ lượt.

* 1. **Phát triển**

Android có một [hạt nhân](https://vi.wikipedia.org/wiki/Nh%C3%A2n_h%E1%BB%87_%C4%91i%E1%BB%81u_h%C3%A0nh) dựa trên [nhân Linux](https://vi.wikipedia.org/wiki/H%E1%BA%A1t_nh%C3%A2n_Linux) phiên bản 2.6, kể từ Android 4.0 Ice Cream Sandwich (bánh ngọt kẹp kem) trở về sau, là phiên bản 3.x, với [middleware](https://vi.wikipedia.org/wiki/Middleware), [thư viện](https://vi.wikipedia.org/w/index.php?title=Th%C6%B0_vi%E1%BB%87n_ph%E1%BA%A7n_m%E1%BB%81m&action=edit&redlink=1) và [API](https://vi.wikipedia.org/wiki/Giao_di%E1%BB%87n_l%E1%BA%ADp_tr%C3%ACnh_%E1%BB%A9ng_d%E1%BB%A5ng) viết bằng [C](https://vi.wikipedia.org/wiki/C_(ng%C3%B4n_ng%E1%BB%AF_l%E1%BA%ADp_tr%C3%ACnh)), còn [phần mềm ứng dụng](https://vi.wikipedia.org/wiki/Ph%E1%BA%A7n_m%E1%BB%81m_%E1%BB%A9ng_d%E1%BB%A5ng) chạy trên một [nền tảng ứng dụng](https://vi.wikipedia.org/w/index.php?title=N%E1%BB%81n_t%E1%BA%A3ng_%E1%BB%A9ng_d%E1%BB%A5ng&action=edit&redlink=1) gồm các thư viện tương thích với Java dựa trên [Apache Harmony](https://vi.wikipedia.org/w/index.php?title=Apache_Harmony&action=edit&redlink=1). Android sử dụng [máy ảo Dalvik](https://vi.wikipedia.org/w/index.php?title=Dalvik_(ph%E1%BA%A7n_m%E1%BB%81m)&action=edit&redlink=1) với một [trình biên dịch động](https://vi.wikipedia.org/w/index.php?title=Tr%C3%ACnh_bi%C3%AAn_d%E1%BB%8Bch_JIT&action=edit&redlink=1) để chạy 'mã dex' (Dalvik Executable) của Dalvik, thường được biên dịch sang [Java bytecode](https://vi.wikipedia.org/w/index.php?title=Java_bytecode&action=edit&redlink=1). Nền tảng phần cứng chính của Android là [kiến trúc ARM](https://vi.wikipedia.org/wiki/C%E1%BA%A5u_tr%C3%BAc_ARM). Người ta cũng hỗ trợ [x86](https://vi.wikipedia.org/wiki/X86) thông qua dự án [Android x86](https://vi.wikipedia.org/w/index.php?title=Android_x86&action=edit&redlink=1), và [Google TV](https://vi.wikipedia.org/wiki/Google_TV) cũng sử dụng một phiên bản x86 đặc biệt của Android.

Nhân Linux dùng cho Android đã được Google thực hiện nhiều thay đổi về kiến trúc so với nhân Linux gốc. Android không có sẵn [X Window System](https://vi.wikipedia.org/w/index.php?title=X_Window_System&action=edit&redlink=1) cũng không hỗ trợ các thư viện [GNU](https://vi.wikipedia.org/wiki/D%E1%BB%B1_%C3%A1n_GNU) chuẩn, nên việc chuyển các ứng dụng hoặc thư viện Linux có sẵn sang Android rất khó khăn. Các ứng dụng C đơn giản và [SDL](https://vi.wikipedia.org/wiki/Simple_DirectMedia_Layer) cũng được hỗ trợ bằng cách chèn những đoạn [shim](https://vi.wikipedia.org/w/index.php?title=Shim&action=edit&redlink=1) Java và sử dụng tương tự [JNI](https://vi.wikipedia.org/w/index.php?title=JNI&action=edit&redlink=1), như khi người ta chuyển [Jagged Alliance 2](https://vi.wikipedia.org/wiki/Jagged_Alliance_2) sang Android.

[Bộ lưu trữ flash](https://vi.wikipedia.org/wiki/B%E1%BB%99_nh%E1%BB%9B_flash) trên các thiết bị Android được chia thành nhiều phân vùng, như "/system" dành cho hệ điều hành và "/data" dành cho dữ liệu người dùng và cài đặt ứng dụng. Khác với các bản phân phối Linux cho máy tính để bàn, người sở hữu thiết bị Android không được trao quyền truy cập [root](https://vi.wikipedia.org/w/index.php?title=Superuser&action=edit&redlink=1) vào hệ điều hành và các phân vùng nhạy cảm như /system được thiết lập [chỉ đọc](https://vi.wikipedia.org/wiki/B%E1%BB%99_nh%E1%BB%9B_ch%E1%BB%89_%C4%91%E1%BB%8Dc). Tuy nhiên, quyền truy cập root có thể chiếm được bằng cách tận dụng những [lỗ hổng bảo mật](https://vi.wikipedia.org/w/index.php?title=L%E1%BB%97_h%E1%BB%95ng_(m%C3%A1y_t%C3%ADnh)&action=edit&redlink=1) trong Android, điều mà [cộng đồng mã nguồn mở](https://vi.wikipedia.org/wiki/Android_(h%E1%BB%87_%C4%91i%E1%BB%81u_h%C3%A0nh)#C.E1.BB.99ng_.C4.91.E1.BB.93ng_m.C3.A3_ngu.E1.BB.93n_m.E1.BB.9F) thường xuyên sử dụng để nâng cao tính năng thiết bị của họ, kể cả bị những người ác ý sử dụng để cài [virus](https://vi.wikipedia.org/wiki/Virus_(m%C3%A1y_t%C3%ADnh)) và [phần mềm ác ý](https://vi.wikipedia.org/wiki/Ph%E1%BA%A7n_m%E1%BB%81m_%C3%A1c_%C3%BD).

Việc Android có được xem là một bản phân phối Linux hay không vẫn còn là vấn đề gây tranh cãi, tuy được [Linux Foundation](https://vi.wikipedia.org/w/index.php?title=Linux_Foundation&action=edit&redlink=1) và [Chris DiBona](https://vi.wikipedia.org/w/index.php?title=Chris_DiBona&action=edit&redlink=1), trưởng nhóm mã nguồn mở Google, ủng hộ. Một số khác, như *linux-magazine.com* thì không đồng ý, do Android không không hỗ trợ nhiều công cụ GNU, trong đó có [glibc](https://vi.wikipedia.org/w/index.php?title=Glibc&action=edit&redlink=1).

* 1. **Cộng đồng mã nguồn mở**

Android có một cộng đồng các lập trình viên và những người đam mê rất năng động. Họ sử dụng mã nguồn Android để phát triển và phân phối những phiên bản chỉnh sửa của hệ điều hành. Các bản Android do cộng đồng phát triển thường đem những tính năng và cập nhật mới vào nhanh hơn các kênh chính thức của nhà sản xuất/nhà mạng, tuy không được kiểm thử kỹ lưỡng cũng như không có đảm bảo chất lượng; cung cấp sự hỗ trợ liên tục cho các thiết bị cũ không còn nhận được bản cập nhật chính thức; hoặc mang Android vào những thiết bị ban đầu chạy một hệ điều hành khác, như [HP Touchpad](https://vi.wikipedia.org/w/index.php?title=HP_Touchpad&action=edit&redlink=1). Các bản Android của cộng đồng thường được [root](https://vi.wikipedia.org/wiki/Root_(Android)) sẵn và có những điều chỉnh không phù hợp với những người dùng không rành rẽ, như khả năng [ép xung](https://vi.wikipedia.org/wiki/T%C4%83ng_t%E1%BB%91c_ph%E1%BA%A7n_c%E1%BB%A9ng_m%C3%A1y_t%C3%ADnh) hoặc [tăng/giảm áp](https://vi.wikipedia.org/w/index.php?title=Thay_%C4%91%E1%BB%95i_%C4%91i%E1%BB%87n_%C3%A1p_%C4%91%E1%BB%99ng&action=edit&redlink=1) bộ xử lý của thiết bị. [CyanogenMod](https://vi.wikipedia.org/wiki/CyanogenMod) là firmware của cộng đồng được sử dụng phổ biến nhất, và hoạt động như một tổ chức của số đông khác.

Trước đây, nhà sản xuất thiết bị và nhà mạng tỏ ra thiếu thiện chí với việc phát triển [firmware](https://vi.wikipedia.org/wiki/Firmware) của bên thứ ba. Những nhà sản xuất còn thể hiện lo ngại rằng các thiết bị chạy phần mềm không chính thức sẽ hoạt động không tốt và dẫn đến tốn tiền hỗ trợ. Hơn nữa, các firmware đã thay đổi như CyanogenMod đôi khi còn cung cấp những tính năng, như [truyền tải mạng](https://vi.wikipedia.org/w/index.php?title=Truy%E1%BB%81n_t%E1%BA%A3i_m%E1%BA%A1ng&action=edit&redlink=1) (tethering), mà người dùng bình thường phải trả tiền nhà mạng mới được sử dụng. Kết quả là nhiều thiết bị bắt đầu đặt ra hàng rào kỹ thuật như khóa [bootloader](https://vi.wikipedia.org/w/index.php?title=Bootloader&action=edit&redlink=1) hay hạn chế quyền truy cập root. Tuy nhiên, khi phần mềm do cộng đồng phát triển ngày càng trở nên phổ biến, và sau một thông cáo của Thư viện Quốc hội Hoa Kỳ cho phép "[jailbreak](https://vi.wikipedia.org/w/index.php?title=IOS_jailbreaking&action=edit&redlink=1)" (vượt ngục) thiết bị di động, các nhà sản xuất và nhà mạng đã tỏ ra mềm mỏng hơn với các nhà phát triển thứ ba, thậm chí một số hãng như [HTC](https://vi.wikipedia.org/wiki/HTC), [Motorola](https://vi.wikipedia.org/wiki/Motorola), [Samsung](https://vi.wikipedia.org/wiki/T%E1%BA%ADp_%C4%91o%C3%A0n_Samsung) và [Sony](https://vi.wikipedia.org/wiki/Sony_Mobile_Communications), còn hỗ trợ và khuyến khích phát triển. Kết quả của việc này là dần dần nhu cầu tìm ra các hạn chế phần cứng để cài đặt được firmware không chính thức đã bớt đi do ngày càng nhiều thiết bị được phát hành với [bootloader](https://vi.wikipedia.org/w/index.php?title=Bootloader&action=edit&redlink=1) đã mở khóa sẵn hoặc có thể mở khóa, tương tự như điện thoại dòng [Nexus](https://vi.wikipedia.org/wiki/Google_Nexus), tuy rằng thông thường họ sẽ yêu cầu người dùng từ bỏ chế độ bảo hành nếu họ làm như vậy. Tuy nhiên, tuy được sự chấp thuận của nhà sản xuất, một số nhà mạng tại Mỹ vẫn bắt buộc điện thoại phải bị khóa.

Việc mở khóa và "hack" điện thoại thông minh và máy tính bảng vẫn còn là tác nhân gây căng thẳng giữa cộng đồng và công nghiệp. Cộng đồng luôn biện hộ rằng sự hỗ trợ không chính thức ngày càng trở nên quan trọng trước việc nền công nghiệp không cung cấp các bản cập nhật thường xuyên và/hoặc ngưng hỗ trợ cho chính các thiết bị của họ.

* 1. **Tỷ lệ sử dụng các phiên bản Android**

Tỷ lệ sử dụng các phiên bản khác nhau tính đến tháng 4 năm 2014. Phần lớn các thiết bị Android cho tới nay vẫn chạy hệ điều hành phiên bản 4.1.x Jelly Bean được phát hành ngày 9 tháng 7 năm 2012 nhờ tính ổn định và hỗ trợ tốt các máy có cấu hình thấp.

****

****

1. **Công cụ phát triển**
   1. **IDE Android Studio**

Trước năm 2015, để lập trình ứng dụng Android bạn cần cài đặt Eclipse (Vốn là một công cụ lập trình ứng dụng Java), sau đó bạn cần phải cài đặt thêm plugin vào Eclipse, nó sẽ cho phép bạn lập trình các ứng dụng Android trên Eclipse.

Năm 2015 Google cho ra mắt **Android Studio**, một công cụ lập trình dành riêng cho ứng dụng **Android**, và chính thức không hỗ trợ Plugin cho Eclipse nữa.

Về mặt logic những công cụ gì được tạo ra dành cho một nhiệm vụ chuyên biệt bao giờ cũng tốt hơn.  Điều đó có nghĩa là để lập trình **Android** bạn  nên chọn **Android Studio**.

* 1. **Genymotion**

 Genymotion là một máy ảo chạy nhanh rất nhiều so với Android Emulator và giúp tiết kiệm thời gian test khi chạy ứng dụng khi bạn [lập trình android](https://hocthietkeweb.net.vn/khoa-hoc-lap-trinh-game-android-co-ban.html).

**Tính năng đặc trưng của Genymotion :**

* Genymotion shell cho phép tương tác với máy ảo bằng cách sử dụng dòng lệnh.
* Hỗ trợ ADB.
* Eclipse và Android Studio plugin.
* Hỗ trợ Linux, Windows và Mac.
  1. **Sqlite manager**

Bất kì một ứng dụng nào dù lớn hay nhỏ chúng ta hầu như đều phải sử dụng cơ sở dữ liệu.

 Khi sử dụng SQLite các bạn có thể tạo cơ sở dữ liệu trực tiếp trong code nhưng việc làm này khá tốn thời gian, và để giải quyết điều này chúng ta có thể sử dụng Sqlite Manager được tích hợp như là plug-in của Firefox

Sqlite Manager cho phép chúng ta Thêm, Sửa, Xóa các trường, các hàng, các cột thậm chí là các bảng một cách trực quan trên giao diện, điều này giúp lập trình viên dễ mường tượng và thao tác dễ dàng hơn với cơ sở dữ liệu

1. **Các thư viện, công nghệ sử dụng**
   1. **ButterKnife**

ButterKnife là một sản phẩm của Jake Wharton, một người sau khi đã phát chán với việc phải tự tay viết các câu findViewById khi lập trình Android và anh đã nảy ra ý tưởng về một API giúp giảm bớt thời gian và sự ức chế đó mỗi khi lập trình giao diện trên Android. Nói cơ bản thì khi sử dụng API này, các lập trình viên Android có thể quên luôn cái method findViewById đi, không cần sử dụng đến nó nữa

Việc sử dụng Butter Knife cực kì đơn giản, gồm 3 bước :

**Bước 1: Thêm thư viện vào file build.gradle**

Rất đơn giản, chỉ cần thêm 2 dòng sau vào phần dependencies trong file build.gradle của module

****

### Bước 2: Bind target

### Việc bind target là để butter knife được xem bạn định sử dụng layout nào và trong layout đó có các thành phần nào để từ đó tự động gán các field mà bạn khai báo ở bước 3 với các thành phần tương ứng trong layout. Lưu ý là bước này các bạn chỉ cần thực hiện 1 lần đối với mỗi 1 activity, fragment hay ViewHolder thôi

### Bước 3: Gán view hoặc resouce

### *Đối với view :*

### Butter Knife hỗ trợ bạn 1 cách rất đơn giản để có thể gán thuộc tính (field) tương ứng với view trên layout, điều này sẽ giúp bạn giảm đi được n lần lặp đi lặp lại để viết findViewById

### Để làm điều này, đơn giản chỉ thêm tiền tố(annotation) [@BindView](https://kipalog.com/users/BindView/mypage)(id\_view) vào trước khai báo của thuộc tính tương ứng với view đó.

### C:\Users\hnc\Desktop\2017-04-26_085028.jpg

### *Đối với resource khác như string, drawable, color….*

### Rất đơn giản, tương tự như với view, các bạn cũng có thể sử dụng các tiền tố (annotation) [@BindBool](https://kipalog.com/users/BindBool/mypage), [@BindColor](https://kipalog.com/users/BindColor/mypage), [@BindDimen](https://kipalog.com/users/BindDimen/mypage), [@BindDrawable](https://kipalog.com/users/BindDrawable/mypage), [@BindInt](https://kipalog.com/users/BindInt/mypage), [@BindString](https://kipalog.com/users/BindString/mypage), … để gán resource với field tương ứng

### 

### Bước 4: Gán listener cho các view

### Không chỉ hỗ trợ các bạn trong việc gán (bind) view hay resource mà Butter Knife còn hỗ trợ các bạn trong việc gán (bind) các listener cho các view tương ứng

### 

### Các bạn có thể để ý thấy là ta chỉ việc thêm [@OnClick](https://kipalog.com/users/OnClick/mypage)(view\_id) sau đó viết hàm handler sự kiện onClick là xong chứ không cần thiết phải qua từng bước getViewById, implement OnClickListener rồi setOnClickListener như trước nữa, rất tiện

* 1. **Jsoup**

Như đã nói **Jsoup** là một thư viện của **Java** để cung câp một bộ các API cho việc trích xuất và thao tác các file **HTML** . Có một vài phương thức cho việc đọc và phân tích một trang **HTML** , trong trường hợp của chúng ta, chúng ta muốn lấy một vài thông tin từ một trang HTML trên server ở xa và chúng ta phải cung cấp một **URL**. Nếu chúng ta muốn phân tích trang như là DOM, chúng ta có:

Document doc = Jsoup.connect(URL).get();

**doc** là một biến thuộc class **Document** để lưu trữ những tài liệu đã được lấy về. Bây giờ chúng ta đã có **Document** và chúng ta có thể tự do trích xuất thông tin. Chúng ta có thể lấy title và những thông tin khác đang được sử dụng trong các\*\* Tag HTML\*\*  
Cho ví dụ, nếu chúng ta muốn lấy tất cả các tag có tên là\*\* meta \*\*:

Elements metaElems = doc.select("meta");

Phương thức **select** được sử dụng khi chúng ta muốn lấy về những tag đang sử dụng truy vấn **CSS** . Cho ví dụ, nếu chúng ta muốn lấy giá trị thuộc tính từ một tag :

String name = metaElem.attr("name");

**name** ở đây là tên thuộc tính. Hơn nữa, Chúng ta có thể lựa chọn tất cả các thành phần trong một trang HTML có một giá trị CSS xác định. Cho ví dụ, trong website có một vài thành phần có một class CSS tên là \*\*“topic" \*\*, vậy chúng tao có.

Elements topicList = doc.select("h2.topic");

* 1. **Retrofit**

[Retrofit](https://square.github.io/retrofit/) là một HTTP client type-safe cho Android và Java. Retrofit giúp dễ dàng kết nối đến một dịch vụ REST trên web bằng cách chyển đổi API thành Java Interface. Retrofit là một trong những thư viện HTTP phổ biến nhất và được khuyến dùng nhất cho Android.

Thư viện mạnh mẽ này giúp bạn dễ dàng xử lý dữ liệu JSON hoặc XML sau đó phân tích cú pháp thành Plain Old Java Objects (POJOs). Tất cả các yêu cầu GET, POST, PUT, PATCH, và DELETE đều có thể được thực thi.

Giống như hầu hết các phần mềm mã nguồn mở khác, Retrofit được xây dựng dựa trên một số thư viện mạnh mẽ và công cụ khác. Đằng sau nó, Retrofit làm cho việc sử dụng [OkHttp](http://square.github.io/okhttp/) (từ cùng một nhà phát triển) để xử lý các yêu cầu trên mạng. Ngoài ra, Retrofit không tích hợp bất kỳ một bộ chuyển đổi JSON nào để phân tích từ JSON thành các đối tượng Java. Thay vào đó nó đi kèm với các thư viện chuyển đổi JSON sau đây để xử lý điều đó:

* Gson: com.squareup.retrofit:converter-gson
* Jackson: com.squareup.retrofit:converter-jackson
* Moshi: com.squareup.retrofit:converter-moshi
* Protobuf: com.squareup.retrofit2:converter-protobuf
* Wire: com.squareup.retrofit2:converter-wire

Và đối với XML, Retrofit hỗ trợ:

* SimpleFramework: com.squareup.retrofit2:converter-simpleframework

Việc phát triển thư viện HTTP type-safe của chính bạn để giao tiếp với REST API có thể thật sự là một điều khó khăn: bạn phải xử lý nhiều chức năng chẳng hạn như tạo các kết nối, lưu trữ cach, thử lại các yêu cầu thất bại, phân tích luồng phản hồi, xử lý lỗi, và nhiều hơn nữa. Retrofit, mặt khác, được hoạch định rất tốt, có tài liệu hướng dẫn và kiểm đã được kiểm thử—một thư viện đã được thử nghiệm rất nhiều trong thực tế sẽ giúp bạn tiết kiệm rất nhiều thời gian quý báu và giảm bớt căng thẳng.

* 1. **Gson**

Gson là một thư viện java cho phép người sử dụng có thể chuyển đổi từ một đối tượng Java sang JSON và cũng có thể chuyển đổi từ một đối tượng JSON sang java.Gson có thể làm việc với đối tượng java tùy ý bao gồm các đối tượng tồn tại sẵn mà bạn không có source-code của chúng.

* 1. **Picasso**

Khi phát triển các ứng dụng có nhiều ảnh hoặc yêu cầu phải load và hiển thị ảnh từ internet thì rất nhiều bạn sẽ băn khoăn về cách làm sao cho vừa không bị “đơ” UI (không gây khó chịu cho người dùng) lại vừa dễ dàng trong việc viết code và quản lý code. Trước đây mình cũng đã từng rất băn khoăn về vấn đề này bởi nếu dùng Loader hay AsyncTask thì code sẽ khá phức tạp, còn nếu cứ load thẳng ảnh trong Main Thread thì người dùng sẽ khó chịu vì không thể tương tác với UI, nhưng từ khi chuyển sang sử dụng Picasso thì các vấn đề này đã không còn nữa bởi vì chính Picasso đã lo hết những phần này rồi. Chính vì vậy, trong bài viết này mình muốn giới thiệu tới các bạn về Picasso, một thư viện chuyên về download và hiển thị ảnh từ internet rất tuyệt vời cho Android.

Picasso là một thư viện chuyên về load ảnh với các ưu điểm sau:

* **Đơn giản hóa code** chính là ưu điểm hàng đầu của Picasso. Thay vì phải viết hàng chục thậm chí là cả trăm dòng code thì với thư viện này bạn sẽ chỉ mất vài dòng thậm chí là chỉ 1 dòng code là bạn đã hoàn thành việc load ảnh tử internet vào ImageView của bạn. Đây chính là điểm tuyệt vời nhất khi bạn sử dụng thư viện load ảnh này.
* **Tối ưu hóa bộ nhớ** sử dụng của ứng dụng
* **Tự động sử dụng RAM cache và Disk cache** để tối ưu hóa lưu lượng mạng và tăng tốc độ load ảnh
* **Tự động recycle bitmap và cancel download theo ngữ cảnh**. Nhờ ưu điểm này mà bạn sẽ không còn phải để ý xem mình đã recycler bitmap hay chưa, hay có cần bắt exception khi quá trình download bị cancel hay không… Tất tần các công việc này hãy để thư viện lo và bạn chỉ cần viết code để xử lý cho luồng chính mà thôi.

Không chỉ dừng lại ở việc load ảnh, Picasso còn cung cấp cho chúng ta 1 số tùy biến khác để việc load ảnh và handle các tình huống được đơn giản và dễ dàng hơn. Các tùy biến đó bao gồm:

* Placeholder() : Tùy chọn placeholder() cho phép chúng ta hiển thị 1 ảnh nào đó trên imageView trong quá trình mà Picasso đang thực hiện load bức ảnh mà ta cần load vào ImageView, ví dụ như bạn có thể hiển thị hình ảnh loading… hoặc biểu tượng bất kì nào đó để thể hiện cho người dùng biết là ảnh đang trong quá trình loading.
* Error() : Tùy chọn error() cho phép bạn hiển thị một bức ảnh nào đó nếu như việc load bức ảnh vào imageView không thể hoàn thành được. Với tùy chọn này, các bạn có thể dễ dàng thể hiện cho người dùng việc load ảnh hiện đang gặp vấn đề và cần phải tải lại.
* Load ảnh từ Resource, Assets, file … : Không chỉ cho phép load ảnh từ internet, Picasso còn cho phép chúng ta load ảnh từ Resources, Assets của application hoặc từ files nằm trên bộ nhớ trong hoặc thẻ nhớ của máy.
  1. **Material Design**

Đội ngũ thiết kế của Google đã tạo nên một loạt nguyên tắc về diện mạo của các ứng dụng. Nó bao gồm việc đơn giản hóa và tiêu chuẩn hóa các thành phần đồ họa của Android, từ việc tạo các công cụ chọn bảng màu cho tới nâng cấp font chữ Roboto mà Google đã áp dụng trên Android trong vòng 1 năm qua. Tất cả đã rõ ràng hơn, xuyên suốt trên nhiều thiết bị khác nhau.

Không chỉ về giao diện ứng dụng mà còn là tạo nên một tiêu chuẩn chung cho tất cả, điều này rất quan trọng cho Google trong tương lai. Bởi vì Material Design không chỉ áp dụng cho không gian chữ nhật trên điện thoại hay tablet, nó sẽ được vận dụng cho màn hình dạng tròn trên đồng hồ đeo tay thông minh. Hay thậm chí là trên kính mắt (Google Glass). Nó đơn giản hơn, sáng hơn và rõ ràng hơn. Nó rất quan trọng bởi Material Design sẽ xuất hiện trên nhiều thiết bị hơn bất kỳ giao diện người dùng (UI) nào khác trong lịch sử.

Như Mattias Duarte trưởng bộ phận thiết kế của Google đã giải thích, Material Design là tính ba chiều của mọi sự vật: hãy nghĩ tới các điểm ảnh (pixel) không chỉ là điểm 2 chiều của màu sắc, nó là các khối hình 3 chiều với chiều sâu và chiều cao. Lấy ví dụ với một thẻ Google Now. Khi nó xuất hiện trên màn hình điện thoại, cửa sổ đó không chỉ là tập hợp của những khung hình vuông màu trắng mà nó có những đặc tính giống như một thẻ thật ngoài đời khi di chuyển trên màn hình của bạn. Điều tương tự với các hiệu ứng như đổ bóng hay phối cảnh. Material Design sẽ biến thẻ Google Now như thể nó đang nổi trên màn hình đi kèm với hiệu ứng đổ bóng vật lý và độ sâu.

Như vậy có thể thấy Material Design rất khác so với Flat Design mà chúng ta đã nghe nhiều về nó. Thiết kế phẳng đề cao sự đơn giản với ít tính ba chiều nhưng với Material Design thì Google đã mang rất nhiều tính ba chiều vào Android. Và nó cũng có nguyên nhân cả.

#### Một số đặc điểm có thể dùng để nhận biết những ứng dụng Material Design như sau:

* Sử dụng các màu nổi bật, thường có một mảng màu chủ đạo nằm ở cạnh trên ứng dụng
* Các biểu tượng phẳng, đơn giản nhưng dễ hiểu
* Một số ứng dụng sẽ có một nút tròn to nằm ở góc dưới bên phải, thường có chức năng tạo mới
* Giao diện phẳng, ít hoặc không có hiệu ứng chuyển màu, có hoặc không có hiệu ứng đổ bóng đen
* Menu, nút nhấn, chữ viết… có nhiều khoảng cách trắng nên trông thoáng đãng
* Có các hiệu ứng chuyển động tự nhiên, dễ hiểu, có thể gợi ý cho một tính năng nào đó

#### ****Lợi ích của Material Design là gì?****

Như đã nói ở trên, Material Design là một ngôn ngữ được Google tạo ra nhằm “phát triển một hệ thống nền duy nhất cho phép xây dựng các trải nghiệm đồng bộ giữa nhiều loại và kích thước thiết bị khác nhau. Việc thiết kế cho thiết bị di động được đặt lên hàng đầu, nó còn phải chú trọng đến việc nhập liệu bằng cảm ứng, giọng nói, chuột, bàn phím”.

Thông qua lời nói trên, chúng ta có thể thấy là Google muốn mang lại trải nghiệm thống nhất cho người dùng trong hệ sinh thái của hãng. Người dùng cuối như chúng ta sẽ có cảm giác quen thuộc hơn, dành ít thời gian để học hỏi hơn khi sử dụng cùng một app nhưng trên nhiều máy khác nhau. Các lập trình viên thì có thể đảm bảo rằng cách người dùng trải nghiệm app của họ là như nhau không quan trọng thiết bị đang chạy là gì.

Theo thiết kế của Google, nội dung trên trang web và [ứng dụng android](http://devpro.edu.vn/khoa-hoc-lap-trinh-android/) sẽ đồng nhất về hình ảnh và giao diện, trong đó có tùy chỉnh về mặt hiển thị trên các thiết bị khác nhau. Material Design đã cung cấp một giải pháp dễ dàng để thực hiện điều này, thiết kế ưu việt và tối ưu hóa tốt.

**CHƯƠNG 2 : PHÂN TÍCH**

1. **Mô tả yêu cầu người dùng**

* Người dùng cần gì ở một ứng dụng Nghe nhạc
  + Giao diện đẹp mắt, thân thiện mà dễ sử dụng
  + Đa dạng các thể loại nhạc
  + Ứng dụng nhẹ mà lại nhanh và mượt
  + Nhiều hình ảnh của ca sĩ, bài hát
  + Hiển thị lời bài hát theo từng câu hát
  + Đồng bộ, lưu trữ nhạc trên cloud hoặc driver
  + Cập nhật những bài hát hit, những album mới hàng ngày
  + Equalizer : bộ chỉnh âm sắc cho âm thanh
  + Và nhiều tiện ích khác nữa

1. **Phân tích cơ sở dữ liệu**
   1. **Giải thích các thành phần**
   2. **Giải thích các mối quan hệ của các bảng trong csdl**
2. **Phân tích chức năng**
   1. **Nghe nhạc Online**
   2. **Nghe nhạc Offline**
   3. **Android Interface Define Langugae (AIDL)**

**CHƯƠNG 3 : THIẾT KẾ**

1. **Thiết kế giao diện**
2. **Thiết kế ứng dụng**
   1. **Mô hình code**
   2. **Cấu trúc code**
   3. **Permission Android 6 API 23**

**CHƯƠNG 4 : CÀI ĐẶT VÀ KIỂM THỬ ỨNG DỤNG**

1. **Cài đặt ứng dụng**
2. **Kiểm thử ứng dụng**