



TRUNG TÂM TIN HỌC - ĐẠI HỌC KHOA HỌC TỰ NHIÊN

ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH

**CHƯƠNG TRÌNH ĐÀO TẠO
LẬP TRÌNH VIÊN CHUYÊN NGHIỆP TRÊN THIẾT BỊ DI ĐỘNG**

TÀI LIỆU



**LẬP TRÌNH THIẾT BỊ DI ĐỘNG TRÊN
ANDROID**

MODULE 02:

KIẾN TRÚC VÀ XÂY DỰNG ỨNG DỤNG ANDROID

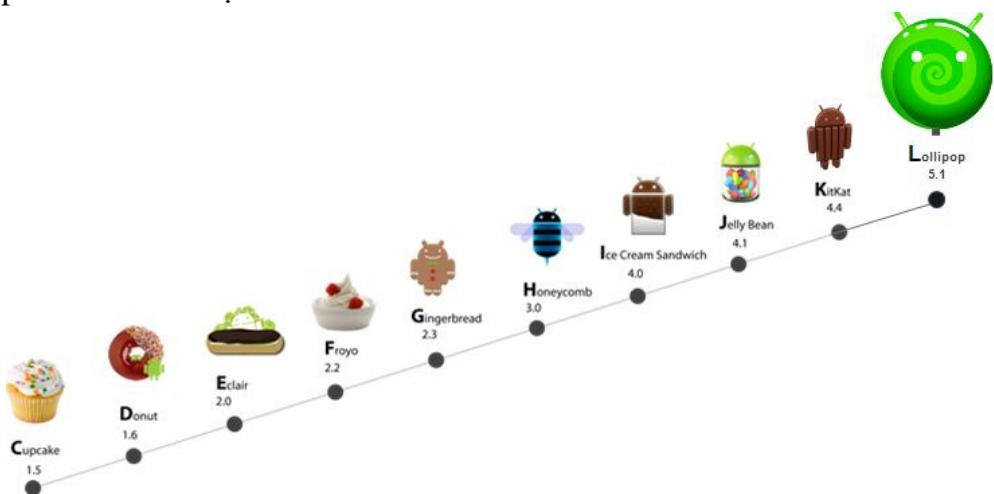
Bài 1

TỔNG QUAN VỀ LẬP TRÌNH ANDROID VÀ MÔI TRƯỜNG PHÁT TRIỂN

1. TỔNG QUAN ANDROID

1.1. Hệ điều hành Android

- Năm 2003, Android Inc. được thành lập bởi Andy Rubin, Rich Miner, Nick Sears và Chris White tại California.
- Năm 2005, Google mua lại Android Inc và bắt đầu nuôi ý tưởng tự sản xuất điện thoại di động.
- Năm 2007, tổ chức OHA (Open Handset Alliance) được thành lập với hơn 80 công ty trong lĩnh vực kỹ thuật điện tử bao gồm các công ty chuyên về phần cứng, phân phối thiết bị di động đến các công ty phần mềm, sản xuất chất bán dẫn... Có thể kể đến một số công ty nổi tiếng như Samsung, Motorola, LG, HTC, T-Mobile, Vodafone, ARM và Qualcomm...
- Năm 2008, Google ra mắt chiếc điện thoại đầu tiên đồng thời open source bản SDK (Software Development Kit) phiên bản 1.0.
- Năm 2010, Google khởi đầu dòng thiết bị Nexus với thiết bị đầu tiên của HTC là Nexus One.
- Năm 2013, ra mắt loạt thiết bị phiên bản GPE.
- Năm 2014, Google công bố Android Wear, hệ điều hành dành cho các thiết bị đeo được.
- Các phiên bản của hệ điều hành Android:



Hình 1.1. Các phiên bản hệ điều hành Android.

Phiên bản	API	Thời gian phát hành	Đặc điểm nổi bật
Android 1.0	1	10/2008	Thanh thông báo kéo từ trên xuống, màn hình chính phong phú, tích hợp chật chẽ với Gmail, giao diện đơn giản nhưng khá đẹp mắt thời bấy giờ.
Android 1.1	2	02/2009	Tải và cài đặt bản cập nhật ngay trên thiết bị, không cần kết nối với máy tính (phương thức Over The Air).
Android 1.5 Cupcake	3	05/2009	Là bản Android đầu tiên được Google gọi tên theo các món đồ ăn với chữ cái bắt đầu được xếp theo thứ tự alphabet. Với các tính năng: bàn phím ảo, mở rộng khả năng cho widget – khả năng tùy biến giao diện của Android được đẩy mạnh, cải tiến clipboard, có khả năng quay phim, cho phép tải ảnh, video lên YouTube, truy cập danh bạ trong Google Talk,...
Android 1.5 Donut	4	09/2009	Một vài điểm trong giao diện được cải thiện, hỗ trợ mạng CDMA, hỗ trợ các thành phần đồ họa độc lập với độ phân giải, tính năng Quick Search Box – tìm kiếm tất cả mọi thứ chỉ trong một hộp tìm kiếm, Android Market – hiển thị các ứng dụng miễn phí và trả phí hàng đầu. Camera được cải thiện, tích hợp trình xem ảnh tốt hơn.
Android 2.0 – Android 2.0.1 Eclair	5, 6	11/2009 - 12/2009	Hỗ trợ nhiều tài khoản người dùng, tính năng Quick Contact, cải tiến bàn phím ảo. Trình duyệt mới: hỗ trợ HTML5, phát video ở chế độ toàn màn hình, hộp địa chỉ kết hợp với thanh tìm kiếm, cho phép phóng to, thu nhỏ bằng cách chạm tay vào màn hình. Giao diện đẹp, sang trọng và gọn gàng hơn trước.
Android 2.1 Eclair	7	01/2010	Hỗ trợ Live Wallpaper, chuyển giọng nói thành văn bản và một màn hình khóa mới.
Android 2.2 Froyo	8	06/2010	Từ 3 màn hình chính tăng lên thành 5 màn hình. Dãy nút kích hoạt nhanh chế độ gọi điện, web và App Drawer xuất hiện. Khả năng hiển thị hình ảnh 3D. Tính năng trạm phát Wifi xuất hiện. Hỗ trợ duyệt web với Flash. Tính năng di chuyển một phần ứng dụng từ bộ nhớ máy sang thẻ nhớ.

Android 2.3.3 – Android 2.3.7 Gingerbread	9, 10	11/2010 – 02/2011	Có hai thanh chặn khi chọn văn bản. Bàn phím được cải thiện đẹp hơn và dễ sử dụng hơn. Xuất hiện công cụ quản lý pin và ứng dụng. Hỗ trợ máy ảnh trước. Cung cấp nhiều tính năng mới, tập trung vào việc phát triển game, đa phương tiện và phương thức truyền thông mới.
Android 3.0 – Android 3.1 – Android 3.2 Honeycomb	11, 12, 13	02/2011 – 05/2011 – 06/2011	Phiên bản này dành riêng cho máy tính bảng. Sử dụng màu đen và màu xanh dương làm tông màu chủ đạo. Homescreen và widget được thiết kế lại. Không còn nút nhấn vật lý. Cải thiện đa nhiệm – xuất hiện nút Recent Apps giúp chuyển đổi ứng dụng dễ dàng và nhanh chóng. Thanh Action Bar xuất hiện. Phiên bản Android 3.1 và 3.2 sửa lỗi và thêm vài tính năng mới như thay đổi kích thước widget ngay trên homescreen, hỗ trợ thẻ SD,...
Android 4.0 – Android 4.0.3 Ice Cream Sandwich	14, 15	10/2011 – 12/2011	Hỗ trợ bộ font Roboto. Hệ thống thông báo (Notification) được làm mới hoàn toàn, đẹp và thuận tiện hơn. Bàn phím tiếp tục được cải thiện. Hệ điều hành dành cho smartphone và cho máy tính bảng được hợp nhất làm một. Duyệt web nhanh hơn, tối ưu hóa hiệu suất hoạt động của thiết bị, kéo dài thời gian dùng pin...
Android 4.1 – Jelly Bean	16	07/2012	Màn hình Lockscreen - vòng trượt mở khóa dùng kính hoạt 3 tính năng khác nhau : mở khóa máy, sử dụng camera, kích hoạt Google Now (Android 4.0 có 2 tính năng: mở khóa máy, sử dụng camera). Khả năng tìm kiếm bằng giọng nói, kết quả trả về được thiết kế theo dạng thẻ đồ họa, thông minh hơn, trực quan hơn. Project butter giúp mang lại độ mượt mà chưa từng có cho Android.
Android 4.2 Jelly Bean	17	11/2012	Hỗ trợ Miracast - một chuẩn chia sẻ nội dung số thông qua kết nối Wi-Fi , bàn phím có thể nhập liệu bằng cách vẽ các đường nét từ ký tự này đến ký tự khác, chế độ chụp ảnh toàn cảnh Photo Sphere. Hỗ trợ nhiều tài khoản người dùng trên máy tính bảng.
Android 4.3	18	07/2013	Hỗ trợ kết nối Bluetooth Smart, bộ API OpenGL

Jelly Bean			ES 3.0, bổ sung tính năng sử dụng Wi-Fi để định vị ngay cả khi người dùng tắt kết nối này. Hỗ trợ tính năng Restricted Profile, Notification Access,...
Android 4.4 KitKat Android 4.4W KitKat Watch	19, 20	10/2013	Hiệu suất được cải tiến: RAM 512 MB cũng có thể chạy KitKat, hiệu năng cao gấp 1,6 lần phiên bản trước. Giao diện nhìn chung được làm phẳng hơn. Kết hợp dịch vụ tìm kiếm Google Search cho phép dò và tự động liên kết các danh bạ có sẵn trên internet vào số điện thoại mới. Gộp chung Text Messages và Hangouts. Bổ sung các biểu tượng Emoji vào bàn phím mặc định. Tính năng Screen Recording cho phép ghi các hoạt động diễn ra trên màn hình thành các đoạn video MP4.
Android 5.0 Lollipop Android 5.1 Lollipop	21, 22	10/2014	Hình ảnh phẳng hơn, nhiều màu sắc hơn so với trước đây. Xuất hiện chức năng “T action”: tùy vào động tác lắc máy sẽ tương ứng với chức năng cụ thể. Tính năng tự động trả lời điện thoại khi đưa máy lên tai hay tự động giảm nhạc chuông khi nhắc máy lên khỏi mặt bàn... Người dùng có thể nhận và trả lời các thông báo bao gồm email, tin nhắn, cuộc gọi nhỡ ngay ngoài màn hình khoá. Tiết kiệm pin và bảo mật hơn.
Android 6.0 Marshmallow	23	Công bố ngày 28/05/2015	Thiết kế lại quyền hạn của các ứng dụng. Trang bị cho trình duyệt Chrome tính năng mới với tên gọi Chrome Custom Tabs, cho phép các nhà phát triển có thể chèn trực tiếp nội dung trang web đầy đủ vào các ứng dụng. Cho phép mở trực tiếp nội dung bằng các ứng dụng liên quan. Trang bị hệ thống thanh toán mới trên di động, với tên gọi Android Pay. Cải thiện độ ổn định, hiệu suất cũng như khả năng tiết kiệm pin. Hỗ trợ cổng kết nối USB Type-C thế hệ mới.

Bảng 1-1. Các phiên bản hệ điều hành Android.

1.2. Tại sao lập trình trên Android

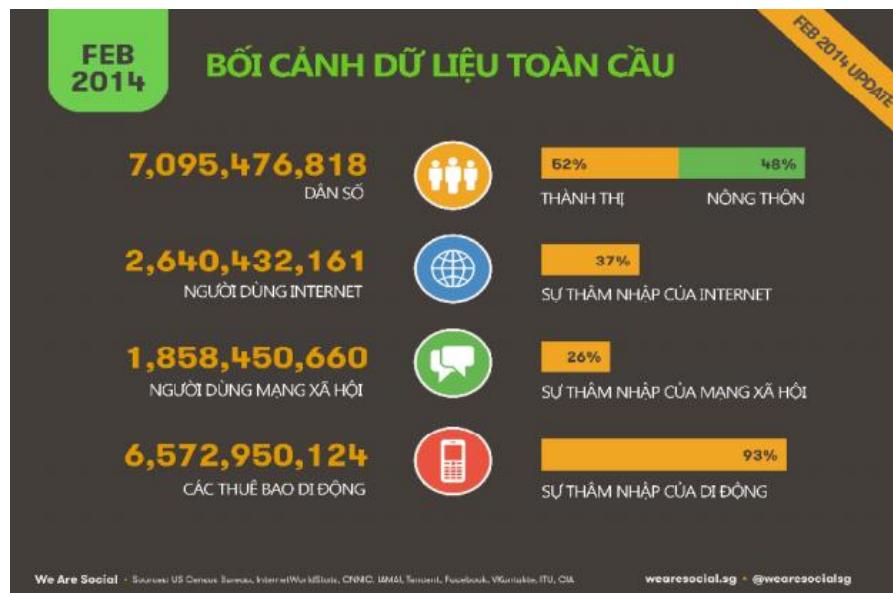
1.2.1. Xu thế phát triển công nghệ di động

- Theo nhận định của nhiều chuyên gia công nghệ từ các hãng công nghệ hàng đầu như Microsoft, Google, IBM, ... Ba xu hướng tắt trên toàn cầu hiện nay là: Social and

Security (mạng xã hội và bảo mật), Mobility (công nghệ di động), Analytics Big Data (phân tích dữ liệu lớn), Cloud (Điện toán đám mây).

- Trên thế giới:

- o Tháng 01/2014, trang WeAreSocial đã đưa ra báo cáo về “Bối cảnh dữ liệu toàn cầu” với những chỉ số phát triển rất đáng kinh ngạc của Thế Giới Số. Cụ thể số liệu thống kê của WeAreSocial cho thấy:
 - Số lượng đăng ký sử dụng di động đang hoạt động vào khoảng 93% của dân số thế giới.
 - Tỉ lệ người kết nối Internet toàn cầu đạt 35%, tương đương 2,5 tỉ người.
 - Các kênh Mạng xã hội tiếp tục phát triển mạnh mẽ trong 12 tháng qua, khi đạt tỉ lệ thâm nhập người dùng là 26%.
 - Hơn 4 tỉ người trên khắp thế giới hiện đang sở hữu ít nhất một chiếc điện thoại di động.



Hình 1.2. Bối cảnh dữ liệu toàn cầu - Nguồn: WeAreSocial

- Tại Việt Nam:

- o Trong giai đoạn 2014 - 2016, xu hướng Mobile và lượng người dùng Internet 3G sẽ tiếp tục tăng mạnh. Các dịch vụ kết nối OTT và truyền thông xã hội đóng góp hơn 80% phương thức giao tiếp online, video online và nội dung số mobile. Điều này góp phần đẩy mạnh xu hướng truyền thông số đa phương tiện, đa màn hình sẽ bùng nổ với độ phủ hơn 50% dân số Việt Nam.
- o Doanh thu điện thoại thông minh và máy tính bảng giờ đây đã vượt qua PC và laptop. Các doanh nghiệp hoạt động trong lĩnh vực sản xuất công nghệ cũng đang cố gắng hướng tới các dịch vụ như “thanh toán di động, nội dung di động, dịch vụ xác định địa điểm hay khai thác dữ liệu sử dụng của người dùng thiết bị di động”.



Hình 1.3. Chỉ số về Internet trên di động tại Việt Nam năm 2014 - Nguồn: Theo thống kê Cục Thương mại điện tử và công nghệ thông tin (VECITA), Bộ Công thương.

- Tỉ lệ truy cập Internet qua các thiết bị di động: 36% tổng số dân.
- Tỉ lệ truy cập Internet có tham gia mua sắm online 57%.
- Một người Việt Nam truy cập Internet 5,6 giờ/ngày, 6,4 ngày/tuần - Tổng số thời gian 36 giờ/tuần.



Hình 1.4. Thống kê các chỉ số người dùng smartphone tại Việt Nam năm 2014 - Nguồn: Theo thống kê Cục Thương mại điện tử và công nghệ thông tin (VECITA), Bộ Công thương.



Hình 1.5. Thống kê chỉ số điện thoại di động và các loại hình thanh toán tại Việt Nam năm 2014 - Nguồn: Theo thống kê Cục Thương mại điện tử và công nghệ thông tin (VECITA), Bộ Công thương.

1.2.2. Thị trường thiết bị Android

- Trong tất cả các hệ điều hành dành cho di động hiện nay, có thể nói: Android đã mang lại một cuộc cách mạng thật sự cho các lập trình viên. Nổi bật với tính mở, đơn giản nhưng mạnh mẽ, không tốn phí cho bất cứ bản quyền nào và đặc biệt cộng đồng lập trình viên vô cùng lớn mạnh. Android thật sự là một nền tảng mạnh mẽ cho phép các lập trình viên, những người chưa từng lập trình trên thiết bị di động có thể tạo ra các ứng dụng một cách nhanh chóng và dễ dàng. Có thể nói Android đang dần mang lại phong cách mới trong thói quen sử dụng điện thoại của người dùng.
- Kể từ khi bắt đầu được thương mại hóa, ước tính mỗi ngày có khoảng 850 ngàn thiết bị Android được kích hoạt. Nhiều cuộc nghiên cứu đã chỉ ra rằng phần lớn các thiết bị smartphone mới được xuất xưởng chạy hệ điều hành Android.
- Có tới 53 triệu thiết bị sử dụng mã nguồn mở Android (AOSP) được bán ra, chiếm tới 11% tổng số smartphone trong quý 1/2014. Điều này một lần nữa cho thấy, Android đã có được một địa vị cực kỳ vững chắc trên thị trường smartphone.
- Công ty nghiên cứu thị trường Strategy Analytics công bố hôm 31/10/2014 cho biết, số lượng smartphone chạy Android bán ra trên phạm vi toàn cầu đạt 268 triệu chiếc, tăng mạnh so với thành tích 206 triệu chiếc của cùng kỳ năm ngoái. Nếu xét về thị phần, chú robot xanh của đại gia công nghệ Google đang giữ 84% thị phần.

1.2.3. Nhu cầu tuyển dụng lập trình viên Android

- Với xu thế phát triển công nghệ di động nhanh và mạnh như hiện nay, thị trường thiết bị Android chiếm vị trí cao nhất không chỉ ở Việt Nam mà trên toàn thế giới, thì nhu cầu sử dụng các ứng dụng cho các thiết bị Android là rất lớn. Vì vậy, nhu cầu tuyển dụng lập trình viên Android cũng rất lớn và sẽ tăng nhanh.

2. KIẾN TRÚC ANDROID

- Có thể hiểu Android Software Stack bao gồm nhân Linux, tập các thư viện C/C++ được truy xuất bởi tầng ứng dụng để sử dụng các dịch vụ, các bộ quản lý thực thi và quản lý ứng dụng. Mỗi tầng đều có chức năng vai trò riêng biệt với nhau:
 - o Linux kernel – lõi chính của toàn hệ thống bao gồm các điều khiển phần cứng, bộ quản lý xử lý và bộ nhớ, bảo mật, kết nối mạng, bộ quản lý năng lượng.
 - o Libraries – thực thi trên tầng nhân Linux, bao gồm các thư viện lõi khác nhau của C/C++ như libc và SSL. Có các dạng sau:
 - Thư viện hỗ trợ phát các tập tin đa truyền thông.
 - Bộ quản lý hiển thị
 - Thư viện hỗ trợ đồ họa OpenGL 2D và 3D
 - SQLite hỗ trợ lưu trữ cơ sở dữ liệu
 - SSL và WebKit cho phép tương tác với trình duyệt và bảo mật Internet.
- Android Run Time – đây chính là điểm làm nên sự khác biệt giữa thiết bị Android và thiết bị Linux. Bên trong thành phần này bao gồm máy ảo Dalvik và thư viện lõi. Android Run Time ngoài tăng tốc độ cho ứng dụng còn làm nền cho tầng Application Framework kết nối đến.
 - o Core Libraries – mặc dù hầu hết các ứng dụng Android viết bằng ngôn ngữ Java nhưng Dalvik không phải là máy ảo Java. Các thư viện lõi Android sẽ cung cấp hầu hết các chức năng chính có thể có trong thư viện Java cũng như thư viện riêng biệt của Android.
 - o Dalvik VM – dạng máy ảo cho phép tối ưu hóa để có thể chạy được nhiều tiến trình một cách hiệu quả, dựa trên nhân Linux các máy ảo cho phép quản lý các tiểu trình và quản lý bộ nhớ ở bậc thấp.
- Application Framework – cung cấp các lớp cho việc tạo ra các ứng dụng. Bên cạnh đó nó cũng chứa các lớp trùu tượng cho phép truy nhập phần cứng, quản lý giao diện người dùng và tài nguyên của ứng dụng.
- Application Layer – gồm các ứng dụng được tích hợp sẵn và các ứng dụng của hãng thứ ba. Tầng ứng dụng trong Android Run Time sử dụng các lớp từ tầng Application Framework để thực thi ứng dụng.



Hình 1.6. Kiến trúc Android.

3. MÔI TRƯỜNG PHÁT TRIỂN ỨNG DỤNG ANDROID

3.1. Giới thiệu Java JDK, Android SDK, Android Studio

- Android SDK (Software Development Kit) và JDK (Java Development Kit) là hai công cụ cần thiết để chúng ta có thể lập trình trên các ứng dụng Android. Và tất nhiên nếu bạn không muốn lập trình trên phần mềm soạn thảo văn bản thì một công cụ lập trình IDE (Integrated development environment) sẽ rất hữu ích và tiện lợi. Eclipse được xem là một công cụ hỗ trợ rất tốt trong việc lập trình ứng dụng Android.
- Android SDK, JDK và Eclipse đều có mặt trên một số phiên bản hệ điều hành Windows, Mac OS và Linux do đó chúng ta có thể lập trình trên hệ điều hành mà chúng ta đã quen sử dụng.Thêm nữa, Android được thực thi trên máy ảo Dalvik nên việc phát triển ứng dụng là như nhau trên cả 3 môi trường.
- Android Studio được Google chính thức phát hành phiên bản đầu tiên Android Studio 0.1 vào tháng 5/ 2013 (Phiên bản hiện nay là 1.2.1 – phát hành vào tháng 5/ 2015 và phiên bản 1.3 đã được công bố tại Google I/O 2015). Là công cụ lập trình dựa trên nền IntelliJ, cung cấp các tính năng mạnh mẽ hơn ADT như:
 - o Hỗ trợ xây dựng dự án dạng Gradle.
 - o Hỗ trợ sửa lỗi nhanh và tái sử dụng cấu trúc phương thức.
 - o Cung cấp các công cụ kiểm tra tính khả dụng, khả năng hoạt động của ứng dụng, tương thích nền tảng...
 - o Hỗ trợ bảo mật mã nguồn và đóng gói ứng dụng.
 - o Trình biên tập giao diện cung cấp tổng quan giao diện ứng dụng và các thành phần, cho phép tùy chỉnh trên nhiều cấu hình khác nhau.
 - o Cho phép tương tác với nền Google Cloud.

- Với mục tiêu tạo ra môi trường phát triển tất cả trong một, trải nghiệm nhanh và mượt hơn các IDE khác, Android Studio không ngừng ra đời các phiên bản cải tiến.

3.2. Thiết lập môi trường phát triển:

- Như đã nói ở trên, ứng dụng Android được thực thi trên máy ảo Dalvik nên chúng ta có thể lập trình trên nhiều phiên bản của các hệ điều hành. Cụ thể như sau:
 - o Microsoft® Windows® 8/7/Vista/2003 (32 or 64-bit).
 - o 2 GB RAM trở lên.
 - o Dung lượng ổ đĩa ứng còn trống ít nhất 400 MB.
 - o Ít nhất 1 GB cho Android SDK, emulator system images và caches.
 - o Độ phân giải tối thiểu 1280 x 800.
 - o Java Development Kit (JDK) 7 trở lên.
 - o Tùy chọn thêm cho accelerated emulator: hỗ trợ bộ xử lý Intel® với các phiên bản: Intel® VT-x, Intel® EM64T (Intel® 64), và tính năng Execute Disable (XD) Bit.
- Để bắt đầu viết ứng dụng với Android Studio, chúng ta cần tải và cài đặt hai bộ phần mềm sau:
 - o **Java JDK:** <http://java.sun.com/javase/downloads/index.jsp> (Cài đặt trước hết và nên chọn phiên bản mới nhất).
 - o **Android Studio:** <http://developer.android.com/sdk/index.html> - tải gói Android Studio, gói này sẽ chứa các thành phần:
 - o Android Studio IDE.
 - o Android SDK tools.
 - o Android 5.0 (Lollipop) Platform.
 - o Android 5.0 emulator system image with Google APIs.



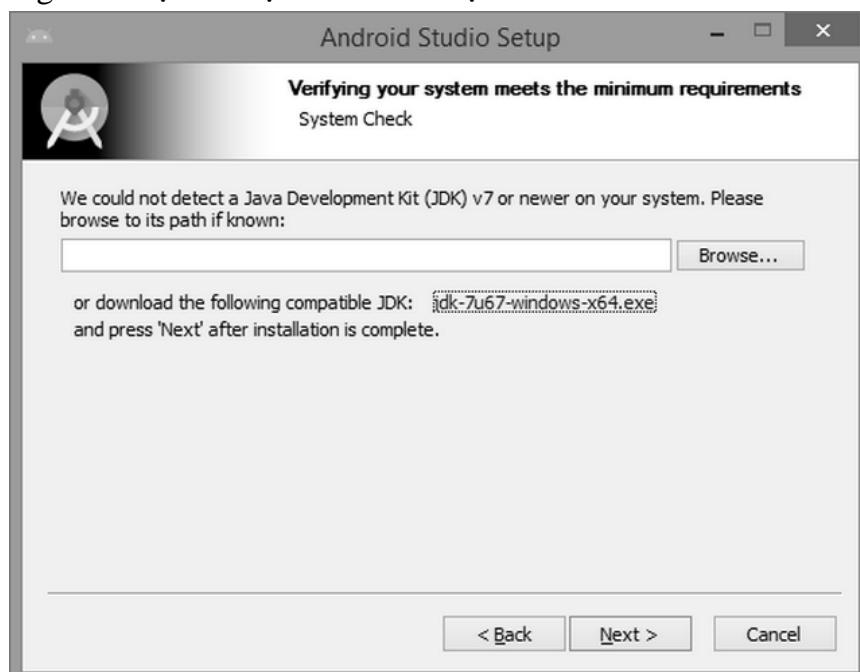
Hình 1.7

- Sau khi tải bộ cài Android Studio thành công, chúng ta chạy tập tin **android – studio – bundle** để tiến hành cài đặt Android Studio → màn hình Welcome to Android Studio Setup xuất hiện → Next:



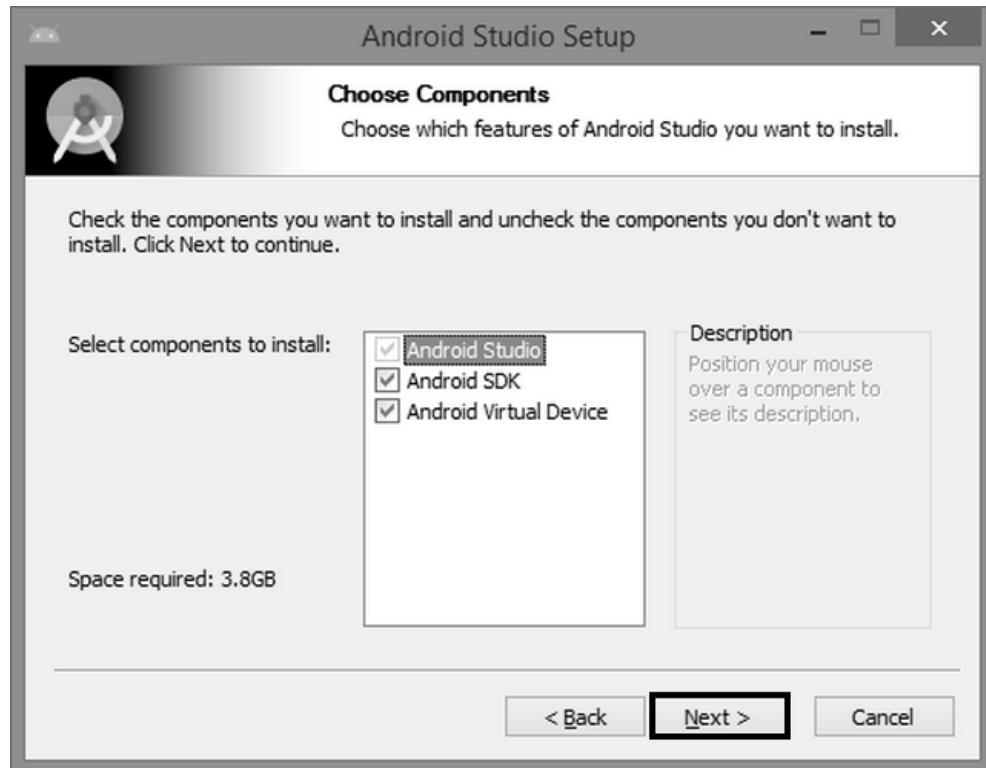
Hình 1.8.

- Nếu hệ thống không phát hiện ra JDK trong máy tính của bạn, một hộp thoại yêu cầu chỉ rõ đường dẫn hoặc cài đặt JDK xuất hiện:



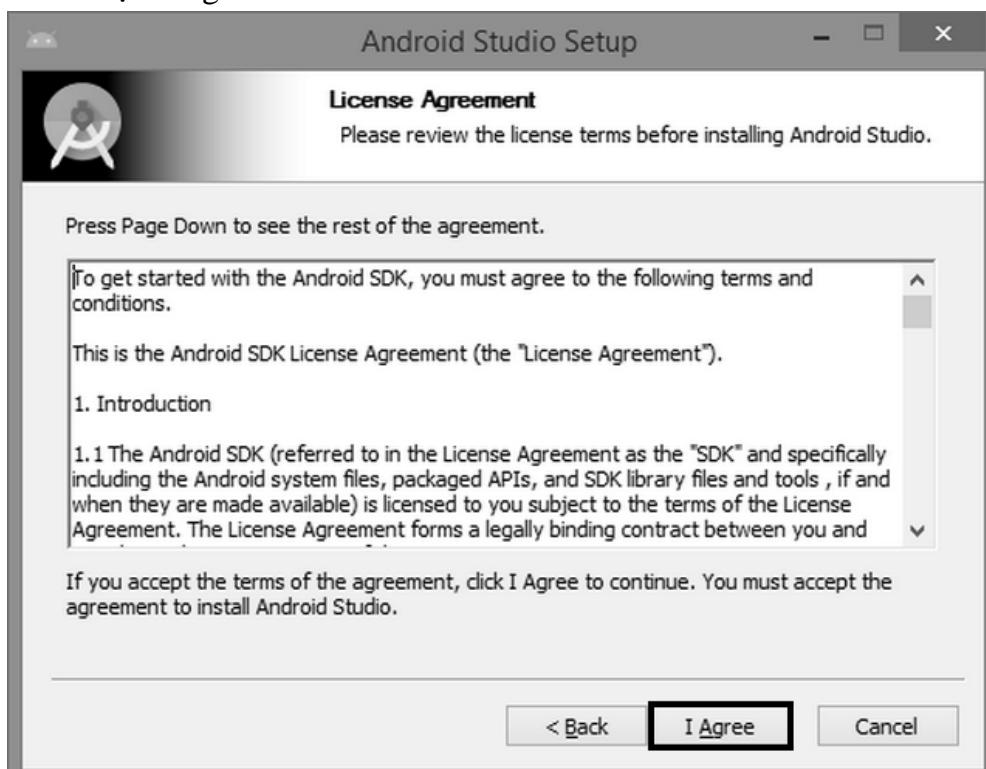
Hình 1.9

Nếu đã có JDK thì tìm đường dẫn đến nơi cài đặt. Sau đó, nhấn Next. Hộp thoại lựa chọn cấu hình cài đặt sẽ mở ra → chọn (check) đủ các thành phần như hình bên dưới → Next:



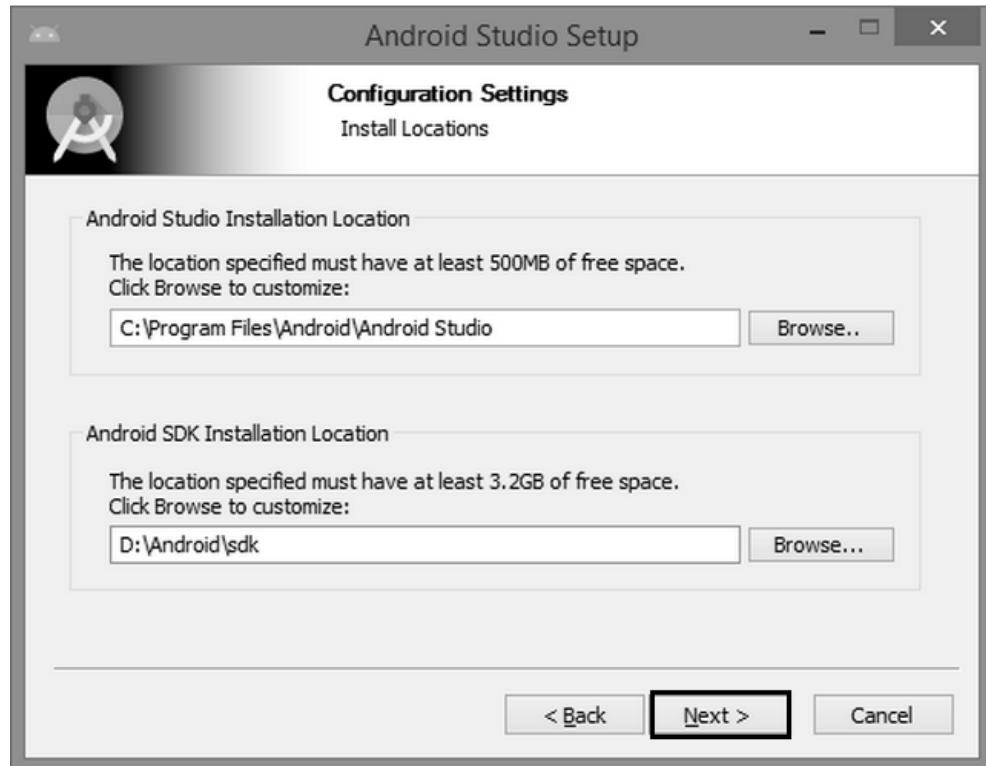
Hình 1.10

- Xuất hiện hộp loại thông báo các điều khoản và một số lưu ý khi sử dụng Android Studio → chọn I Agree:



Hình 1.11.

- Xuất hiện hộp thoại yêu cầu chọn nơi cài đặt Android Studio và Android SDK như sau:



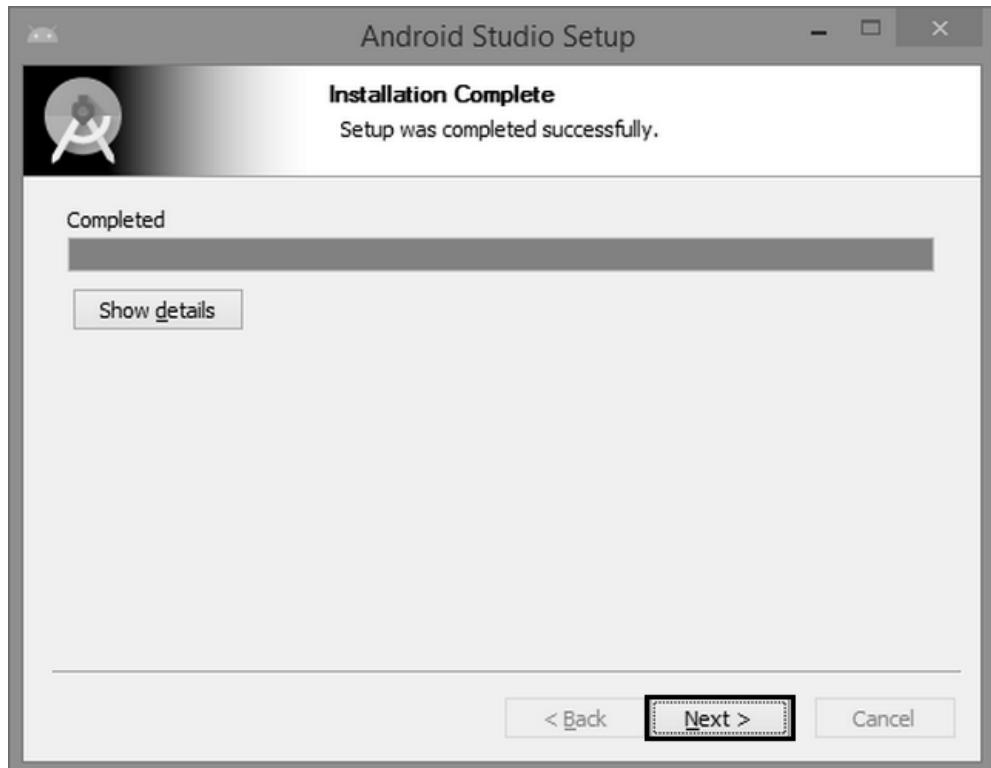
Hình 1.12.

- Sau đó, nhấn Next để tiếp tục quá trình cài đặt. Hộp thoại chọn Start Menu xuất hiện, chọn Android Studio và nhấn vào Install để tiếp tục quá trình cài đặt:



Hình 1.13.

- Khi quá trình cài đặt hoàn tất, nhấn Next để tiếp tục:



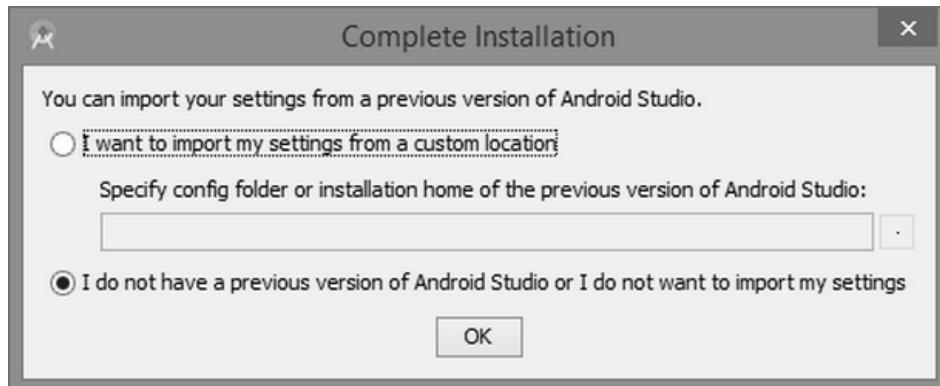
Hình 1.14.

- Hộp thoại cuối cùng của quá trình cài đặt xuất hiện, chọn Finish để kết thúc quá trình cài đặt:



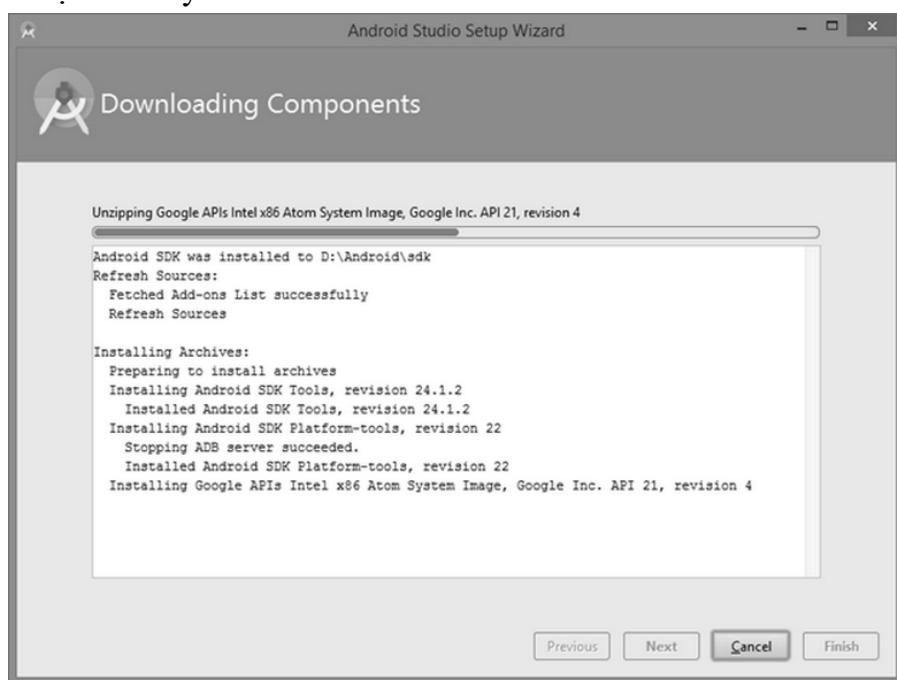
Hình 1.15.

Ở lần khởi động đầu tiên, một hộp thoại “import” bản Android Studio cũ xuất hiện, nếu bạn cài đặt mới thì chọn như hình dưới → chọn OK:



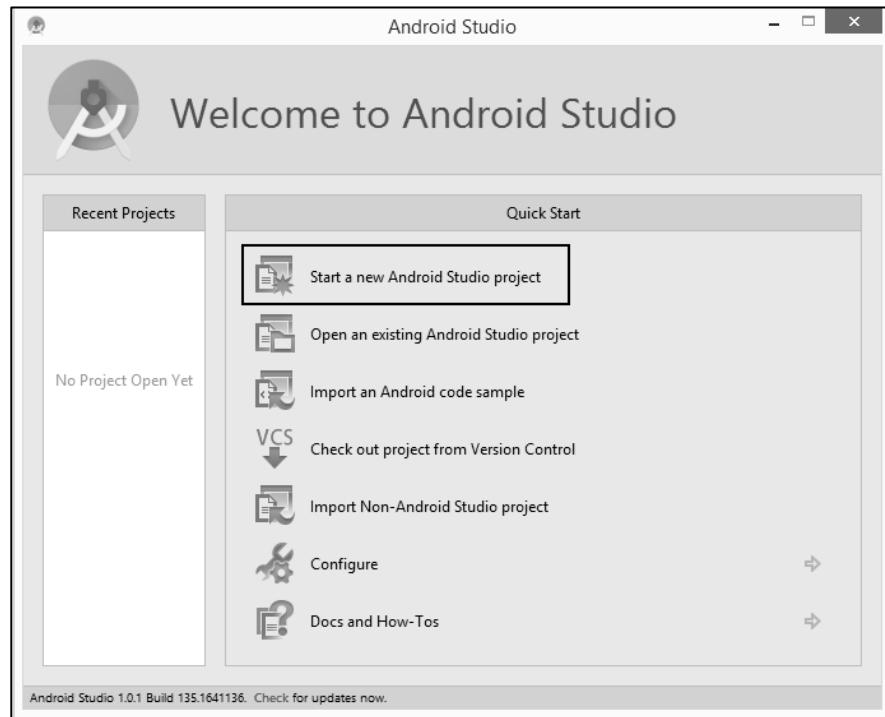
Hình 1.16.

- Vì là lần đầu tiên khởi động nên Android Studio phải cập nhật một vài thứ cho quá trình làm việc sau này:



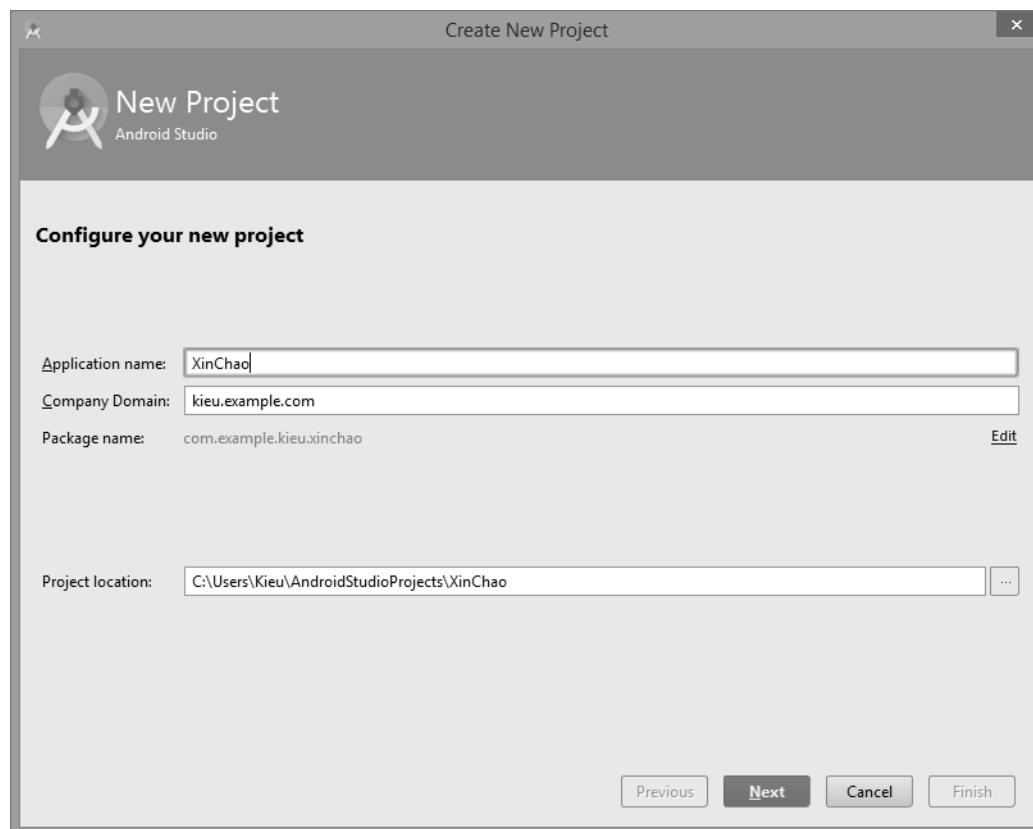
Hình 1.17.

- Khi hoàn tất quá trình cập nhật, nhấn Finish để kết thúc và hoàn thành quá trình cài đặt.
- ### 3.3. Tạo ứng dụng đầu tiên
- #### 3.3.1. Khởi tạo dự án
- Khởi chạy Android Studio, xuất hiện màn hình “Welcome to Android Studio” → chọn Start a new Android Studio project để tạo dự án Android Studio (Android Studio project) mới như sau:



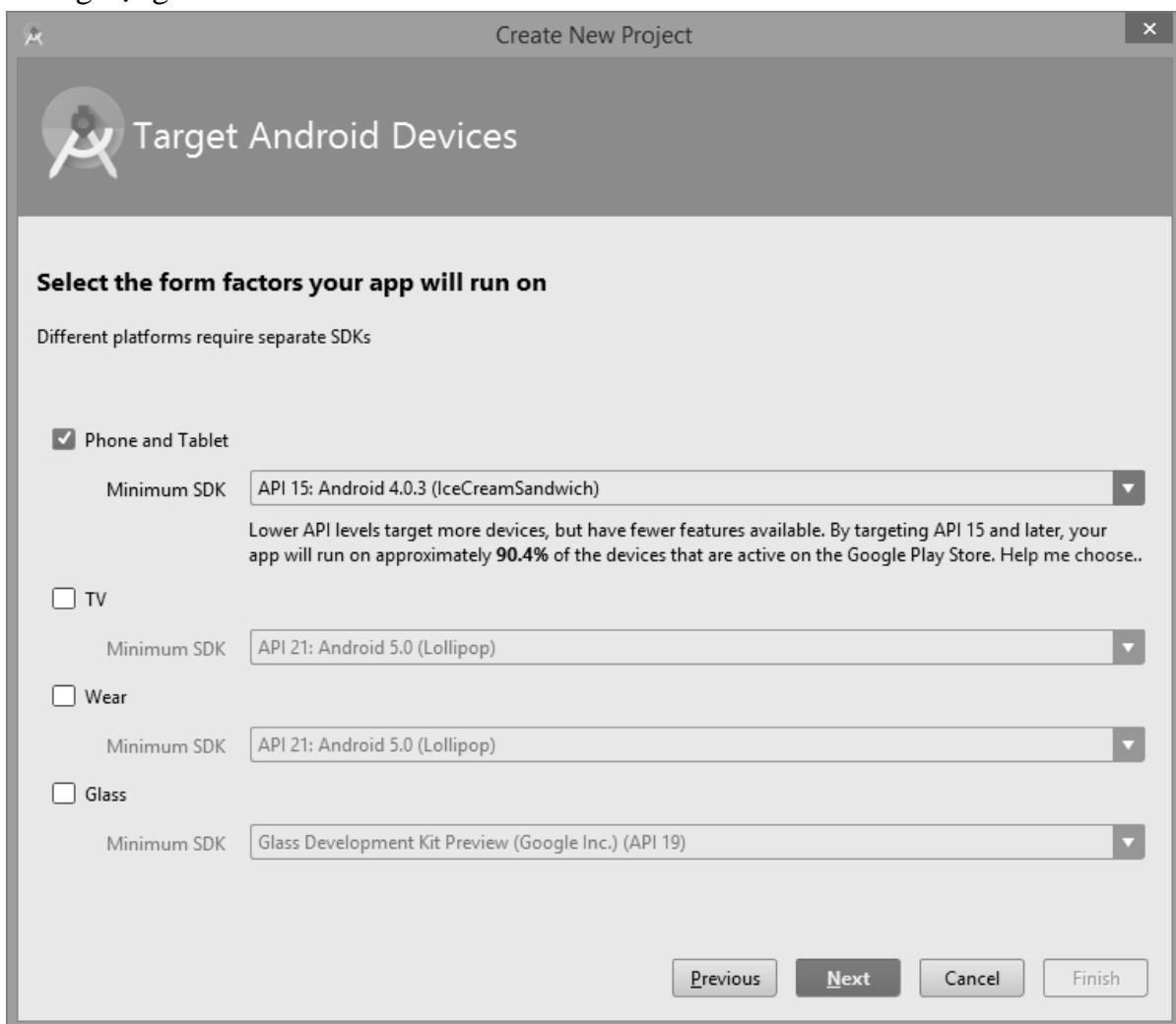
Hình 1.18.

- Hộp thoại tiếp theo xuất hiện yêu cầu đặt tên cho ứng dụng và tên domain như hình sau:



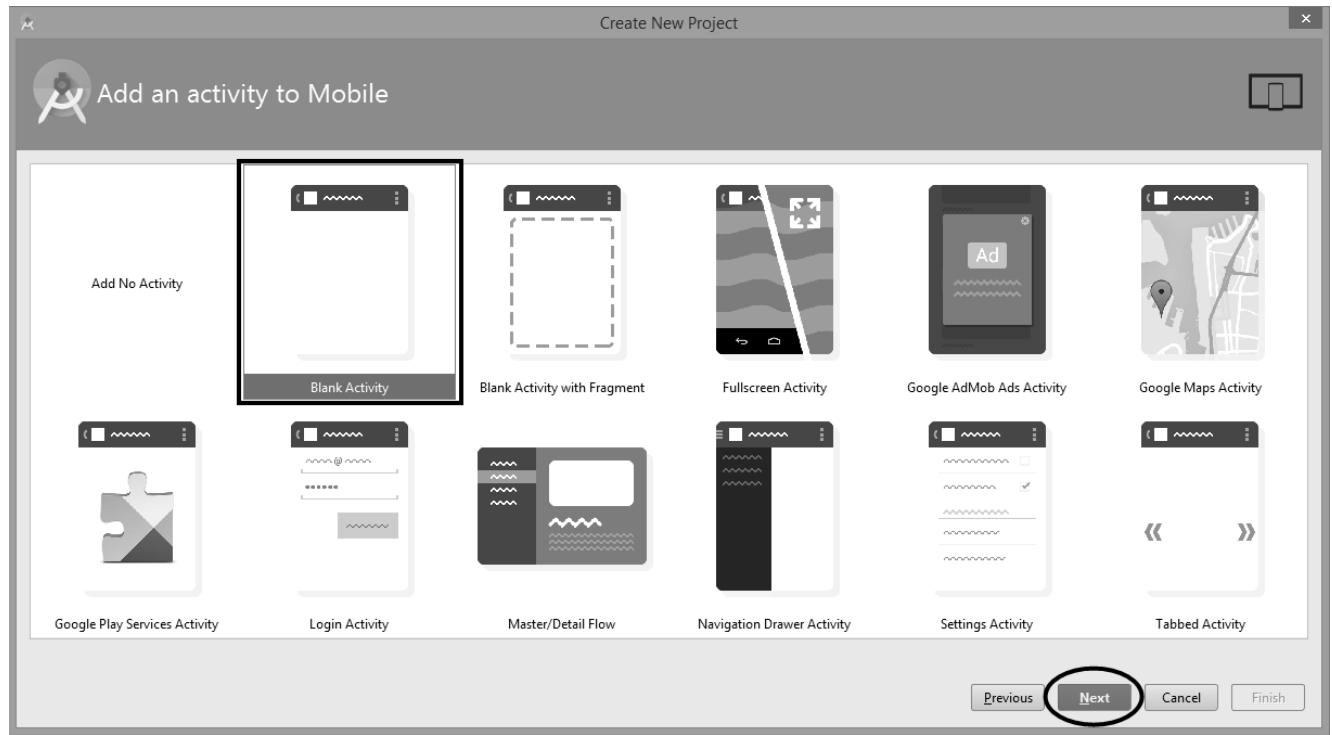
Hình 1.19.

- Trong hộp thoại này:
 - o **Application name:** là tên của ứng dụng.
 - o **Company Domain:** là tên domain của công ty kết hợp với tên của ứng dụng để tạo ra package. Sử dụng để đưa ứng dụng của bạn lên Google Play Store.
 - o **Pakage name:** Tên của gói ứng dụng của bạn. Dùng để phân biệt với các gói khác trên Google Play Store.
 - o **Project location:** Nơi lưu trữ ứng dụng của bạn.
- Sau khi nhập xong thông tin (bao gồm tên ứng dụng, domain và nơi lưu dự án) → nhấn Next để tiếp tục → Xuất hiện hộp thoại chọn thiết bị (device) mà bạn muốn phát triển ứng dụng:



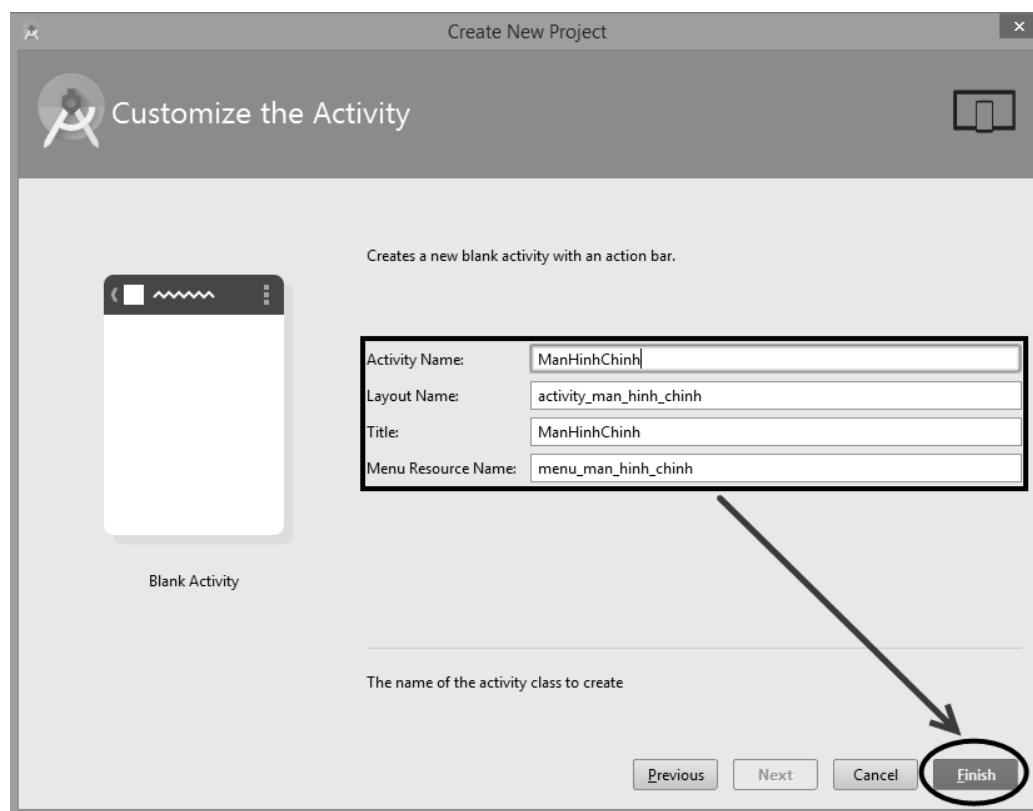
Hình 1.20.

- Chọn như hình trên nếu muốn phát triển ứng dụng trên điện thoại (phone) và tablet. Sau đó nhấn, Next để tiếp tục.
 - o Chú ý: Minimum SDK là phiên bản thấp nhất được chỉ định để chạy ứng dụng và một số các phương thức API sẽ được gọi bổ sung trong phần thư viện hỗ trợ.



Hình 1.21.

- Chọn Blank Activity và nhấn Next. Hộp thoại nhập thông tin của Activity xuất hiện như sau:

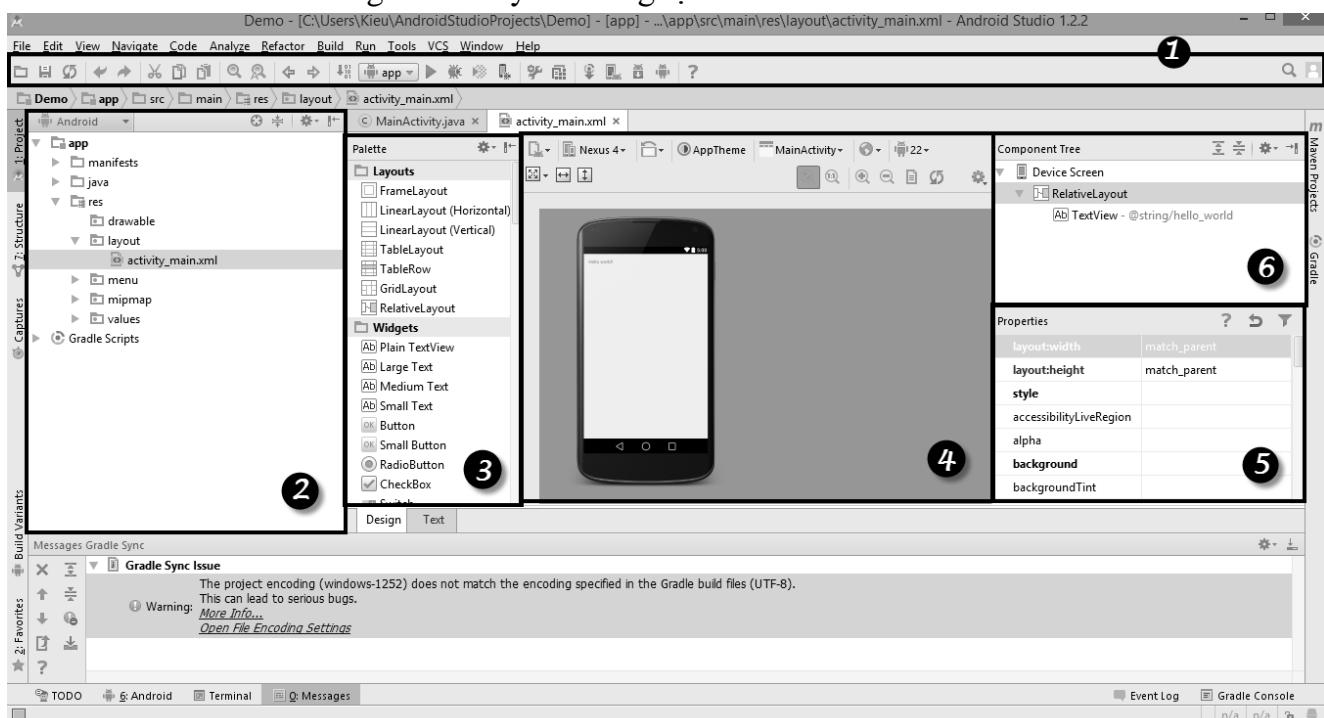


Hình 1.22.

- Trong hộp thoại này:
 - o **Activity Name**: Tên lớp lưu giữ mã nguồn.
 - o **Layout Name**: Tên tập tin XML làm giao diện cho Activity Name.
 - o **Title**: Tên tiêu đề, sẽ hiển thị khi kích hoạt activity trên thiết bị.
 - o **Menu Resource Name**: Tên tập tin xml để tạo menu cho phần mềm.
- Sau khi nhập đầy đủ thông tin, nhấn Finish để kết thúc. Chờ “building” dự án xong là hoàn tất việc tạo dự án.

3.3.2. Cấu trúc dự án

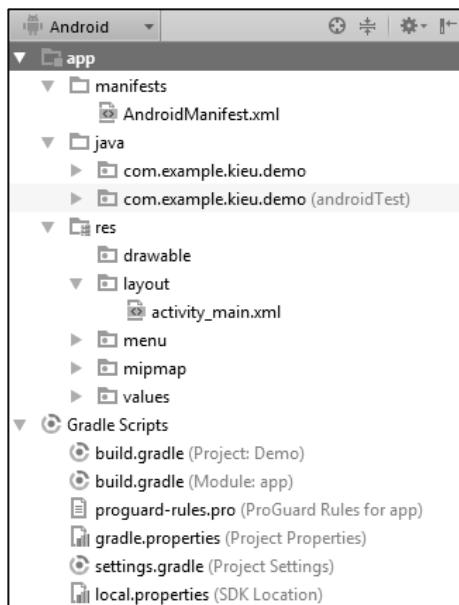
- Sau khi tạo dự án, chúng ta tiến hành mở dự án đã tạo để xem cấu trúc của nó. Và trên màn hình của chúng ta lúc này sẽ tương tự như sau:



Hình 1.23.

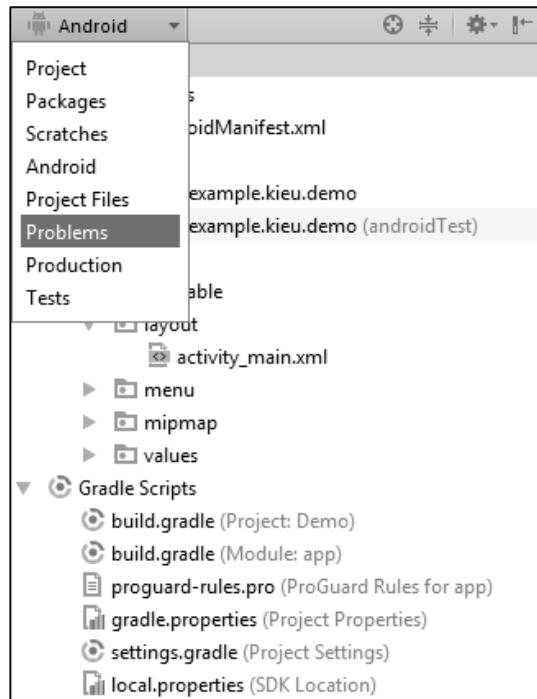
- Ở hình trên, màn hình làm việc được chia thành 7 phần – 7 phần này, chúng ta sẽ tương tác thường với chúng với các chức năng cụ thể như sau:
 - o Phần 1: thanh công cụ này rất tiện lợi. Nó giúp chúng ta thao tác nhanh các chức năng thường dùng khi lập trình, chi tiết như sau:
 - Biểu tượng dùng để mở một tập tin (file) hay một dự án (Project).
 - Biểu tượng dùng để lưu lại tất cả các tập tin trong dự án (Save All). Chúng ta cũng có thể dùng phím Ctrl + S để thực hiện chức năng này thay vì chọn biểu tượng.
 - Biểu tượng cho phép đồng bộ tập tin. Chức năng này sẽ tìm tất cả các tập tin đã thay đổi từ bên ngoài và tải lại (reload) chúng từ ổ đĩa cứng. Chúng ta có thể dùng phím Ctrl + Alt + Y để thực hiện chức năng này thay vì chọn biểu tượng này.

- Hai biểu tượng cho phép chúng ta quay lại hành động đã làm trước hoặc sau (Undo – Redo). Chúng ta cũng có thể sử dụng phím Ctrl + Z và Ctrl + Shift + Z để sử dụng chức năng này.
 - Biểu tượng để “cut” tập tin hoặc hình ảnh đến Clipboard. Có thể sử dụng phím Ctrl + X để thực hiện chức năng này thay vì chọn biểu tượng này.
 - Biểu tượng để sao chép (copy) tập tin hoặc hình ảnh đến Clipboard. Có thể sử dụng phím Ctrl + C để thực hiện chức năng này thay vì chọn biểu tượng này.
 - Biểu tượng để dán (paste) tập tin hoặc hình ảnh từ Clipboard. Có thể sử dụng phím Ctrl + V để thực hiện chức năng này thay vì chọn biểu tượng này.
 - Biểu tượng để chạy (run) ứng dụng. Chúng ta cũng có thể dùng phím Shift + F10 thay vì chọn biểu tượng này.
 - Biểu tượng để “debug” ứng dụng. Chúng ta cũng có thể dùng phím Shift + F9 thay vì chọn biểu tượng này.
 - Biểu tượng là Android Virtual Device Manager (AVD Manager) cho phép tạo và quản lý các thiết bị ảo.
 - Biểu tượng là SDK Manager có chức năng quản lý các phiên bản Android.
- Đó là các biểu tượng với các chức năng thường sử dụng trong khi lập trình, ngoài ra còn các biểu tượng khác, bạn có thể tự tìm hiểu thêm.
- Phần 2: là phần dành cho cấu trúc hệ thống của ứng dụng:



Hình 1.24.

- Chúng ta cũng có thể thay đổi cách hiển thị (thường đặt mặc định là **Android**).



Hình 1.25.

- Cấu trúc dự án có thể chia thành phần sau:

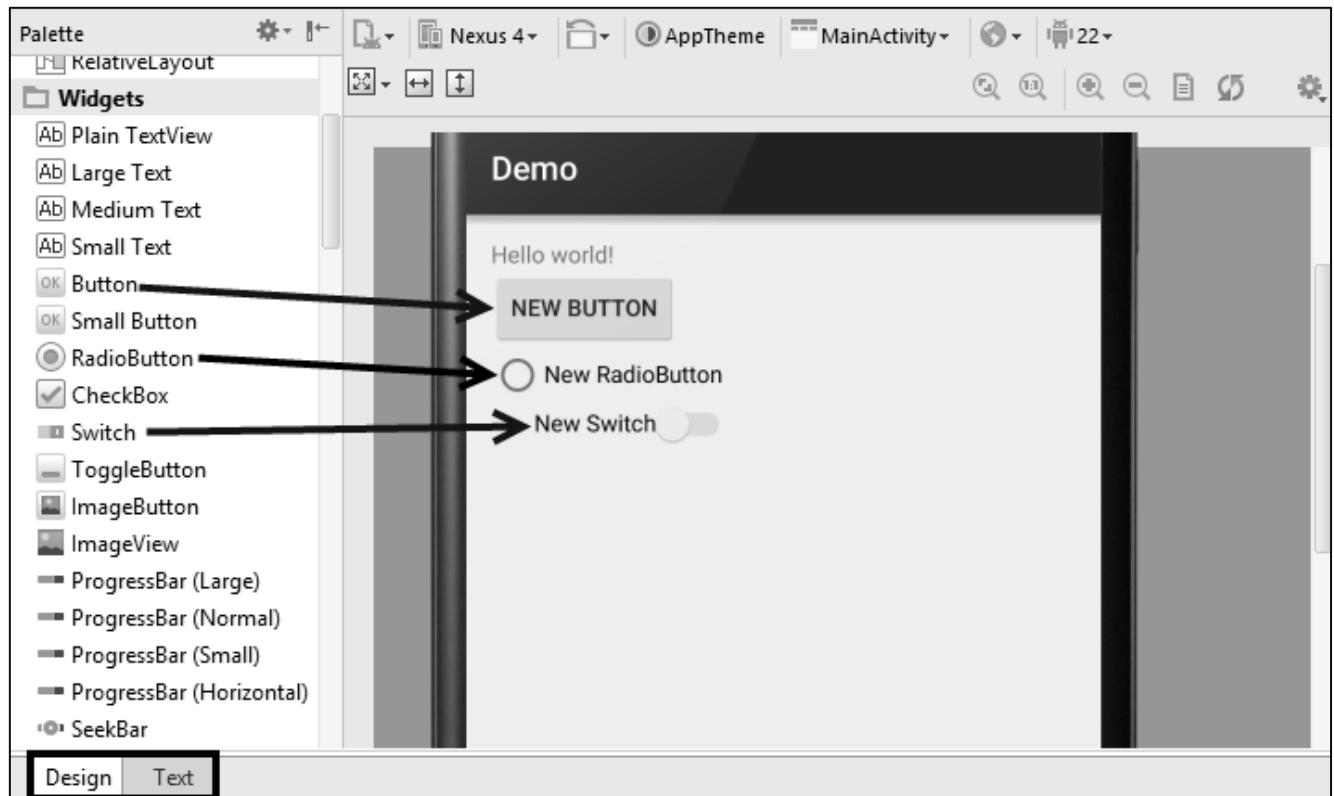
- **Thư mục:**

- ✓ src: Thư mục này chứa các file mã nguồn .java cho dự án của bạn.
- ✓ gen: Thư mục này chứa file R.java - 1 file được trình biên dịch sinh ra có khả năng tham chiếu tới tất cả các tài nguyên trong dự án. Bạn không nên chỉnh sửa file này.
- ✓ bin: Thư mục này chứa các file *.apk (Android Package file) được build bởi ADT.
- ✓ res/drawable-hdpi: Đây là thư mục chứa các đối tượng drawble được thiết kế dành cho các màn hình có độ phân giải cao.
- ✓ res/layout: Đây là thư mục chứa các file layout cho việc thiết kế giao diện.
- ✓ res/values: Đây là thư mục dành cho các file XML khác chứa 1 tập hợp các tài nguyên, ví dụ như: các định nghĩa về strings, colors.

- **Tập tin:**

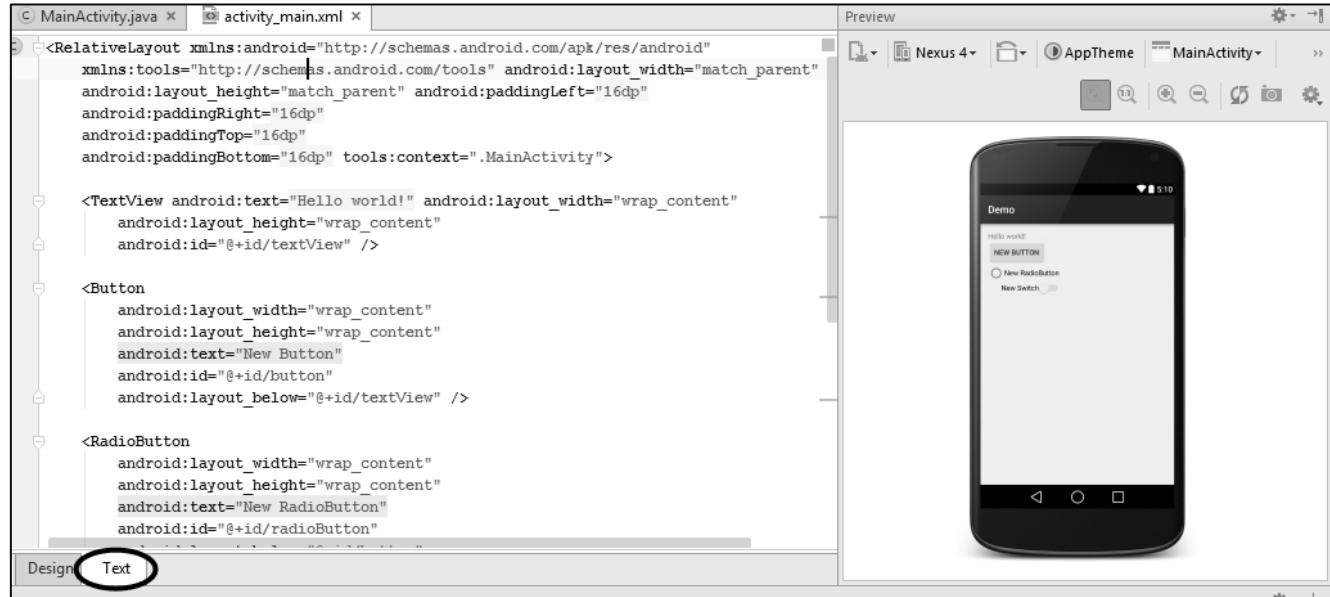
- ✓ AndroidManifest.xml: chứa thông tin cài đặt ứng dụng.

- Ngoài ra còn có thư mục assets chứa tất cả các tập tin không biên dịch như: âm thanh, hình ảnh, tập tin CSDL của ứng dụng...
- Phần 3: nó khá quan trọng cho những bạn mới bắt đầu lập trình. Đây là nơi hiển thị các điều khiển (control) mà Android hỗ trợ, cho phép bạn kéo thả trực tiếp vào phần 4 (Giao diện thiết bị) để thiết kế.



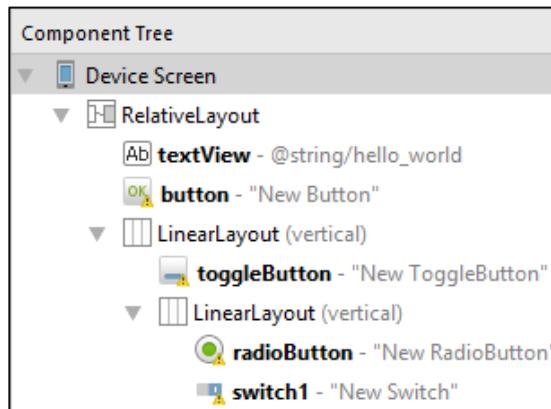
Hình 1.26.

- Chú ý ở góc trái có 2 phần là **Design** và **Text**, với:
 - Phần “Design”: cho phép thiết kế giao diện bằng cách kéo thả.
 - Phần “Text”: phép chúng ta thiết kế giao diện bằng cách viết thẻ XML.



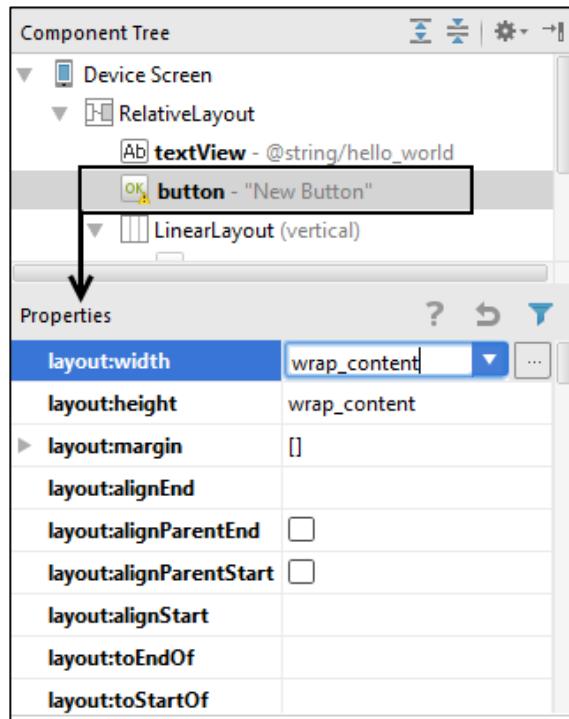
Hình 1.27.

- Phần 4: là vùng giao diện thiết bị, cho phép các điều khiển (control) kéo thả vào đây. Chúng ta có thể chọn cách hiển thị theo nằm ngang - nằm đứng, phóng to - thu nhỏ, căn chỉnh điều khiển, lựa chọn loại thiết bị hiển thị, ...
- Phần 5: Khi màn hình của bạn có nhiều điều khiển (control) thì phần 5 này rất cần thiết. Nó hiển thị giao diện theo dạng cấu trúc cây nên bạn dễ dàng quan sát và lựa chọn điều khiển khi chúng bị chồng lặp trên giao diện.



Hình 1.28.

- Phần 6: cho phép thiết lập trạng thái hay thuộc tính cho các điều khiển trên giao diện.



Hình 1.29.

3.3.3. AndroidManifest

Vai trò của tập tin AndroidManifest.xml:

- Lưu trữ thông tin tên gói ứng dụng, tồn tại duy nhất một tên gói cho mỗi ứng dụng.
 - o Ví dụ: com.htsi.myfirstapp
- Cho biết ứng dụng sử dụng các thành phần nào, mỗi thành phần được khai trong một cặp thẻ.
 - o Ví dụ: <activity>....</activity>
- Định nghĩa tiến trình quản lý các thành phần ứng dụng.
- Định nghĩa các quyền sử dụng API và truy xuất ứng dụng khác.
- Qui định các yêu cầu khi được ứng dụng khác truy xuất.
- Khai báo cấp độ API tối thiểu xây dựng ứng dụng.
- Khai báo các thư viện có liên quan.

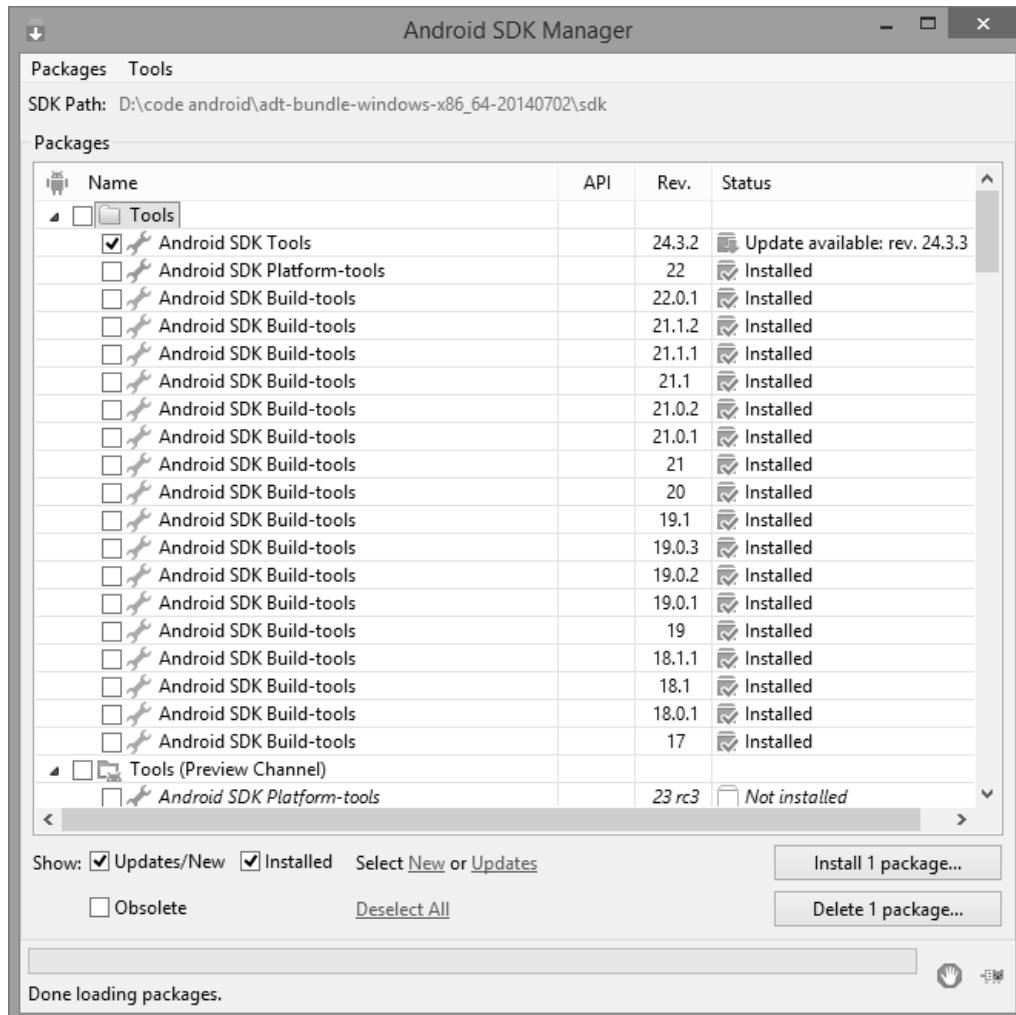
3.4. Cập nhật Android API

- Như đã đề cập ở phần cấu trúc dự án (3.3.2 – Phần 1) để cập nhật Android API, chúng ta sử dụng công cụ SDK Manager.



Hình 1.30.

- SDK Manager quản lý các phiên bản Android. Ứng với từng phiên bản là các tài liệu hướng dẫn, các ví dụ minh họa và các phương thức API... Nhấn vào biểu tượng để khởi động SDK Manager và ADT sẽ tự động cập nhật các gói mới, bạn có thể lựa chọn phiên bản Android hoặc các công cụ cần thiết và ấn Install Packages.



Hình 1.31. Tải và cài đặt bộ SDK với SDK Manager.

- **Chú ý:**

- o Các gói cần được cài đặt:

- Tools:

- Android SDK Tools.
 - Android SDK Platform-tools.
 - Android SDK Build-tools.
 - SDK Platform (phiên bản mới nhất):
 - SDK Platform.
 - ARM EABI v7a System Image.

- Extras:

- Android Support Repository.
 - Android Support Library.
 - Google Repository.
 - Google USB Driver (chỉ yêu cầu trên hệ điều hành Windows).
 - Intel x86 Emulator Accelerator (HAXM installer).

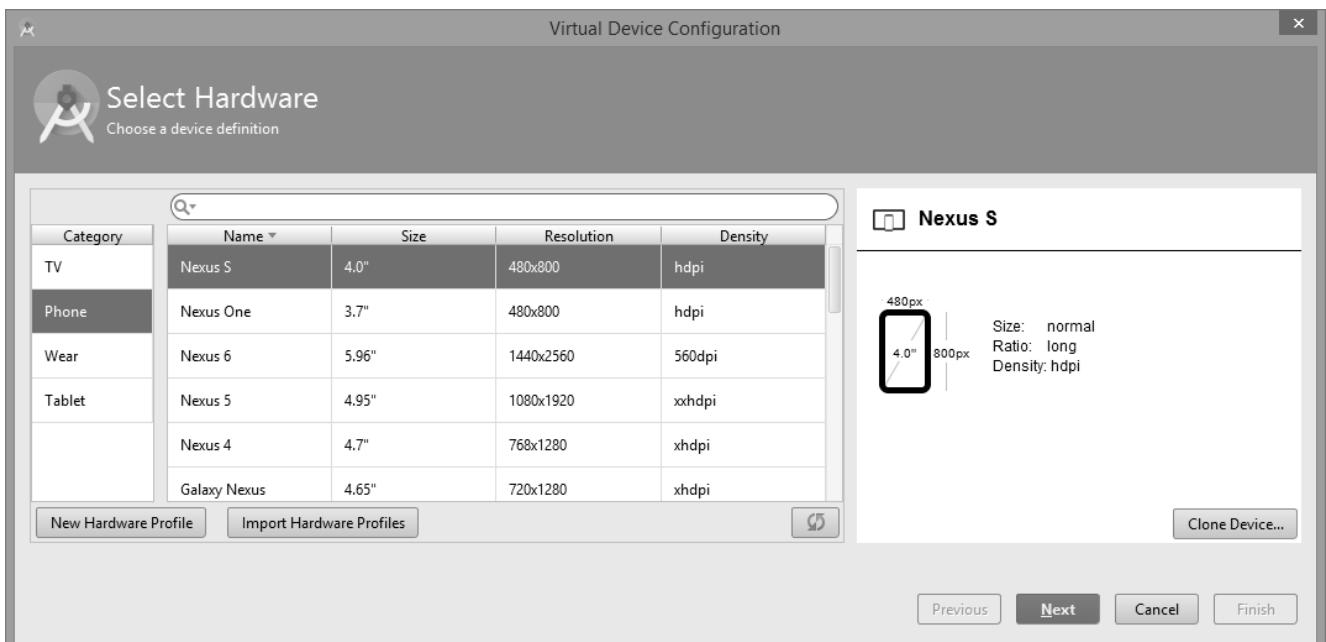
3.5. Cài đặt và sử dụng Android Virtual Device (AVD)

- Android Virtual Device Manager (AVD Manager), cho phép lập trình viên tạo và quản lý các thiết bị ảo. Để tạo thiết bị ảo, ta chọn biểu tượng AVD Manager trên thanh công cụ của Android Studio → một hộp thoại mới hiện ra → Create a virtual device:



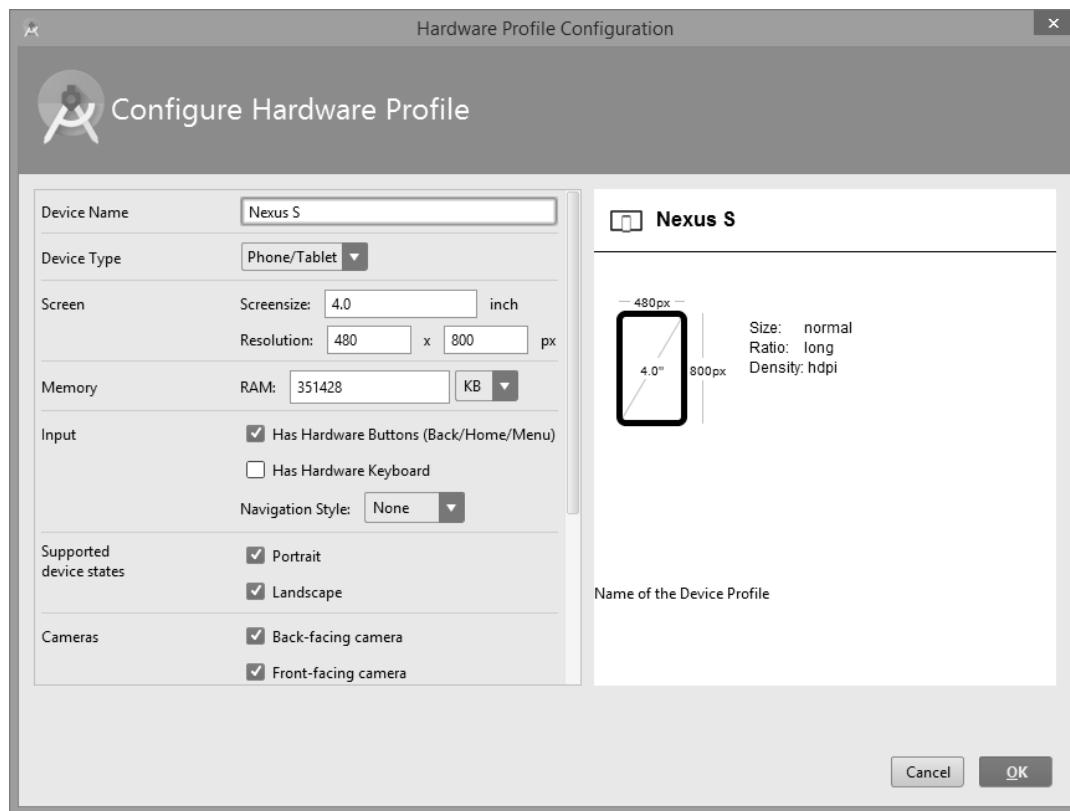
Hình 1.32.

- Xuất hiện hộp thoại “Virtual Device Configuration” cho phép bạn chọn thiết bị muốn tạo như sau:



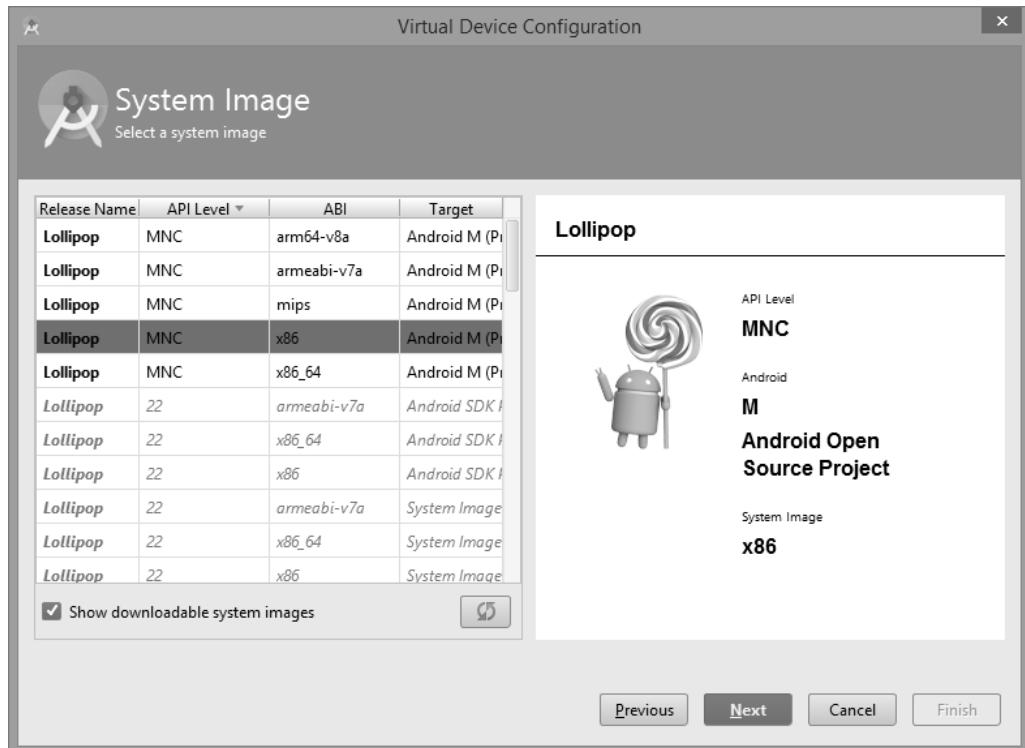
Hình 1.33.

- Chú ý: chúng ta có thể tùy chỉnh lại tên, loại thiết bị, dung lượng bộ nhớ, ... cho thiết bị muốn tạo bằng cách chọn nút Clone Device → xuất hiện hộp thoại Configure Hardware Profile như sau:



Hình 1.34.

- Trong đó:
 - o **Device Name**: tên thiết ảo cần tạo (thường đặt tên theo phiên bản).
 - o **Device Type**: loại thiết bị muốn tạo, có các lựa chọn: Phone/ Tablet, Android Wear, Android TV.
 - o **Screen**: độ rộng và độ phân giải cho màn hình.
 - o **Memory**: bộ nhớ thiết bị.
- Tiếp theo, chúng ta chọn Next để tiếp tục quá trình tạo máy ảo → xuất hiện hộp thoại cho phép chọn phiên bản hệ điều hành muốn tạo:



Hình 1.35.



Hình 1.36.

- Để cấu hình nâng cao, chúng ta chọn Show Advanced Settings.
- Chọn Finish để hoàn thành quá trình tạo máy ảo.

Bài 2

CÁC THÀNH PHẦN ỦNG DỤNG ANDROID

1. CÁC THÀNH PHẦN TRONG ỦNG DỤNG ANDROID

- Trong một ứng dụng Android thường chúng ta sẽ gặp 8 thành phần cơ bản sau:

1.1. Activity

- Trong ứng dụng Android, Activity đóng vai trò là một màn hình, nơi người dùng có thể tương tác với ứng dụng, ví dụ: chụp hình, xem bản đồ, gửi mail...
- Một ứng dụng có thể có một hoặc nhiều Activity, Activity được khởi chạy đầu tiên khi ứng dụng hoạt động được gọi là “MainActivity”.
- Activity có thể hiển thị ở chế toàn màn hình, hoặc ở dạng cửa sổ với một kích thước nhất định.
- Các Activity có thể gọi đến các Activity khác, Activity được gọi sẽ nhận được tương tác ở thời điểm đó.

1.2. Intent

- Intent là đối tượng mang thông điệp, cho phép tạo ra các yêu cầu hành động giữa các thành phần trong ứng dụng, hoặc giữa các ứng dụng với nhau.
- Được sử dụng nhiều trong 3 trường hợp sau:
 - o Khởi động Activity.
 - o Khởi động Service.
 - o Chuyển phát thông tin cho Broadcast Receiver.

1.3. View

- View được sử dụng để tạo ra các điều khiển trên màn hình cho phép nhận các tương tác từ người dùng cũng như hiển thị các thông tin cần thiết.
- View bao gồm hai dạng:
 - o View: các điều khiển đơn lẻ.
 - o ViewGroup: tập hợp nhiều điều khiển đơn lẻ.



Hình 2.1. View.

1.4. Broadcast Receiver

- Thành phần ứng dụng cho phép truyền tải các thông báo trên phạm vi toàn hệ thống. Không có giao diện nhưng có thể thực hiện thông báo qua thanh trạng thái.
- Broadcast Receiver truyền thông báo ở hai dạng:
 - o Hệ thống: các thông báo được truyền trực tiếp từ hệ thống như: tắt màn hình, pin yếu, thay đổi kết nối...
 - o Ứng dụng: xây dựng các truyền thông báo đến các thành phần trong ứng dụng như: khởi động Service, tải nội dung đến ứng dụng...

1.5. Service

- Service được sử dụng để thực thi các tác vụ cần nhiều thời gian, thực hiện ở chế độ ngầm và thường không cần giao diện hiển thị.
- Service được chạy ngầm, tức là ứng dụng vẫn chạy nhưng không hiển thị giao diện tương tác.
- Một số tác vụ cần thực hiện bằng Service:
 - o Trình diễn các tập tin đa truyền thông như nhạc, phim...
 - o Kết nối và thực hiện tải các nội dung thông qua Internet.
 - o Truy xuất đọc ghi tập tin.

1.6. Content Provider

- Content Provider xây dựng cách thức truy xuất tập hợp các dữ liệu ứng dụng, dữ liệu có thể lưu trữ ở nhiều dạng như: SQLite, tập tin, tài nguyên Web hoặc bất kỳ thư mục lưu trữ nào.
- Có thể sử dụng Content Provider để xây dựng các ứng dụng sử dụng chung nguồn tài nguyên hoặc sử dụng riêng.
- Trong Android, một số Content Provider được xây dựng sẵn:
 - o Danh bạ.
 - o Tài nguyên đa truyền thông.
 - o Lịch.

1.7. Context

- Context thuộc gói android.content (android.content.Context).
- Là một lớp cơ bản chứa hầu hết thông tin về môi trường ứng dụng của android, có nghĩa là mọi thao tác, tương tác với hệ điều hành đều phải qua lớp này.
- Nó cung cấp các phương thức để các lớp khác có thể tương tác với hệ thống Android.
- Nó cho phép truy cập tới các nguồn tài nguyên (resources) đã được định nghĩa và các lớp khác. Có thể nói Context là một lớp ở mức ứng dụng (Application level- liên quan tới hệ thống).
- Hầu hết các lớp có liên quan tới UI (layout, button, textview, imageview, listview,...) đều phải “super” tới Context vì bản thân Context đảm nhiệm việc truy cập tài nguyên (R.id, R.layout,...). Nếu chúng ta không tham chiếu tới lớp Context thì không thể dùng tới các tài nguyên mà chúng ta đã tạo ra.
- Tóm lại, Context giúp chúng ta dễ dàng truy cập và tương tác tới các tài nguyên của hệ thống, các thông tin, các dịch vụ (services), các thông số cấu hình, dữ liệu, danh bạ, cuộc gọi, kết nối, chế độ rung (vibrator), ...

1.8. Notification

- Thành phần cho phép gửi các thông báo đến người dùng mà không chiếm quá nhiều việc điều khiển của người dùng trên thiết bị. Ví dụ, khi nhận được một tin nhắn hay một thư điện tử, thiết bị sẽ sử dụng Notification để thông báo người dùng thông qua nhạc chuông, đèn nền, thể hiện biểu tượng trên thanh tác vụ...
- Notification được xây dựng cho mục đích gửi các thông báo đến người dùng thông qua thanh trạng thái.
- Giao diện Notification không thuộc giao diện ứng dụng, nhưng có thể tùy chỉnh giao diện Notification thông qua các phương thức có sẵn.



Hình 2.2. Notification.

2. TƯƠNG THÍCH THIẾT BỊ

- Ứng dụng Android được viết bằng ngôn ngữ Java và biên dịch, đóng gói cùng các tập tin tài nguyên thành tập tin *.apk.
- Cài đặt trên thiết bị theo đường dẫn data/app/<Tên đóng gói>, được chứa trong Sandbox và được hiểu:
 - o Mỗi ứng dụng là một dạng “người dùng” khác nhau.
 - o Mỗi ứng dụng được cấp một ID, do đó chỉ duy nhất ứng dụng mới có thể truy xuất các tập tin liên quan đến ứng dụng đó.
 - o Ứng dụng thực thi riêng biệt trên từng máy ảo.
 - o Tiết trình Linux được cấp phát khi bắt đầu thành phần ứng dụng được gọi thực thi, và thu hồi khi chấm dứt hoạt động.
 - o Các ứng dụng có cùng ID và chứng chỉ (Certificate) có thể truy xuất tài nguyên của nhau, hoặc xin quyền nếu truy xuất hệ thống.
- Tính tương thích ứng dụng với thiết bị bao gồm:
 - o Trang bị tính năng của thiết bị.
 - o Phiên bản hệ điều hành.
 - o Kích thước màn hình.
- Trang bị tính năng của thiết bị:
 - o Mỗi tính năng phần cứng và phần mềm trên thiết bị Android được cung cấp một ID, qui định thiết bị đó có được trang bị tính năng đó hay không. Ví dụ:
 - FEATURE_SENSOR_COMPASS: tính năng la bàn.
 - FEATURE_APP_WIDGET: tính năng gắn Widget.
- Phiên bản hệ điều hành:
 - o Mỗi phiên bản kế tiếp được bổ sung hoặc lược bỏ các phương thức API, cần thực thi khai báo các thông tin phiên bản để sử dụng đầy đủ các tính năng cho ứng dụng.
 - o Ví dụ:
 - Android ICS 4.0: bổ sung API cho Calendar
 - Android Kit Kat 4.4: bổ sung API cho WirelessPrint
- Kích thước màn hình:
 - o Ứng dụng Android cần tương thích với nhiều kích cỡ thiết bị, được phân chia thành hai thuộc tính:
 - Kích thước vật lý màn hình.
 - Độ phân giải (mật độ điểm ảnh).

3. ACTIVITY VÀ ĐỘ ƯU TIÊN ỨNG DỤNG

3.1. Xây dựng Activity

Trong một ứng dụng Android thì người dùng thao tác với các Activity, và một ứng dụng thì không thể chỉ có một Activity mà cần có nhiều Activity. Và trong lập trình, bạn cần chuyển

đổi qua lại giữ các Activity với nhau đồng thời truyền dữ liệu trừ Activity trước sang Activity mới để có thể xử lý.

- Mỗi Activity đại diện cho một màn hình ứng dụng. Ta sẽ thực hiện tạo Activity cho ứng dụng theo các bước sau:
 - o Tạo mới lớp kế thừa từ lớp Activity.
 - o Thực thi các phương thức quản lý trạng thái Activity.
 - o Xây dựng giao diện trong tài nguyên res/layout.
 - o Khai báo Activity trong tập tin AndroidManifest.xml.

Ví dụ:

```
public class MyFirstActivity extends Activity{  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
    }  
}
```

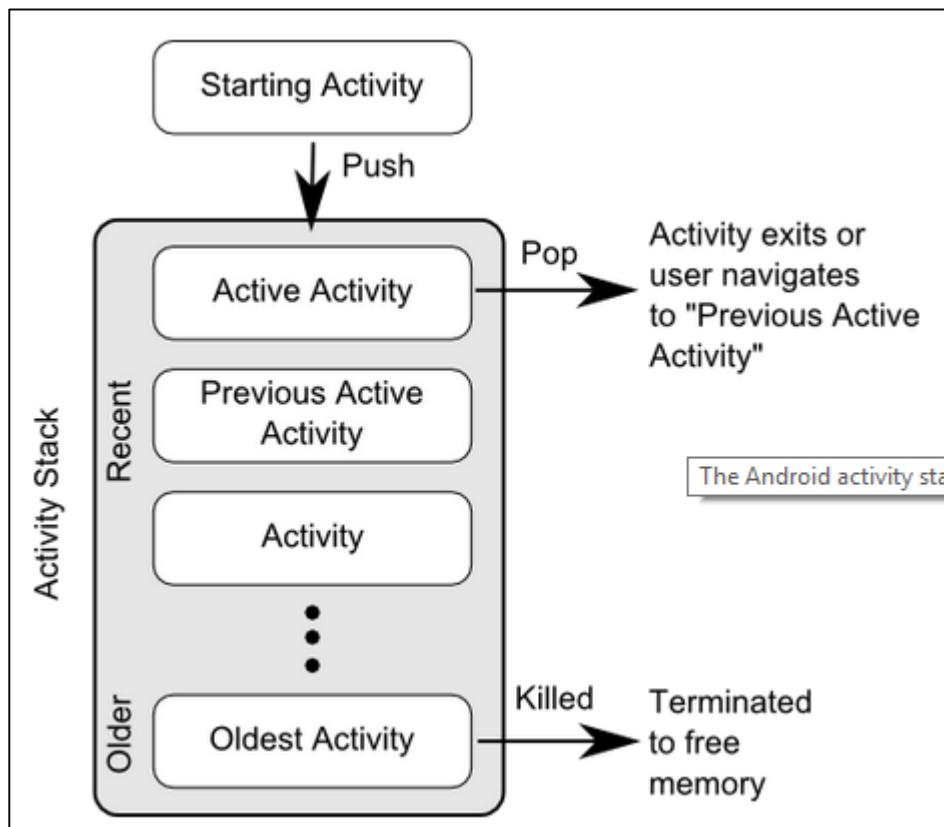
- Đăng kí Activity trong tập tin AndroidManifest.xml:

```
<application  
        android:allowBackup="true"  
        android:icon="@drawable/ic_launcher"  
        android:label="@string/app_name"  
        android:theme="@style/AppTheme" >  
  
        <activity android:name="com.your-company.MyFirstActivity" >  
            <intent-filter>  
                <action android:name="android.intent.action.MAIN" />  
                <category android:name="android.intent.category.LAUNCHER" />  
            </intent-filter>  
        </activity>  
</application>
```

3.2. Quản lý trạng thái Activity (Managing the Activity Lifecycle)

- Activity là thành phần quan trọng nhất và đóng vai trò chính trong xây dựng ứng dụng Android. Hệ điều hành Android quản lý Activity theo dạng stack: khi một Activity mới được khởi tạo, nó sẽ được xếp lên đầu của stack và trở thành running activity, các Activity trước đó sẽ bị tạm dừng và chỉ hoạt động trở lại khi Activity mới được giải phóng.

- Activity Stack là gì?
 - o Tương tự như các ngôn ngữ lập trình khác, Activity Stack hoạt động theo cơ chế LIFO (LAST IN FIRST OUT).
 - o Mỗi một Activity mới được mở lên nó sẽ ở bên trên Activity cũ, để trở về Activity thì bạn chỉ cần nhấn nút “Back” để trở về hoặc viết lệnh. Tuy nhiên nếu bạn nhấn nút Home rồi thì sẽ không thể dùng nút “Back” để quay lại màn hình cũ được.

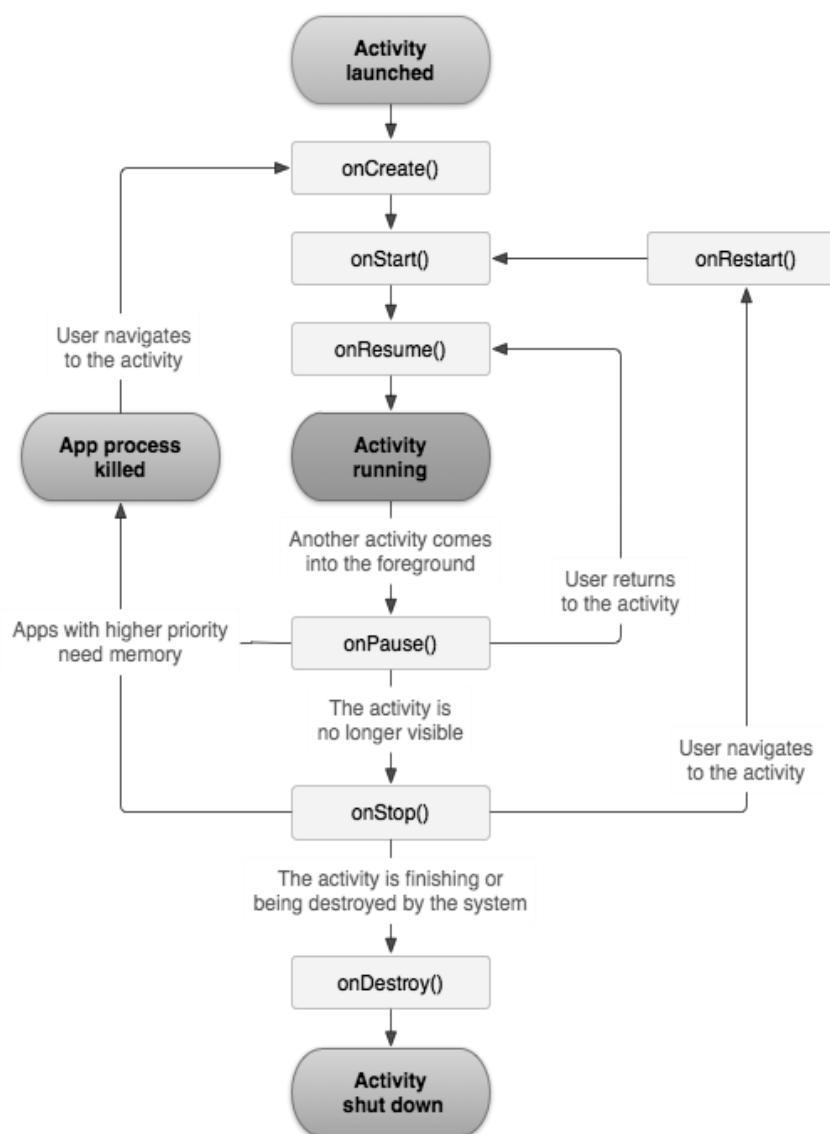


Hình 2.3. Activity Stack

- Một Activity cơ bản có 4 trạng thái:
 - o **Active**: Activity đang hiển thị trên màn hình (foreground).
 - o **Paused**: Activity vẫn hiển thị (visible) nhưng không thể tương tác (lost focus). VD: một activity mới xuất hiện hiển thị giao diện đè lên trên activity cũ, nhưng giao diện này nhỏ hơn giao diện của activity cũ, do đó ta vẫn thấy được 1 phần giao diện của activity cũ nhưng lại không thể tương tác với nó.
 - o **Stopped**: Activity bị thay thế hoàn toàn bởi Activity mới sẽ tiến đến trạng thái stopped.
 - o **Inactive**: Khi hệ thống bị thiếu bộ nhớ, nó sẽ giải phóng các tiến trình theo nguyên tắc ưu tiên. Các Activity ở trạng thái stopped hoặc paused cũng có thể bị giải phóng và khi nó được hiển thị lại thì các Activity này phải khởi động lại hoàn toàn và phục hồi lại trạng thái trước đó.

3.3. Vòng đời của Activity (Activity Lifecycle)

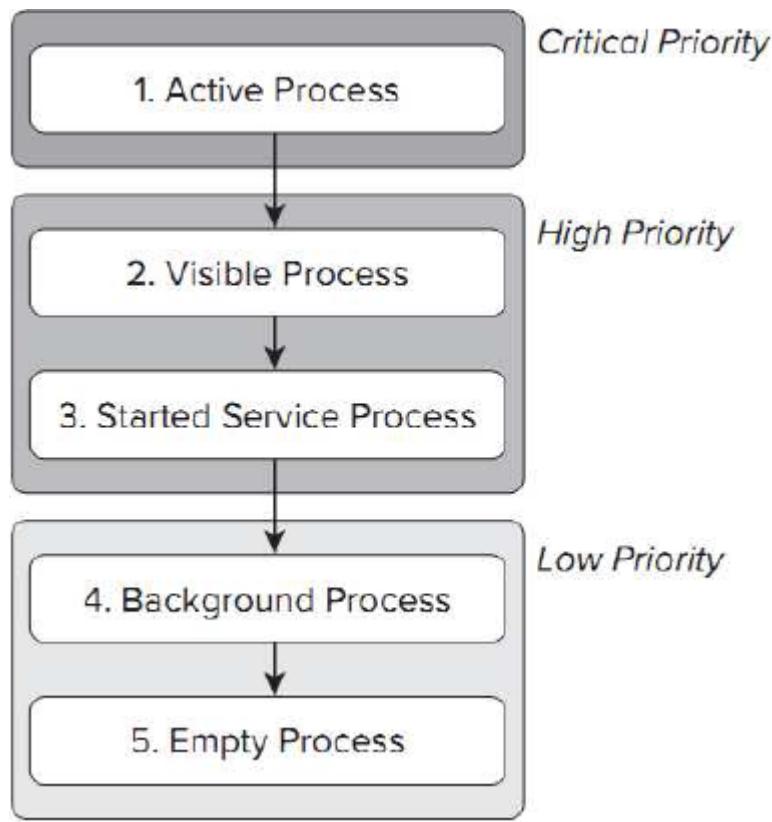
- **Entire lifetime:**
 - o Từ phương thức onCreate() cho tới onDestroy(): từ lúc Activity được gọi ra cho đến lúc bị huỷ.
- **Visible lifetime:**
 - o Từ sau khi gọi onStart() cho tới lúc gọi onStop(): trong trường hợp này ta vẫn có thể thấy màn hình Activity.
- **Foreground lifetime:**
 - o Xảy ra từ khi gọi onResume() cho tới lúc gọi onPause(): trong suốt thời gian này Activity luôn nằm ở trên cùng và ta có thể tương tác được với nó.



Hình 2.4. Vòng đời của Activity.

3.4. Độ ưu tiên trong ứng dụng

- Android quản lý các Ứng dụng dựa trên độ ưu tiên.
- Nếu hai ứng dụng có cùng trạng thái thì ứng dụng nào đã chạy lâu hơn sẽ có độ ưu tiên thấp hơn.
- Nếu ứng dụng đang chạy một Service hay Content Provider do một ứng dụng khác hỗ trợ thì sẽ có cùng độ ưu tiên với ứng dụng đó.
- Các ứng dụng sẽ bị đóng mà không có sự báo trước.



Hình 2.4. Cách phân định độ ưu tiên.

- **Foreground process**: là process của ứng dụng hiện thời đang được người dùng tương tác.
 - **Visible process**: là process của ứng dụng mà activity đang hiển thị đối với người dùng (onPaused() của activity được gọi).
 - **Service process**: là Service đang chạy.
 - **Background process**: là process của ứng dụng mà các activity của nó không hiển thị với người dùng (onStopped() của activity được gọi).
 - **Empty process**: process không có bất cứ 1 thành phần nào active.
-
- Theo chế độ ưu tiên thì khi cần tài nguyên, Android sẽ tự động kill process, trước tiên là các empty process.

Bài 3

GIAO DIỆN NGƯỜI DÙNG VÀ XỬ LÝ SỰ KIỆN

1. GIAO DIỆN NGƯỜI DÙNG

Giao diện người dùng là một trong những yếu tố quan trọng, quyết định sự thành công của một ứng dụng Android. Ứng dụng Android muốn thành công thì phải có giao diện trực quan, dễ hiểu và dễ sử dụng. Ở bài này, chúng ta sẽ tìm hiểu cấu trúc giao diện, các thành phần trên giao diện và các thuộc tính của chúng.

Giao diện được tạo bởi nhiều tài nguyên như: layout, các điều khiển (control), tài nguyên hình ảnh, màu sắc, v.v... Và tất cả các tài nguyên dùng để thiết kế giao diện sẽ được lưu trong thư mục res/.

1.1. Layout

- Các tài nguyên layout được lưu trữ trong thư mục res/layout.
- Có thể có nhiều thư mục layout theo từ hạn định khác nhau:
- Ví dụ: layout-land, layout-xhdpi...
- Truy xuất: bao gồm 2 cách thức:
 - o Java: R.layout.<tên tài nguyên>.
 - o XML: @[package:]layout/<tên tài nguyên>.

- Các định dạng Layout

Layout bao gồm những lớp được mở rộng từ lớp ViewGroup mà ở đó ta có thể sắp xếp và bố trí các điều khiển cho một giao diện. Tùy thuộc vào cách bố trí của mỗi người mà sẽ có các giao diện khác nhau và tất nhiên cần cân bằng giữa tính tiện dụng và tính thẩm mỹ. Android cũng cấp một số các lớp Layout cho phép chúng ta có thể sử dụng hoặc tùy chỉnh tùy thuộc vào từng ứng dụng.

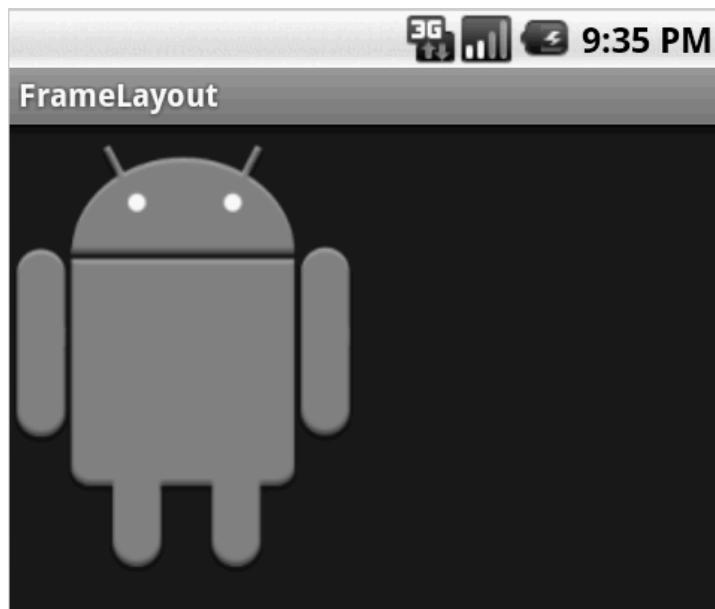
Bao gồm các lớp kế thừa từ ViewGroup: AbsoluteLayout (Deprecated), AdapView (ListView, Gridview...), DrawerLayout, FragmentBreadCrumbs, FrameLayout, GridLayout, LinearLayout, PagerTitleStrip, RelativeLayout, SlidingDrawer, SlidingPaneLayout, SwipeRefreshLayout, ViewPager.

1.1.1. Frame Layout

- Sử dụng trong các trường hợp xây dựng bố cục tổ chức hiển thị một đối tượng duy nhất.
- Đối tượng mặc định vị trí top-left trên FrameLayout, có thể sử dụng thuộc tính Gravity để thiết lập lại vị trí.

- Frame Layout có thể chứa nhiều View và các View này có thể sắp chồng lên nhau. Và các View nằm dưới có thể bị các View nằm trên che khuất. Vì vậy, thường Frame Layout chỉ chứa một View.
- Ví dụ khai báo:

```
<?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="match_parent" >
    <ImageView
        android:id="@+id/imageView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:src="@drawable/android" />
</FrameLayout>
```



Hình 3.1. FrameLayout.

- Các đối tượng kế thừa phổ biến của FrameLayout:
 - o **ViewFlipper**: đối tượng cho phép thực hiện hiển thị các đối tượng ở chế độ phân trang, chỉ hiển thị một đối tượng ở một thời điểm.
 - Ví dụ khai báo:

```
<ViewFlipper
    android:layout_width="match_parent"
    android:layout_height="match_parent" >
</ViewFlipper>
```

- Các phương thức sử dụng:

- startFlipping
- setAutoStart
- showNext
- showPrevious

- **ScrollView**: đối tượng cho phép thực hiện hiển thị các đối tượng ở chế độ cuộn màn hình, chỉ cho phép chứa một đối tượng ở một thời điểm.

- Ví dụ khai báo:

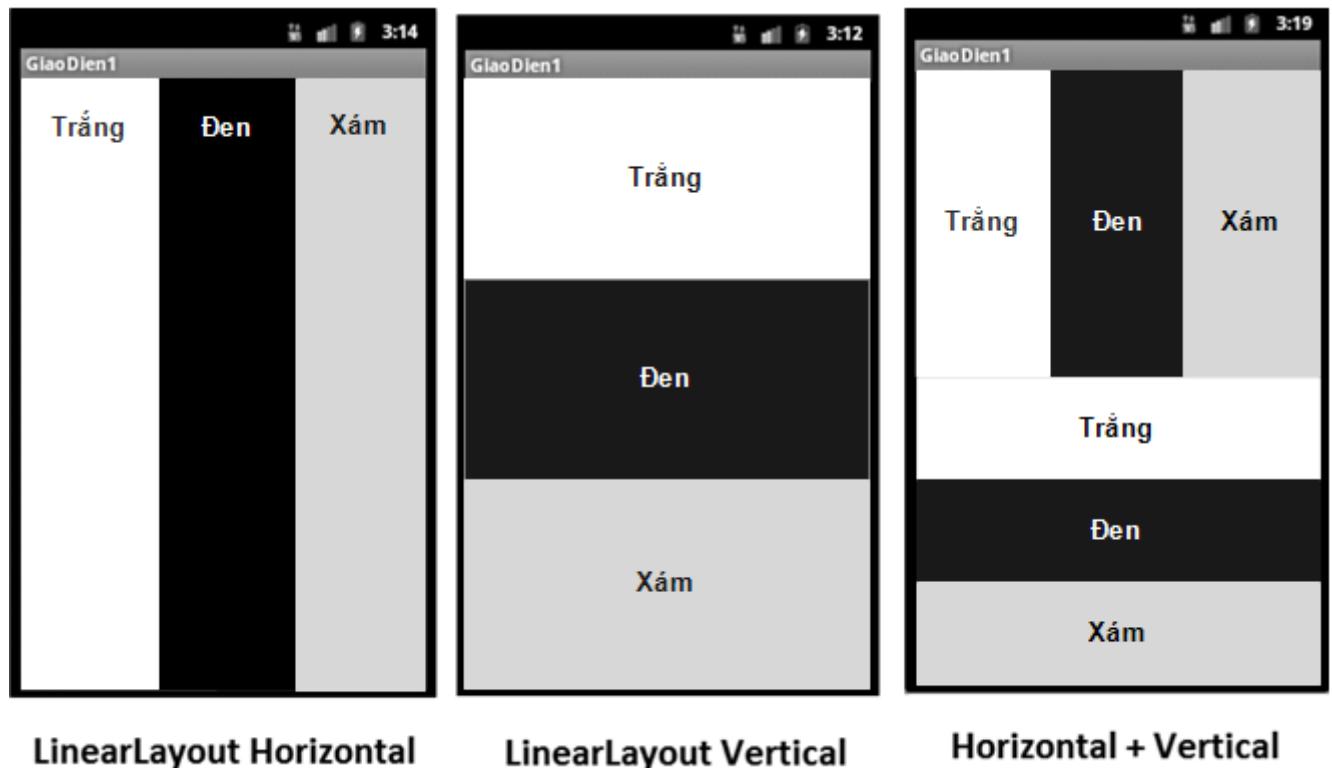
```
<ScrollView  
    android:layout_width="match_parent"  
    android:layout_height="match_parent" >  
</ScrollView>
```

- Các phương thức sử dụng:

- setFillViewport
- scrollBy
- scrollTo
- smoothScrollBy
- smoothScrollTo

1.1.2. Linear Layout

- Giao diện một hướng, có nghĩa là các View sẽ được sắp xếp lên Layout theo một hướng nhất định hoặc theo chiều ngang (từ trái qua phải) hoặc theo chiều dọc (từ trên xuống dưới). Thuộc tính weight của Layout sẽ cho phép điều chỉnh kích thước của các View dựa trên mối quan hệ của chúng (cùng một hàng chẵng hạn).
- Được xem dạng là Layout dễ sử dụng nhất, chúng cho phép chúng ta tạo ra các giao diện đơn giản bằng cách sắp xếp các View theo một chiều duy nhất. Điều này gây khó khăn khi chúng ta cần thiết kế các giao diện phức tạp, do đó LinearLayout thường được sử dụng trong một Layout khác. Ví dụ, ở đây ta thiết kế khác LinearLayout lồng nhau, một Layout dạng Horizontal để chứa hai Button và một Layout dạng Vertical cho phép hai Button nằm trên ListView.



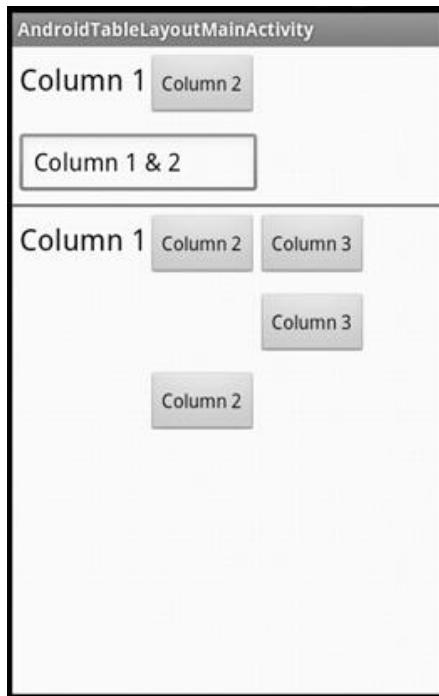
Hình 3.2. Các dạng LinearLayout.

- Ví dụ khai báo:

```
<LinearLayout  
    xmlns:android="http://schemas.android.com/apk/res/android"  
    android:layout_width="match_parent"  
    android:layout_height="match_parent"  
    android:orientation="vertical" >  
</LinearLayout>
```

1.1.3. Table Layout

- Table Layout là đối tượng layout kế thừa từ LinearLayout, cho phép hiển thị các đối tượng theo nhiều dòng (TableRow).
- Mỗi dòng có thể chứa nhiều View, mỗi View được xem là một cột.



Hình 3.3. TableLayout.

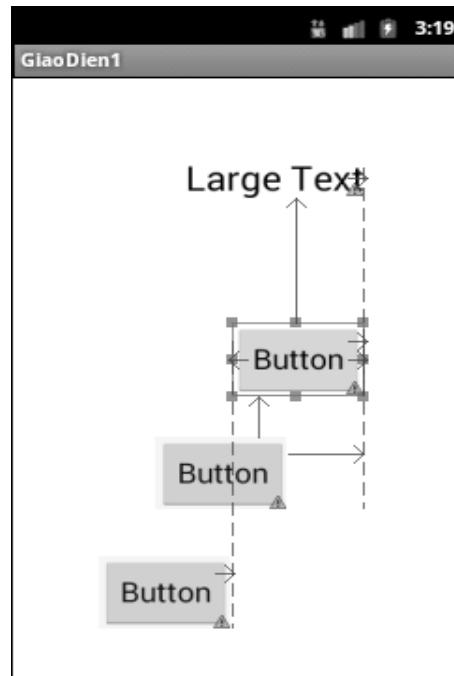
- Ví dụ khai báo:

```
<TableLayout  
    xmlns:android="http://schemas.android.com/apk/res/android"  
        android:layout_width="match_parent"  
        android:layout_height="match_parent" >  
    <Tablerow>  
        <Button/>  
    </Tablerow>  
        <Tablerow>  
            <Button/>  
        </Tablerow>  
    </TableLayout>
```

- Có thể kéo dãn độ rộng một cột với thuộc tính: android: layout_span.

1.1.4. Relative Layout

- Các View khi được đặt lên Layout sẽ có vị trí phụ thuộc vào View đã đặt vào trước nó, do đó khi thay đổi vị trí của một View sẽ làm thay đổi vị trí của các View còn lại.
- Đối với các giao diện phức tạp hơn thì việc dùng Relative Layout sẽ dễ dàng hơn. Các View khi được đặt lên Layout sẽ có vị trí phụ thuộc vào vị trí các View đã đặt vào trước đó và đối tượng đang chứa nó.
- Ví dụ ở đây ta có thể xây dựng Layout cho TextView nằm phía trên ba Button theo vị trí tùy ý.



Hình 3.4. RelativeLayout.

Chúng ta có thể kết hợp các loại layout lại với nhau để thiết kế giao diện. Tuy nhiên, việc tổ chức nhiều layout lồng nhau trên một giao diện sẽ tạo ra độ phức tạp và có thể làm giảm tốc độ xử lý. Vì vậy, khi tạo giao diện, chúng ta nên chú ý và tránh sử dụng nhiều layout chồng nhau.

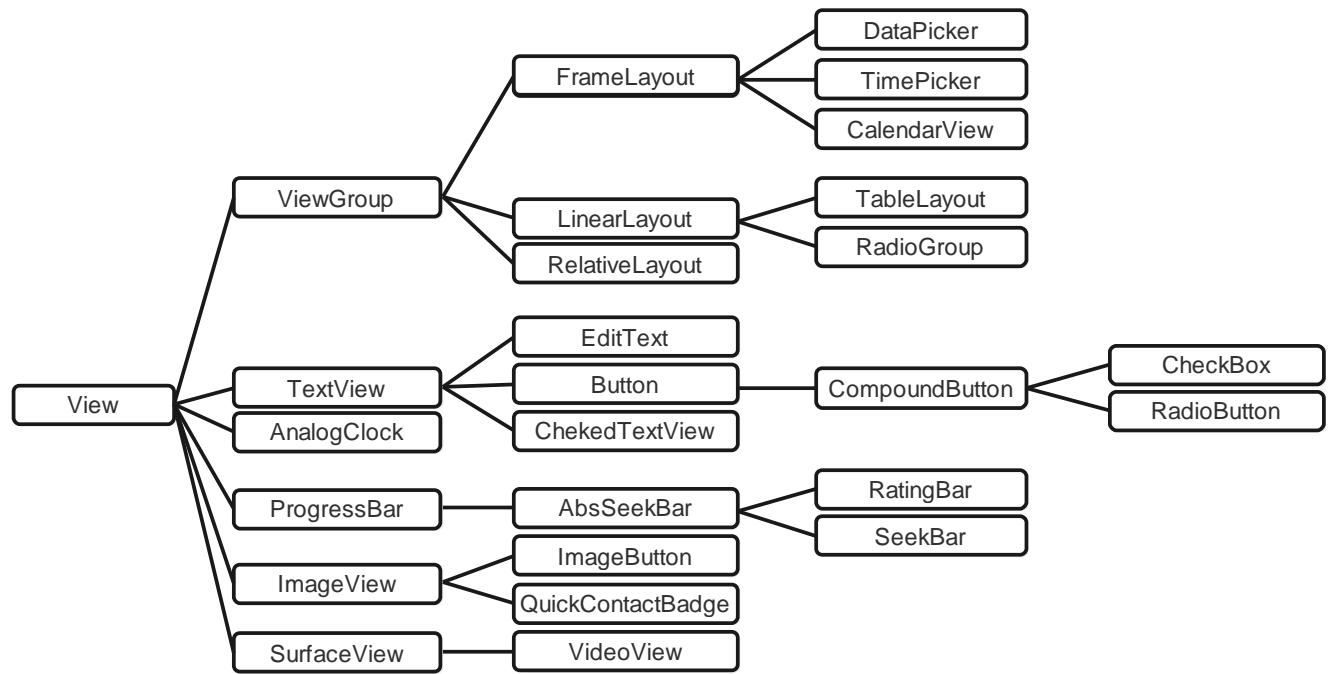
2. VIEW VÀ VIEWGROUP

2.1. Tổng quan

Trong Android, để tạo giao diện, chúng ta phải sử dụng hai lớp cốt lõi là: View và ViewGroup. Chúng rất quan trọng và được sử dụng thường xuyên trong quá trình phát triển ứng dụng Android.

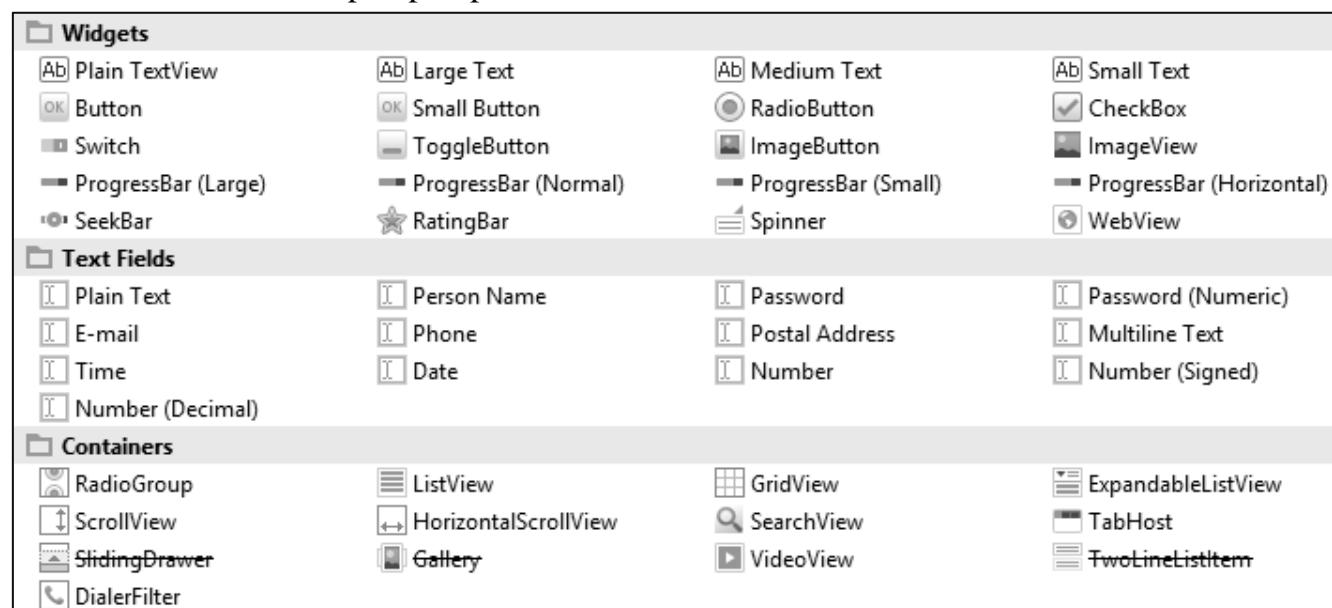
View và ViewGroup là hai lớp Java kế thừa từ lớp Java Object.

View là một lớp cha trên cùng. ViewGroup, TextView, AnalogClock, ImageView,... là các lớp con kế thừa từ lớp cha View. Và các lớp: Frame Layout, Linear layout và Relative Layout lại kế thừa từ ViewGroup. Tương tự, TextView cũng có 3 lớp con: EditText, Button và CheckedTextView.



Hình 3.5. Các đối tượng View trong thư viện Android.

- View được sử dụng để tạo ra các điều khiển trên màn hình cho phép nhận các tương tác từ người dùng cũng như hiển thị các thông tin cần thiết.
- View bao gồm hai dạng:
 - o View: các điều khiển đơn lẻ .
 - o ViewGroup: tập hợp nhiều điều khiển đơn lẻ.



Hình 3.6. Một số View và ViewGroup.

- Các đối tượng View được thể hiện trên màn hình giao diện như một hình chữ nhật tùy thuộc vị trí, kích thước, màu sắc và nhện vào cũng như xử lý các tương tác có liên quan.

- Một số thể hiện của lớp View: TextView, ImageView, SurfaceView...
- ViewGroup cũng là một thể hiện của View.
- Có thể xây dựng đối tượng View theo 2 cách:
 - o Kéo thả và tùy chỉnh thuộc tính trong XML.
 - o Thiết lập thông số và truy xuất trực tiếp trong Java Code.
- Thao tác với View
 - o Các đối tượng View được xây dựng và thiết lập với bốn thao tác chính:
 - Hiển thị nội dung thông qua phương thức set<TT>(TS).
 - Ví dụ: TextView hiển thị văn bản, ImageView hiển thị hình ảnh...
 - Yêu cầu tương tác
 - Ví dụ: sử dụng requestFocus để yêu cầu tương tác với điều khiển.
 - Thiết lập chế độ hiển thị thông qua phương thức setVisibility (hoặc thuộc tính visibility: trong XML)
 - VISIBLE: Trạng thái hiển thị
 - INVISIBLE: Trạng thái ẩn (vẫn giữ lại vị trí trên màn hình)
 - GONE: Trạng thái mất đi (mất vị trí trên màn hình)
 - Xây dựng phương thức “lắng nghe”
Một số sự kiện lắng trên đối tượng View:
 - OnClickListener
 - OnTouchListener
 - OnLongClickListener
 - OnDragListener
 - OnKeyListener
- Ví dụ: bắt lại các sự kiện nhấn xảy ra trên điều khiển:

```
view.setOnClickListener(new OnClickListener() {
    @Override
    public void onClick(View v) {
        Log.i("View", "onClick");
    }
});
```

2.2. Các thuộc tính của View

2.2.1. Id:

- Khai báo kiểu số nguyên int, đánh dấu vùng nhớ của đối tượng View.
- Id có thể giống nhau cho các điều khiển khác nhau trong cùng một tập tin giao diện.
- Phương thức thiết lập
 - setId()
- Phương thức truy xuất
 - getId()

- Thuộc tính Id được đi kèm với đối tượng View khi khai báo trong XML cho phép truy xuất trong Java Code khi cần. Ví dụ:

Khai báo id trong XML

```
<Button  
    android:id="@+id(btnAdd"  
    android:layout_width="200dp"  
    android:layout_height="wrap_content"  
    android:layout_below="@+id edtClass"  
    android:layout_centerHorizontal="true"  
    android:layout_marginTop="42dp"  
    android:background="#4696BE"  
    android:textColor="#FFFFFF" />
```

Truy xuất trong JavaCode

```
Button myBtn = (Button) findViewById(R.id.btnAdd);
```

2.2.2. Thuộc tính vị trí: cho biết toạ độ hiển thị cho View trên giao diện.

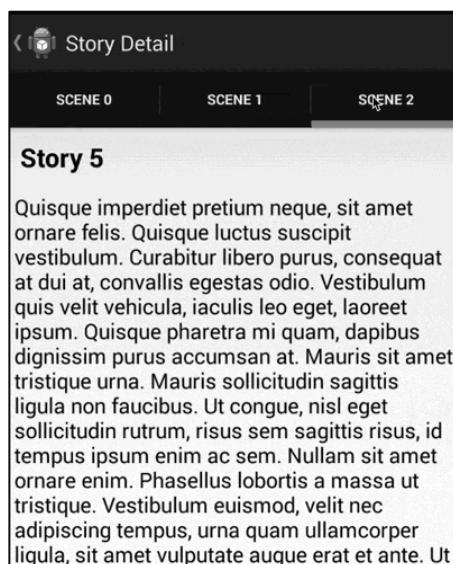
- Phương thức thiết lập :
 - o layout
 - o setLeft
 - o setTop
 - o setRight
 - o setBottom
- Phương thức truy xuất:
 - o getLeft
 - o getTop
 - o getRight
 - o getBottom
- Vị trí của View tuỳ thuộc vào thuộc tính của đối tượng Layout.
 - o Kích thước: bao gồm chiều ngang và chiều cao của một đối tượng View. Kích thước của đối tượng View có thể thiết lập qua các thông số:
 - WRAP_CONTENT
 - MATCH_PARENT (API 8 trở lên)
 - FILL_PARENT
 - Các thông số trên là một con số bất kỳ (tính theo dp/px/dip).
- Phương thức thiết lập:
 - o Thiết lập thông qua đối tượng LayoutParams
 - Thuộc tính thiết lập trong XML:

- layout_width
- layout_height
- Phương thức truy xuất:
 - getWidth
 - getHeight
 - getMeasuredWidth
 - getMeasureHeight
- Canh lè nội dung trong JavaCode:
 - Phương thức thiết lập:
 - setPadding
 - Phương thức truy xuất:
 - getPaddingTop
 - getPaddingLeft
 - getPaddingRight
 - getPaddingBottom

3. CÁC ĐIỀU KHIỂN CƠ BẢN

3.1. TextView:

- TextView là đối tượng cho phép hiển thị các nội dung văn bản ở 4 dạng:
 - **Normal**: : dạng văn bản kích thước font chữ mặc định.
 - **SmallText**: dạng văn bản kích thước font chữ nhỏ.
 - **MediumText**: dạng văn bản kích thước font chữ vừa.
 - **LargeText**: dạng văn bản kích thước font chữ to.



Hình 3.7. Ví dụ về TextView.

- Thiết lập nội dung hiển thị:

Trong Java code:

```
textView.setText("Đối tượng TextView");
```

Trong XML:

```
android:text="Đối tượng TextView"
```

- TextView được hỗ trợ các phương thức để thực hiện các siêu liên kết (Linkify). Cơ chế tự động thiết lập hành động cho các siêu liên kết, bao gồm:
 - o Web
 - o Email
 - o Phone
 - o Map
- Phương thức thiết lập:

Trong Java code:

```
textView.setAutoLinkMask(Linkify.PHONE_NUMBERS);
```

Trong XML:

```
android:autoLink="phone"
```

- TextView cho phép hiển thị hình ảnh theo văn bản ở hai dạng:
 - o Theo bộ cục văn bản: Left, Top, Right, Bottom.
 - o Theo đoạn văn bản: Start, End.
- Phương thức thiết lập:

Trong Java code:

```
textView.setCompoundDrawables(Left, Top, Right, Bottom);
```

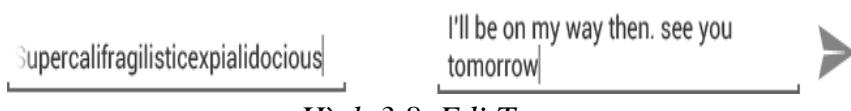
Trong XML:

```
android:drawableLeft="@drawable/ic_launcher"
```

- Một số phương thức quan trọng khác:
 - o setTextColor – android:textColor: thiết lập màu chữ.
 - o setTextSize – android:textSize: thiết lập kích cỡ chữ.
 - o setTypeFace – android:typeface: thiết lập các tùy chọn khác về font hay áp dụng một định dạng font ngoài cho TextView.

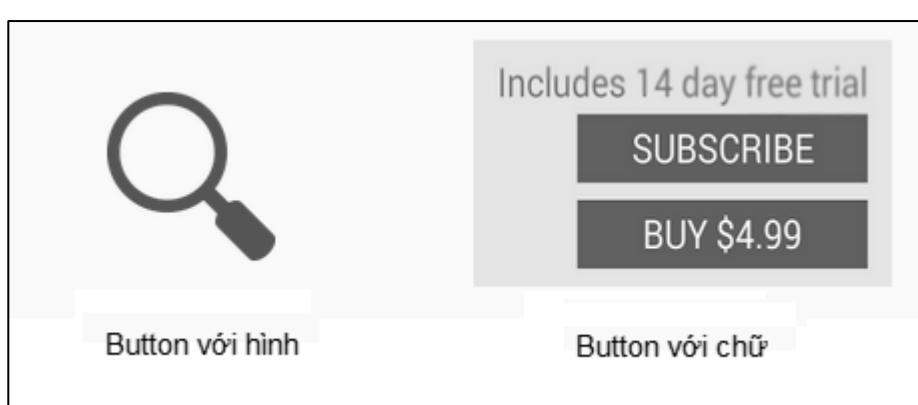
3.2. EditText:

- EditText là lớp kế thừa từ TextView, được dùng thay đổi nội dung text, chứa tất cả thuộc tính của TextView.



3.3. Button:

- Đối tượng Button được xây dựng từ TextView, cho phép thể hiện các nội dung văn bản, hình ảnh – nhận và phản hồi tương tác nhấn từ người dùng.



Hình 3.9. Button.

- Các dạng Button:
 - o Button
 - o CompoundButton:
 - CheckBox
 - RadioButton
 - ToggleButton
 - Switch
- Bắt sự kiện khi người dùng ấn vào Buton: có 3 cách chính như sau:
 - o Cách 1:

```
Button btnHello = (Button) findViewById(R.id.btnHello);
btnHello.setOnClickListener(new OnClickListener() {

    @Override
    public void onClick(View v) {
        //do something
    }
});
```

- o Cách 2: Cho Activity **implement** OnClickListener của android.view.View

```
import android.view.View.OnClickListener;
```

```
public class MainActivity extends ActionBarActivity implements  
OnClickListener {  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
        Button btnHello = (Button) findViewById(R.id.btnHello);  
        btnHello.setOnClickListener(this);  
    }  
    @Override  
    public void onClick(View v) {  
        switch (v.getId()) {  
            case R.id.btnHello:  
                //do something  
                break;  
            default:  
                break;  
        }  
    }  
}
```

- Cách 3: Thêm thuộc tính **onClick** vào thẻ Button:

```
<Button  
    android:id="@+id	btnHello"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:onClick="doHello"  
    android:text="Hello" />
```

- Viết phương thức public void doHello() trong MainActivity

```
@Override  
protected void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.activity_main);  
}  
public void doHello(View v){  
    switch (v.getId()) {  
        case R.id.btnHello:  
            //do something  
            break;  
  
        default:  
            break;  
    }  
}
```

3.4. Checkbox:

- CheckBox là đối tượng nút bấm hai trạng thái “được chọn” và “bỏ chọn”.
- Phương thức lắng nghe sự kiện thay đổi trạng thái:

```
checkBox.setOnCheckedChangeListener(new OnCheckedChangeListener() {  
    @Override  
    public void onCheckedChange(CompoundButton v, boolean isChecked)  
    {  
        Log.i("CompoundButton", "onChecked");  
    }  
});
```

3.5. RadioButton: đối tượng nút bấm hai trạng thái “được chọn” và “bỏ chọn”, không thể “bỏ chọn” khi đã “được chọn”.

- Thường xử lý trên nhóm nút nhiều trạng thái.
- Phương thức lắng nghe sự kiện thay đổi trạng thái trên nhóm nút:

```
radioGroup.setOnCheckedChangeListener(new OnCheckedChangeListener() {  
    @Override  
    public void onCheckedChange(RadioGroup group, int checkedId)  
    {  
        Log.i("CompoundButton", "Checked at id:" + checkedId);  
    }  
});
```

3.6. ToggleButton: đối tượng nút bấm hai trạng thái “bật” và “tắt”, thể hiện trạng thái trên đối tượng.

- Thuộc tính quan trọng:
 - o textOn: trạng thái nút đang bật
 - o textOff: trạng thái nút đang tắt
- Phương thức lắng nghe sự kiện thay đổi trạng thái trên nhóm nút:

```
toggleButton.setOnCheckedChangeListener(new OnCheckedChangeListener() {  
    @Override  
    public void onCheckedChange(CompoundButton v, boolean isChecked)  
    {  
        Log.i("CompoundButton", "onChecked");  
    }  
});
```

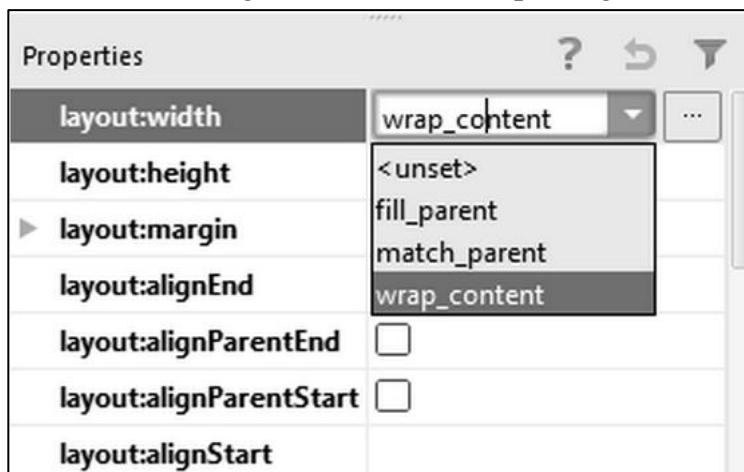
3.7. Switch: đối tượng nút bấm hai trạng thái “bật” và “tắt”, có thể thao tác bằng cách trượt ngón tay trên đối tượng.

- Thuộc tính quan trọng:
 - o textOn: trạng thái nút đang bật
 - o textOff: trạng thái nút đang tắt

- Phương thức lắng nghe sự kiện thay đổi trạng thái trên nhóm nút:

```
switchButton.setOnCheckedChangeListener(new OnCheckedChangeListener() {
    @Override
    public void onCheckedChange(CompoundButton v, boolean isChecked) {
        Log.i("CompoundButton", "onChecked");
    }
});
```

Ngoài cách thiết lập các thuộc tính cho layout và các điều khiển trong tập tin xml như các ví dụ trên, chúng ta có thể điền các giá trị muốn thiết lập vào giao diện được hỗ trợ sẵn.

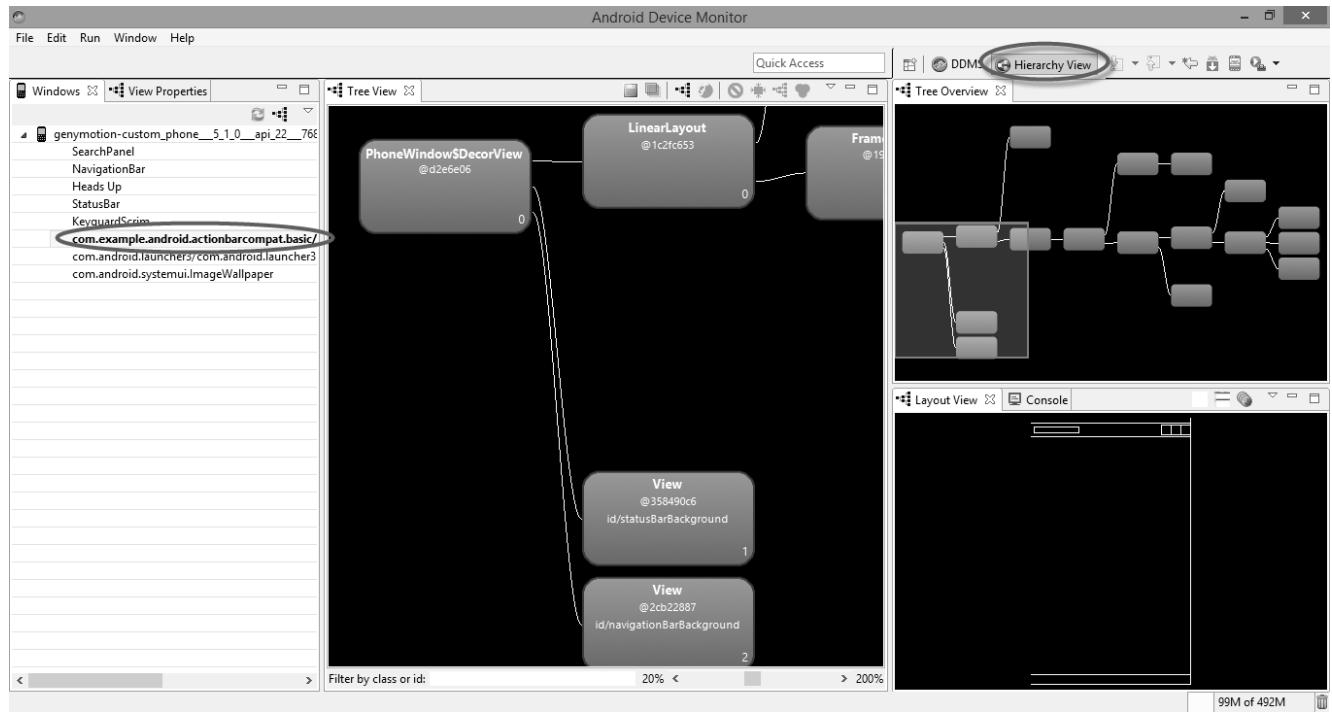


Hình 3.10. Minh họa thiết lập giá trị layout width trên giao diện tương tác.

4. KIỂM TRA LỖI VÀ TỐI ƯU LẠI GIAO DIỆN

Để kiểm tra lỗi và tối ưu lại giao diện, Android Studio có hỗ trợ công cụ Hierarchy Viewer giúp chúng ta làm việc này.

- Để sử dụng Hierarchy Viewer, chúng ta chọn biểu tượng Android Device Monitor trên thanh công cụ hoặc chọn Tools → Android → Android Device Monitor. Sau đó, xuất hiện màn hình Android Device Monitor, chúng ta tìm đến và chọn biểu tượng Open Perspectives → chọn Hierarchy Viewer → chọn OK.
- Chúng ta có thể sử dụng Hierarchy Viewer để xem cấu trúc giao diện như sau:
 - o Kết nối thiết bị thật hoặc khởi động thiết bị ảo.
 - o Mở Hierarchy Viewer.
 - o Khi ứng dụng vừa chạy ta sẽ thấy danh sách các thiết bị đang kết nối. Lựa chọn thiết bị muốn kiểm tra và nhấn đôi vào tên gói của ứng dụng muốn xem cấu trúc giao diện để thấy cấu trúc cây của giao diện ứng dụng đang chạy trên thiết bị.



Hình 3.11. Minh họa sử dụng công cụ Hierarchy Viewer

Ngoài ra, ta còn có thể khảo sát được sự phóng to, thu nhỏ của màn hình (sử dụng công cụ Pixel Perfect). Công cụ này có ích trong việc phát triển giao diện ứng dụng đòi hỏi độ chính xác cao về hình ảnh hay tinh chỉnh lại hình ảnh, màu sắc, độ trong suốt cho hợp lý hơn.

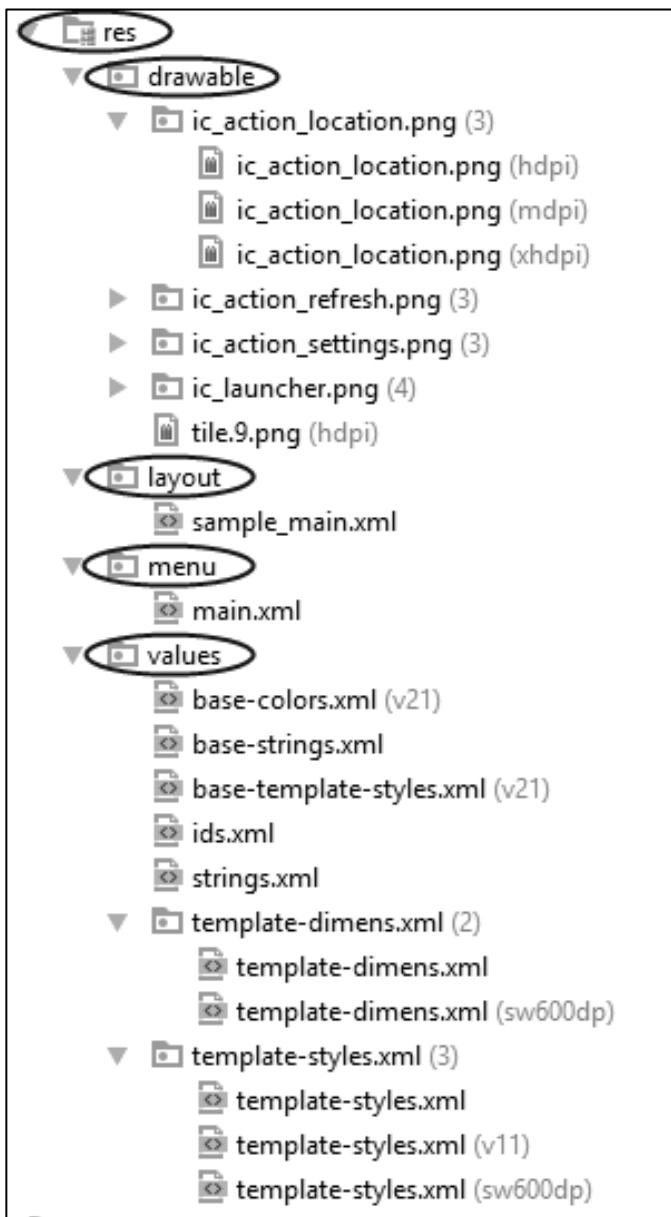
Bài 4

TÀI NGUYÊN ÚNG DỤNG TRONG ANDROID

1. TỔNG QUAN

1.1. Tài nguyên

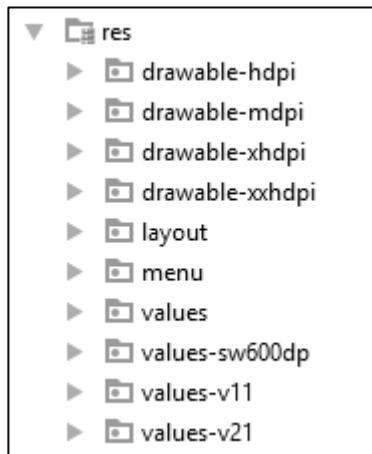
- Tài nguyên là một dạng dữ liệu được xây dựng nhằm đáp ứng các yêu cầu về hiển thị bao gồm hình ảnh, âm thanh, văn bản, các bộ cục...tương thích cho từng thiết bị riêng biệt. Cho phép khai báo một lần và sử dụng trong phạm vi toàn ứng dụng, dễ dàng thay đổi theo ngữ cảnh.
- Android cung cấp cho chúng ta khá nhiều cách thức để có thể sử dụng đa dạng tài nguyên như hình ảnh, các chuỗi hằng, chuyển hoạt, bộ cục giao diện...mà không cần phụ thuộc quá nhiều vào các dòng mã lệnh. Điều này giúp chúng ta đơn giản hóa lưu trữ, quản lý và cập nhật. Trong thiết kế cũng giúp chúng ta linh động hơn trong việc điều chỉnh cho nhiều thiết bị phân cứng khác nhau.
- Hệ thống tự động tạo thư mục /res dùng để chứa các thư mục của từng loại tài nguyên.
- Các file tài nguyên sẽ được chứa trong các thư mục tùy thuộc vào định dạng của tài nguyên đó.
- Chúng ta cần tạo ra các thư mục tài nguyên theo đúng định dạng của một ứng dụng Android. Tài nguyên ứng dụng được định nghĩa trong thư mục res của dự án, bao gồm các dạng tài nguyên sau:
 - **Animator:** Các dạng tài nguyên chuyển hoạt. (property animation)
 - **Anim:** Các dạng tài nguyên chuyển hoạt. (property animation)
 - **Color:** Các dạng tài nguyên màu sắc.
 - **Drawable:** Các dạng tài nguyên hình ảnh.
 - **Layout:** Các dạng tài nguyên giao diện.
 - **Menu:** Các dạng tài nguyên chỉ mục.
 - **Raw:** Các dạng tài nguyên không biên dịch.
 - **Values:** Các dạng tài nguyên cơ bản. Ví dụ: String, Integer, Boolean...
 - **XML:** Các dạng tài nguyên xml khác.



Hình 4.1. Thư mục chứa tài nguyên ứng dụng Android.

1.2. Định nghĩa tài nguyên

- Có quá nhiều thiết bị có cấu hình khác nhau về kích thước màn hình, độ phân giải, phím vật lý...
- Mỗi thiết bị có thể hoạt động ở nhiều chế độ khác nhau: nằm ngang, nằm đứng, thay đổi ngôn ngữ...
- **Tùy chọn định**: dùng để tạo ra các tài nguyên khác nhau cho nhiều thiết bị có cấu hình khác nhau hoạt động ở các chế độ khác nhau. Khi vừa tạo mới một Project, trong thư mục res ta sẽ thấy có đến bốn thư mục drawable và ba thư mục values chỉ khác nhau ở phần ngăn cách bởi “-“. Phần hậu tố này chúng ta gọi là tùy chọn định, dùng để phân biệt tài nguyên dành cho các cấu hình khác nhau. Khi ứng dụng được cài lên thiết bị hệ thống sẽ tự tìm các tài nguyên phù hợp với phần cứng thiết bị đó.



Hình 4.2. Ví dụ về từ hạn định.

- Tất cả thư mục tài nguyên đều có thể sử dụng từ hạn định. Bao gồm các từ hạn định sau:
 - o MCC/MNC (Mobile Country Code – Mobile Network Code) - các tài nguyên trong thư mục có từ hạn định này chỉ được dùng cho quốc gia hay nhà cung cấp dịch vụ mạng nào đó. Để sử dụng chỉ cần khai báo mmc theo sau là chữ số của quốc gia, hoặc chữ số nhà mạng nếu là mnc. Ví dụ mmc452-mnc04 là Việt Nam và mạng Viettel, (có thể tham khảo thêm tại trang http://en.wikipedia.org/wiki/Mobile_network_code).
 - o Language/Region – các từ hạn định phân biệt tài nguyên cho các thiết bị các thiết lập ngôn ngữ khác nhau.Các từ hạn định này bao gồm hai kí tự thường tiêu chuẩn ISO 639-1 và ISO 3166-1-alpha-2. Ví dụ vn; en; hoặc en-rUS cho tiếng Anh ở Mĩ, en-rGB tiếng anh ở Anh.
 - o Smallest Screen Width – từ hạn định cho chiều ngắn nhất của màn hình thiết bị dùng trong việc thiết kế nhiều giao diện. Ví dụ sw600dp, sw320dp hoặc sw720dp.
 - o Available Screen Width/Height – từ hạn định cho chiều ngang và chiều cao tối thiểu để chứa các tài nguyên được sử dụng khi quay màn hình. Ví dụ w600dp, h720dp...
 - o Screen size – được dùng cho để chỉ định các kích thước màn hình khác nhau bao gồm small (từ HVGA trở xuống), medium (từ HVGA đến VGA), large (VGA trở lên).
 - o Screen Aspect Ratio – bao gồm 2 lựa chọn long và notlong để chỉ định các màn rộng. Ví dụ WVGA là long và QVGA là notlong.
 - o Screen Orientation – từ hạn định cho biết hướng cầm của thiết bị bao gồm port (portrait) khi thiết bị đứng và land (landscape) khi thiết bị nằm ngang.
 - o Dock Mode – bao gồm car chế độ thiết bị dùng cho tình trạng lái xe và desk ở chế độ thường.

- Screen Pixel Density – mật độ điểm ảnh trên một inch (dpi – dot per inch). Dùng để phân biệt các dạng tài nguyên cho các màn hình có độ phân giải khác nhau bao gồm ldpi (low - 120dpi), mdpi (medium - 160dpi), hdpi (high - 240dpi) và xhdpi (extra high – 320dpi).
 - Touchscreen Type – từ hạn định dành cho các tài nguyên liên quan đến điều khiển cảm ứng gồm notouch (không hỗ trợ cảm ứng), stylus (dùng bút vẽ) và finger (cảm ứng điện dung).
 - Keyboard Availability - từ hạn định dành cho các tài nguyên liên quan đến bàn phím nhập liệu gồm keyexposed (bàn phím cứng), keyhidden (bàn phím trượt) và keysoft (bàn phím ảo).
 - Keyboard Input Type - từ hạn định dành cho các tài nguyên liên quan đến kiểu nhập liệu bao gồm nokeys (không bàn phím), qwertyp (bàn phím máy tính cá nhân) và 12key (bàn phím số).
 - Navigation Key Availability – từ hạn định dành cho các tài nguyên liên quan đến phím điều hướng bao gồm navexposed (phím cứng) và navhidden (phím trượt).
 - UI Navigation Type - từ hạn định dành cho các tài nguyên liên quan đến kiểu điều hướng bao gồm nonav (không hỗ trợ điều hướng), dpad (bàn rê), trackball (bi lăn) và wheel (bánh xe).
 - Platform Version – cho phép chỉ định các tài nguyên chạy trên các phiên bản có API khác nhau. Ví dụ: v7 chỉ định các phiên bản có API 7 (Android 2.1).
- Chúng ta có thể sử dụng cùng lúc nhiều từ hạn định cho các thư mục tài nguyên khác nhau và ngăn cách bởi dấu “-“. Tuy nhiên cần lưu ý là các từ hạn định phải theo độ ưu tiên (trong tài liệu này độ ưu tiên từ trên xuống dưới) và không được phép sử dụng hai từ hạn định cùng loại ở cùng một thư mục.
 - Ví dụ: các thư mục tài nguyên sau là không được phép:
 - layout-rUS-en (không theo thứ tự)
 - layout-vn-en (hai từ hạn định cùng loại)
- **Quy tắc đặt tên thư mục có từ hạn định:**
 - Có thể sử dụng nhiều từ hạn định trong thư mục lựa chọn, ngăn cách bởi dấu “-“.
 - Ví dụ: drawable-en-rUS-land.
 - Phải tuân theo quyền ưu tiên từ trên xuống dưới.
 - Ví dụ:
 - drawable-hdpi-port (Sai)
 - drawable-port-hdpi (Đúng)
 - Thư mục tài nguyên được lựa chọn không được lồng vào nhau.
 - Ví dụ: Không được phép /res/drawable/drawable-en.
 - Chỉ một giá trị cho loại từ hạn định được hỗ trợ cho một thư mục.

- Ví dụ: Không được phép drawable-hdpi-mdpi.

- Không phân biệt chữ hoa, chữ thường.

1.3. Tính tương thích

- Để có thể tối ưu hóa tính tương thích thiết bị tài nguyên được chia làm hai dạng:
 - Tài nguyên mặc định: không quan tâm đến cấu hình của thiết bị hoặc không có tài nguyên để lựa chọn.
 - Tài nguyên đặc trưng: được sử dụng trên thiết bị riêng biệt thông qua các từ hạn định và đường dẫn.
 - Trong quá trình chạy ứng dụng, hệ thống sẽ dò ra cấu hình hiện thời của thiết bị để lựa chọn tài nguyên phù hợp cho ứng dụng với điều kiện trong ứng dụng có khai báo nguồn tài nguyên cho thiết bị đó.



Hình 4.3. Lựa chọn tài nguyên thích hợp.

1.4. Truy xuất tài nguyên

- Tất cả tài nguyên ứng dụng được truy xuất thông qua lớp R.

1.4.1. Lớp R:

- Lớp tĩnh.
- Chứa trong thư mục gen, tự động tạo các định danh cho tài nguyên (ID) thông qua AAPT (Android Application Project Tool).
- Chứa các lớp tài nguyên, mỗi dạng tài nguyên là một lớp tĩnh.
 - Ví dụ:
 - Truy xuất tài nguyên hình ảnh:
 - Java code: R.drawable.ic_launcher
 - XML: @drawable/ic_launcher

- Cú pháp dùng chung khi truy xuất:

- Java Code:

```
[<package_name.>]R.<resource_type>.<resource_name>
```

- XML:

```
@[<package_name.>:]<resource_type>/<resource_name>
```

- Trong đó:

- Package_name: tên gói ứng dụng
- Resource_type: dạng tài nguyên
- Resource_name:
 - Tên tài nguyên cần truy xuất không bao gồm phần mở rộng tập tin
 - Thuộc tính android:name dành cho các tài nguyên cơ bản (string, color...).

1.4.2. Tài nguyên Alias

- Tài nguyên Alias cho phép tạo ra tài nguyên từ tài nguyên có sẵn, phục vụ cho nhiều cấu hình thiết bị nhưng không phải là tài nguyên mặc định.
- Ví dụ:

Vấn đề: tạo biểu tượng ứng dụng khác nhau cho các ngôn ngữ khác nhau, đối với tiếng Anh và tiếng Việt thì cùng biểu tượng.

- Giải quyết vấn đề (không dùng Alias):
 - Tạo thư mục tài nguyên cho từng ngôn ngữ.
 - Chép hình ảnh khác nhau cho từng thư mục, hai thư mục có từ hạn định en và vi có hình ảnh giống nhau.
- Giải quyết vấn đề (dùng Alias):
 - Tạo thư mục tài nguyên cho từng ngôn ngữ.
 - Chép hình ảnh khác nhau cho từng thư mục.
 - Thư mục có từ hạn định en tạo tài nguyên Alias từ thư mục vi.

2. CÁC TÀI NGUYÊN ỨNG DỤNG CƠ BẢN

- Các tài nguyên cơ bản được lưu trữ trong thư mục res/values.
- Định danh tài nguyên được khởi tạo thông qua thuộc tính name, không phải tên tập tin.
- Có thể lưu trữ nhiều tài nguyên vào trong một tập tin.
- Một số tên tập tin đề xuất trong values: string.xml, arrays.xml, colors.xml, dimens.xml, ...
- Tất cả các tập tin xml trong values, được mở đầu và kết thúc bằng cặp thẻ <resource></resource>.

2.1. String

- Cung cấp tài nguyên dạng văn bản cho ứng dụng, cho phép thực hiện các thao tác định dạng và thiết kế khác nhau, bao gồm ba dạng:
 - String.
 - StringArray.
 - QuantityString (Plural).
- Khai báo:

```
<string name="string_name">Text_string</string>
```

- Trong đó:

- string_name: định danh dùng để truy xuất trong XML và Java Code.
- text_string: nội dung lưu trữ.
 - Ví dụ: tạo tập tin strings.xml

```
<resource>
    <string name="hello">Hello world!</string>
</resource>
```

- Truy xuất và sử dụng:
Khai báo TextView và gắn văn bản cho thuộc tính text.

- Truy xuất trong XML:

```
<TextView
...
    android:text="@string/hello" />
```

- Truy xuất trong Java Code:
 - Truy xuất trực tiếp:

```
textView.setText(R.string.hello);
```

- Dùng phương thức getString:

```
textView.setText(getString(R.string.hello));
```

2.2. Bool

- Cú pháp

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <boolname="bool_name">[true | false]</bool>
</resources>
```

- Ví dụ: res/values-small/bools.xml

```
<resources>
    <bool name="screen_small">true</bool>
    <bool name="adjust_view_bounds">true</bool>
</resources>
```

2.3. Color

- Thư mục tài nguyên:

```
/res/values/filename.xml
```

- Cú pháp khai báo:

```
<resources>
    <color name="color_name">Hexa_color</color>
    ...
</resources>
```

- Tham chiếu tài nguyên trong xml

```
[package]:@color/color_name
```

- Tham chiếu tài nguyên trong Java code

```
Resources res = getResources();
int color = res.getColor(R.color.color_name);
```

2.4. Id

- Thư mục tài nguyên:

```
/res/values/filename.xml
```

- Cú pháp khai báo:

```
<resources>
    <item type="id" name="id_name"/>
    ...
</resources>
```

- Tham chiếu tài nguyên trong xml:

```
[package]:@id/id_name
```

- Tham chiếu tài nguyên trong Java code

```
TextView tv = (TextView) findViewById(R.id.id_name);
```

2.5. Dimen

- Khai báo tài nguyên sử dụng cho các đại lượng kích thước trong ứng dụng.
- Có thể sử dụng các đại lượng kích thước sau: dp – dip, sp, pt, px, mm, in.
- Khai báo:

```
<dimen name="dimen_name">size</dimen>
```

- Trong đó:

- o dimen_name: định danh dùng để truy xuất trong XML và Java Code.

- o size: đại lượng đi kèm định dạng
- Ví dụ: tập tin dimens.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resource>
    <dimen name="text_size">20sp</dimen>
</resource>
```

- Ví dụ truy xuất và sử dụng: Khai báo Button và thiết lập kích thước văn bản thông qua thuộc tính textSize.

- o Truy xuất trong XML:

```
<TextView
    ...
    android:textSize="@dimen/text_size" />
```

- o Truy xuất trong Java Code: Dùng phương thức getDimension:

```
Resources res = getResources();
float textSize = res.getDimension(R.dimen.text_size);
textView.setTextSize(textSize);
```

2.6. Integer

- Thư mục tài nguyên:

```
/res/values/filename.xml
```

- Cú pháp khai báo:

```
<resources>
    <integer name="name">value</integer>
    ...
</resources>
```

- Tham chiếu tài nguyên trong Java code

```
Resource res = getResources();
int value = res.getIntegers(R.integer.name);
```

2.7. Integer Array

- Thư mục tài nguyên:

```
/res/values/filename.xml
```

- Cú pháp khai báo:

```
<resources>
    <integer-array name="integer_name">
        <item>value1</item>
        ...
    </integer>
</resources>
```

- Tham chiếu tài nguyên trong Java code

```
Resource res = getResources();
int[] value = res.getIntArray(R.array.integer_name);
```

2.8. Typed Array

- Cú pháp

```
<resources>
    <array name="array_name">
        <item>value</item>
    </array>
</resources>
```

- Ví dụ: res/values/arrays.xml

```
<resources>
    <array name="icons">
        <item>@drawable/home</item>
        <item>@drawable/settings</item>
        <item>@drawable/logout</item> </array>
    <array name="colors">

        <item>#FFFF0000</item>
        <item>#FF00FF00</item>
        <item>#FF0000FF</item>
    </array>
</resources>
```

- Tham chiếu tài nguyên trong Java code

```
Resources res = getResources();
TypedArray icons = res.obtainTypedArray(R.array.icons);
Drawable drawable = icons.getDrawable(0);
TypedArray colors = res.obtainTypedArray(R.array.colors);
int color = colors.getColor(0,0);
```

3. CÁC TÀI NGUYÊN ÚNG DỤNG NÂNG CAO

3.1. Tài nguyên hình ảnh

3.1.1. Thư mục lưu trữ:

- Các tài nguyên hình ảnh được lưu trữ trong thư mục res/drawable.
- Có thể có nhiều thư mục drawable theo từ hạn định khác nhau.
- Ví dụ: drawable-hdpi, drawable-xhdpi...

3.1.2. Định dạng:

- Tài nguyên hình ảnh bao gồm cả định dạng *.xml và định dạng hình ảnh (.png, .gif, .jpg).

3.1.3. Truy xuất: bao gồm 2 cách thức:

- Java: R.drawable.<tên tài nguyên>.
- XML: @[pakage:]drawable/<tên tài nguyên>.
- Ví dụ: truy xuất tài nguyên hình ảnh
 - Trong Java Code:

```
Resources res = getResources();
Drawable drawable = res.getDrawable(R.drawable.ic_launcher);
```

- Trong XML:

```
<ImageView
    android:layout_width="50dp"
    android:layout_height="50dp"
    android:src="@drawable/ic_launcher" />
<ImageButton
    android:layout_width="50dp"
    android:layout_height="50dp"
    android:background="@drawable/ic_launcher" />
```

3.2. Các dạng tài nguyên hình ảnh

3.2.1. Bitmap

- Các thực thi của Bitmap bao gồm:
 - Sử dụng như tài nguyên thông qua R.drawable.filename
 - Tham chiếu biến dịch tài nguyên thông qua đối tượng BitmapDrawable.
- Các thuộc tính thường sử dụng của Bitmap trong XML:
 - Antialias (XML)
 - Dither
 - Filter
 - Gravity
 - Mipmap
 - Tilemode
 - Automirrored
- Ví dụ: xây dựng Bitmap trong XML: mipmap.xml

```
<bitmap xmlns:android="http://schemas.android.com/apk/res/android"
        android:mipMap="false"
```

```
    android:src="@drawable/caro"  
    android:tileMode="repeat" > </bitmap>
```

- Truy xuất trong Java code:

```
BitmapDrawable drawable = (BitmapDrawable)getResources().  
                           getDrawable(R.drawable.mipmap);
```

3.3. Shape

- Tài nguyên hình ảnh cho các đối tượng đa giác được vẽ bằng XML, bao gồm:
 - o Rectangle
 - o Oval
 - o Line
 - o Ring
- Tham chiếu biến dịch tài nguyên thông qua đối tượng GradientDrawable.
- Sử dụng các thuộc tính để cấu tạo đối tượng:
 - o Corners (Rectangle) - Integer
 - radius
 - topLeftRadius
 - topRightRadius
 - bottomLeftRadius
 - o bottomRadius Padding (Rectangle) – Integer
 - left
 - top
 - right
 - bottom
 - o Gradient
 - angle - integer
 - centerX - integer
 - centerY - integer
 - centerColor - integer
 - endColor - color
 - gradientRadius – integer
 - startColor – color
 - type – linear | radial | sweep
 - useLevel – true | false
 - o Size – integer
 - width – integer
 - height – integer

- Solid – integer
- color – color
- Stroke – integer
 - width - integer
 - color – color
 - dashWith – integer
 - dashGap – integer
- Một số thuộc tính chỉ sử dụng cho đối tượng Ring:
 - innerRadius
 - innerRadiusRatio
 - thickness
 - thicknessRatio
 - useLevel (false)

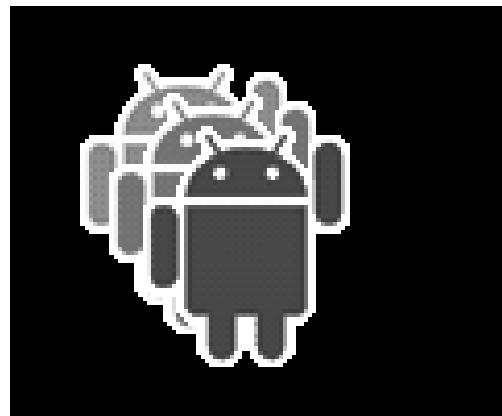
- Ví dụ:

```
<shape xmlns:android="http://schemas.android.com/apk/res/android
    android:shape="rectangle" >
    <gradient
        android:angle="90"
        android:startColor="@android:color/holo_blue_bright"
        android:type="linear" />
    <corners android:radius="20dp"/>

    <size
        android:height="80dp"
        android:width="80dp" />
</shape>
```

3.4. LayerList

- Tài nguyên hình ảnh cho phép quản lý mảng các đối tượng hình ảnh được vẽ chồng lên nhau, mỗi đối tượng hình ảnh được qui ước là một item.
- Mỗi item bao gồm:
 - drawable – resource
 - Id – resource id
 - top - integer
 - right - integer
 - bottom - integer
 - left - integer
- Tham chiếu biến dịch tài nguyên thông qua đối tượng LayerDrawable.



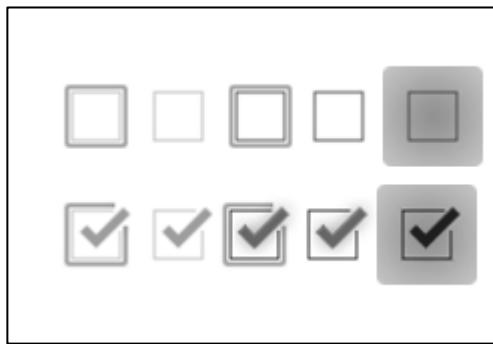
Hình 4.4. LayerList.

- Ví dụ:

```
<layer-list xmlns:android="http://schemas.android.com/apk/res/android" >
    <item android:drawable="@drawable/caro"
        android:left="10dp"
        android:top="10dp"> </item>
    <item android:drawable="@drawable/my_shape"
        android:left="30dp"
        android:top="30dp"></item>
    <item android:drawable="@drawable/chrysanthemum"
        android:left="50dp"
        android:top="50dp"></item>
</layer-list>
```

3.5. StateList

- Tài nguyên hình ảnh cho phép quản lý mảng các đối tượng hình ảnh được vẽ theo trạng thái của đối tượng thể hiện.
- Một item bao gồm: drawable – resource
- Tập các trạng thái có thể có:
 - Pressed
 - Focused
 - Hovered
 - Selected
 - Checkable
 - Enable
 - Activated
 - Window focused



Hình 4.5. Ví dụ về các trạng thái có thể có của một Checkbox.

- Tham chiếu biến dịch tài nguyên thông qua đối tượng StateListDrawable.
- Ví dụ:

```
<?xml version="1.0" encoding="utf-8"?>
<selector xmlns:android="http://schemas.android.com/apk/res/android" >
    <item android:drawable="@drawable/on"
        android:state_pressed="true"></item>
    <item android:drawable="@drawable/off" ></item>
</selector>
```

3.6. LevelList

- Tài nguyên hình ảnh cho phép quản lý mảng các đối tượng hình ảnh, mỗi đối tượng hình ảnh được qui ước là một item, hiển thị ảnh theo cấp độ tương ứng đã khai báo.
- Một item bao gồm:
 - o drawable – resource
 - o maxLevel
 - o minLevel
- Tham chiếu biến dịch tài nguyên thông qua đối tượng LevelListDrawable.
- Ví dụ:

```
<?xml version="1.0" encoding="utf-8"?>
<level-list
    xmlns:android="http://schemas.android.com/apk/res/android" >
    <item    android:drawable="@drawable/status_off" android:maxLevel="0" />
    <item    android:drawable="@drawable/status_on"   android:maxLevel="1" />
</level-list>
```

3.7. Transition

- Tài nguyên hình ảnh cho phép thực hiện chuyển đổi (hiệu ứng “biến bóng”) giữa hai đối tượng hình ảnh.
- Mỗi item bao gồm:
 - o drawable – resource

- Id – resource id
 - top - integer
 - right - integer
 - bottom - integer
 - left – integer
- Các phương thức xử lý chính:
 - startTransition
 - reverseTransition
 - resetTransition.
 - Ví dụ:

```
<?xml version="1.0" encoding="utf-8"?>
<transition
    xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:drawable="@drawable/on" />
    <item android:drawable="@drawable/off" />
</transition>
```

3.8. Inset

- Tài nguyên hình ảnh cho phép thực hiện lồng đối tượng hình ảnh theo một ví trí cho trước.



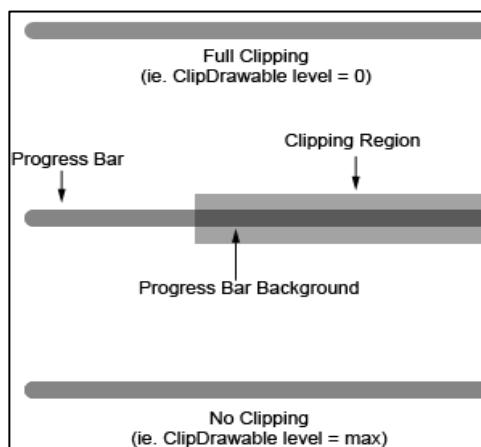
Hình 4.6. Inset Drawable.

- Các thuộc tính bao gồm:
 - drawable – resource
 - insetTop - integer
 - insetRight - integer

- insetBottom - integer
 - insetLeft – integer
- Tham chiếu biên dịch tài nguyên thông qua đối tượng InsetDrawable.

3.9. Clip

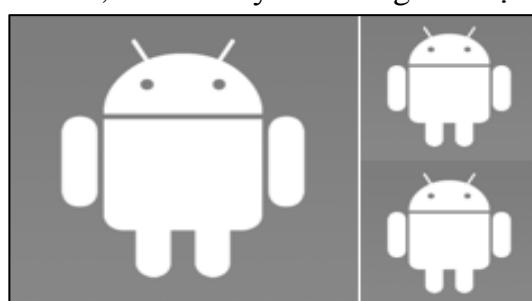
- Tài nguyên hình ảnh cho phép thực hiện cắt một đối tượng hình ảnh theo thông số vị trí cho trước, có thể thay đổi thông số cắt trong quá trình hoạt động.
- Tham chiếu biên dịch tài nguyên thông qua đối tượng ClipDrawable.
- Các thuộc tính bao gồm:
 - drawable – resource
 - clipOrientation – integer
 - Gravity
- Các phương thức xử lý chính:
 - setLevel (min:0 – max: 10.000)
 - getLevel



Hình 4.7. Ví dụ dùng Clip Drawable để tạo thanh ProgressBar.

3.10. Scale

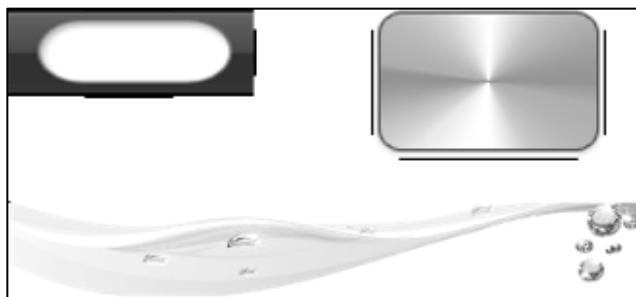
- Tài nguyên hình ảnh cho phép thực hiện phóng to hoặc thu nhỏ một đối tượng hình ảnh theo thông số tỉ lệ cho trước, có thể thay đổi thông số tỉ lệ trong quá trình hoạt động.



Hình 4.8. Scale Drawable.

- Các thuộc tính bao gồm:
 - drawable – resource
 - scaleGravity – integer

- scaleWidth - %
 - scaleHeight - %
- Tham chiếu biến dịch tài nguyên thông qua đối tượng ScaleDrawable.
- 3.11. Nine-Patch
- Tài nguyên hình ảnh cho phép thực hiện tạo đối tượng hình ảnh (PNG) có kích thước co giãn theo tỉ lệ đối tượng thể hiện.



Hình 4.9. Một số hình Nine-Patch.

- Các thuộc tính bao gồm:
 - src – resource
 - dither – integer
- Tham chiếu biến dịch tài nguyên thông qua đối tượng NinePatchDrawable.

Bài 5 INTENT

1. KHÁI NIỆM VỀ INTENT

1.1. Cơ chế hoạt động

- Intent được sử dụng để truyền tải thông điệp, yêu cầu một hành động xử lý từ thành phần được gọi.
- Intent được sử dụng trong ba trường hợp chính:
 - o Khởi động Activity thông qua phương thức startActivity.
 - o Khởi động Service thông qua phương thức startService.
 - o Chuyển thông điệp đến BroadcastReceiver thông qua phương thức sendBroadcast.
- Intent được chia làm hai dạng:
 - o **Explicit Intent:** (Intent tường minh) chỉ định rõ thành phần xử lý thông qua tên lớp, thường được dùng để gọi đến các thành phần trong cùng ứng dụng.
 - o **Implicit Intent:** (Intent không tường minh) không chỉ định rõ thành phần xử lý, thay vào đó bổ sung các thuộc tính như: mô tả hành động, dạng dữ liệu...

1.2. Xây dựng và truy xuất Intent

- Đối tượng Intent khởi động các thành phần trong ứng dụng đồng thời mang các thông tin về dữ liệu được xử lý, bao gồm các thành phần sau:
 - o **Component:** tên thành phần nhận và xử lý Intent
 - o **Action:** hành động yêu cầu thực thi
 - o **Data:** dữ liệu yêu cầu nhận và xử lý
 - o **Category:** mô tả lĩnh vực hoạt động
 - o **Extras:** bộ key/value cho phép gửi nhận thông tin
 - o **Flag:** biến cờ mô tả cách thức hoạt động
 - o **Explicit Intent:** chỉ cần sử dụng thuộc tính Component.
- Khai báo:

```
Intent intent = new Intent(this, <Component>);
```

- Ví dụ: khởi động Activity có tên SecondActivity từ MainActivity

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

- Implicit Intent: chỉ cần sử dụng thuộc tính Action.
- Khai báo:

```
Intent intent = new Intent(<Action>);
```

- Ví dụ: khởi động Activity có thể thực hiện ACTION_VIEW.

```
Intent intent = new Intent(Intent.ACTION_VIEW);
startActivity(intent);
```

- Action: một số Action thường dùng trong Intent.
 - o ACTION_VIEW
 - o ACTION_DIAL
 - o ACTION_CALL
 - o ACTION_EDIT
 - o ACTION_DELETE
 - o ACTION_SEND
 - o ACTION_SENDTO
- Data: một dạng đường dẫn URI, cho phép trả về bảng dữ liệu và truy xuất thông tin bao gồm:
 - o type
 - o scheme
 - o authority
 - o path
- Data có thể chỉ định thông qua ba phương thức:
 - o setData
 - o setType
 - o setDataAndType
- Ví dụ: thực hiện cuộc gọi thông qua dữ liệu số điện thoại

```
Intent callPhone = new Intent(Intent.ACTION_CALL);
callPhone.setData(Uri.parse("tel:01234-56789"));
startActivity(callPhone);
```

- Extras: bao gồm biến Bundle chứa các giá trị bổ sung cần thiết cho thành phần nhận xử lý Intent.
- Có hai cách gửi dữ liệu vào Intent:
 - o Trực tiếp: dùng phương thức putExtra(Key, Value) thiết lập trực tiếp vào Intent.
 - o Thông qua Bundle: tạo một đối tượng Bundle, dùng phương thức set<KDL>(Key, Value) vào đối tượng Bundle. Sau đó, ta dùng phương thức putExtras() gửi Bundle vào Intent.

- Ví dụ: gửi số nguyên x vào Intent
 - o Trực tiếp:

```
Intent intent = new Intent();
intent.putExtra("SoNguyenX", x);
```

- o Thông qua Bundle:

```
Intent intent = new Intent();
Bundle bundle = new Bundle();
bundle.putInt("SoNguyenX", x);
intent.putExtras(bundle);
```

- Truy xuất:
 - o Truy xuất dữ liệu trực tiếp Extras:
 - Dùng phương thức `get<KDL>Extra(Key, DefaultValue)` để truy xuất dữ liệu Intent.
 - o Thông qua Bundle:
 - Dùng phương thức `getExtras()` để truy xuất đối tượng Bundle trong Intent.
 - Dùng phương thức `get<KDL>(Key, DefaultValue)` để truy xuất dữ liệu trong Bundle.
 - o Ví dụ: truy xuất số nguyên được gửi trong Intent
 - Trực tiếp:

```
Intent intent = getIntent();
int soNguyenX = intent.getIntExtra("SoNguyenX", 0);
```

- Thông qua Bundle:

```
Intent intent = getIntent();
Bundle bundle = intent.getExtras();
int soNguyenX = bundle.getInt("SoNguyenX", 0);
```

- Gửi và Phản hồi Intent trong Activity
 - o Việc gửi và phản hồi Intent trong Activity được chia làm 3 bước
 - Bước 1: Gửi Intent thông qua phương thức `startActivityForResult()` bao gồm 2 tham số:
 - **Intent:** dữ liệu cần gửi để xử lý.
 - **requestCode:** mã yêu cầu xử lý từ phía gửi.
 - Bước 2: Nhận và xử lý Intent, sau đó xác nhận thông tin phản hồi thông qua phương thức `setResult()` trong thành phần ứng dụng phản hồi.

- Khởi tạo đối tượng Intent, thiết lập các thuộc tính cần thiết: action, category...
 - Gửi dữ liệu phản hồi trực tiếp vào Intent hoặc thông qua biến Bundle.
 - Gọi phương thức setResult với tham số truyền vào là Intent.
- **Bước 3:** Gọi phương thức **onActivityResult()** truy xuất ba tham số:
 - **requestCode:** mã yêu cầu giải quyết với intent tương ứng.
 - **resultCode:** mã kết quả nhận về từ phía phản hồi.
 - **Intent:** dữ liệu nhận về từ phía phản hồi.



Hình 5.1. Gửi và phản hồi Intent.

2. INTENT FILTER

2.1. Mô tả

- Thực hiện mô tả cấu trúc Intent, cho phép thực hiện chỉ nhận các Intent theo đúng cấu trúc đã mô tả.
- Có thể lọc Intent theo ba thuộc tính:
 - Action
 - Data (type, scheme, authority & path)
 - Category

2.2. Quy tắc thiết lập

- IntentFilter thực hiện lọc Intent theo thứ tự ưu tiên khi có nhiều thuộc tính được thiết lập và có những qui tắc nhất định:
 - Nếu không thiết lập Action, chỉ nhận các Intent không có Action.
 - Nếu thiết lập thuộc tính Action và không thiết lập thuộc tính Data, chỉ cho phép lọc các Intent không có Data.
- IntentFilter cho phép nhận các Intent có bất kỳ dữ liệu nào có liên quan đến thuộc tính Action.

2.3. Xây dựng IntentFilter

- Có thể khởi tạo đối tượng IntentFilter như sau:
- Trong java Code:
 - Các phương thức khởi tạo:

- IntentFilter()
 - IntentFilter(String Action)
 - IntentFilter(String Action, Uri data)
 - IntentFilter(IntentFilter o)
- Trong tập tin AndroidManifest.xml:
 - o Khai báo thẻ cặp thẻ <intentfilter></intentfilter>
 - o Trong cặp thẻ có thể chứa các thẻ sau: <action/>, <data/>, <category/>

2.3.1. Action:

- Các thuộc tính:

```
<action android:name="string" />
```

- o Trong đó:
 - **android:name**: sử dụng các thuộc tính trong lớp **Intent.ACTION_string** hoặc tự định nghĩa chuỗi action.
- Ví dụ: khai báo:

```
<action android:name="android.intent.action.MAIN" />
<action android:name="com.nghialt.t3h.action.ShowImage" />
```

2.3.2. Data:

- Các thuộc tính:

```
<data
    android:scheme="string"
    android:host="string"
    android:port="string"
    android:path="string"
    android:pathPattern="string"
    android:pathPrefix="string"
    android:mimeType="string" />
```

- Ví dụ: khai báo:

```
<data
    android:scheme="http" android:mimeType="video/mpeg" />
<data android:mimeType="image/*" />
<data android:mimeType="*/*" />
```

2.3.3. Category

- Các thuộc tính:

```
<category android:name="string" />
```

- Trong đó:
 - **android:name:** Intent.CATEGORY_string. Khai báo theo cấu trúc android.intent.category.string
- Ví dụ khai báo:

```
<category android:name="android.intent.category.DEFAULT" />
<category android:name="android.intent.category.LAUNCHER" />
```

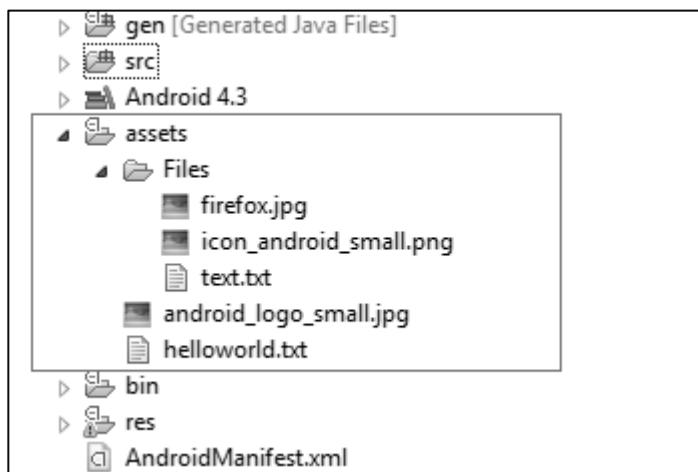
Bài 6

QUẢN LÝ ASSET – SHARED PREFERENCES – BỘ NHỚ THIẾT BỊ

1. BỘ QUẢN LÝ ASSET

1.1. Giới thiệu

- Asset là thư mục chứa dữ liệu đầu vào. Ví dụ: âm thanh, hình ảnh, tập tin CSDL hoặc tập tin có đuôi khác ... Những tập tin này sẽ không được biên dịch khi ứng dụng được đóng gói.



Hình 6.1. Thư mục assets.

- Truy xuất đối tượng quản lý: getAssets() – AssetManager
- Các phương thức xử lý chính:
 - o list(String path) – String[]
 - o open(String fileName) – InputStream
 - o open(String fileName, int mode) – InputStream
 - o close()
- Các chế độ mở Asset
 - o ACCESS_BUFFER
 - o ACCESS_RANDOM
 - o ACCESS_STREAMING
 - o ACCESS_UNKNOWN
- Để truy xuất được những tập tin trong thư mục assets, ta phải sử dụng bộ quản lý asset do Android cung cấp. Lớp này tên là AssetManager.

- AssetManager là lớp cung cấp quyền truy cập vào các tập tin không biên dịch của ứng dụng. AssetManager cho phép bạn mở và đọc file nguyên gốc dưới dạng một luồng byte.
- Cú pháp khởi tạo AssetManager:

```
AssetManager assetManager = Context.getAssets();
```

- Để truy xuất phông chữ thông qua thư mục Asset ta sử dụng phương thức createFromAsset.
- Ví dụ:

```
Typeface arial = Typeface.createFromAsset(getAssets(), "font/arial.ttf");
textView.setTypeface(arial);
```

1.2. Đọc dữ liệu Asset

- Ví dụ sau đây sẽ cho ta cái nhìn tổng quát về cách thức sử dụng bộ quản lý asset để truy xuất tài nguyên tập tin trên thư mục dự án.
- Ta có thư mục assets với các tập tin như Hình 6.1.
- Xây dựng Activity với giao diện layout như sau:

```
<ScrollView xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">
    <LinearLayout
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:orientation="vertical" >
        <TextView
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:text="@string/hello"
            android:id="@+id/txtContent" />
        <ImageView
            android:layout_width="fill_parent"

            android:layout_height="wrap_content"
            android:id="@+id/imgAssets"/>
        <TextView
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:id="@+id/txtFileName" />
    </LinearLayout>
</ScrollView>
```

- Trong Activity tiến hành đọc dữ liệu asset:

```
public class ReadFileAssetsActivity extends Activity {
    @Override
```

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

    TextView txtContent = (TextView) findViewById(R.id.txtContent);
    TextView txtFileName = (TextView) findViewById(R.id.txtFileName);
    ImageView imgAssets = (ImageView) findViewById(R.id.imgAssets);

    AssetManager assetManager = getAssets();

    // To get names of all files inside the "Files" folder
    try {
        String[] files = assetManager.list("Files");
        for(int i=0; i<files.length; i++){
            txtFileName.append("\n File :" + i + " Name => " + files[i]);
        }
    } catch (IOException e1) {
        e1.printStackTrace();
    }
    // To load text file
    InputStream input;
    try {
        input = assetManager.open("helloworld.txt");
        int size = input.available();
        byte[] buffer = new byte[size];
        input.read(buffer);

        input.close();
        // byte buffer into a string
        String text = new String(buffer)
        txtContent.setText(text);
    } catch (IOException e) {
        e.printStackTrace();
    }
    // To load image
    try {
        InputStream ims = assetManager.open("android_logo_small.jpg");

        // create drawable from stream
        Drawable d = Drawable.createFromStream(ims, null);
        // set the drawable to imageview
        imgAssets.setImageDrawable(d);
    }catch(IOException ex) {
        return;
    }
}
```

2. SHARED PREFERENCES

2.1. Giới thiệu

- Shared Preferences là một dạng lưu trữ nhanh bằng file XML.
- SharedPreferences dùng lưu trữ dữ liệu dạng cơ bản.
- Chỉ hỗ trợ một số kiểu dữ liệu cơ bản : booleans, float, int, long, string.
- Hoạt động theo kiểu lưu trữ cũng như truy xuất thông qua cặp giá trị key/value.
- Dữ liệu vẫn được bảo toàn ngay cả khi ứng dụng bị đóng hoàn toàn.
- Các dữ liệu được lưu trữ và truy xuất qua từng phiên (Session) tương tác của ứng dụng với người dùng.
- Lưu trữ và truy xuất trạng thái ứng dụng

2.2. Lưu trữ

- Sử dụng đối tượng của lớp SharedPreferences thông qua phương thức:
GetSharedPreferences(String, int);
- Tạo đối tượng SharedPreferences: sử dụng phương thức **getSharedPreferences()**.

```
SharedPreferences sharedPre = getSharedPreferences(PREFS_NAME, mode);
```

- o Trong đó:
 - **PREFS_NAME**: tên của sharedPreference cần tạo
 - **Mode**: kiểu ghi dữ liệu
 - **MODE_PRIVATE**: mode ghi mặc định, chỉ có duy nhất ứng dụng tạo ra tập tin được phép truy cập vào.
 - **MODE_WORLD_READABLE**: cho phép các ứng dụng khác truy cập vào.
 - **MODE_WORLD_WRITEABLE**: cho phép các ứng dụng khác truy cập và sửa đổi.
 - **MODE_MULTI_PROCESS**: cho phép nhiều tiến trình trên ứng dụng cùng truy xuất vào cùng một thời điểm.
- Bổ sung thông tin cho đối tượng SharedPreferences: thông qua đối tượng **Editor** và **put<Type>** tùy theo dữ liệu.
- Tạo đối tượng Editor từ phương thức **edit()** của Preference.
- Các giá trị lưu trữ được truy xuất thông qua phương thức **get<KDL>**:
 - o **getBoolean(Key, DefaultValue)**
 - o **getFloat(Key, DefaultValue)**
 - o **getInt(Key, DefaultValue)**
 - o **getLong(Key, DefaultValue)**
 - o **getString(Key, DefaultValue)**

- Ví dụ:

```
SharedPreferences.Editor editor = sharedPre.edit();
editor.putBoolean("isTrue", true);
editor.putFloat("lastFloat", 1f);
editor.putInt("number", 2);
editor.putLong("aNumber", 3l);
editor.putString("textEntryValue", "Not Empty");
//Sau cùng, để lưu SharedPreference dùng lệnh commit()
editor.commit();
```

2.3. Truy xuất

- Các định dạng mở tập tin khi truy xuất Preference:
 - o MODE_PRIVATE
 - o MODE_APPEND
 - o MODE_WORLD_READABLE (Deprecated in API 17)
 - o MODE_WORLD_WRITEABLE (Deprecated in API 17)
 - o MODE_MULTI_PROCESS
- Các giá trị lưu trữ được truy xuất thông qua phương thức get<KDL>:
 - o getBoolean(Key, DefValue)
 - o getFloat(Key, DefValue)
 - o getInt(Key, DefValue)
 - o getLong(Key, DefValue)
 - o getString(Key, DefValue)
- Lấy dữ liệu trong SharedPreferences(): sử dụng phương thức get<Type> tùy theo từng loại dữ liệu.

```
boolean isTrue = sharedPre.getBoolean("isTrue", false);
float lastFloat = sharedPre.getFloat("lastFloat", 0f);
int number = sharedPre.getInt("number", 1);
long aNumber = sharedPre.getLong("aNumber", 0);
String stringPreference = sharedPre.getString("textEntryValue", "");
```

- Giá trị mặc định: là giá trị sẽ được trả về mặc định khi get<Type> không trả về được value thích hợp từ key truyền vào.
 - o Ví dụ:

```
//Nếu không tìm thấy giá trị tương ứng với từ khóa "number" truyền vào thì
Default Value là 1
    int number = sharedPre.getInt("number", 1);
```

3. BỘ NHỚ THIẾT BỊ

3.1. Bộ lưu trữ trong

- Lưu dữ liệu riêng trên bộ nhớ thiết bị.
- Khi ứng dụng được cài đặt, hệ thống cung cấp 1 vùng bộ nhớ trong dành cho việc lưu trữ dữ liệu cho mỗi ứng dụng riêng (sandbox).
- Khi người dùng gỡ bỏ ứng dụng, vùng bộ nhớ này sẽ được xóa.
- Chỉ có ứng dụng mới có thể truy xuất được bộ nhớ của ứng dụng đó.
- Đường dẫn các tập tin được lưu trữ:

```
data/data/<package_name_app>/files
```

- Phương thức truy xuất:

```
getFilesDir - File
```

- Tạo và lưu trữ file vào bộ lưu trữ trong
 - o Gọi **openFileOutput()** với tên file và chế độ truy cập file.
 - o Ghi lên file sử dụng phương thức **write()**.
 - o Đóng stream với phương thức **close()**.
 - o Đọc file từ bộ lưu trữ trong.
 - o Gọi **openFileInput()** với tên file cần đọc.
 - o Đọc nội dung từ file sử dụng phương thức **read()**.
 - o Đóng stream với phương thức **close()**
 - o Ví dụ:

```
String FILENAME = "hello_file";
String string = "hello world!";
FileOutputStreamfos = openFileOutput(FILENAME, Context.MODE_PRIVATE);
fos.write(string.getBytes());
fos.close();
```

3.2. Bộ lưu trữ ngoài

- Mỗi thiết bị Android cung cấp bộ nhớ ngoài cho phép người dùng có thể lưu trữ và truy xuất trực tiếp trên thiết bị hoặc thông qua máy tính khi kết nối USB Storage. Hay nói cách khác, mỗi thiết bị Android có thể hỗ trợ bộ lưu trữ ngoài sử dụng lưu file (như là SDcard).
- Lưu trữ dữ liệu **public** trên bộ nhớ ngoài.
- Bộ lưu trữ ngoài bao gồm hai dạng:
 - o Bộ nhớ thiết bị (non-removable)
 - o Bộ nhớ ngoài (sd-card, usb...)

- Cần xin cấp quyền để truy xuất bộ nhớ này:

```
android.permission.WRITE_EXTERNAL_STORAGE
```

- Ứng dụng được cấp phát bộ nhớ ngoài để lưu trữ dữ liệu, mặc định không thể truy xuất như tập tin Media.
- Địa chỉ lưu trữ:

```
/mnt/sdcard/Android/<package_name>/files
```

- Phương thức truy xuất:
 - o getExternalFilesDir(String) – File
 - o getExternalFilesDirs(String) – File[] (API 19 - KITKAT)
 - o ContextCompat.getExternalFilesDirs(String) – File[] (Support Library)
- Cấu hình cài đặt ứng dụng trên bộ nhớ trong hoặc ngoài thông qua cặp thẻ <manifest> trong AndroidManifest.
 - o installLocation
 - o internalOnly
 - o Auto
 - o preferExternal
- Dữ liệu được lưu trữ trên bộ lưu trữ ngoài có thể được thay đổi bởi người dùng, có thể đồng bộ với máy tính cá nhân.
- Trước khi làm việc với bộ lưu trữ ngoài cần kiểm tra trạng thái của nó trước khi thực hiện các thao tác.
- Truy xuất trạng thái bộ nhớ ngoài thông qua phương thức **getExternalStorageState** trong đối tượng Environment.
- Các thông số được trả về bao gồm:
 - o MEDIA_UNKNOWN
 - o MEDIA_REMOVED
 - o MEDIA_UNMOUNTED
 - o MEDIA_CHECKING
 - o MEDIA_NOFS
 - o MEDIA_MOUNTED
 - o MEDIA_MOUNTED_READ_ONLY
 - o MEDIA_SHARED
 - o MEDIA_BAD_REMOVAL
 - o MEDIA_UNMOUNTABLE

- Ví dụ: kiểm tra trạng thái bộ lưu trữ ngoài

```
public boolean isExternalStorageWritable() {  
    String state = Environment.getExternalStorageState();  
    if( Environment.MEDIA_MOUNTED.equals(state))  
        return true;  
    return false;  
}
```

- Một số phương thức hỗ trợ duyệt tập tin trong bộ nhớ:
 - o isFile() – boolean
 - o isDirectory() – boolean
 - o getAbsolutePath() - String
 - o getPath() - String
 - o list() – String[]
 - o mkdir() - boolean
 - o mkdirs() – boolean
 - o delete() – boolean
- Một số thư mục dùng chung:
 - o Thư mục:
 - DIRECTORY_ALARMS
 - DIRECTORY_DCIM
 - DIRECTORY_DOCUMENTS (API 19)
 - DIRECTORY_DOWNLOADS
 - DIRECTORY_MOVIES
 - DIRECTORY_MUSIC
 - DIRECTORY_NOTIFICATIONS
 - DIRECTORY_PICTURES
 - DIRECTORY_POSTCASTS
 - DIRECTORY_RINGTONES

Bài 7

CÁC ĐIỀU KHIỂN HIỂN THỊ DANH SÁCH

1. KHÁI NIỆM CƠ BẢN

- Adapter giữ vai trò như cầu nối giữa các đối tượng điều khiển dạng danh sách (AdapterView) và các dữ liệu bên dưới, cung cấp các phương thức truy xuất dữ liệu. Cho phép thực hiện quản lý giao diện, số lượng chỉ mục trên AdapterView và thực hiện truy vấn dữ liệu, sắp xếp dữ liệu. Bên cạnh đó, Adapter cũng chịu trách nhiệm tạo ra các View con trong tập các dữ liệu.
- AdapterView là các đối tượng điều khiển dạng tập hợp, cho phép hiển thị thông tin cơ bản theo dạng danh sách và thực hiện quản lý thông tin theo từng mục riêng biệt.

2. CÁC DẠNG ADAPTER

- Bao gồm các lớp thực thi giao thức Adapter:
 - ArrayAdapter
 - BaseAdapter
 - CursorAdapter
 - HeaderViewListAdapter
 - ResourceCursorAdapter
 - SimpleAdapter
 - SimpleCursorAdapter
 - SpinnerAdapter
- Các phương thức xử lý quan trọng trên Adapter:
 - getCount - int
 - getItems(int position) - Objects
 - getItemsId(int position) – long
 - getView(int position, View convertView, ViewGroup parent) – View

2.1. BaseAdapter

- BaseAdapter là lớp adapter cơ sở cho các Adapter thường dùng khác như ArrayAdapter<T>, CursorAdapter, SimpleAdapter...
- Ví dụ:
 - Tạo một ArrayAdapter chứa các dữ liệu chuỗi.

```
String arr[]={"One", "Two", "Three", "Four"};
ArrayAdapter<String> adapter = new
    ArrayAdapter<String>(this, android.R.layout.simple_list_item_1, arr);
```

- Trong đó:
 - **android.R.layout.simple_list_item_1**: là layout mặc định do Android xây dựng sẵn cho 1 item của ListView.
 - Tuy nhiên, nếu muốn sử dụng với những giao diện phức tạp hơn như ImageView, hoặc nhiều TextView ta nên ghi đè lại getView (int, View, ViewGroup) để trả lại kiểu giao diện mà bạn muốn.
- 2.2. SimpleCursorAdapter
- SimpleCursorAdapter là lớp Adapter lấy dữ liệu từ một Cursor ra một điều khiển hiển thị ví dụ ListView, GridView, ... Cursor phải có chứa tên định danh cho các cột, nếu không sẽ không thực hiện được.

3. CÁC ĐIỀU KHIỂN DANH SÁCH

3.1. AbsListView

- AbsListView: đối tượng điều khiển hiển thị danh sách các danh mục với thông tin cơ bản, cho phép thực hiện các thao tác khác nhau trên từng danh mục. Bao gồm hai chế độ hiển thị:
 - ListView
 - GridView
- Thuộc tính XML quan trọng:
 - listSelector: drawable
 - choiceMode: none | singleChoice | multipleChoice | multipleChoiceModal
 - smoothScrollbar: boolean
 - fastScrollEnable: boolean
- Một số Interface đã được khai báo sử dụng :
 - TextWatcher
 - ViewTreeObserver.OnGlobalLayoutListener
 - ViewTreeObserver.OnTouchModeChangeListener
 - Filter.FilterListener

3.2. Listview

- ListView là một dạng điều khiển nâng cao có chức năng hỗ trợ hiển thị dữ liệu theo dạng từng dòng. ListView cần một đối tượng Adapter để quản lý và giúp hiển thị dữ liệu.
- ArrayAdapter là gì?
 - ArrayAdapter được hiểu như cái modem mạng. Nó giúp ListView có thể “đọc và hiểu” dữ liệu từ mảng dữ liệu để hiện lên giao diện.
 - ArrayAdapter là một lớp kế thừa từ BaseAdapter và được hỗ trợ bởi một mảng các đối tượng tùy ý. Mặc định lớp này sẽ được cung cấp Id tham chiếu đến một TextView duy nhất để làm điều khiển hiển thị dữ liệu.

- Các bước để tạo một ListView
 - o Bước 1: Đưa dữ liệu cần hiển thị lên ListView vào một mảng hoặc danh sách (ArrayList, mảng thông thường, mảng đối tượng,...).
 - o Bước 2: Tạo một ListView trên giao diện.
 - o Bước 3: Tạo một đối tượng ArrayAdapter để liên kết giữ ListView và mảng hoặc danh sách dữ liệu.
- Các xử lý trên ListView
 - o Cách lấy ListView thông qua Id của ListView

```
ListView lvTenLV=(ListView)findViewById(R.id.idcuaListView);
```
 - o Cách tạo ArrayAdapter và đưa dữ liệu từ mảng vào ArrayAdapter

```
ArrayAdapter<[Kiểu mảng]>[Tên mảng adapter]  
= new ArrayAdapter<Kiểumảng>  
(this, android.R.layout.simple_list_item_1, [tenMangDuLieu]);
```
- Dữ liệu từ mảng chứa dữ liệu sẽ được gắn vào ArrayAdapter, ArrayAdapter sẽ gắn vào ListView.
- Giải thích ý nghĩa các đối số trong phương thức ArrayAdapter:
 - o Đối số thứ 1: **this**
 - Đại diện cho context của Activity hiện tại, bạn có thể viết MainActivity.this (nếu bạn viết như thế này thì ở bất kỳ vị trí nào nó cũng hiểu là context của MainActivity).
 - o Đối số thứ 2: **android.R.layout.simple_list_item_1**
 - Là layout Listview mà được Android xây dựng sẵn. Nó được lưu trong SDK/ platforms/ android-api(x)/ data/ res/ layout/ simple_list_item_1.xml.
 - o Đối số thứ 3: **[tenMangDuLieu]**
 - Là mảng chứa dữ liệu cần hiển thị lên ListView.
- Thuộc tính XML quan trọng:
 - o listSelector: drawable
 - o divider: drawable
 - o dividerHeight: dimen
 - o entries: string-array
- Một số phương thức quan trọng:
 - o setAdapter(Class Extends <T implements Adapter>)
 - o addHeaderView(View) – removeHeaderView(View)
 - o addFooterView(View) – removeFooterView(View)
 - o setSelection(int)
 - o smoothScrollToPosition(int)

- Ví dụ: Tạo ListView hiển thị danh sách tên các quốc gia.

Bước 1: Tạo dữ liệu

- Chúng ta tạo mảng dữ liệu để hiển thị trên listview thông qua một tập tin xml bằng cách tạo mới một tập tin list_data.xml trong / res/ values với nội dung sau:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string-array name="countries">
        <item>Sri Lanka</item>
        <item>Tokelau</item>
        <item>United Kingdom</item>
        <item>Uruguay</item>
        <item>Vatican</item>
        <item>Vietnam</item>
    </string-array>
</resources>
```

Bước 2: Tạo một ListView trên giao diện

- Kéo vào một ListView trong thẻ tab Composite hoặc khai báo 1 cặp thẻ <listview> trong file giao diện xml:

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >
    <ListView
        android:id="@+id/lvCountry"
        android:layout_width="match_parent"
        android:layout_height="match_parent" >
    </ListView>
</LinearLayout>
```

Bước 3: Đưa dữ liệu vào mảng, tạo đối tượng ArrayAdapter, đưa dữ liệu từ mảng vào ArrayAdapter, hiển thị nội dung lên ListView

- Chúng ta vào Activity để thao tác như sau:

```
//Đưa dữ liệu vào mảng
String[] countries = getResources().getStringArray(R.array.countries);
//lấy listview thông qua id của listview
ListView lvCountry=(ListView)findViewById(R.id.lvCountry);
//Tạo đối tượng ArrayAdapter và đưa dữ liệu từ mảng vào ArrayAdapter
ArrayAdapter<String> adapter = new ArrayAdapter<String>
    (this, android.R.layout.simple_list_item_1, countries);
//đổ dữ liệu lên listview
lvCountry.setAdapter(adapter);
```

Bước 4: Thiết lập sự kiện cho ListView

- Sau khi dữ liệu đã được hiển thị lên ListView, tiếp theo, để bắt sự kiện khi click lên mỗi dòng của listview, chúng ta gán sự kiện **setOnItemClickListener** cho ListView như sau:

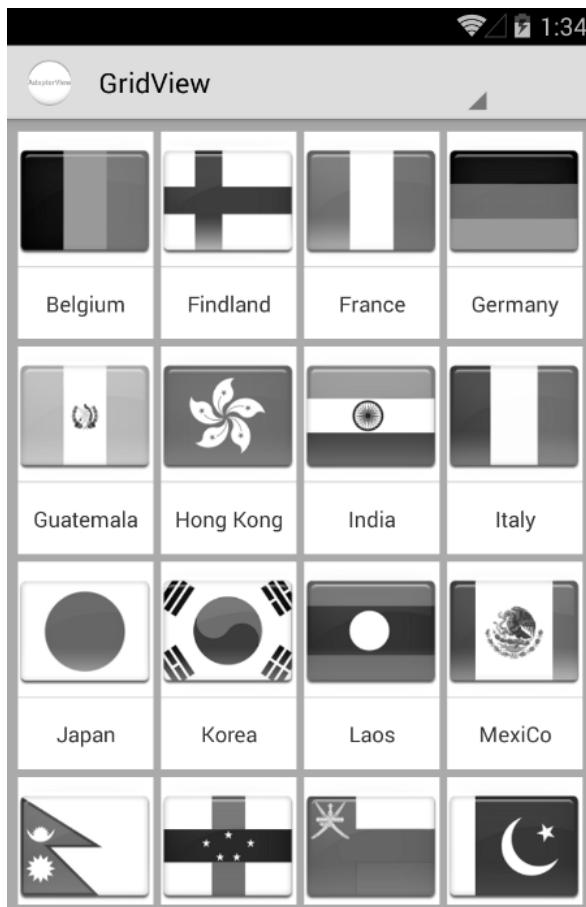
```
lvCountry.setOnItemClickListener(new OnItemClickListener(){
@Override
public void onItemClick(AdapterView<?> adapterView, View v, int position, long
id) {
    //Ghi Log vị trí của Item được click vào trên ListView
    Log.d("ListView","onItemClick-POS:"+position+ " "+ID+" "+id);
}
});
```

Cách khác: chúng ta còn có thể thiết lập sự kiện trên cho listView bằng cách cho Activity kế thừa lớp ListActivity để trở thành một Activity dạng list:

```
public class ListViewActivity extends ListActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        String[] countries =getResources().getStringArray(R.array.countries);
        setListAdapter(new ArrayAdapter<String>
(this,android.R.layout.simple_list_item_1,android.R.id.text1,countries));
        getListView().setOnItemClickListener(new OnItemClickListener() {
            @Override
            public void onItemClick(AdapterView<?> parent, View view, int
position, long id) {
                // Ghi log
                Log.d("ListView","onItemClick- POS :" +position+" "+ ID : " "+id);
            }
        });
    }
}
```

3.3. GridView

- GridView giống với ListView nhưng có chức năng hỗ trợ hiển thị dữ liệu theo dạng lưới. GridView cũng cần một đối tượng Adapter để quản lý và giúp hiển thị dữ liệu.



Hình 7.1. Ví dụ về Custom GridView.

- Gridview cũng dựa vào Adapter để gắn kết các dữ liệu bên dưới.

```
String[] adobe_products =  
getResources().getStringArray(R.array.adobe_products);  
GridView gvAdobe = (GridView) findViewById(R.id.gvAdobe);  
  
ArrayAdapter<String> adapter = new ArrayAdapter<String>  
(ListViewActivity.this, android.R.layout.simple_list_item_1, adobe_products);  
gvAdobe.setAdapter(adapter);
```

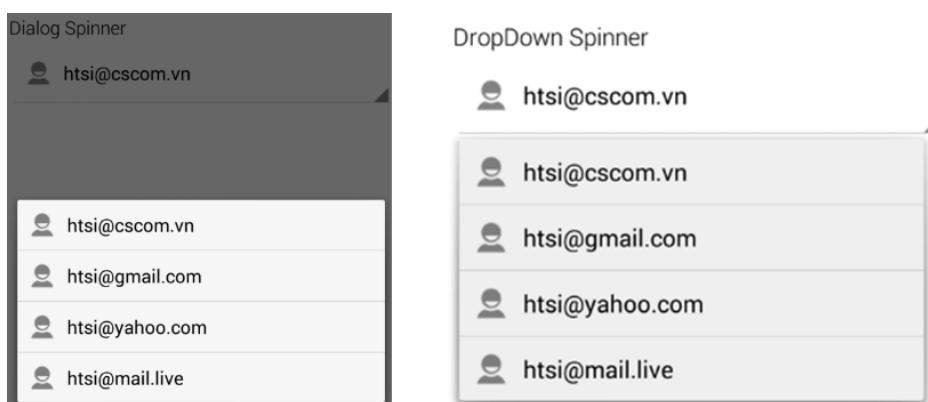
- Điểm khác nhau là GridView có thiết lập số cột. Dữ liệu luôn đưa vào dưới dạng hình ống (mảng, list một chiều), nhưng dựa vào số cột ta thiết lập mà nó tự động ngắt hàng dựa vào thuộc tính numColums trong layout.

```
<GridView  
    android:id="@+id/gvAdobe"  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    android:numColumns="3" >  
</GridView>
```

- Thuộc tính XML quan trọng:
 - o columnWidth: dimen
 - o gravity: Gravity
 - o horizontalSpacing: dimen
 - o verticalSpacing: dimen
 - o numColumns: integer
 - o stretchMode: none | spacingWidth | columnWidth | spacingWidthUniform
- Một số phương thức quan trọng:
 - o setColumnWidth(int) – getColumnWidth()
 - o setNumColumn(int) – getNumColumn()
 - o setSelection(int)
 - o smoothScrollToPosition(int)

3.4. Spinner

- Spinner là đối tượng điều khiển hiển thị một danh mục ở một thời điểm, người dùng có thể lựa chọn một trong nhiều danh mục để hiển thị.
- Bao gồm hai chế độ hiển thị pop-up lựa chọn (spinnerMode):
 - o Dialog
 - o Dropdown



Hình 7.2. Hai chế độ hiển thị dữ liệu với Spinner.

- Thuộc tính XML quan trọng:
 - o spinnerMode: dialog | dropdown
 - o prompt: string
 - o popupBackground: drawable | color

- gravity
 - entries: string-array
- Một số phương thức quan trọng:
 - setAdapter(SpinnerAdapter)
 - setPrompt(CharSequence) – setPrompt(int resId) (Dialog Mode)
 - setPopupBackgroundResource(int)
 - setPopupBackgroundDrawable(Drawable)
 - Ví dụ: xây dựng Spinner:

```
// Xây dựng Adapter thông qua dữ liệu tài nguyên và giao diện mẫu
SpinnerAdapter adapter = new ArrayAdapter.createFromResource(this,
    R.array.countries,
    android.R.layout.simple_dropdown_item_1line);
// Tham chiếu điều khiển
Spinner spinner = (Spinner)findViewById(R.id.spinner);
// Thiết lập Adapter cho điều khiển
spinner.setAdapter(adapter);
```

3.5. Tao Custom ListView

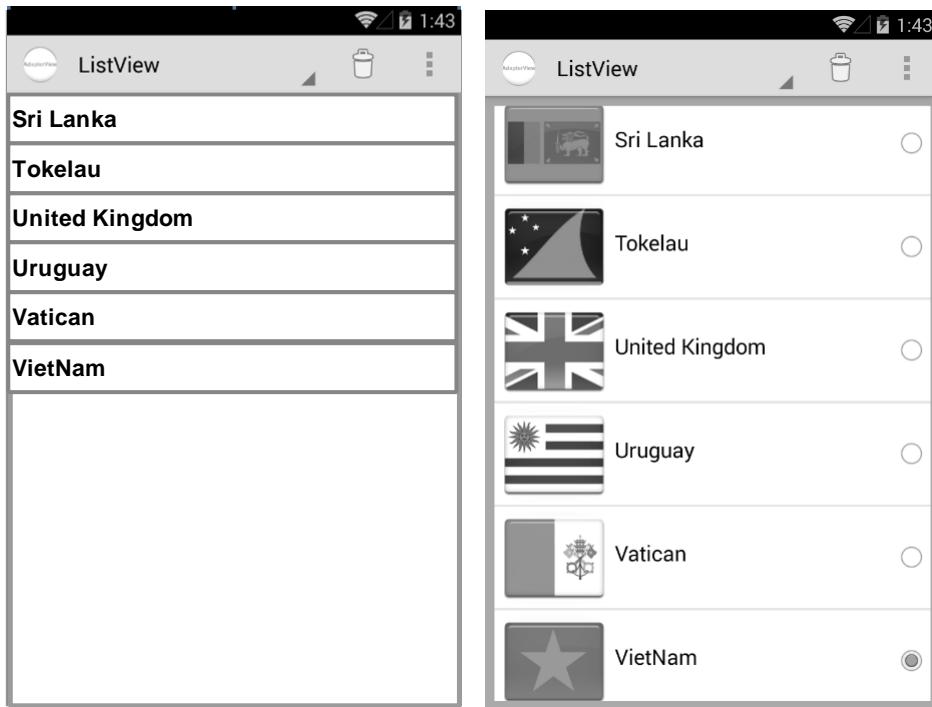
- Trong thực tế, các ứng dụng Android có liên quan đến ListView thì đa phần chúng ta phải điều chỉnh lại layout cho đúng với yêu cầu của khách hàng. Và cách điều chỉnh lại layout là tạo Custom Layout cho ListView. Vì vậy, học cách tạo Custom Layout cho ListView là cần thiết và thực tế.
- Sự khác nhau giữa ListView thông thường và ListView có layout được điều chỉnh lại là ở việc khởi tạo Adapter.
- Sau khi đã có ListView trên giao diện, chúng ta có thể tạo Custom Layout cho ListView như sau:

Bước 1: Tạo thêm một layout cho một item của ListView.

Bước 2: Tạo lớp Custom Adapter kế thừa từ lớp ArrayAdapter.

Bước 3: Tạo một lớp dùng để quản lý dữ liệu.

Bước 4: Hiển thị dữ liệu lên ListView.



Hình 7.3. Minh họa Custom ListView.

Sau đây là ví dụ về cách tạo Custom Layout cho ListView (lấy ví dụ ở phần III.1).

- Tạo layout trong Activity như sau:

```
<RelativeLayout 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:paddingBottom="@dimen/activity_vertical_margin"  
    android:paddingLeft="@dimen/activity_horizontal_margin"  
    android:paddingRight="@dimen/activity_horizontal_margin"  
    android:paddingTop="@dimen/activity_vertical_margin"  
    tools:context="nghialt.demo.democustomlistview.MainActivity" >  
  
    <ListView  
        android:id="@+id/lvCountries"  
        android:layout_width="match_parent"  
        android:layout_height="match_parent" >  
    </ListView>  
</RelativeLayout>
```

- Tạo layout cho một item của ListView gọi là list_row.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="match_parent"
    android:orientation="horizontal"
    android:padding="10dp" >

    <ImageView
        android:id="@+id/imgvFlag"
        android:layout_width="80dp"
        android:layout_height="80dp"
        android:contentDescription="@null"
        android:src="@drawable/flag_japan" />

    <LinearLayout
        android:layout_width="120dp"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        android:layout_marginLeft="20dp"
        android:padding="10dp" >

        <TextView
            android:id="@+id/tvEnName"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@null"
            android:textColor="#357B6A"
            android:textAppearance="?android:attr/textAppearanceMedium" />

        <TextView
            android:id="@+id/tvViName"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@null"
        />
    </LinearLayout>

    <RelativeLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent" >

        <CheckBox
            android:id="@+id/checkBox1"
            android:layout_width="wrap_content"
```

```
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="16dp" />

    </RelativeLayout>

</LinearLayout>
```

- Custom lại Adapter bằng cách tạo một class kế thừa từ class ArrayAdapter gọi là CountryAdapter

```
public class CountryAdapter extends ArrayAdapter<Country> {

    Context context;
    int resId;
    ArrayList<Country> countries;

    public CountryAdapter(Context context, int resId,
                          ArrayList<Country> countries) {
        super(context, resId, countries);
        this.context = context;
        this.resId = resId;
        this.countries = countries;
    }

    @Override
    public View getView(int position, View convertView, ViewGroup parent) {
        View view = null;
        LayoutInflater inflater = (LayoutInflater) context
            .getSystemService(Context.LAYOUT_INFLATER_SERVICE);

        view = inflater.inflate(resId, null);

        TextView tvEnName = (TextView) view.findViewById(R.id.tvEnName);
        TextView tvViName = (TextView) view.findViewById(R.id.tvViName);
        ImageView imgvFlag = (ImageView) view.findViewById(R.id.imgvFlag);

        Country c = countries.get(position);
        if (c != null) {
            tvEnName.setText(c.getEnName());
            tvViName.setText(c.getViName());
            imgvFlag.setImageResource(c.getImg_id());
        }
        return view;
    }
}
```

- Xây dựng đối tượng Country:

```
public class Country {  
    private String enName;  
    private String viName;  
    private int img_id;  
    public String getEnName() {  
        return enName;  
    }  
    public void setEnName(String enName) {  
        this.enName = enName;  
    }  
    public String getViName() {  
        return viName;  
    }  
    public void setViName(String viName) {  
        this.viName = viName;  
    }  
    public int getImg_id() {  
        return img_id;  
    }  
    public void setImg_id(int img_id) {  
        this.img_id = img_id;  
    }  
    public Country(String enName, String viName, int img_id) {  
        this.enName = enName;  
        this.viName = viName;  
        this.img_id = img_id;  
    }  
    public Country() {  
        // TODO Auto-generated constructor stub  
    }  
}
```

- Tạo class CountryModel để quản lý dữ liệu:

```
public class CountryModel {  
  
    public static String[] en_countries = new String[] { "Vietnam", "France",  
    "Belgium", "Italy", "Germany", "Spain", "Japan", "Laos" };  
  
    public static String[] vi_countries = new String[] { "Việt Nam", "Pháp",  
    "Bỉ", "Ý", "Đức", "Tây Ban Nha", "Nhật Bản", "Lào" };  
  
    public static int[] img_ids = new int[] { R.drawable.flag_vietnam,  
    R.drawable.flag_france, R.drawable.flag_belgium,  
    R.drawable.flag_italy, R.drawable.flag_germany,  
    R.drawable.flag_spain, R.drawable.flag_japan, R.drawable.flag_laos };  
  
    private static ArrayList<Country> countries;
```

```
public static ArrayList<Country> getCountries() {
    if (countries == null) {
        CountryModel countryModel = new CountryModel();
        countryModel.initCountryList();
    }
    return countries;
}

private void initCountryList() {

    countries = new ArrayList<Country>();
    for (int i = 0; i < en_countries.length; i++) {
        Country c = new Country(en_countries[i], vi_countries[i],
                               img_ids[i]);
        countries.add(c);
    }
}
```

- Tạo và setAdapter cho ListView trong MainActivity

```
public class MainActivity extends Activity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

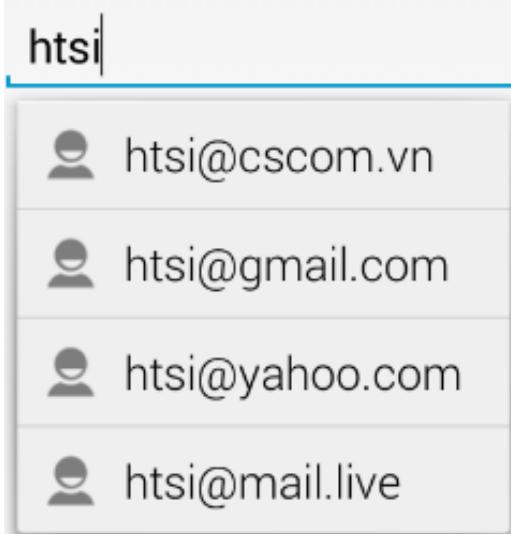
        ListView lvCountries = (ListView) findViewById(R.id.lvCountries);

        ArrayList<Country> countries = CountryModel.getCountries();

        CountryAdapter countryAdapter = new CountryAdapter(this,
            R.layout.my_item_layout, countries);
        lvCountries.setAdapter(countryAdapter);
    }
}
```

3.6. AutoCompleteTextView

- Đổi tượng kế thừa từ EditText.
- Cho phép xây dựng dữ liệu mẫu hỗ trợ người dùng hoàn chỉnh quá trình nhập liệu trên EditText.
- Thực hiện xây dựng AutoCompleteTextView:
- Khai báo dữ liệu mẫu.
- Khai báo giao diện hiển thị cho dữ liệu.
- Xây dựng Adapter thông qua phương thức khởi tạo tương ứng với dữ liệu và giao diện hiển thị.
- Thiết lập Adapter cho đối tượng AutoCompleteTextView



Hình 7.4. Minh họa AutoComplexTextView.

- Ví dụ xây dựng AutoCompleteTextView:

```
// Khởi tạo dữ liệu mẫu
private static final String[] COUNTRIES = new String[]
{“VietNam”, “Belgium”, “France”, “Italy”, “Germny”, “Spain”};
// Xây dựng Adapter thông qua dữ liệu mẫu và giao diện mẫu
ArrayAdapter<String> adapter = new ArrayAdapter<String> (this,
    android.R.layout.simple_dropdown_item1line, COUNTRIES);
// Tham chiếu điều khiển
AutoCompleteTextView editCountry = (AutoCompleteTextView)
    findViewById(R.id.editCountry);
// Thiết lập Adapter cho điều khiển
editCountry.setAdapter(adapter);
```

3.7. MultiAutoCompleteTextView

- Đổi tượng kế thừa từ đối tượng AutoCompleteTextView.
- Cho phép xây dựng dữ liệu mẫu hỗ trợ người dùng hoàn chỉnh quá trình nhập liệu trên EditText.
- Dữ liệu được hỗ trợ hoàn chỉnh nhiều lần, cách nhau bằng một Tokenizer.
- Thực hiện xây dựng MultiAutoCompleteTextView:
- Khai báo dữ liệu mẫu.
- Khai báo giao diện hiển thị cho dữ liệu.
- Xây dựng Adapter thông qua phương thức khởi tạo tương ứng với dữ liệu và giao diện hiển thị.
- Thiết lập Adapter cho đối tượng MultiAutoCompleteTextView.
- Thiết lập đối tượng Tokenizer.

- Ví dụ xây dựng MultiAutoCompleteTextView:

```
// Khởi tạo dữ liệu mẫu
private static final String[] COUNTRIES = new String[]
{“VietNam”, “Belgium”, “France”, “Italy”, “Germny”, “Spain”};
// Xây dựng Adapter thông qua dữ liệu mẫu và giao diện mẫu
ArrayAdapter<String> adapter = new ArrayAdapter<String> (this,
    android.R.layout.simple_dropdown_item1line, COUNTRIES);
// Tham chiếu điều khiển
MultiAutoCompleteTextView editCountry = (MultiAutoCompleteTextView)
    findViewById(R.id.editCountry);
// Thiết lập Adapter cho điều khiển
editCountry.setAdapter(adapter);
// Thiết lập Tokenizer
editCountry.setTokenizer(new MultiCompleteTextView.CommaTokenizer());
```

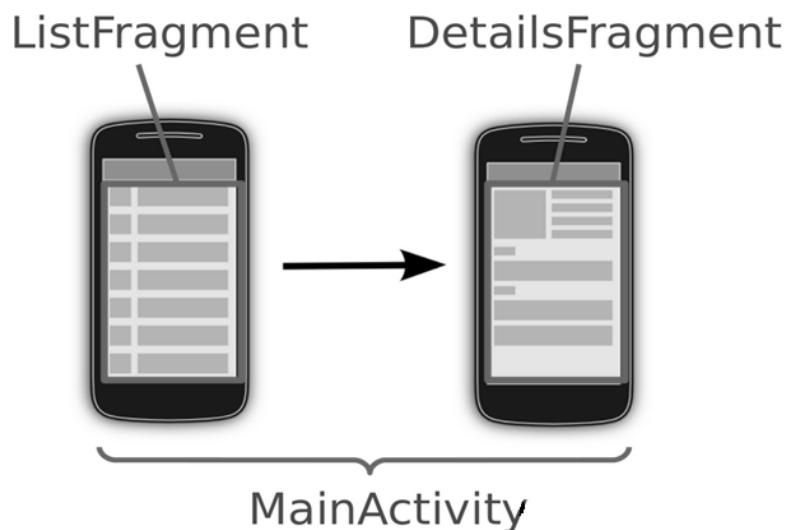
Bài 8

XÂY DỰNG GIAO DIỆN VỚI FRAGMENT VÀN ĐỀ ACTIVITY VÀ FRAGMENT

1. CÁC KHÁI NIỆM CƠ BẢN

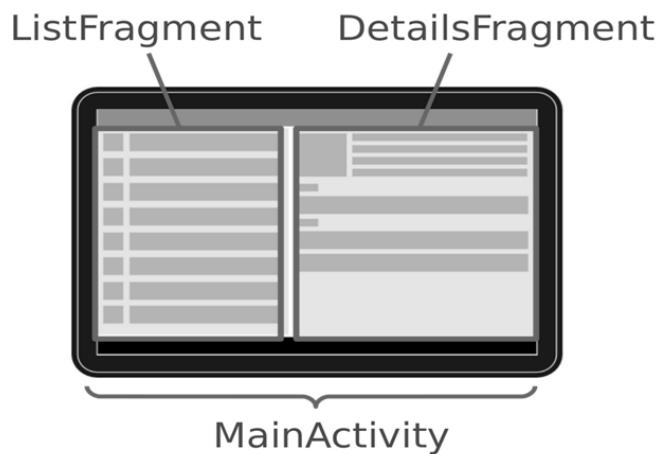
1.1. Fragment và phiên bản hỗ trợ

- Fragment được giới thiệu trong phiên bản Android 3.0 Honeycomb, tuy nhiên do được hỗ trợ trong gói Android Support Library nên ta có thể sử dụng trong các phiên bản từ Android 1.6 trở lên.
- Fragment là đối tượng được nhúng trong Activity, cho phép thực hiện nhận tương tác, có vòng đời riêng và thực hiện trao đổi thông tin với Activity và các Fragment khác.
- Yếu tố chính trong việc sử dụng Fragment là nó có thể giúp chúng ta xây dựng các giao diện một cách chủ động và linh hoạt để có thể thích ứng với các kiểu màn hình có kích thước khác nhau từ điện thoại cho đến máy tính bảng, nói chính xác hơn là tối ưu hóa giao diện cho từng loại thiết bị, kích thước và độ phân giải.



Hình 8.1. Xây dựng giao diện ứng dụng trên điện thoại với Fragment.

- Tính linh động của Fragment cho phép chúng ta bố cục lại các phần giao diện của một ứng dụng theo cách hợp lý nhất.



Hình 8.2. Xây dựng giao diện ứng dụng trên máy tính bảng với Fragment.

- Mỗi Fragment sẽ không bị bó buộc trong một Activity nhất định và có thể tái sử dụng nhiều lần, giúp chủ động hơn trong việc xây dựng giao diện từ việc thêm hoặc xóa các thành phần vào các cửa sổ khác nhau.
- Các lớp Fragment:
 - Fragment
 - o ListFragment
 - o DialogFragment
 - o PreferenceFragment
 - o WebViewFragment
 - Trong bộ Android SDK tích hợp một số lớp con từ lớp Fragment cho phép đóng gói và thực thi một số Fragment thường dùng. Ta có thể liệt kê một số lớp như sau:
 - DialogFragment – Fragment cho phép thể hiện ở dạng hộp thoại trên Activity. Chúng ta có thể thực hiện các tùy chỉnh về giao diện cho Fragment dạng này thông qua các phương thức có sẵn trong lớp Fragment.
 - ListFragment – dạng Fragment tích hợp sẵn điều khiển ListView và các phương thức dùng để thực kết nối dữ liệu.
 - WebViewFragment – dùng để hiện các kết nối và hiển thị nội dung trang web trên đối tượng WebView.

1.2. Giao diện

- Để tạo mới đối tượng Fragment ta cần tạo lớp kế thừa từ lớp Fragment, khai báo các điều khiển và thực thi các chức năng.
- Hầu như khi làm việc với Fragment, ta cần tạo ra các giao diện cho nó bằng cách gắn các điều khiển, tuy nhiên ta cũng có thể cung cấp giao diện cho Fragment một cách trực tiếp từ XML giống như ta làm việc với Activity bằng phương thức setContentView().

- Các thuộc tính quan trọng:
 - o **class**: <Class extends Fragment> (Pakage Name + Class Name).
 - o **name**: <Class extends Fragment> (Pakage Name + Class Name).
 - o **id**: Identifier Resource.
 - o **tag**: String Resource.
 - o **layout_width – layout_height**: Dimen.
 - o **layout_weight**: Integer.
 - o **lauout_gravity**: Gravity.
 - o **layout_margin<Postion>**: Dimen.

Ở đây, ta có thể dùng phương thức onCreateView() để thực hiện điều tương tự.

```
public class MyFragment extends Fragment{  
    @Override  
    public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) {  
        return inflater.inflate(R.layout.activity_main, container, false);  
    }  
}
```

- Ví dụ: giao diện Fragment trong XML:

```
fragment_layout.xml:  
<fragment xmlns:android="http://schemas.android.com/apk/res/android"  
class="com.htsi.demofragment.ImageFragment"  
android:id="@+id/imageFragment"  
android:tag="ImageFramgent"  
android:layout_width="match_parent"  
android:layout_height="match_parent"  
android:layout_margin="@dimen/marginFragment" />
```

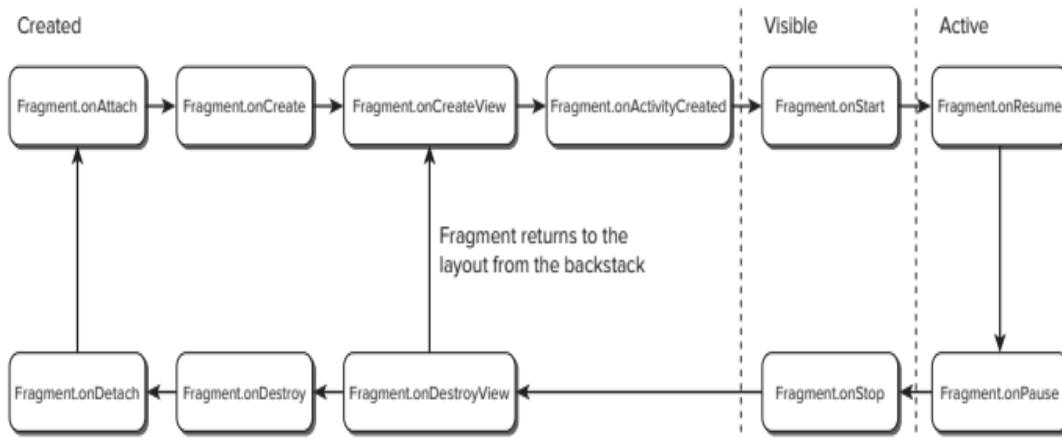
- Sử dụng trong MainActivity.java như một layout bình thường:

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

- Chúng ta hoàn có thể xây dựng giao diện trong mã Java thông qua đối tượng ViewGroup, tuy nhiên cách tốt hơn hết là dùng xây dựng tập tin giao diện XML và gán vào Fragment.
- Không giống như Activity, Fragment không cần phải khai báo trong Manifest bởi vì Fragment chỉ tồn tại khi nhúng nó vào một Activity và vòng đời sống cũng phụ thuộc Activity này.

1.3. Vòng đời của một Fragment

- Vòng đời của một Fragment sẽ được quản lý bởi các phương thức giống như một Activity. Chúng cũng có các phương thức khởi tạo (created), bắt đầu thực thi (started), phục hồi trạng thái (resumed), tạm dừng tương tác (paused), đóng (stopped) và hủy (destroyed). Bên cạnh đó các phương thức được gọi thực thi lại từ lớp Activity sẽ được dùng để cho biết việc kết nối và hủy kết nối Fragment đến Activity, khởi tạo và hủy khởi tạo các điều khiển trên Fragment, thông báo việc hoàn thành việc khởi tạo Activity chứa Fragment. Có thể xem lược đồ sau để hiểu thêm chi tiết.



Hình 8.3. Vòng đời của Fragment.

- Hầu hết các phương thức này đều khá giống so với các phương thức quản lý vòng đời của Activity mà ta đã tìm hiểu trong chương 3. Chúng có vai trò chỉ định cách thức hoạt động của Fragment và cách Fragment được gắn vào một Activity.
- Vòng đời của một Fragment bắt đầu từ lúc chúng được gắn lên một Activity và kết thúc khi được gỡ bỏ. Hai sự kiện này sẽ được giám sát bởi hai phương thức tương ứng là onAttach và onDetach.
- Tất cả phương thức quản lý sẽ được gọi sau khi Fragment/Activity vào trạng thái tương tác, tuy nhiên Phương thức onDetach có thể sẽ không được gọi lên nếu Activity chứa Fragment bị hủy mà chưa hoàn thành vòng đời của nó.
- Phương thức onAttach đã được gọi lên trước khi giao diện Fragment được khởi tạo, trước cả sự khởi tạo của Fragment và Activity chứa Fragment. Thường thì phương thức này được dùng để tạo có tham chiếu đến Activity để chuẩn bị các khởi tạo xử lý khác.
- Việc tạo ra một Fragment được thực hiện bởi phương thức onCreate và kết thúc bằng phương thức onDestroy. Khi dùng trong Activity, chúng ta nên khởi tạo Fragment trong onCreate của Activity.

1.4. Lưu trữ Fragment (BackStack)

- Việc lưu trữ Fragment bao gồm hai giai đoạn:
 - o Lưu trữ trạng thái Fragment thông qua biến Bundle:
 - Truy xuất biến Bundle trong phương thức onSaveInstanceState

- Truyền các thông tin cần lưu trữ vào biến Bundle
- Dưa Fragment vào Stack:
 - Thực hiện khai báo Fragment
 - Cho phép nhúng Fragment vào Activity
 - Dùng phương thức addToBackStack(String Tag) trong đối tượng quản lý Fragment để đưa Fragment vào Stack.
 - Có thể truy xuất lại đối tượng Fragment thông qua Tag, hoặc khi phím “Back” trên thiết bị được nhấn.

2. XÂY DỰNG VÀ SỬ DỤNG FRAGMENT

2.1. Thực hiện xây dựng Fragment:

- Khai báo lớp kế thừa từ lớp Fragment
 - Gọi phương thức onCreateView thực hiện tạo giao diện cho Fragment.
 - Ví dụ về tạo lớp Fragment và giao diện:

```
public class ImageFragment extends Fragment {  
    public View onCreateView(LayoutInflater inflater, ViewGroup parent, Bundle b )  
    {  
        View rootView = inflater.inflate(R.layout.fragment_main, null);  
        return rootView;  
    }  
}
```

2.2. Sử dụng Fragment

- Sử dụng Fragment trong Activity, bao gồm hai cách:
 - Thực hiện tham chiếu Fragment từ giao diện XML của Activity.
 - Khai báo đối tượng FragmentManager, cho phép nhúng Fragment vào Activity từ JavaCode.
- Mỗi Activity sẽ có một Fragment Manager dùng để quản lý các Fragment chứa trong nó. Để sử dụng ta cần gọi phương thức getFragmentManager.

```
FragmentManager frManager = getFragmentManager();
```

- Đối tượng này sẽ cung cấp một số phương thức như thêm, gỡ bỏ và cập nhật Fragment vào Activity, quản lý các phiên làm việc của Fragment...
- Các đơn giản nhất để thêm một vào một Activity là khai báo thẻ fragment trong tập tin giao diện của Activity đó. Fragment được xem như một đối tượng ViewGroup khi được gắn vào trong Layout. Thường chúng ta sẽ sử dụng cách này để định nghĩa một tập các giao diện tĩnh cho các màn hình có kích cỡ khác nhau.
- Trong trường hợp ta cần thay đổi giao diện dựa trên trạng thái của ứng dụng thì cách tốt nhất ta nên sử dụng các giao diện động dựa trên FragmentTransaction.

Lớp FragmentTransaction hỗ trợ chúng ta trong việc thêm, gỡ bỏ hoặc thay thế các Fragment trên Activity ngay khi ứng dụng đang thực thi. Dựa vào các tương tác của người dùng mà ta có thể xây dựng các giao diện hợp lý. Ngoài ra, để giúp cho ứng dụng có thể đáp ứng các trải nghiệm người dùng, các chuyển hoạt cũng được sử dụng giữa các phiên làm việc của Fragment Transaction và các Transaction sẽ được lưu trữ trong stack để tái sử dụng.

- Mỗi phiên làm việc của FragmentTransaction được khởi tạo bằng phương thức beginTransaction. Các hoạt động của Fragment sẽ được định nghĩa bằng các phương thức add, remove hoặc replace. Sau đó chúng ta có thể tùy chỉnh các chuyển hoạt hoặc cách thức hoạt động trong stack trước khi thực hiện phương thức commit để thêm Transaction vào hàng đợi.

```
FragmentTransaction frTransaction = frManager.beginTransaction();
//Thiết lập add,remove hoặc replace, addToBackStack...
//Thiết lập các Animation cho Fragment
//Cuối cùng gọi phương thức commit()
frTransaction.commit()
```

- Ví dụ, để thực hiện thêm một Fragment mới đơn giản ta chỉ cần chỉ định Fragment cần thêm và đối tượng sẽ chứa Fragment này.

```
FragmentTransaction frTransaction = frManager.beginTransaction();
frTransaction.add(R.id.ui_container, new MyFragment());
frTransaction.commit()
```

- Đối với các phương thức remove và replace ta cũng thực hiện tương tự.
- Để thực hiện các chuyển hoạt cho các Fragment trong một Transaction ta có 2 cách, một là dùng phương thức setTransaction và truyền vào các tham số có sẵn trong lớp FragmentTransaction. Ví dụ:

```
frTransaction.setTransition(FragmentTransaction.TRANSIT_FRAGMENT_OPEN);
```

- Cách thứ hai là chúng ta sẽ sử dụng phương thức setCustomAnimations và thực hiện tạo các chuyển hoạt ở các tập tin XML . Phương thức yêu cầu truyền vào hai tham số, một dành cho khi Fragment được thêm vào và một cho Fragment được gỡ bỏ. Ví dụ:

```
frTransaction.setCustomAnimations(android.R.anim.fade_in,
android.R.anim.fade_out);
```

- Trong lớp Fragment hỗ trợ phương thức getActivity cho phép chúng có thể truy cập vào các đối tượng của Activity mà chúng được nhúng vào. Điều này giúp chúng ta có thể lấy về các View mà ta cần tương tác ngay trong lớp Fragment. Ví dụ:

```
TextView textView = (TextView) getActivity().findViewById(R.id.fragEdit);
```

- Mặc dù chúng ta vẫn có dùng lớp FragmentManager để liên lạc giữa Fragment với Activity nhưng cách tốt hơn là xây dựng một Interface mà ở đó Activity như một thành phần trung gian và các Fragment sẽ không phụ thuộc vào Activity mà nó được nhúng vào. Thường thì chúng ta sẽ thực hiện tạo kết nối Interface trong phương thức onAttach của Fragment. Có thể xem ví dụ sau để hiểu thêm.

- o Xây dựng Interface

```
public interface OnNewItemAddedListener{  
    public void OnNewItemAdded(String newItem);  
}
```

- o Thực hiện lấy Activity làm trung gian trong phương thức onAttach.

```
@Override  
public void onAttach(Activity activity){  
    super.onAttach(activity);  
    try{  
        onNewItemAddedListener = (OnNewItemAddedListener) activity;  
    }catch(ClassCastException e){  
        Throw new ClassCastExeption(activity.toString() + " must  
implement  
        onNewItemAddedListener.");  
    }  
}
```

- o Thực thi Interface trong Activity

```
@Override  
public void onNewItemAdded(String newItem){  
    //Do Something with newItem  
}
```

Mục lục

Bài 1: Tổng quan về Lập trình Android và môi trường phát triển	1
1. Tổng quan Android	1
2. Kiến trúc Android	8
3. Môi trường phát triển ứng dụng Android	9
Bài 2: Các thành phần ứng dụng Android	29
1. Các thành phần trong ứng dụng Android	29
2. Tương thích thiết bị	32
3. Activity và độ ưu tiên ứng dụng	32
Bài 3: Giao diện người dùng và xử lý sự kiện	37
1. Giao diện người dùng	37
2. View và Viewgroup	42
3. Các điều khiển cơ bản.....	46
4. Kiểm tra lỗi và tối ưu lại giao diện	51
Bài 4: Tài nguyên ứng dụng trong Android.....	53
1. Tổng quan	53
2. Các tài nguyên ứng dụng cơ bản.....	58
3. Các tài nguyên ứng dụng nâng cao	62
Bài 5: Intent	71
1. Khái niệm về Intent.....	71
2. Intent filter	74
Bài 6: Quản lý Asset – Shared Preferences – Bộ nhớ thiết bị.....	77
1. Bộ quản lý Asset	77

2. Shared Preferences.....	80
3. Bộ nhớ thiết bị	82
Bài 7: Các điều khiển hiển thị danh sách	85
1. Khái niệm cơ bản	85
2. Các dạng Adapter.....	85
3. Các điều khiển danh sách	86
Bài 8: Xây dựng giao diện với Fragment – Vấn đề Activity và Fragment	100
1. Các khái niệm cơ bản.....	100
2. Xây dựng và sử dụng Fragment.....	104