

Giáo trình Android

Hồ Thị Thảo Trang

Published
with GitBook



Table of Contents

1. Giới thiệu
2. Những kiến thức cơ bản về thiết bị di động và lập trình cho thiết bị di động
 - i. Các thiết bị di động
 - ii. Tổng quan về lập trình cho thiết bị di động
3. Nhập môn lập trình Android
 - i. Thiết bị Android - hệ điều hành và máy ảo Dalvik
 - ii. Lập trình cho thiết bị Android
4. Các Activity, Fragment và Intent
 - i. Activity
 - ii. Intent và việc tương tác giữa các Activity
 - iii. Fragment
5. Giao diện người dùng của ứng dụng Android
 - i. View và ViewGroup
 - ii. Bố cục giao diện thích nghi với hướng màn hình
 - iii. Sử dụng trình đơn (Menu)
 - iv. Sử dụng thanh tác vụ (ActionBar)
 - v. Xử lý sự kiện tương tác với các thành phần đồ họa
6. Thiết kế giao diện người dùng với các View cơ bản
 - i. Sử dụng các View cơ bản trong Android
 - ii. TimePicker và DatePicker
 - iii. Hiển thị ảnh với ImageView và Gallery
 - iv. Sử dụng ListView để hiển thị danh sách dài
 - v. Hiển thị nội dung trang web với WebView
7. Lưu trữ dữ liệu
 - i. Lưu trữ dữ liệu cố định với shared preferences
 - ii. Lưu trữ dữ liệu với file trên bộ nhớ trong và bộ nhớ ngoài
 - iii. CSDL SQLite trong ứng dụng Android
8. Lập trình mạng với Android
 - i. Sử dụng web services thông qua giao thức HTTP
 - ii. Tải dữ liệu nhị phân thông qua HTTP
 - iii. Tải dữ liệu dạng text thông qua HTTP
 - iv. Web service với dữ liệu XML
 - v. Web service với dữ liệu JSON
9. Google Play Store và việc phân phối ứng dụng
 - i. Chuẩn bị ứng dụng trước khi phân phối
 - ii. Phân phối ứng dụng

Giáo trình Android

Giáo trình dùng cho sinh viên khoa Công Nghệ Thông Tin, đại học Mỏ Địa Chất, Hà Nội

Hồ Thị Thảo Trang thaotrang1011@gmail.com

Bộ môn **Công Nghệ Phần Mềm**

Khoa **Công Nghệ Thông Tin**

Đại học **Mỏ Địa Chất**

Những kiến thức cơ bản về thiết bị di động và lập trình cho thiết bị di động

Các thiết bị di động

Phân loại các thiết bị di động

Các thiết bị di động đã trải qua rất nhiều năm phát triển với rất nhiều loại thiết bị khác nhau, có thể kể đến như máy nhắn tin di động, điện thoại di động, thiết bị trợ giúp cá nhân (PDA, Palm...), điện thoại thông minh, máy tính bảng... Các thiết bị nghe nhìn khác như máy ảnh, máy quay kỹ thuật số, máy nghe nhạc... cũng có thể được xếp vào "thiết bị di động". Giáo trình này sẽ không đề cập đến các thiết bị đã là lịch sử (không còn hoặc gần như không còn được sử dụng trong thực tế) và các thiết bị nghe nhìn và chỉ đề cập đến các loại thiết bị điện toán cầm tay hiện đang phổ biến trên thị trường tiêu dùng. Các thiết bị này thường được phân thành các loại như sau:

Điện thoại di động cơ bản (basic phones và featured phones) - là các điện thoại di động với các tính năng cơ bản như nghe, gọi, danh bạ... và một số ứng dụng dựng sẵn đơn giản. Các thiết bị này thường có kích thước nhỏ, màn hình độ phân giải thấp, có hoặc không có bàn phím, pin dùng được lâu, ít khả năng kết nối, khả năng phát triển cũng như cài đặt thêm phần mềm của nhà phát triển là (gần như) không có.

Điện thoại di động thông minh (smartphones) - là các điện thoại được trang bị cấu hình tốt hơn, chạy hệ điều hành thông minh với các bộ công cụ phát triển phần mềm (Software Development Kit - SDK) cho phép lập trình viên phát triển đa dạng các ứng dụng phục vụ mọi mục đích của cuộc sống và công việc. Các thiết bị này thường có kích thước và màn hình lớn hơn nhiều so với featured phones, cấu hình phần cứng (CPU, RAM, GPU, camera...) cao, đa dạng các kết nối (Wifi, Bluetooth, 3G/4G, GPS, Glonass, NFC...), có thể có một số loại cảm biến (cảm biến gia tốc, la bàn, cảm biến tiệm cận, cảm biến ánh sáng, con quay hồi chuyển, cảm biến áp suất...). Với tính năng đa dạng như vậy nên dù thường được trang bị thời pin lớn hơn so với các máy điện thoại cơ bản, thời lượng pin của điện thoại thông minh thường hạn chế hơn so với featured phones.

Máy tính bảng (tablets) - là các thiết bị thông minh, tương tự như smartphones nhưng có kích thước màn hình lớn hơn rất nhiều (thông thường từ 7"-13"), có thể có khe cắm thẻ SIM (phục vụ việc nhắn tin, gọi điện hoặc truy cập Internet qua mạng 3G).

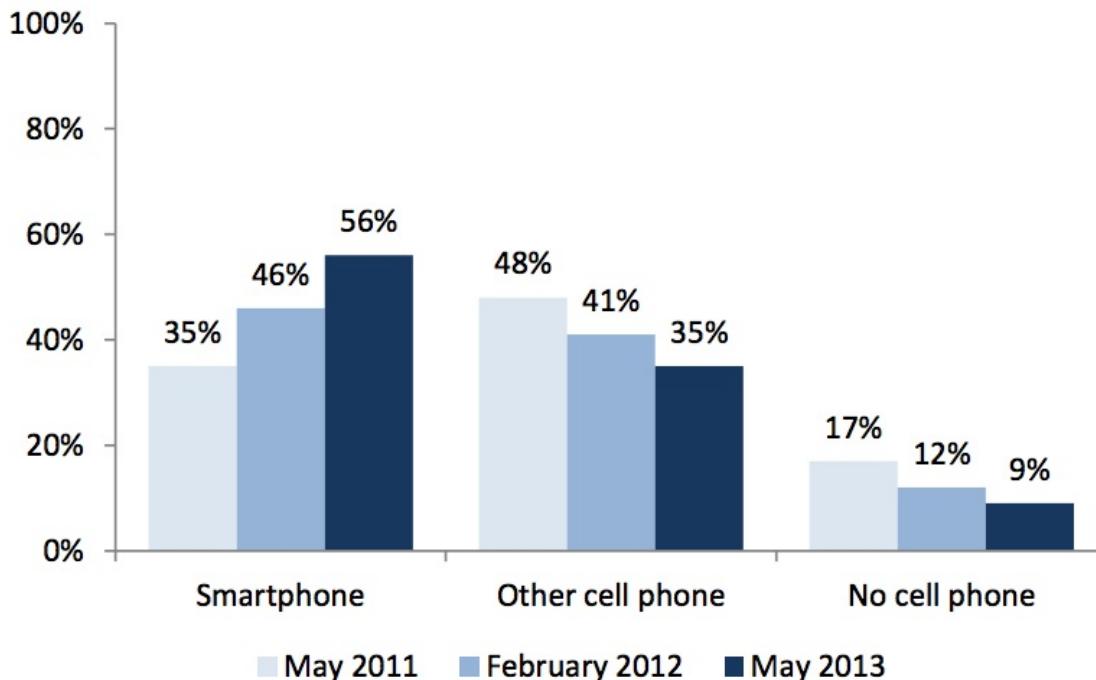
Điện thoại thông minh lai máy tính bảng (phablet) - là loại thiết bị lai giữa smartphone và máy tính bảng, về tính năng nó là một smartphone, nhưng được trang bị màn hình cỡ lớn hơn smartphone thông thường và nhỏ hơn kích thước phổ biến của màn hình tablet. Màn hình của phablet thường có kích thước 5.0"-6.9". Trong lập trình, các phablet thường được xếp cùng với các smartphones.

Do khả năng lập trình cho các điện thoại cơ bản rất hạn chế (thường phải làm việc với mức thấp hơn, không có bộ công cụ phát triển tiện dụng) nên mặc dù điện thoại cơ bản vẫn chiếm thị phần lớn, giáo trình chỉ tập trung vào việc phát triển ứng dụng cho các thiết bị thông minh (điện thoại thông minh, máy tính bảng và điện thoại lai). Về mặt phần mềm, các thiết bị này hầu như giống nhau nên có thể gọi chung là điện thoại thông minh.

Biểu đồ dưới đây thể hiện tỷ lệ thị phần giữa Smart Phones và Featured Phones trong 3 năm gần đây, dữ liệu khảo sát trên tập người trưởng thành tại Hoa Kỳ (theo số liệu của PewSearchCenter, tháng 6 năm 2013):

Changes in smartphone ownership, 2011–2013

% of all U.S. adults who own...



Hình: Thị phần các loại điện thoại tại Hoa Kỳ 2011-2013

Các hệ điều hành thiết bị di động thông minh

Những hệ điều hành di động phổ biến nhất hiện nay gồm có:

Android

Android là hệ điều hành miễn phí, mã nguồn mở, được phát triển bởi "gã khổng lồ" [Google](#). Android dựa trên nền tảng Linux, được thiết kế dành cho các thiết bị di động có màn hình cảm ứng như điện thoại thông minh và máy tính bảng. Phiên bản mới nhất của Android có hỗ trợ thêm các thiết bị khác như ô-tô, đồng hồ thông minh và TV. Ban đầu, Android được phát triển bởi Android Corporation, với sự hỗ trợ tài chính từ Google và sau này được chính Google mua lại vào năm 2005. Android ra mắt vào năm 2007 cùng với tuyên bố thành lập Liên minh thiết bị cầm tay mở (Open Handset Alliance - <http://www.openhandsetalliance.com/>), một hiệp hội gồm các công ty phần cứng, phần mềm, và viễn thông với mục tiêu đẩy mạnh các tiêu chuẩn mở cho các thiết bị di động. Chiếc điện thoại đầu tiên chạy Android được bán vào tháng 10 năm 2008.

Android có *mã nguồn mở* và Google đã phát hành mã nguồn theo Giấy phép Apache. Chính mã nguồn mở cùng với một giấy phép không có nhiều ràng buộc đã cho phép các nhà phát triển thiết bị, mạng di động và các lập trình viên nhiệt huyết được tự biến và phân phối Android một cách tự do. Ngoài ra, Android còn có một cộng đồng lập trình viên đông đảo, chuyên viết các ứng dụng để mở rộng chức năng của thiết bị bằng ngôn ngữ lập trình Java có sửa đổi. Vào tháng 7 năm 2013, có hơn một triệu ứng dụng cho thiết bị Android được công bố, và số lượt tải ứng dụng từ Google Play, cửa hàng ứng dụng chính của Android, ước tính đạt khoảng 50 tỷ. Một cuộc khảo sát các nhà phát triển vào tháng 4 - 5 năm 2013 cho thấy hơn 71% lập trình viên cho di động đã phát triển ứng dụng cho Android. Tại sự kiện Google I/O 2014 (<https://www.google.com/events/io>), "gã khổng lồ" công bố đã có hơn một tỉ người dùng Android hàng tháng, trong khi con số này vào tháng 6 năm 2013 mới chỉ là 538 triệu.

Những yếu tố trên đã giúp Android trở thành nền tảng điện thoại thông minh phổ biến nhất thế giới, vượt qua Symbian vào quý 4 năm 2010 (chiếm 33%), và được các công ty công nghệ lựa chọn khi họ cần một hệ điều hành không nặng nề, có

khả năng tinh chỉnh, giá rẻ để chạy trên các thiết bị công nghệ cao thay vì xây dựng từ đầu. Kết quả là mặc dù được thiết kế để chạy trên điện thoại và máy tính bảng, Android đã xuất hiện trên TV, máy chơi game, đồng hồ thông minh, ô-tô và các thiết bị điện tử khác. Bản chất mở của Android cũng khích lệ một đội ngũ đông đảo lập trình viên và những người đam mê sử dụng mã nguồn mở tạo ra những dự án do cộng đồng quản lý. Những dự án này bổ sung các tính năng cao cấp cho những người dùng thích tìm tòi hoặc đưa Android vào các thiết bị ban đầu chạy hệ điều hành khác.

Android chiếm 75% thị phần điện thoại thông minh trên toàn thế giới vào thời điểm quý 3 năm 2012, với tổng cộng 750 triệu thiết bị đã được kích hoạt. Vào tháng 4 năm 2013 đã có 1,5 triệu lượt kích hoạt mỗi ngày và tổng số thiết bị Android được kích hoạt lên đến 1 tỷ vào tháng 9 năm 2013. Sự thành công của hệ điều hành này cũng khiến nó trở thành mục tiêu trong các vụ kiện liên quan đến bằng phát minh, góp mặt trong cái gọi là "*cuộc chiến điện thoại thông minh*" giữa các công ty công nghệ.

iOS

iOS là hệ điều hành trên các thiết bị di động của **Apple**. Ban đầu hệ điều hành này chỉ được phát triển để chạy trên iPhone (gọi là iPhone OS), nhưng sau đó nó đã được mở rộng để chạy trên các thiết bị của Apple như iPod touch, iPad và Apple TV. Ngày 31 tháng 5, 2011, App Store của Apple chứa khoảng 500 000 ứng dụng chạy trên iOS, và được tải về tổng cộng khoảng 15 tỷ lượt. Trong quý 4 năm 2010, có khoảng 26% điện thoại thông minh chạy hệ điều hành iOS, đứng sau hệ điều hành Android của Google và Symbian của Nokia.

Giao diện người dùng của iOS dựa trên cơ sở thao tác bằng tay. Người dùng có thể tương tác với hệ điều hành này thông qua rất nhiều động tác bằng tay trên màn hình cảm ứng của các thiết bị của Apple.

Phiên bản mới nhất là iOS 8.1 (ra ngày 20/10/2014) dành riêng cho các thiết bị iPhone, iPad, iPad Mini và iPod Touch và iOS 7.0.1 (ra ngày 20/10/2014) cho các thiết bị Apple TV thế hệ 3.

Blackberry

BlackBerry OS là nền tảng phần mềm tư hữu do RIM (Research In Motion) phát triển cho dòng sản phẩm cầm tay BlackBerry. BlackBerry OS cung cấp khả năng đa nhiệm và được thiết kế cho các thiết bị sử dụng phương pháp nhập thông tin đặc biệt, thường là bì điều khiển trackball hoặc màn hình cảm ứng. Hệ điều hành này hỗ trợ MIDP 1.0 và WAP 1.2. Các phiên bản trước đó cho phép đồng bộ hóa không dây thư điện tử và lịch với Microsoft Exchange Server, và với cả Lotus Domino. Phiên bản OS 4 hiện tại hỗ trợ MIDP 2.0, có khả năng kích hoạt không dây hoàn toàn và đồng bộ thư điện tử, lịch công việc, ghi chú và danh bạ với Exchange, và hỗ trợ Novell GroupWise, Lotus Notes khi kết hợp với BlackBerry Enterprise Server.

Các bản cập nhật cho BlackBerry OS có thể được tải về thông qua dịch vụ BlackBerry OTASL.

Các bên thứ ba có thể phát triển ứng dụng sử dụng các API tư hữu của BlackBerry, nhưng bất kỳ ứng dụng nào sử dụng các chức năng này đều cần phải chứng thực trước khi cài đặt. Việc chứng thực này xác nhận tác giả của phần mềm, nhưng không bảo đảm tính an toàn và bảo mật của ứng dụng.

BlackBerry 10

Là thế hệ tiếp theo của hệ điều hành BlackBerry OS, được phát triển bởi BlackBerry Limited (Research In Motion đổi tên), dành cho cả điện thoại lẫn máy tính bảng. Thiết bị gần đây nhất sử dụng hệ điều hành là smartphone cao cấp BlackBerry Q10.

Windows phone

Windows Phone là hệ điều hành của Microsoft dành cho smartphone kế tục nền tảng Windows Mobile, mặc dù chúng không tương thích với nhau. Khác với Windows Mobile, Windows Phone tập trung vào sự phát triển của Marketplace - nơi các nhà phát triển có thể cung cấp sản phẩm (miễn phí hoặc có phí) tới người dùng. Windows Phone được bán vào tháng 10 năm 2010 và đầu năm 2011 tại Châu Á.

Phiên bản mới nhất hiện tại là Windows Phone 8. Microsoft còn đang phát triển bản Windows Phone Apollo Plus, và trong tương lai có thể còn có Windows Blue (hay có thể là Windows 9) giúp tương thích với hệ điều hành Windows trên máy tính. Với Windows Phone, Microsoft đã phát triển giao diện người dùng mới mang tên Modern (trước đây gọi là Metro) - tích hợp khả năng liên kết với các phần cứng và phần mềm của hãng thứ ba một cách dễ dàng.

Ngày 11 tháng 2 năm 2011, trước mặt báo giới, CEO Microsoft Steve Ballmer và CEO Nokia Stephen Elop công bố trở thành đối tác của nhau, đồng nghĩa với việc Windows Phone trở thành hệ điều hành chính của Nokia, thay thế Symbian đã già cỗi. Sự kiện này cũng đánh dấu một mốc quan trọng trong cuộc chiến với Android và iOS, được ví như là "cuộc đua giữa 3 con ngựa". Theo đó, công ty này sẽ hợp tác với Microsoft trong việc sản xuất những điện thoại Windows Phone 7 (hiện tại là Windows Phone 8). Nokia hứa hẹn sẽ:

- **Tập trung vào Windows Phone 7/8:** Đưa ra những thiết kế mới, bổ sung những gói ngôn ngữ và phổ biến chúng nhiều hơn cho người tiêu dùng thông qua những thiết kế mới về phần cứng, nhiều phân khúc giá và thị trường hơn
- Hợp tác trong lĩnh vực marketing, phát triển phần mềm cho điện thoại di động. Bing sẽ trở thành nền tảng tìm kiếm trong các thiết bị và dịch vụ của Nokia. Kho ứng dụng riêng của Nokia sẽ được tích hợp chung với Marketplace.

Ngoài các hệ điều hành trên, trên thị trường hiện nay còn có các thiết bị chạy các hệ điều hành khác với thi phần không đáng kể như: **Bada** (của Samsung), **BlackBerry Tablet OS** (cho máy tính bảng BlackBerry PlayBook), **GridOS** (do Fusion Garage phát triển dựa trên Android), **Linux, Brew** (của Qualcomm), **webOS** (của Palm, sau HP mua lại, rồi lại bán lại cho LG hồi tháng 2 năm 2013), **Tizen** (do Samsung và Intel phối hợp hỗ trợ, dựa trên **LiMo** - Linux for Mobile), **Windows RT** (của Microsoft cho các thiết bị sử dụng chip kiến trúc ARM). Một số tổ chức, công ty cũng đang nỗ lực phát triển các hệ điều hành di động mới, được nhắc đến nhiều nhất trong số đó có thể kể đến:

Aliyun OS

Aliyun OS ra đời tháng 7 năm 2011, là hệ điều hành dựa trên Linux, được phát triển bởi AliCloud, một công ty con của Alibaba Group, Trung Quốc. Ý tưởng chung của hệ điều hành Aliyun là "đám mây hóa" các tính năng của thiết bị di động (cloud functionality). Theo Google, Aliyun được phát triển từ hệ điều hành mã nguồn mở Android của mình, Alibaba thì phủ nhận điều này, tuy nhiên hệ điều hành này có thể chạy được hầu hết các ứng dụng của Android. Trên chợ ứng dụng của Aliyun thậm chí còn chứa rất nhiều ứng dụng Android vi phạm bản quyền. Thiết bị đầu tiên chạy hệ điều hành này là điện thoại K-Touch W700.

FireFox OS

Là hệ điều hành cho điện thoại di động và máy tính bảng, được phát triển dựa trên nền Linux, được phát triển bởi tổ chức phi lợi nhuận Mozilla Foundation (tổ chức làm ra trình duyệt FireFox nổi tiếng). Được thiết kế để cung cấp một hệ thống toàn diện cho thiết bị di động, sử dụng các công nghệ mở và phổ biến như HTML5, Javascript, web API... với khả năng truy cập trực tiếp vào phần cứng thiết bị, FireFox OS nhằm đến cạnh tranh với các ông lớn khác như Apple's iOS, Google's Android, Microsoft's Windows Phone, cũng như các hệ điều hành mới xuất hiện như Ubuntu Touch OS. FireFox OS được giới thiệu tháng 2 năm 2012 trên vài điện thoại chạy Android, và trên bộ kit Raspberry Pi vào năm 2013. Tháng 1 năm 2013, tại triển lãm thiết bị điện tử tiêu dùng quốc tế CES 2013, hãng ZTE xác nhận sẽ sản xuất điện thoại chạy hệ điều hành này và đến 2 tháng 7 năm 2013, Telefónica chính thức giới thiệu điện thoại thương mại đầu tiên chạy FireFox OS tại Tây Ban Nha: điện thoại ZTE Open.

Ubuntu Touch OS

Là giao diện di động của hệ điều hành mã nguồn mở nổi tiếng - Ubuntu của Canonical Ltd., được thiết kế dành cho các thiết bị di động với màn hình cảm ứng như điện thoại thông minh và máy tính bảng. Điểm mạnh của Ubuntu Touch là việc dựa trên cùng một công nghệ lõi như hệ điều hành Ubuntu cho máy tính, giúp các ứng dụng dành cho Ubuntu và Ubuntu Touch có thể chạy lẫn nhau mà không cần phát triển lại phiên bản riêng. Ngoài ra, thiết bị chạy Ubuntu Touch có thể biến thành Ubuntu bản cho máy tính với đầy đủ tính năng của máy tính cá nhân khi kết nối vào màn hình ngoài hoặc kết nối qua bộ đế (dock). Với sự phổ biến của hệ điều hành Ubuntu và khả năng tương thích của Ubuntu Touch, đây có thể sẽ trở thành một trong những hệ điều hành phổ biến trong tương lai.

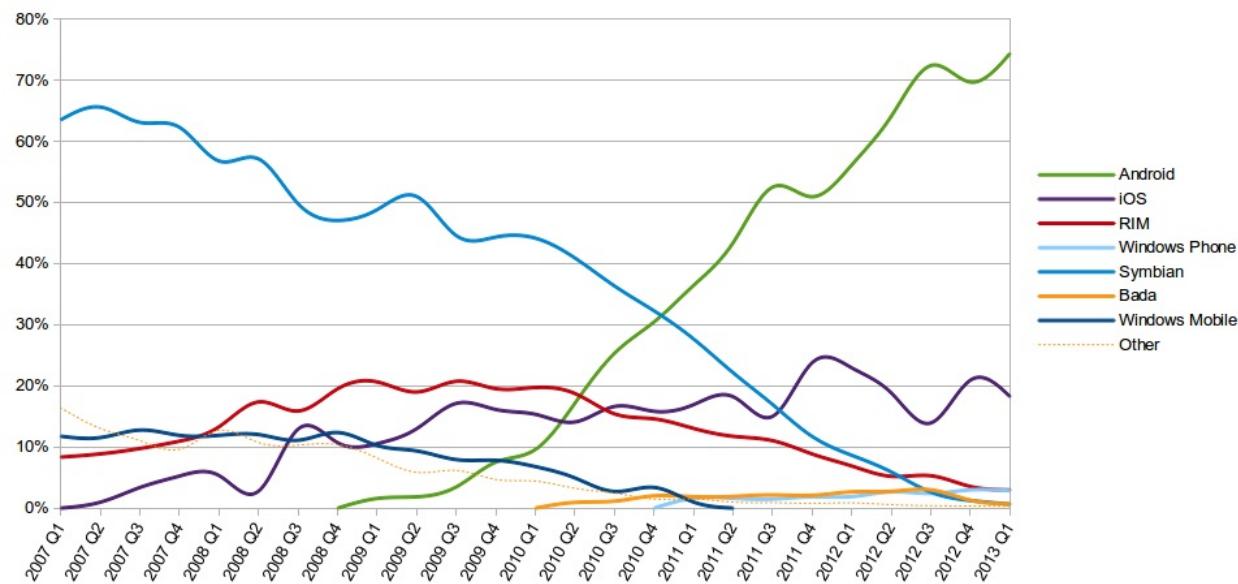
Bảng dưới đây thể hiện tương quan thị phần giữa các hệ điều hành di động thông minh phổ biến từ năm 2012-2013, theo số liệu ngày 16/05/2013 của IDC Worldwide Quarterly Mobile Phone Tracker - <http://www.idc.com/prodserv/smartphone-os-market-share.jsp> (đơn vị: triệu thiết bị).

HĐH	Số lượng Q1-2013	Thị phần Q1-2013	Số lượng Q1-2012	Thị phần Q1-2012	Tỷ lệ tăng trưởng
Android	162.1	75.00%	90.3	59.10%	79.50%
iOS	37.4	17.30%	35.1	23.00%	6.60%
Windows Phone	7	3.20%	3	2.00%	133.30%
BlackBerry OS	6.3	2.90%	9.7	6.40%	-35.10%
Linux	2.1	1.00%	3.6	2.40%	-41.70%
Symbian	1.2	0.60%	10.4	6.80%	-88.50%
Others	0.1	0.00%	0.6	0.40%	-83.30%
Total	216.2	100.00%	152.7	100.00%	41.60%

Bảng: Thị phần giữa các hệ điều hành di động thông minh phổ biến từ năm 2012-2013

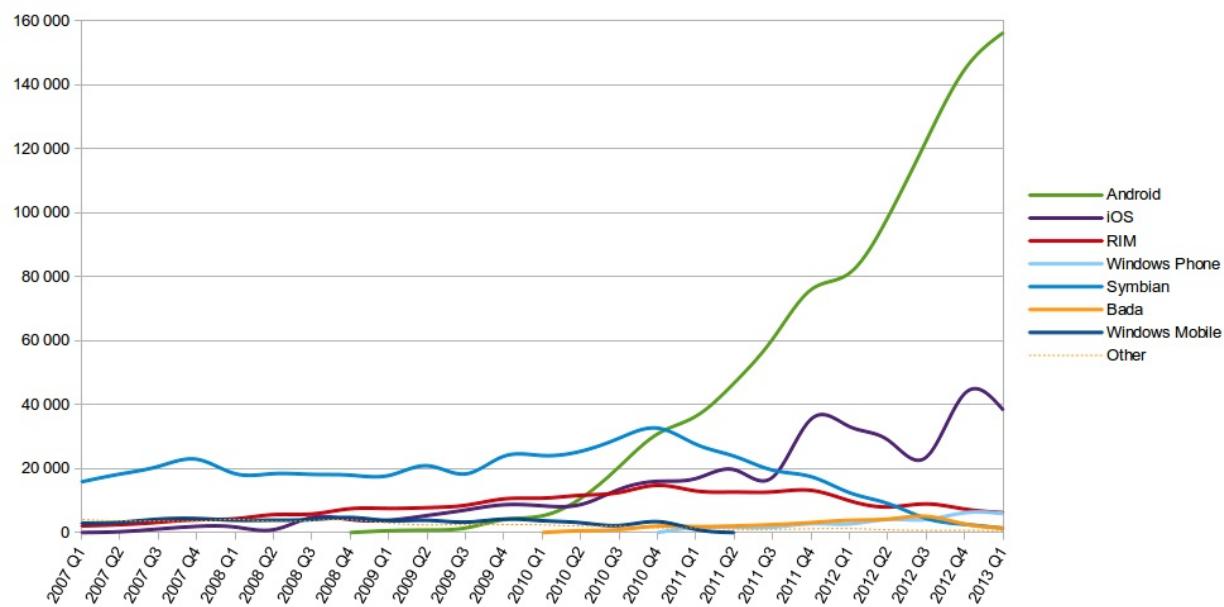
Biểu đồ dưới đây cho thấy sự thay đổi về sản lượng cũng như thị phần của các hệ điều hành di động từ năm 2007 đến nay (nguồn [wikipedia.org](http://en.wikipedia.org))

World-Wide Smartphone Sales (%)



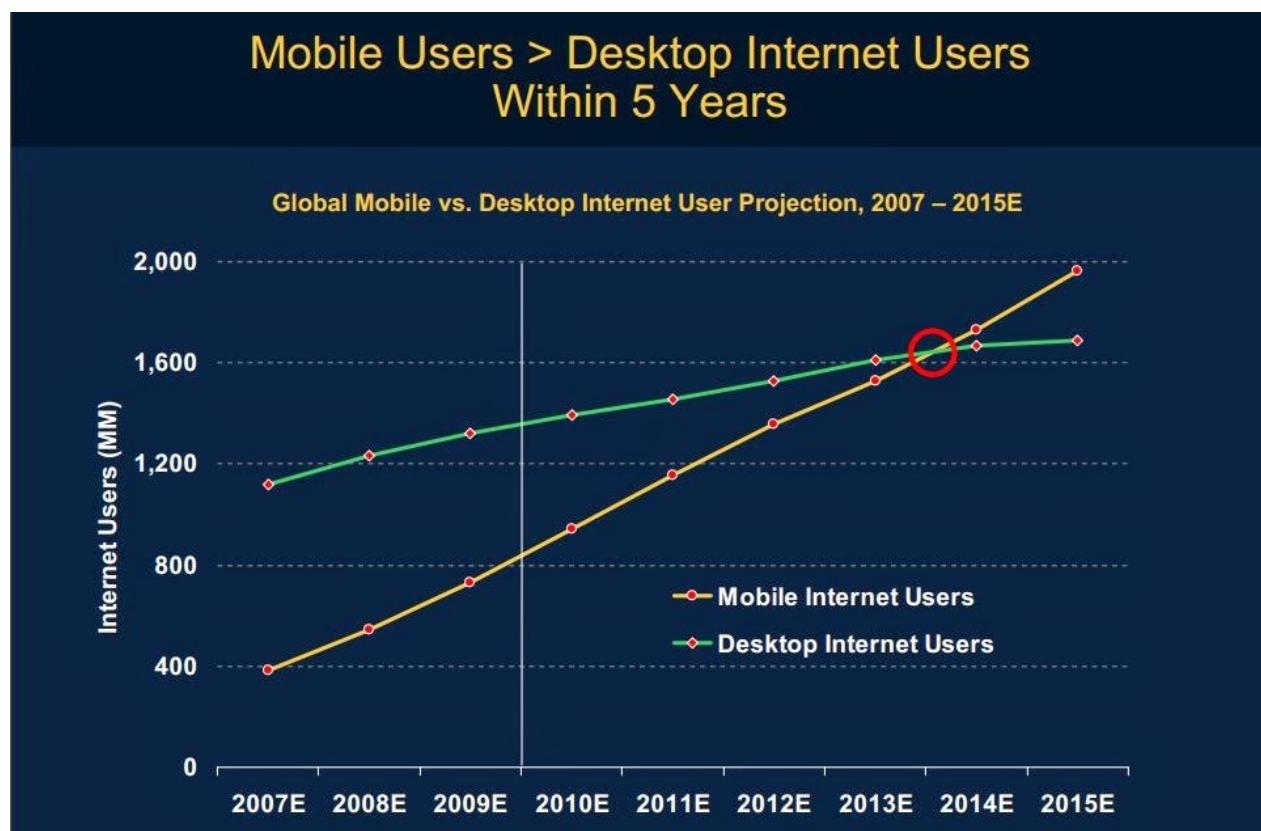
Hình: Thị phần (phần trăm) điện thoại toàn cầu theo hệ điều hành

World-Wide Smartphone Sales (Thousands of Units)



Hình: Thị phần (sản lượng) điện thoại toàn cầu theo hệ điều hành

Xu hướng di động hóa



Tổng quan về lập trình cho thiết bị di động

Người lập trình ứng dụng cho thiết bị di động truyền thống luôn luôn phải nhớ trong đầu nguyên tắc "tiết kiệm tối đa tài nguyên" của thiết bị, dùng mọi cách để tối ưu hóa độ phức tạp tính toán cũng như lượng bộ nhớ cần sử dụng. Tuy nhiên, cùng với sự phát triển nhanh chóng của phần cứng, các thiết bị di động hiện đại thường có cấu hình rất tốt, với chip xử lý mạnh mẽ, bộ nhớ (RAM) lớn, khiến việc lập trình cho thiết bị di động trở nên dễ dàng hơn bao giờ hết. Các bộ kit phát triển của các hãng sản xuất hệ điều hành di động hiện nay cũng thường làm trong suốt hầu hết các tác vụ liên quan đến quản lý bộ nhớ, quản lý tiến trình... Lập trình viên có thể ít quan tâm hơn đến việc tối ưu hóa sử dụng tài nguyên và tập trung vào việc "*lập trình*", phát triển tính năng cho ứng dụng như khi lập trình cho máy tính cá nhân.

Tuy nhiên, đặc trưng di động của các thiết bị này cũng đem đến nhiều vấn đề mà người lập trình cần phải quan tâm như:

- Tính "*di động*" của các thiết bị này khiến kết nối mạng trở nên rất bất ổn định và khó kiểm soát. Các ứng dụng phụ thuộc nhiều vào kết nối Internet cần chú ý điểm này.
- Lưu lượng Internet trên thiết bị di động thường có chi phí cao hơn so với Internet cố định. Điều này cũng cần lưu ý khi phát triển các ứng dụng sử dụng nhiều tài nguyên Internet như phim trực tuyến, nhạc trực tuyến...
- So với máy tính cá nhân, các thiết bị di động hiện đại được trang bị thêm rất nhiều tính năng giúp việc tương tác với người dùng trở nên thuận tiện (màn hình cảm ứng đa điểm, tương tác giọng nói, cử chỉ...), các loại kết nối đa dạng (NFC, GPS, 3G, 4G, bluetooth, IR...), các cảm biến phong phú giúp trải nghiệm rất đa dạng (cảm biến ánh sáng, cảm biến tiệm cận, la bàn, cảm biến chuyển động, gia tốc kế...). Người lập trình, tùy thuộc vào ứng dụng cụ thể, có thể sử dụng đến các tính năng đặc biệt này để đem đến cho người dùng trải nghiệm tốt nhất trên thiết bị di động của mình.

Ngoài ra các hãng phát triển hệ điều hành di động đều làm ra bộ công cụ phát triển (SDK) và môi trường phát triển tích hợp (IDE) rất thuận tiện cho việc viết mã nguồn, biên dịch, gỡ rối, kiểm thử cũng như xuất bản phần mềm.

Xét theo thị phần trên thị trường, ba hệ điều hành phổ biến nhất cho thiết bị di động hiện nay là Google's Android, Apple's iOS và Microsoft's Windows Phone. Mỗi ứng dụng thành công thường được phát triển cho cả 3 hệ nền này. Mỗi hệ nền đều có một chợ ứng dụng chính hãng (Google có Google Play Store, Apple có AppStore, Microsoft có Windows Phone Store) với rất nhiều khách hàng tiềm năng, giúp người phát triển có thể phân phối ứng dụng miễn phí hoặc có phí với chi phí nhất định.

Bảng dưới đây liệt kê các hệ điều hành cùng với ngôn ngữ lập trình và IDE phổ biến nhất của nó.

HĐH	Ngôn ngữ lập trình	IDE
Android	Java	IBM's Eclipse với Google's ADT plugins
iOS	Objective-C	Apple's Xcode
Windows Phone	C#	Microsoft's Visual Studio cho Windows phone

Ngoài việc phát triển ứng dụng cho từng hệ điều hành như kể trên, lập trình viên có thể lựa chọn các thư viện lập trình đa nền tảng để phát triển ứng dụng, phổ biến nhất trong các ứng dụng đa nền là các ứng dụng viết bằng ngôn ngữ Web (HTML5, CSS & Javascript). Trình duyệt web của các thiết bị di động hiện nay có đầy đủ tính năng lẫm hiệu năng để chạy tốt các ứng dụng web hiện đại. Một ứng dụng Web có thể được đặt trên máy chủ hoặc được đóng gói thành native app (ứng dụng cho từng hệ điều hành) qua một số công cụ đóng gói của các hãng thứ ba.

Công cụ đóng gói ứng dụng Web cho thiết bị di động phổ biến nhất hiện nay là PhoneGap (<http://phonegap.com/>), được phát triển bởi Nitobi, sau được Adobe mua lại. PhoneGap cho phép lập trình viên phát triển ứng dụng di động sử dụng ngôn ngữ Web phổ biến (HTML5, CSS3 và Javascript), với các tính năng bổ sung, cho phép ứng dụng truy cập vào phần cứng của thiết bị như gia tốc kế, máy ảnh, GPS... và đóng gói thành ứng dụng cho nhiều hệ điều hành khác nhau, bao gồm Android, iOS, Blackberry, BlackBerry 10, Windows Phone, Windows 8, Tizen, Bada. Tuy nhiên, nhược điểm của các ứng dụng loại này là hiệu suất thấp (chạy không được "mượt mà" như ứng dụng native) và không đồng nhất giữa tất cả các trình duyệt Web di động (có thể chạy hoặc hiển thị khác nhau trên các hệ điều hành với các trình duyệt khác nhau).

Việc chọn hệ điều hành/thư viện nào để phát triển tùy thuộc vào nhiều yếu tố khác nhau như mục đích của ứng dụng, đối tượng sử dụng, tiềm năng của hệ điều hành, các yêu cầu kỹ thuật cụ thể cũng như thói quen và kỹ năng của lập trình viên. Trong khuôn khổ giáo trình này, chúng tôi tập trung đi sâu vào việc phát triển ứng dụng cho hệ điều hành di động phổ biến nhất hiện nay - **Google's Android**.

Nhập môn lập trình Android

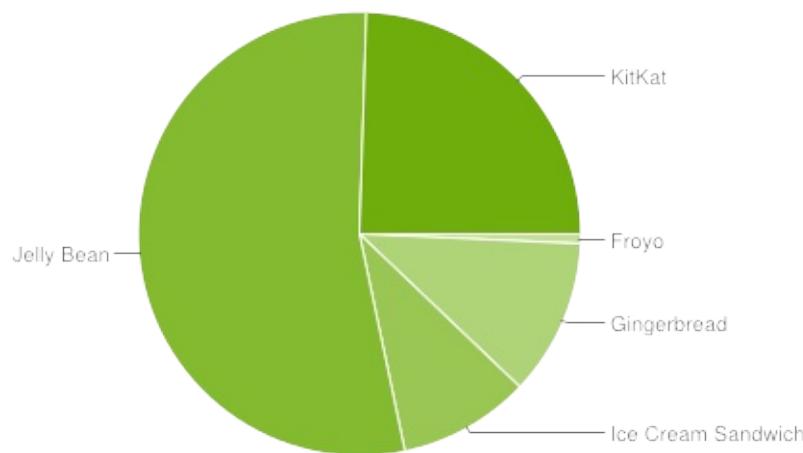
Thiết bị Android - hệ điều hành và máy ảo Dalvik

Ngày 5 tháng 11 năm 2007, Liên minh thiết bị cầm tay mở (Open Handset Alliance), một hiệp hội bao gồm nhiều công ty trong đó có Texas Instruments, Tập đoàn Broadcom, Google, HTC, Intel, LG, Tập đoàn Marvell Technology, Motorola, Nvidia, Qualcomm, Samsung Electronics, Sprint Nextel và T-Mobile được thành lập với mục đích phát triển các chuẩn mở cho thiết bị di động. Cùng ngày, Android cũng được ra mắt với vai trò là sản phẩm đầu tiên của Liên minh, một nền tảng thiết bị di động được xây dựng trên nhân Linux phiên bản 2.6. Chiếc điện thoại chạy Android đầu tiên được bán ra là HTC Dream, phát hành ngày 22 tháng 10 năm 2008. Biểu trưng của hệ điều hành Android mới là một con rô-bốt màu xanh lá cây do hãng thiết kế Irina Blok tại California vẽ.

Từ năm 2008, Android đã trải qua nhiều lần cập nhật để dần dần cải tiến hệ điều hành, bổ sung các tính năng mới và sửa các lỗi trong những lần phát hành trước. Mỗi bản nâng cấp được đặt tên lần lượt theo thứ tự bảng chữ cái, theo tên của một món ăn tráng miệng; ví dụ như phiên bản 1.5 Cupcake (bánh bông lan nhỏ có kem) tiếp nối bằng phiên bản 1.6 Donut (bánh vòng). Phiên bản mới nhất cho đến thời điểm cuối năm 2014 là **Android 5.0 Lollipop** (kẹo mút). Vào năm 2010, Google ra mắt loạt thiết bị Nexus—một dòng sản phẩm bao gồm điện thoại thông minh và máy tính bảng chạy hệ điều hành Android, do các đối tác phần cứng sản xuất. HTC đã hợp tác với Google trong chiếc điện thoại thông minh Nexus đầu tiên, Nexus One. Kể từ đó nhiều thiết bị mới hơn đã gia nhập vào dòng sản phẩm này, như điện thoại Nexus 4 và máy tính bảng Nexus 10, lần lượt do LG và Samsung sản xuất, kế tiếp là Nexus 5 cũng do LG sản xuất, và mới nhất là Nexus 6 (Motorola) và Nexus 9 (HTC) vừa được ra mắt ngày 03/11/2014 vừa qua. Google xem điện thoại và máy tính bảng Nexus là những thiết bị Android chủ lực của mình, với những tính năng phần cứng và phần mềm mới nhất của Android.

Bảng dưới đây thể hiện tỷ lệ sử dụng các phiên bản Android tính đến ngày 09/09/2014 (các phiên bản có thị phần dưới 0.1% không được liệt kê tại đây).

Phiên bản	Tên Mã	API	Tỷ lệ
2.2	Froyo	8	0.7%
2.3.3 -2.3.7	Gingerbread	10	11.4%
4.0.3 -4.0.4	Ice Cream Sandwich	15	9.6%
4.1.x	Jelly Bean	16	25.1%
4.2.x	Jelly Bean	17	20.7%
4.3	Jelly Bean	18	8.0%
4.4	KitKat	19	24.5%

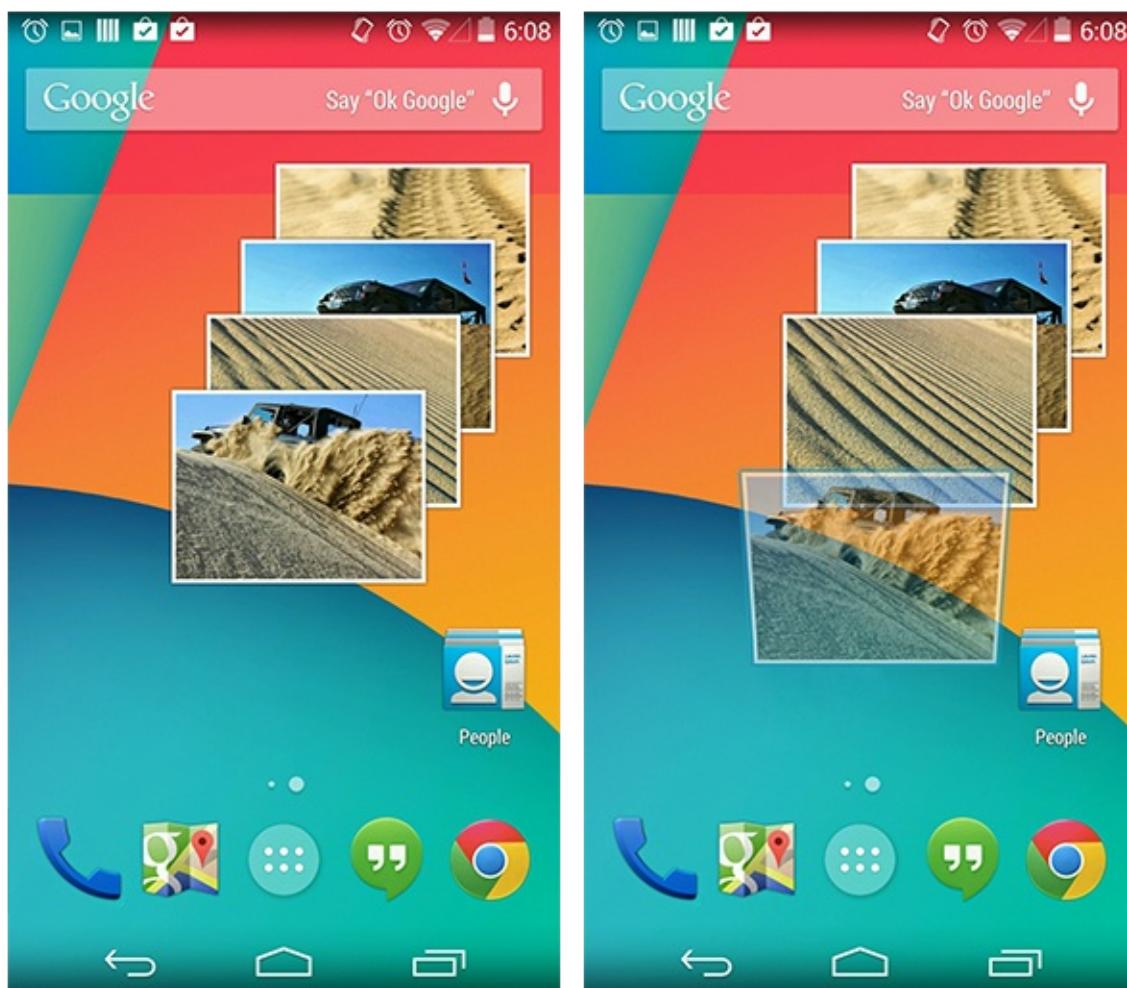


Giao diện và ứng dụng

Giao diện

Giao diện người dùng của Android dựa trên nguyên tắc tác động trực tiếp, sử dụng cảm ứng chạm tương tự như những động tác ngoài đời thực như vuốt, chạm, kéo dãn và thu lại để xử lý các đối tượng trên màn hình. Sự phản ứng với tác động của người dùng diễn ra gần như ngay lập tức, nhằm tạo ra giao diện cảm ứng mượt mà, thường dùng tính năng rung của thiết bị để tạo phản hồi rung cho người dùng. Những thiết bị phần cứng bên trong như gia tốc kế, con quay hồi chuyển và cảm biến khoảng cách được một số ứng dụng sử dụng để phản hồi một số hành động khác của người dùng, ví dụ như điều chỉnh màn hình từ chế độ hiển thị dọc sang chế độ hiển thị ngang tùy theo vị trí của thiết bị, hoặc cho phép người dùng lái xe đua bằng xoay thiết bị, giống như đang điều khiển vô-lăng.

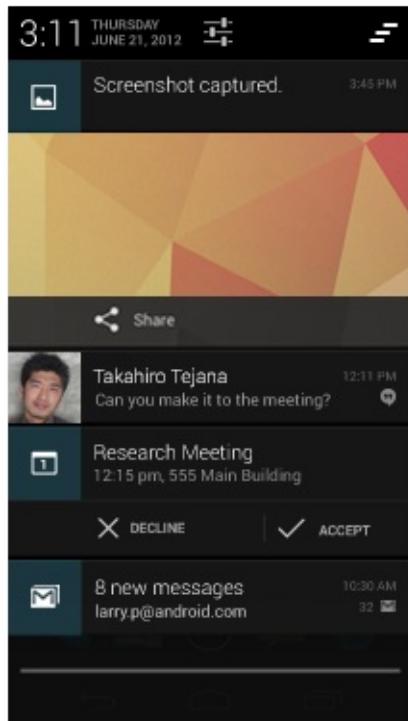
Các thiết bị Android sau khi khởi động sẽ hiển thị màn hình chính, điểm khởi đầu với các thông tin chính trên thiết bị, tương tự như khái niệm desktop (bàn làm việc) trên máy tính để bàn. Màn hình chính của Android thường gồm nhiều biểu tượng (icon) và tiện ích (widget); biểu tượng ứng dụng sẽ mở ứng dụng tương ứng, còn tiện ích hiển thị những nội dung sống động, cập nhật tự động như dự báo thời tiết, hộp thư của người dùng, hoặc những mẫu tin thời sự ngay trên màn hình chính. Màn hình chính có thể gồm nhiều trang được chuyển bằng cách vuốt ra trước hoặc sau. Giao diện màn hình chính của Android có thể tùy chỉnh ở mức cao, cho phép người dùng tự do sắp đặt hình dáng cũng như hành vi của thiết bị theo sở thích. Những ứng dụng do các hãng thứ ba có trên Google Play và các kho ứng dụng khác còn cho phép người dùng thay đổi "chủ đề" của màn hình chính, thậm chí bắt chước hình dáng của hệ điều hành khác như Windows Phone chẳng hạn. Phần lớn những nhà sản xuất, và một số nhà mạng, thực hiện thay đổi hình dáng và hành vi của các thiết bị Android của họ để phân biệt với các hãng cạnh tranh.



Hình . Màn hình chính với các biểu tượng (icon) và tiện ích (widget)

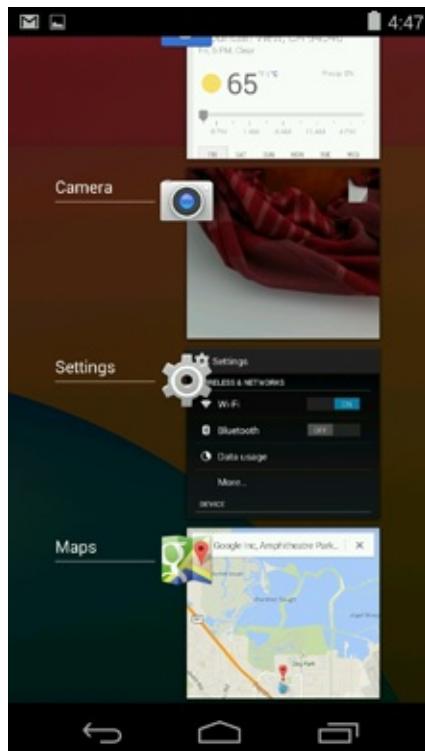
Ở phía trên cùng màn hình là thanh trạng thái, hiển thị thông tin về thiết bị và tình trạng kết nối. Thanh trạng thái này có thể

"kéo" xuống để xem màn hình thông báo gồm thông tin quan trọng hoặc cập nhật của các ứng dụng, như email hay tin nhắn SMS mới nhận, mà không làm gián đoạn hoặc khiến người dùng cảm thấy bất tiện. Trong các phiên bản đầu tiên, người dùng có thể nhấn vào thông báo để mở ra ứng dụng tương ứng, về sau này các thông tin cập nhật được bổ sung theo tính năng, như có khả năng lập tức gọi ngược lại khi có cuộc gọi nhỡ mà không cần phải mở ứng dụng gọi điện ra. Thông báo sẽ luôn nằm đó cho đến khi người dùng đã đọc hoặc xóa chúng đi.



Hình . Thanh trạng thái đang được "kéo" xuống với các thông báo bên trong

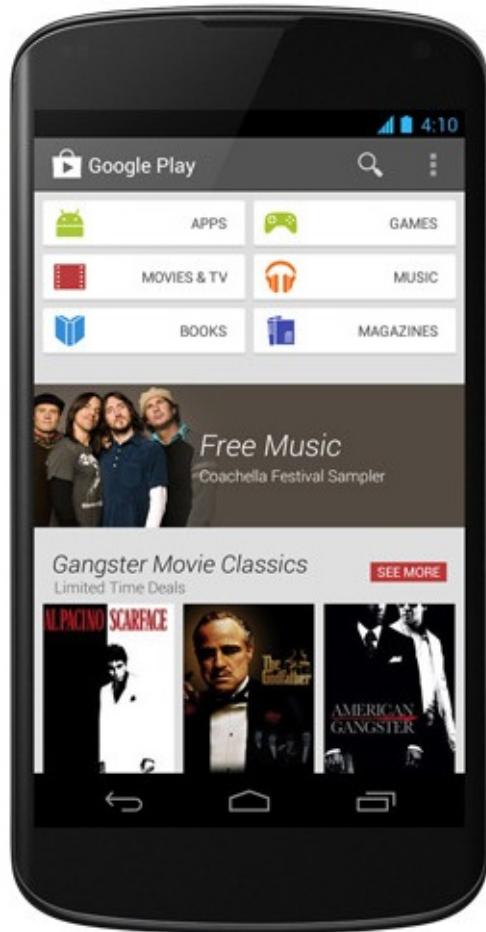
Một điểm mạnh của hệ điều hành Android là cho phép người dùng chạy nhiều ứng dụng cùng lúc (đa nhiệm - multi-tasking). Việc chuyển qua chuyển lại giữa các ứng dụng được thực hiện một cách dễ dàng bằng cách bấm và giữ phím Home để hiển thị danh sách ứng dụng đang chạy.



Hình . Danh sách ứng dụng đang chạy trên thiết bị (đa nhiệm)

Ứng dụng

Android có lượng ứng dụng của bên thứ ba ngày càng nhiều, được chọn lọc và đặt trên một cửa hàng ứng dụng như Google Play hay Amazon Appstore để người dùng lấy về, hoặc bằng cách tải xuống rồi cài đặt tập tin APK từ trang web khác. Các ứng dụng trên Cửa hàng Google Play cho phép người dùng duyệt, tải về và cập nhật các ứng dụng do Google và các nhà phát triển thứ ba phát hành. Google Play được cài đặt sẵn trên các thiết bị thỏa mãn điều kiện tương thích của Google. Google Play sẽ tự động lọc ra một danh sách các ứng dụng tương thích với thiết bị của người dùng, và nhà phát triển có thể giới hạn ứng dụng của họ chỉ dành cho những nhà mạng cố định hoặc những quốc gia cố định vì lý do kinh doanh. Nếu người dùng mua một ứng dụng mà họ cảm thấy không thích, họ có thể được hoàn trả tiền sau 15 phút kể từ lúc tải về. Một vài nhà mạng còn có khả năng mua giúp các ứng dụng trên Google Play, sau đó tính tiền vào trong hóa đơn sử dụng hàng tháng của người dùng. Tính đến tháng 8 năm 2013, có hơn 975.000 ứng dụng dành cho Android trên cửa hàng Google Play, và số lượt ứng dụng tải về từ Cửa hàng Play ước tính đạt gần 30 tỷ.



Hình: Giao diện cửa hàng Google Play trên điện thoại Google Nexus 4

Các ứng dụng cho Android được phát triển bằng ngôn ngữ Java sử dụng Bộ phát triển phần mềm Android (Android SDK). Android SDK bao gồm một bộ đầy đủ các công cụ dùng để phát triển, bao gồm công cụ gỡ lỗi, thư viện phần mềm, bộ giả lập điện thoại dựa trên QEMU, tài liệu hướng dẫn, mã nguồn mẫu, và hướng dẫn từng bước. Môi trường phát triển tích hợp (IDE) được hỗ trợ chính thức là Eclipse sử dụng phần bổ sung Android Development Tools (ADT). Các công cụ phát triển khác cũng có sẵn, gồm có Bộ phát triển gốc dành cho các ứng dụng hoặc phần mở rộng viết bằng C hoặc C++, Google App Inventor, một môi trường đồ họa cho những người mới bắt đầu lập trình, và nhiều nền tảng ứng dụng Web di động đa nền tảng phong phú.

Phát triển

Android được Google tự phát triển riêng cho đến khi những thay đổi và cập nhật đã hoàn thiện, khi đó mã nguồn mới được công khai. Mã nguồn này, nếu không sửa đổi, chỉ chạy được trên một số thiết bị, thường là thiết bị thuộc dòng Nexus. Có nhiều thiết bị có chứa những thành phần được giữ bản quyền do nhà sản xuất đặt vào thiết bị Android của họ.

Hạt nhân Linux

Android có hạt nhân dựa trên nhân Linux phiên bản 2.6. Kể từ Android 4.0 Ice Cream Sandwich (bánh ngọt kẹp kem) trở về sau, là phiên bản 3.x, với middleware, thư viện và API viết bằng C, còn phần mềm ứng dụng chạy trên một nền tảng ứng dụng gồm các thư viện tương thích với Java dựa trên Apache Harmony. Android sử dụng máy ảo Dalvik với một trình biên dịch động để chạy 'mã dex' (Dalvik Executable) của Dalvik, thường được biên dịch sang Java bytecode. Nền tảng phần cứng chính của Android là kiến trúc ARM. Người ta cũng hỗ trợ x86 thông qua dự án Android x86 (chip Intel Atom), và Google TV cũng sử dụng một phiên bản x86 đặc biệt của Android.

Nhân Linux dùng cho Android đã được Google thực hiện nhiều thay đổi về kiến trúc so với nhân Linux gốc. Android không có sẵn X Window System cũng như không hỗ trợ các thư viện GNU chuẩn, nên việc chuyển các ứng dụng hoặc thư viện Linux có sẵn sang Android rất khó khăn. Các ứng dụng C đơn giản và SDL cũng được hỗ trợ bằng cách chèn những đoạn shim Java và sử dụng tương tự như nền tảng lập trình JNI, như khi người ta chuyển trò chơi Jagged Alliance 2 sang Android.

Một số tính năng cũng được Google đóng góp ngược vào nhân Linux, đáng chú ý là tính năng quản lý nguồn điện có tên wakelock, nhưng bị những người lập trình chính cho nhân từ chối vì họ cảm thấy Google không có định sẽ tiếp tục bảo trì đoạn mã do họ viết. Google thông báo vào tháng 4 năm 2010 rằng họ sẽ thuê hai nhân viên để làm việc với cộng đồng nhân Linux, nhưng Greg Kroah-Hartman, người bảo trì nhân Linux hiện tại của nhánh ổn định, đã nói vào tháng 12 năm 2010 rằng ông lo ngại Google không còn muốn đưa những thay đổi của mình vào Linux dòng chính nữa. Một số lập trình viên Android của Google tỏ ý rằng "nhóm Android thấy chán với quy trình đó," vì nhóm họ không có nhiều người và có nhiều việc khẩn cấp cần làm với Android hơn.

Vào tháng 8 năm 2011, Linus Torvalds phát biểu rằng "rốt cuộc thì Android và Linux cũng sẽ trở lại với một bộ nhân chung, nhưng điều đó có thể sẽ không xảy ra trong 4 hoặc 5 năm nữa". Vào tháng 12 năm 2011, Greg Kroah-Hartman thông báo kích hoạt Dự án Dòng chính Android, nhằm tới việc đưa một số driver, bản vá và tính năng của Android ngược vào nhân Linux, bắt đầu từ Linux 3.3. Linux cũng đưa tính năng autosleep (tự động nghỉ) và wakelocks vào nhân 3.5, sau nhiều nỗ lực phối hợp trước đó. Tương tác thì vẫn vậy nhưng bản hiện thực trên Linux dòng chính cho phép hai chế độ nghỉ: bộ nhớ (dạng nghỉ truyền thống mà Android sử dụng), và đĩa (dạng ngủ đông - hibernate trên máy tính để bàn). Việc trộn sẽ hoàn tất kể từ nhân 3.8. Google đã công khai kho mã nguồn trong đó có những đoạn thử nghiệm đưa Android về lại nhân 3.8.

Bộ lưu trữ flash trên các thiết bị Android được chia thành nhiều phân vùng, như "/system" dành cho hệ điều hành và "/data" dành cho dữ liệu người dùng và cài đặt ứng dụng. Khác với các bản phân phối Linux cho máy tính để bàn, người sở hữu thiết bị Android không được trao quyền truy cập root vào hệ điều hành và các phân vùng nhạy cảm như /system được thiết lập chỉ đọc. Tuy nhiên, quyền truy cập root có thể chiếm được bằng cách tận dụng những lỗ hổng bảo mật trong Android, điều mà công đồng mã nguồn mở thường xuyên sử dụng để nâng cao tính năng thiết bị của họ, kể cả bị những người ác ý sử dụng để cài virus và phần mềm gián điệp.

Việc Android có được xem là một bản phân phối Linux hay không vẫn còn là vấn đề gây tranh cãi, tuy được Linux Foundation và Chris DiBona, trưởng nhóm mã nguồn mở Google, ủng hộ. Một số khác, như linux-magazine.com thì không đồng ý, do Android không hỗ trợ nhiều công cụ GNU, trong đó có glibc.

Quản lý bộ nhớ

Vì các thiết bị Android chủ yếu chạy bằng pin, nên Android được thiết kế để quản lý bộ nhớ (RAM) nhằm giảm tối đa tiêu thụ điện năng, trái với hệ điều hành máy tính để bàn luôn cho rằng máy tính sẽ có nguồn điện không giới hạn. Khi một ứng dụng Android không còn được sử dụng, hệ thống sẽ tự động ngưng nó trong bộ nhớ, trong khi ứng dụng về mặt kỹ thuật vẫn "mở", những ứng dụng này sẽ không tiêu thụ bất cứ tài nguyên nào (như năng lượng pin hay năng lượng xử lý) và nằm đó cho đến khi nó được cần đến. Cách làm như vậy có lợi kép là vừa làm tăng khả năng phản hồi nói chung của thiết

bị Android, vì ứng dụng không nhất thiết phải đóng rồi mở lại từ đầu, vừa đảm bảo các ứng dụng nền không làm tiêu hao năng lượng một cách không cần thiết.

Android quản lý các ứng dụng trong bộ nhớ một cách tự động: khi bộ nhớ thấp, hệ thống sẽ bắt đầu diệt ứng dụng và tiến trình không hoạt động trong một thời gian, sắp theo thời điểm cuối mà chúng được sử dụng (cái cũ nhất sẽ bị tắt trước). Quá trình này được thiết kế ẩn đi với người dùng, để người dùng không cần phải quản lý bộ nhớ hoặc tự tay tắt các ứng dụng. Tuy nhiên, sự che giấu này của hệ thống quản lý bộ nhớ Android đã dẫn đến sự thịnh hành của các ứng dụng tắt chương trình của bên thứ ba trên cửa hàng Google Play; những ứng dụng kiểu như vậy được cho là có hại nhiều hơn có lợi.

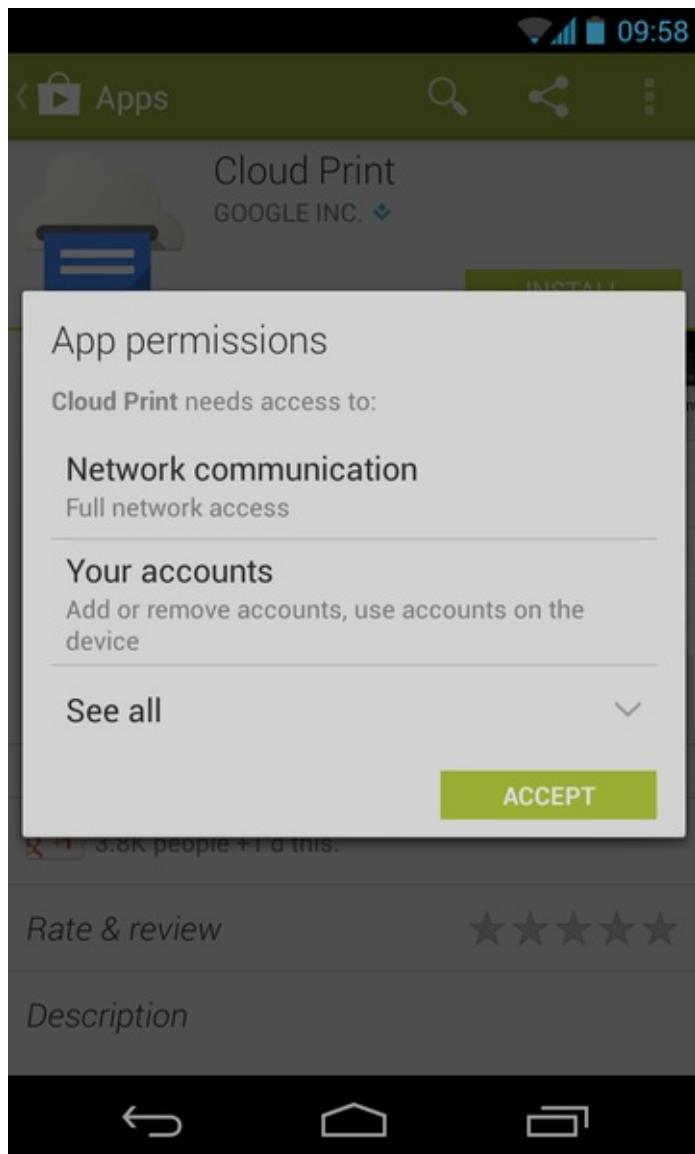
Cộng đồng mã nguồn mở

Android có một cộng đồng các lập trình viên và những người đam mê rất năng động. Họ sử dụng mã nguồn Android để phát triển và phân phối những phiên bản chỉnh sửa của hệ điều hành. Các bản Android do cộng đồng phát triển thường đem những tính năng và cập nhật mới vào nhanh hơn các kênh chính thức của nhà sản xuất/nhà mạng, tuy không được kiểm thử kỹ lưỡng cũng như không có đảm bảo chất lượng; cung cấp sự hỗ trợ liên tục cho các thiết bị cũ không còn nhận được bản cập nhật chính thức; hoặc mang Android vào những thiết bị ban đầu chạy một hệ điều hành khác, như HP Touchpad. Các bản Android của cộng đồng thường được root (mở quyền truy cập toàn hệ thống) sẵn và có những điều chỉnh không phù hợp với những người dùng không rành, như khả năng ép xung hoặc tăng/giảm áp bộ xử lý của thiết bị. CyanogenMod là firmware của cộng đồng được sử dụng phổ biến nhất và là cơ sở cho rất nhiều firmware khác phát triển dựa trên bản firmware này.

Trước đây, nhà sản xuất thiết bị và nhà mạng tỏ ra thiếu thiện chí với việc phát triển firmware của bên thứ ba. Những nhà sản xuất còn thể hiện lo ngại rằng các thiết bị chạy phần mềm không chính thức sẽ hoạt động không tốt và dẫn đến tổn thất kinh tế. Hơn nữa, các firmware đã thay đổi như CyanogenMod đôi khi còn cung cấp những tính năng, như truyền tải mạng (tethering), mà người dùng bình thường phải trả tiền nhà mạng mới được sử dụng. Kết quả là nhiều thiết bị bắt đầu đặt ra hàng rào kỹ thuật như khóa bootloader hay hạn chế quyền truy cập root. Tuy nhiên, khi phần mềm do cộng đồng phát triển ngày càng trở nên phổ biến, và sau một thông cáo của Thư viện Quốc hội Hoa Kỳ cho phép "jailbreak" (vượt ngục) thiết bị di động, các nhà sản xuất và nhà mạng đã tỏ ra mềm mỏng hơn với các nhà phát triển thứ ba, thậm chí một số hãng như HTC, Motorola, Samsung và Sony, còn hỗ trợ và khuyến khích phát triển. Kết quả của việc này là dần dần nhu cầu tìm ra các hạn chế phần cứng để cài đặt được firmware không chính thức đã bớt đi do ngày càng nhiều thiết bị được phát hành với bootloader đã mở khóa sẵn hoặc có thể mở khóa, như điện thoại dòng Nexus, tuy rằng thông thường họ sẽ yêu cầu người dùng từ bỏ chế độ bảo hành nếu họ làm như vậy. Tuy nhiên, dù được sự chấp thuận của nhà sản xuất, một số nhà mạng tại Mỹ vẫn bắt buộc điện thoại phải bị khóa.

Bảo mật và tính riêng tư

Các ứng dụng Android chạy trong một "hộp cá" (sandboxed applications), là một khu vực riêng rẽ với hệ thống và không được tiếp cận đến phần còn lại của tài nguyên hệ thống, trừ khi nó được người dùng trao quyền truy cập một cách công khai khi cài đặt. Trước khi cài đặt ứng dụng, Cửa hàng Play sẽ hiển thị tất cả các quyền mà ứng dụng đòi hỏi: ví dụ như một trò chơi cần phải kích hoạt bộ rung hoặc lưu dữ liệu vào thẻ nhớ SD, nhưng nó không nên cần quyền đọc tin nhắn SMS hoặc tiếp cận danh bạ điện thoại. Sau khi xem xét các quyền này, người dùng có thể chọn đồng ý hoặc từ chối chúng, ứng dụng chỉ được cài đặt khi người dùng đồng ý.



Hình . Yêu cầu chấp nhận các quyền ứng dụng được phép sử dụng trước khi cài đặt

Hệ thống hộp cát và hỏi quyền làm giảm bớt ảnh hưởng của lối bảo mật hoặc lối chương trình có trong ứng dụng, nhưng sự thiếu kinh nghiệm của lập trình viên và tài liệu hướng dẫn còn hạn chế đã dẫn tới những ứng dụng hay đòi hỏi những quyền không cần thiết, do đó làm giảm đi hiệu quả của hệ thống này. Một số công ty bảo mật, như Lookout Mobile Security, AVG Technologies, và McAfee, đã phát hành những phần mềm diệt virus cho các thiết bị Android. Phần mềm này không có hiệu quả vì cơ chế hộp cát vẫn áp dụng vào các ứng dụng này, do vậy làm hạn chế khả năng quét sâu vào hệ thống để tìm nguy cơ.

Một nghiên cứu của công ty bảo mật Trend Micro đã liệt kê tình trạng lạm dụng dịch vụ phải trả tiền là hình thức phần mềm ác ý phổ biến nhất trên Android, trong đó tin nhắn SMS sẽ bị gửi đi từ điện thoại bị nhiễm đến một số điện thoại trả tiền mà người dùng không hề hay biết. Loại phần mềm ác ý khác hiển thị những quảng cáo không mong muốn và gây khó chịu trên thiết bị, hoặc gửi thông tin cá nhân đến bên thứ ba khi chưa được phép. Đe dọa bảo mật trên Android được cho là tăng rất nhanh theo cấp số mũ; tuy nhiên, các kỹ sư Google phản bác rằng hiểm họa từ phần mềm ác ý và virus đã bị thổi phồng bởi các công ty bảo mật nhằm mục đích thương mại, và buộc tội ngành công nghiệp bảo mật đang lợi dụng sự sợ hãi để bán phần mềm diệt virus cho người dùng. Google vẫn giữ quan điểm rằng phần mềm ác ý thật sự nguy hiểm là cực kỳ hiếm, và một cuộc điều tra do F-Secure thực hiện cho thấy chỉ có 0,5% số phần mềm ác ý trên Android là len vào được cửa hàng Google Play.

Google hiện đang sử dụng bộ quét phần mềm ác ý Google Bouncer để theo dõi và quét các ứng dụng trên Cửa hàng Google Play. Nó sẽ đánh dấu các phần mềm bị nghi ngờ và cảnh báo người dùng về những vấn đề có thể xảy ra trước khi

họ tải nó về máy. Android phiên bản 4.2 Jelly Bean được phát hành vào năm 2012 cùng với các tính năng bảo mật được cải thiện, bao gồm một bộ quét phần mềm ác ý được cài sẵn trong hệ thống, hoạt động cùng với Google Play nhưng cũng có thể quét các ứng dụng được cài đặt từ nguồn thứ ba, và một hệ thống cảnh báo sẽ thông báo cho người dùng khi một ứng dụng cố gắng gửi một tin nhắn vào sổ tính tiền, chặn tin nhắn đó lại trừ khi người dùng công khai cho phép nó.

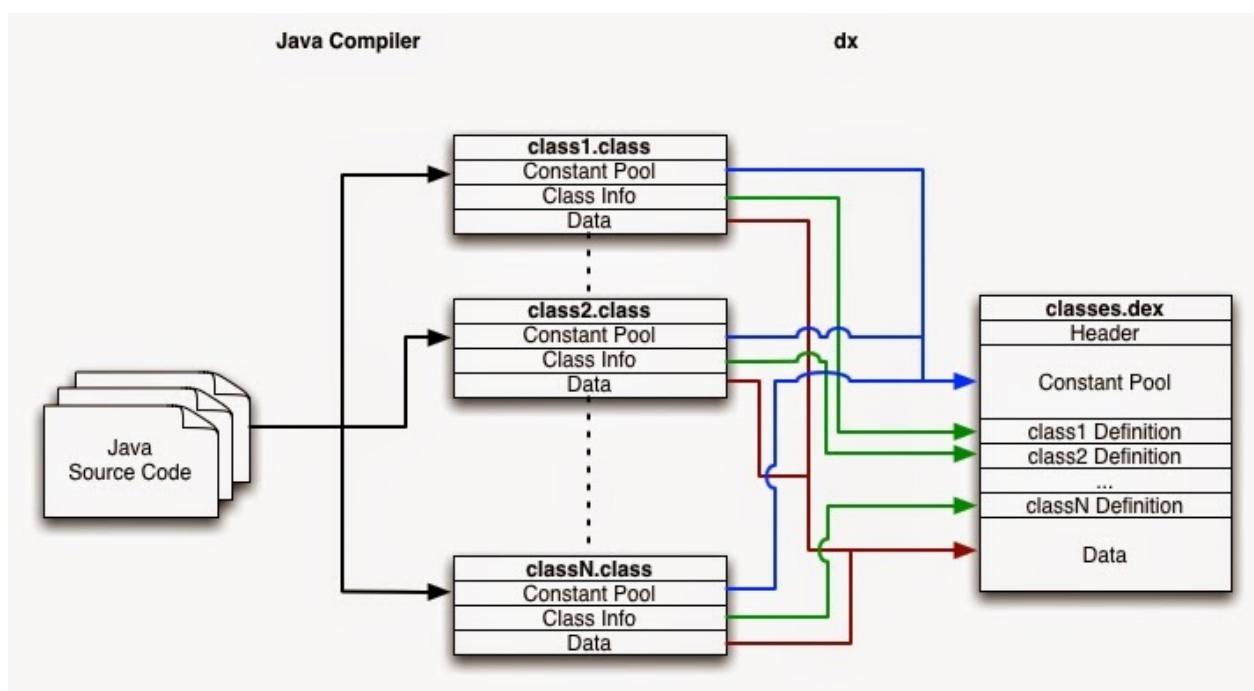
Điện thoại thông minh Android có khả năng báo cáo vị trí của điểm truy cập Wi-Fi, phát hiện ra việc di chuyển của người dùng điện thoại, để xây dựng những cơ sở dữ liệu có chứa vị trí của hàng trăm triệu điểm truy cập. Những cơ sở dữ liệu này tạo nên một bản đồ điện tử để tìm vị trí điện thoại thông minh, cho phép chúng chạy các ứng dụng như Foursquare, Google Latitude, Facebook Places, và gửi những đoạn quảng cáo dựa trên vị trí. Phần mềm theo dõi của bên thứ ba như TaintDroid, một dự án nghiên cứu trong trường đại học, đôi khi có thể biết được khi nào thông tin cá nhân bị gửi đi từ ứng dụng đến các máy chủ đặt ở xa.

Máy ảo Dalvik

Java được biết đến với khẩu hiệu "viết một lần, chạy mọi nơi" (write once, run anywhere). Tính năng này của hệ sinh thái Java có được nhờ một tầng máy ảo đứng giữa tầng ứng dụng (viết bằng Java) và tầng hệ điều hành (nhiều loại khác nhau), gọi là máy ảo Java - JVM (Java Virtual Machine). Máy ảo Java chạy rất ổn định và đồng nhất trên các môi trường máy tính cá nhân (JSE) và máy chủ (JEE), tuy nhiên trong môi trường di động (JME) thì máy ảo Java bị phân mảnh tương đối nhiều với các gói (packages), cấu hình (configurations) và các mẫu (profiles) khác nhau.

Google đã chọn Java làm ngôn ngữ phát triển ứng dụng cho hệ điều hành Android, tuy nhiên lại từ bỏ cả phiên bản Java di động (JME) lẫn máy ảo Java (JVM) để phát triển một máy ảo riêng - máy ảo Dalvik và tập thư viện Java chuẩn được viết lại và thu gọn hơn.

Ứng dụng Android được phát triển bằng ngôn ngữ Java, sau đó được biên dịch thành dạng mã nhị phân Java (Java bytecode). Các file nhị phân .class (tương thích với JVM) này sẽ được chuyển đổi thành 1 file nhị phân dùng cho máy ảo Dalvik - .dex (Dalvik EXecutable) trước khi được cài lên thiết bị. Các tập tin .dex được thiết kế để phù hợp hơn với các thiết bị hạn chế về bộ nhớ cũng như hiệu năng xử lý.



Dalvik là một phần mềm mã nguồn mở, ban đầu được phát triển bởi Dan Bornstein, và được ông đặt tên theo tên ngôi làng Dalvík tại Eyjafjörður, Iceland, quê hương của ông.

Kiến trúc

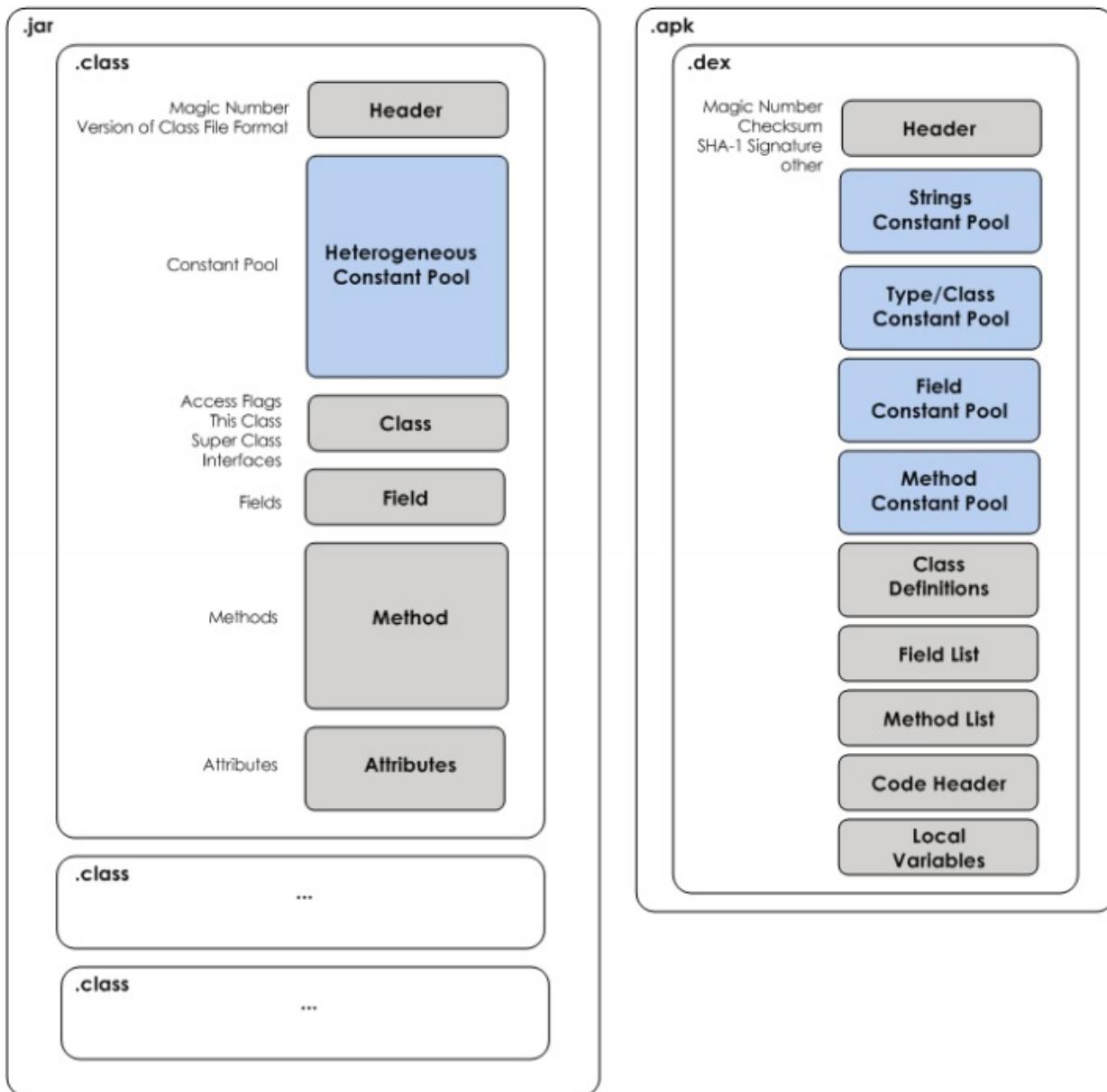
Hệ điều hành Android được thiết kế hướng đến rất nhiều các loại thiết bị khác nhau, từ những thiết bị giá rẻ, cấu hình thấp (low-end) đến những siêu phẩm đắt tiền (high-end). Bên cạnh đó, các ứng dụng Android phải được chạy trong "hộp cá" với độ bảo mật cao, hiệu năng tốt và ổn định, vì vậy việc sử dụng một máy ảo làm môi trường thực thi là điều tất yếu. Tuy nhiên việc sử dụng máy ảo thường sẽ làm giảm hiệu năng của hệ thống. Giải pháp máy ảo Dalvik của Google được đưa ra nhằm cân bằng giữa 2 điểm này:

- Mỗi ứng dụng Android được chạy trong tiến trình riêng, với một thực thể (instance) của máy ảo Dalvik riêng. Dalvik được thiết kế để hệ điều hành có thể chạy nhiều thực thể của máy ảo một cách hiệu quả với file thực thi Dalvik (.dex file) được thiết kế để tối ưu hóa bộ nhớ cần sử dụng. Máy ảo Dalvik là máy ảo dựa trên thanh ghi (register-based), khác với máy ảo Java chạy dựa trên cấu trúc ngăn xếp (stack-based).
- Máy ảo Dalvik sử dụng hạt nhân Linux cho các việc liên quan đến lớp dưới của hệ thống như quản lý tiểu trình (threading), quản lý bộ nhớ cấp thấp... Do mỗi ứng dụng chạy trên một thực thể máy ảo riêng, hệ thống Android không những cần tối ưu việc chạy nhiều máy ảo và còn phải giảm tối đa thời gian khởi tạo máy ảo mới. Phần dưới đây sẽ xem xét 4 điểm chính trong thiết kế máy ảo mà Google đã chọn lựa để đạt được hiệu năng và độ bảo mật cần thiết: cấu trúc file .dex, Zygote, cấu trúc dựa trên thanh ghi và bảo mật.

Định dạng file .dex

Trong môi trường Java truyền thống, mã nguồn Java được biên dịch thành dạng bytecode, ở đó mỗi class trong mã nguồn được biên dịch thành một file .class riêng. Trong hệ thống Android, mã nguồn Java cũng được biên dịch thành các file .class riêng. Tuy nhiên trước khi có thể được cài đặt lên thiết bị, các file này phải được chuyển đổi thành file thực thi Dalvik (.dex) bằng công cụ "dx" có trong Android SDK. File thực thi Dalvik có thể chứa nhiều class bên trong và mỗi ứng dụng Android chỉ chứa một file .dex duy nhất.

File thực thi Dalvik được thiết kế nhằm đến việc tối ưu bộ nhớ trên cơ sở chia sẻ (dùng chung) dữ liệu. Biểu đồ dưới đây phân biệt cấu trúc file .dex với file .class truyền thống.



Hình . Cấu trúc file class của Java và file thực thi Dalvik

Cơ chế chính của file .dex cho việc tiết kiệm bộ nhớ là dùng chung các vùng nhớ chứa hằng số cho tất cả các class. Các vùng nhớ cho hằng số này được chia theo từng loại dữ liệu (hằng số chuỗi, tên trường, tên class, tên method...). Trong mã lệnh của các class bên dưới sẽ không cần ghi lại các tên/hằng số này mà chỉ chứa chỉ số (số nguyên) của nó trong vùng hằng số.

Đối với các file .class truyền thống, mỗi class file chứa vùng nhớ riêng cho các hằng số của class đó. Vùng nhớ này không đồng nhất vì nó chứa tất cả các loại hằng số, không phân biệt theo loại như trong trường hợp của .dex file. Việc lặp lại khai báo các hằng số này ở nhiều file .class khiến tổng kích thước các file .class lớn hơn nhiều so với file thực thi Dalvik.

Theo số liệu thống kê, trung bình trong mỗi file .class, 61% kích thước của file là các hằng số (!!), mã của các phương thức (method) chỉ chứa khoảng 33%, còn lại 5% cho các phần khác của class. Như vậy việc tiết kiệm vùng nhớ cho hằng số dẫn đến việc tiết kiệm đáng kể kích thước file thực thi (thông thường là khoảng 50%), từ đó tiết kiệm được lượng bộ nhớ cần cấp phát lúc thực thi ứng dụng.

Zygote

Các ứng dụng Android được chạy trong các thực thể máy ảo riêng, vì vậy máy ảo phải được "khởi động" nhanh và sử

dụng tối thiểu bộ nhớ cần thiết. Để đạt được điều này, hệ thống Android đưa ra một cơ chế giúp các thực thể máy ảo có thể chia sẻ mã lệnh dùng chung và "khởi động" nhanh, gọi là Zygote.

Hầu hết các ứng dụng Android đều dùng chung một số thư viện lõi và cấu trúc bộ nhớ heap kèm theo nó. Thêm vào đó, vùng nhớ heap của các thư viện lõi này thường chỉ được truy xuất chỉ đọc (read-only). Lợi dụng điểm này, Zygote, một tiến trình máy ảo Dalvik được khởi chạy lúc khởi động thiết bị, nạp sẵn các thư viện lõi của hệ điều hành mà các ứng dụng thường xuyên sử dụng, để sẵn sàng cho các máy ảo sử dụng. Mỗi lần một ứng dụng được chạy, hệ thống sẽ khởi động một máy ảo mới, tuy nhiên máy ảo này sẽ không cần chứa những thư viện lõi, mà sử dụng luôn thư viện lõi trong Zygote, khiến thời gian khởi tạo máy ảo là tối thiểu.

Các thư viện lõi trong hầu hết các trường hợp là "chỉ đọc" chứ không bị ghi đè bởi ứng dụng. Trong trường hợp ứng dụng có nhu cầu ghi vào vùng nhớ heap của các thư viện lõi này, một bản sao của vùng nhớ Zygote sẽ được tạo ra riêng cho tiến trình của ứng dụng đó. Cơ chế "sao chép khi ghi" (copy-on-write) này giúp tối ưu hóa việc sử dụng chung mã lệnh của thư viện lõi mà vẫn đảm bảo tính bảo mật giữa các ứng dụng: việc thay đổi dữ liệu trong bộ nhớ của ứng dụng này không ảnh hưởng đến các ứng dụng khác.

Trong JVM truyền thống, mỗi thực thể của máy ảo Java sẽ có một tập các thư viện lõi riêng cùng vùng nhớ tương ứng cho chúng, không có sự chia sẻ bộ nhớ giữa các thực thể của máy ảo.

Kiến trúc máy ảo dựa trên thanh ghi

Theo truyền thống, các nhà phát triển máy ảo thường lựa chọn kiến trúc dựa trên ngăn xếp (stack-based architecture) hơn là kiến trúc dựa trên thanh ghi (register-based architecture) do việc thiết kế máy ảo và trình biên dịch cho kiến trúc ngăn xếp đơn giản hơn so với trường hợp của kiến trúc thanh ghi. Bù lại cho sự đơn giản là hiệu suất giảm đi.

Các nghiên cứu đã chỉ ra rằng, kiến trúc dựa trên thanh ghi thường đòi hỏi số lượng mã lệnh thực thi cho ứng dụng ít hơn trung bình là 47% so với trường hợp dùng ngăn xếp. Bù lại, kích thước mã lệnh thường lớn hơn khoảng 25%. Tuy nhiên sự tăng về kích thước mã lệnh này chỉ phát sinh thêm khoảng 1.07% tải của CPU. Các trình đánh giá tiêu chuẩn cho thấy hiệu suất tổng thể của máy ảo dựa trên kiến trúc thanh ghi cao hơn khoảng 32.3% so với kiến trúc ngăn xếp.

Với yêu cầu chạy trên các thiết bị với hiệu năng tính toán thấp, việc máy ảo Dalvik lựa chọn kiến trúc thanh ghi là hoàn toàn hợp lý. Mặc dù kích thước mã lệnh lớn hơn trung bình 25%, việc sử dụng vùng nhớ chung cho hằng số của file .dex giúp kích thước mã lệnh giảm đi đến 50% khiến file thực thi Dalvik vẫn hiệu quả hơn đáng kể về mặt bộ nhớ so với các file .class truyền thống.

Bảo mật

Mặc định theo kiến trúc bảo mật của hệ điều hành Android, mỗi ứng dụng không có quyền thực thi bất cứ thao tác nào ảnh hưởng đến các ứng dụng khác, đến hệ điều hành hay người dùng, bao gồm: đọc/ghi dữ liệu cá nhân (như danh bạ, email...), đọc/ghi file dữ liệu của ứng dụng khác, thực hiện kết nối mạng, giữ màn hình luôn sáng...

Tính năng này được thực hiện từ tầng hệ điều hành và tầng khung ứng dụng (application framework) chứ không phải bên trong máy ảo Dalvik. Mỗi ứng dụng được chạy trong một máy ảo riêng, máy ảo này lại được chạy trong một tiến trình riêng. Mỗi tiến trình được chạy dưới một mã người dùng (user ID) riêng, được sinh ra lúc cài đặt và được phân quyền chỉ được truy cập một số tính năng nhất định của hệ điều hành và các tập tin riêng của mình. Nếu ứng dụng có các yêu cầu đặc biệt đến các tính năng khác của hệ thống thì phải khai báo các quyền tương ứng trong file mô tả ứng dụng (AndroidManifest.xml) và sẽ được cấp các quyền này lúc cài đặt nếu được sự đồng ý từ người dùng.

Lập trình cho thiết bị Android

Bộ phát triển phần mềm Android (Android SDK)

Hệ sinh thái Android được cấu thành từ 3 thành phần tổng quan như sau:

- Hệ điều hành mã nguồn mở cho thiết bị nhúng
- Bộ công cụ phát triển phần mềm (mã nguồn mở) - Android SDK
- Các thiết bị (chủ yếu là điện thoại thông minh) chạy hệ điều hành Android với các ứng dụng được phát triển cho chúng. Trong đó Android SDK là bộ phát triển phần mềm Android được Google đưa ra dưới dạng mã nguồn mở, giúp người lập trình có thể dễ dàng truy xuất vào các tính năng của hệ điều hành như:
 - Mạng điện thoại cho việc nghe gọi/SMS cũng như truy cập Internet: GSM/GRPS, EGDE, 3G, 4G, LTE...
 - Tập APIs cho các ứng dụng hướng vị trí (GPS, vị trí dựa trên kết nối mạng)
 - Tích hợp ứng dụng bản đồ thành 1 thành phần của ứng dụng
 - Kết nối mạng WiFi và kết nối điểm-tới-điểm (peer-to-peer)
 - Các ứng dụng đa phương tiện: ghi hình, ghi âm với camera và microphone, phát nhạc, phát video...
 - Thư viện đa phương tiện với khả năng thu và phát nhiều loại audio và video, cũng như các ảnh tĩnh
 - API truy cập đến các cảm biến: cảm biến gia tốc, tiệm cận, la bàn...
 - Thư viện cho việc sử dụng Bluetooth và NFC cho kết nối điểm-tới-điểm
 - Chia sẻ và trao đổi dữ liệu giữa các ứng dụng
 - API cho các kho dữ liệu chia sẻ như danh bạ, lịch, mạng xã hội, đa phương tiện...
 - Các ứng dụng/tác vụ chạy ngầm
 - Các tiện ích (widget) trên màn hình chính và hình nền động
 - Tích hợp khả năng tìm kiếm của ứng dụng vào công cụ tìm kiếm của hệ thống
 - Trình duyệt HTML5 mã nguồn mở, dựa trên Webkit
 - Bộ tăng tốc đồ họa bằng phần cứng (GPU) cho việc tối ưu hiệu năng hiển thị, bao gồm nhưng không giới hạn ở đồ họa 2D và 3D với OpenGL ES 2.0
 - Hỗ trợ đa ngôn ngữ với cơ chế cung cấp tài nguyên đặc trưng
 - Cơ chế sử dụng lại các thành phần của ứng dụng cũng như thay thế các ứng dụng sẵn có
 - Lưu trữ và truy xuất dữ liệu với CSDL Sqlite 3.0
 - Google Cloud Message: cho quản lý việc đẩy thông báo xuống thanh hệ thống
 - ...

Bộ SDK đầy đủ sau khi tải về sẽ bao gồm tất cả các thành phần cần thiết cho việc viết code, gỡ lỗi, kiểm thử, biên dịch và xuất bản phần mềm như:

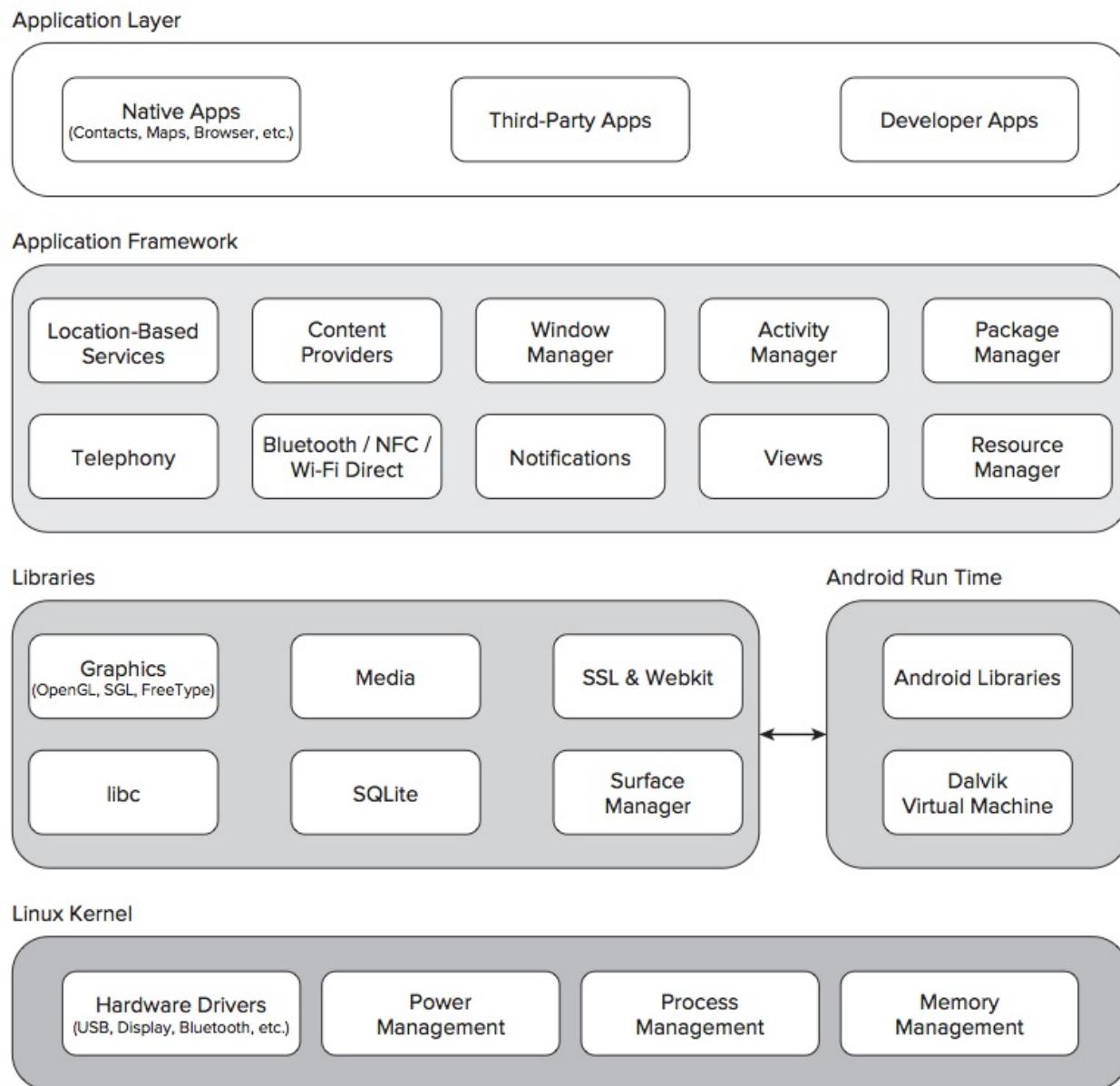
- Android API - là các thư viện lõi của bộ SDK, giúp người lập trình truy xuất các thành phần của hệ điều hành. Các ứng dụng của Google cài sẵn trên thiết bị cũng được phát triển dựa trên các API này.
- Công cụ phát triển: bao gồm các công cụ dùng cho việc biên dịch và gỡ lỗi ứng dụng
- Bộ quản lý các thiết bịảo và bộ giả lập thiết bị: cho phép người lập trình chạy thử ứng dụng trên một hoặc nhiều thiết bị ảo thông qua bộ giả lập trên máy tính cá nhân mà không cần thiết bị di động thật.
- Bộ tài liệu tra cứu rất chi tiết về mô tả và cách sử dụng cho tất cả các gói và các lớp, cũng như các bài viết hữu ích khác.
- Các mã nguồn mẫu: bộ SDK cũng đi kèm với những ứng dụng mẫu, nhằm biểu diễn các tính năng của Android với các ví dụ chi tiết về việc sử dụng các API

Đối với những người dùng Eclipse làm môi trường phát triển tích hợp, Google cũng phát triển một trình cắm thêm (Plugin) cho IDE này gọi là [Android Developer Tools \(ADT\)](#), giúp việc phát triển ứng dụng Android trở nên dễ dàng hơn rất nhiều. Bên cạnh đó, Google cũng phát triển song song và dần chuyển trọng tâm qua một môi trường phát triển mới cho ứng dụng Android, tên là [Android Studio](#), dựa trên phiên bản mã nguồn mở của IDE nổi tiếng [IntelliJ IDEA](#) của JetBrains. Ta sẽ làm quen với trình cắm thêm ADT ở các phần sau của giáo trình. Android Studio đang dần trở nên phổ biến và hỗ trợ tốt hơn ADT về nhiều mặt. Tuy nhiên, đến thời điểm hiện tại, bản mới nhất vẫn đang ở trạng thái Beta 0.8 nên giáo trình sẽ giới

thiệu môi trường phát triển tích hợp này vào lần tái bản sau.

Các tầng phần mềm Android (software stack)

Để đảm bảo sự đơn giản, tính dùng chung và bảo mật trong các ứng dụng trên hệ thống Android, các thư viện cho ứng dụng Android được chia ra thành các phân tầng từ thấp đến cao, các thành phần ở tầng trên chủ yếu sử dụng các dịch vụ do các tầng ngay bên dưới cung cấp mà ít khi phải dùng đến các API của các tầng thấp hơn. Kiến trúc này được chia ra thành các tầng chính như trong hình vẽ dưới đây.



Hình: Các lớp phần mềm trong Android

Ở đó:

Nhân Linux (Linux kernel) - bao gồm các dịch vụ lõi (trình điều khiển phần cứng, quản lý tiến trình, bộ nhớ, mạng, nguồn điện và bảo mật...). Nhân này là lớp trùu tượng giữa phần cứng của thiết bị và các lớp phía trên.

Các thư viện (libraries) - chạy bên trên lớp hạt nhân, bao gồm các các thư viện lõi viết bằng C/C++ như libc và SSL, cũng như các thư viện sau:

- Thư viện cho việc phát các loại dữ liệu âm thanh và hình ảnh

- Bộ quản lý bề mặt (surface manager) cho việc hiển thị đồ họa trên màn hình
- Các thư viện đồ họa bao gồm SGL và OpenGL cho đồ họa 2D và 3D
- SQLite cho các ứng dụng sử dụng CSDL
- SSL và WebKit cho việc tích hợp trình duyệt và bảo mật Internet

Bộ thực thi Android (Android runtime) - là môi trường cho các ứng dụng Android có thể chạy được, bao gồm các thư viện lõi Android (gần như tương đồng với các thư viện lõi của ngôn ngữ Java), các thư viện dành riêng cho Android, và máy ảo Dalvik như đã mô tả ở phần trước.

Bộ khung ứng dụng (Application framework) - cung cấp tập các class để phát triển các ứng dụng Android. Tầng này cung cấp khả năng giao tiếp với phần cứng thông qua các tầng bên dưới cũng như quản lý giao diện và các tài nguyên của ứng dụng

Tầng ứng dụng (Application layer) - hầu hết các ứng dụng (bao gồm cả ứng dụng cài sẵn trong hệ điều hành và các ứng dụng của các bên thứ 3) đều được phát triển dựa trên tầng này (dùng chung tập API). Tầng ứng dụng được chạy trong bộ thực thi Android (với các thư viện lõi và máy ảo Dalvik), sử dụng các lớp và các dịch vụ cung cấp bởi tầng Application Framework.

Môi trường phát triển

Để có thể bắt đầu lập trình ứng dụng cho hệ điều hành Android, chúng ta cần ít nhất các thứ sau:

Bộ phát triển phần mềm Android (Android SDK) như mô tả ở trên, bao gồm các công cụ phát triển do Google phát triển và các Android API (các API được download riêng bởi công cụ

Môi trường phát triển tích hợp (IDE) dùng cho việc viết mã nguồn với các tính năng như tự động định dạng mã nguồn, tự động hoàn thiện mã nguồn (gợi ý)... Các môi trường IDE thường tự động kết nối đến Android SDK giúp các thao tác biên dịch, gỡ lỗi, kiểm thử và xuất bản ứng dụng trở nên dễ dàng hơn rất nhiều. Các IDE phổ biến cho lập trình Android có thể kể đến như Eclipse (với plugin ADT của Google), IntelliJ IDEA, Android Studio...

Thiết bị chạy Android, thông thường là điện thoại thông minh chạy hệ điều hành Android, dùng để chạy thử ứng dụng trong quá trình phát triển. Bạn cũng có thể dùng trình giả lập thiết bị (**Android emulator**) đi kèm theo bộ kit phát triển nếu không có thiết bị thật, tuy nhiên hiệu năng của trình giả lập thiết bị Android tương đối thấp, và một số tác vụ đặc biệt liên quan đến phần cứng như lập trình đa phương tiện hay truy xuất đến một số cảm biến sẽ không chạy được trên trình giả lập. Hầu hết các ví dụ trong giáo trình này sẽ chạy được tốt trên cả thiết bị thật lẫn trình giả lập, trong trường hợp mã nguồn không chạy được trên trình giả lập sẽ có chú thích riêng.

Bên cạnh trình giả lập thiết bị mặc định đi kèm với bộ kit phát triển, bạn đọc có thể sử dụng một loại máy ảo Android khác là GenyMotion (<http://www.genymotion.com/>) với hiệu năng tốt hơn rất nhiều so với *Android Emulator* mặc định.

Tải và cài đặt môi trường SDK, IDE

Cách thức đơn giản nhất để có được bộ công cụ đầy đủ, cấu hình sẵn sàng cho việc phát triển là tải gói Eclipse có đóng gói sẵn trình cắm ADT (Android Developer Tool) của Google tại địa chỉ <http://developer.android.com/sdk/index.html>, chọn *Download the SDK (ADT Bundle for {OS})*.

The screenshot shows the 'Developer Tools' section of the Android Developers website. On the left is a sidebar with a tree view under 'Download' containing links like 'Setting Up the ADT Bundle', 'Android Studio', 'Exploring the SDK', etc. The main content area has a heading 'Get the Android SDK'. It includes a paragraph about the SDK providing API libraries and developer tools, a link to download the ADT Bundle, and a list of included components. An illustration of an Android character standing on a stack of code blocks is shown to the right.

Sau khi tải xong Eclipse + ADT (bộ đóng gói sẵn của Google), giải nén và chạy file eclipse.exe, giao diện màn hình đang khởi động của IDE này như sau:

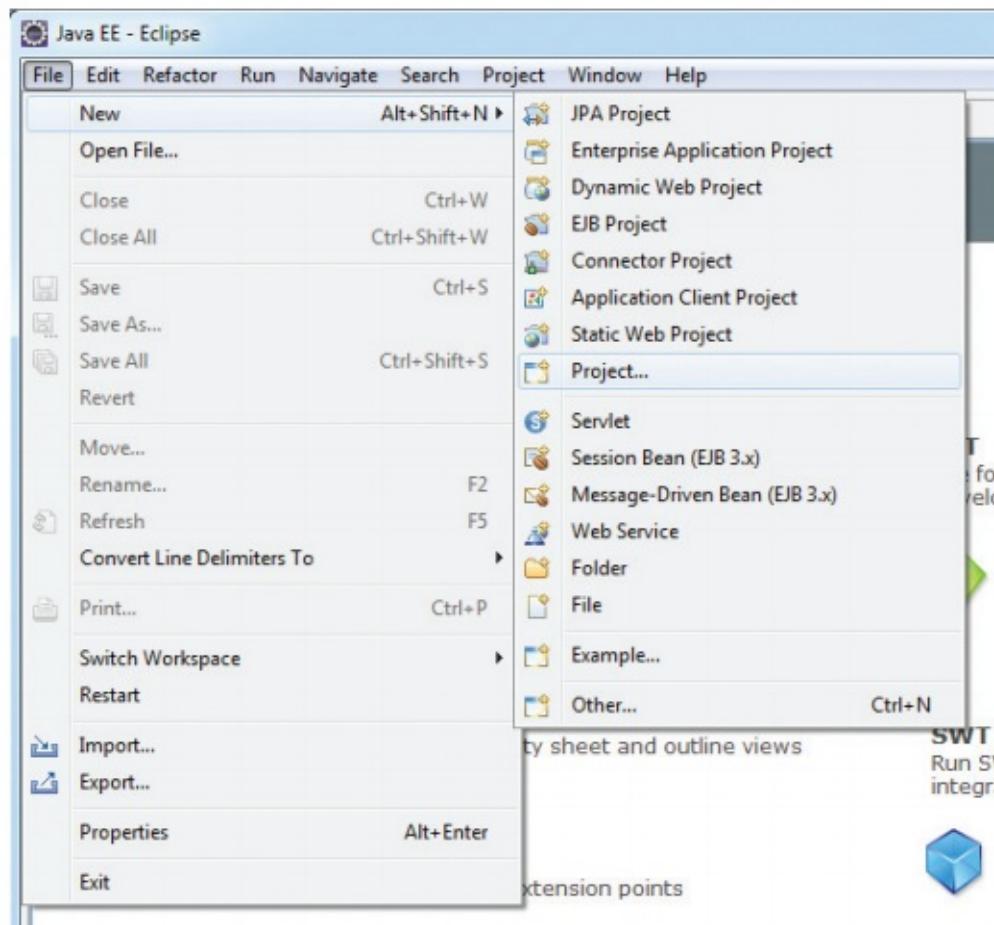


Hình: Eclipse với ADT đang khởi động

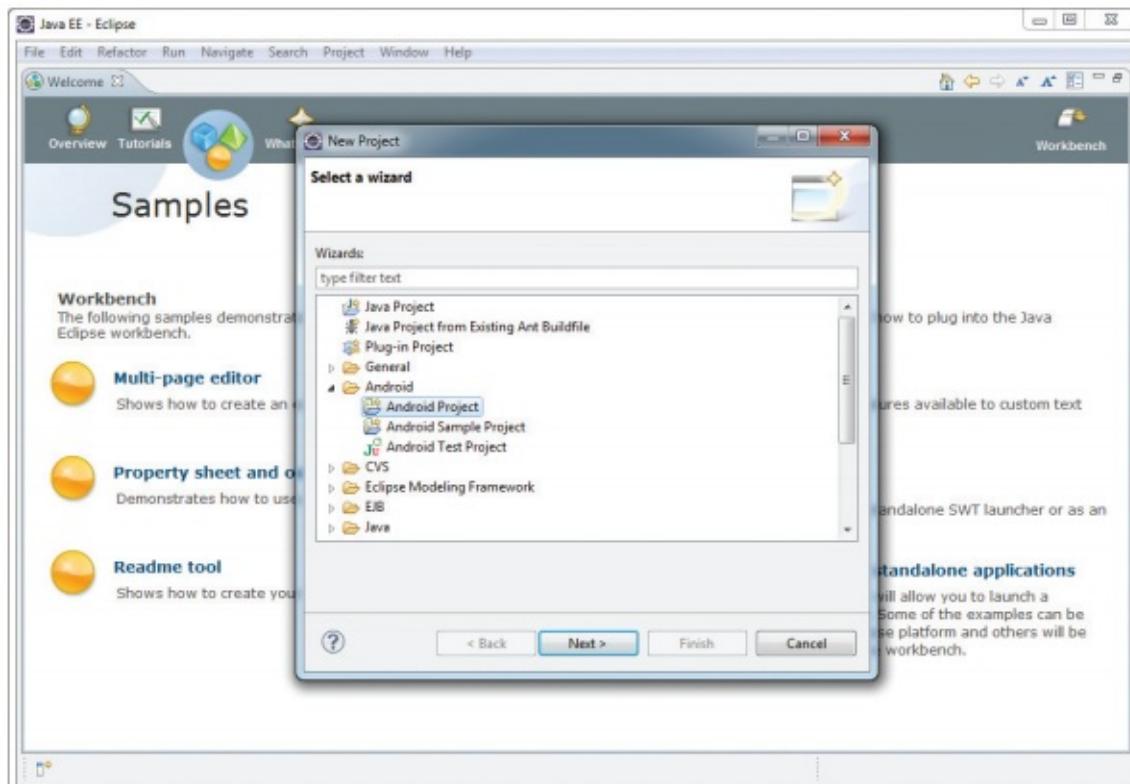
Hello Android (Android “Hello world”)

Để tạo dự án (project) Android mới, chúng ta làm theo các bước sau:

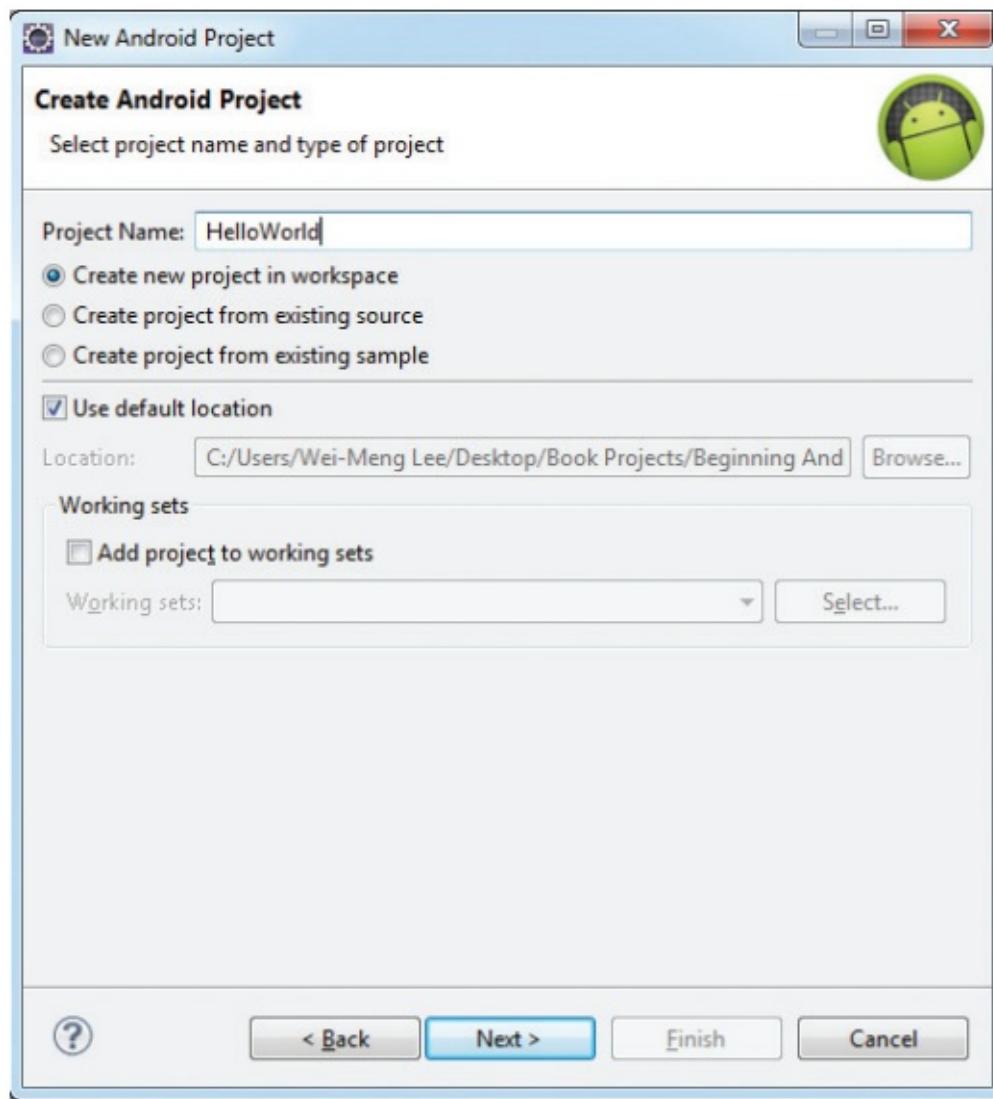
Bước 1. Từ Eclipse, chọn File > New > Project



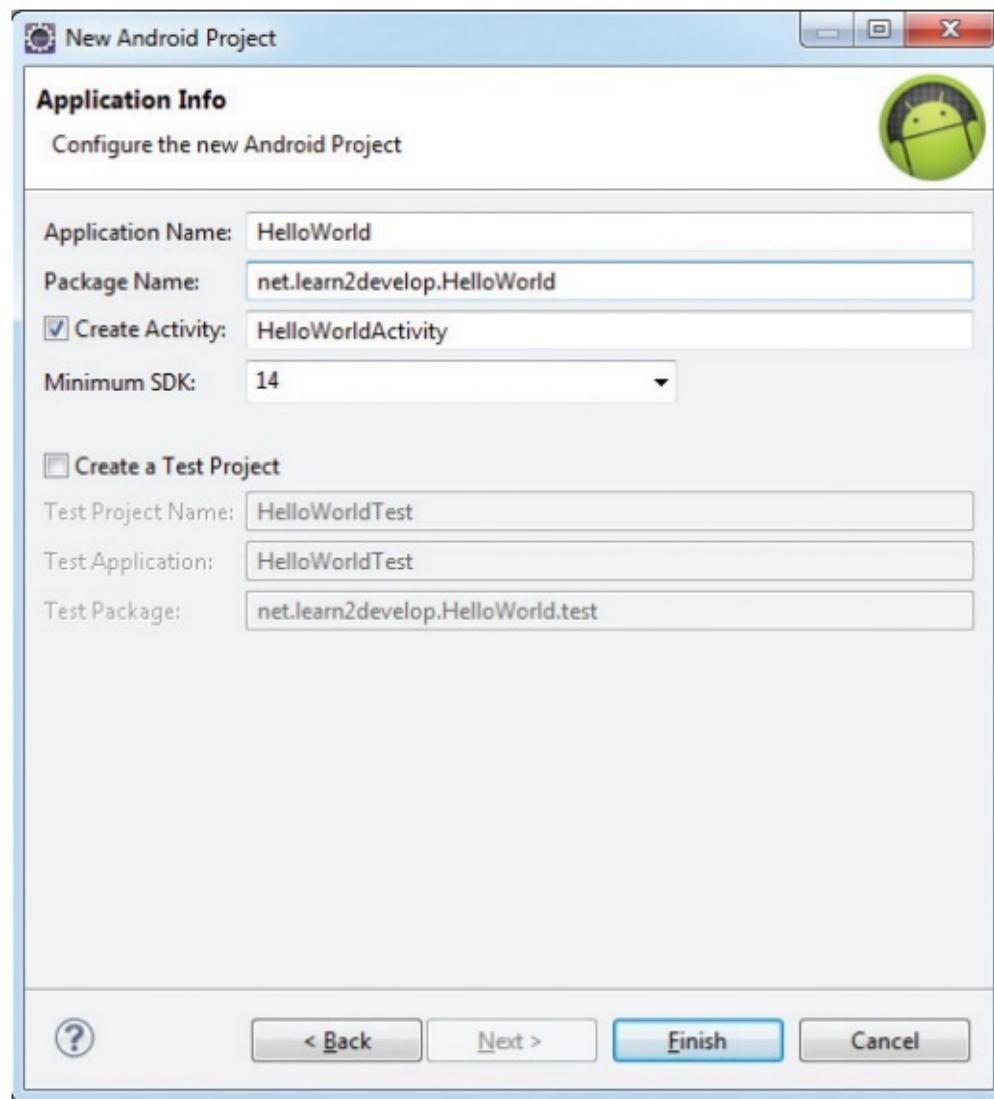
Bước 2. Trong mục Android, chọn “Android project”



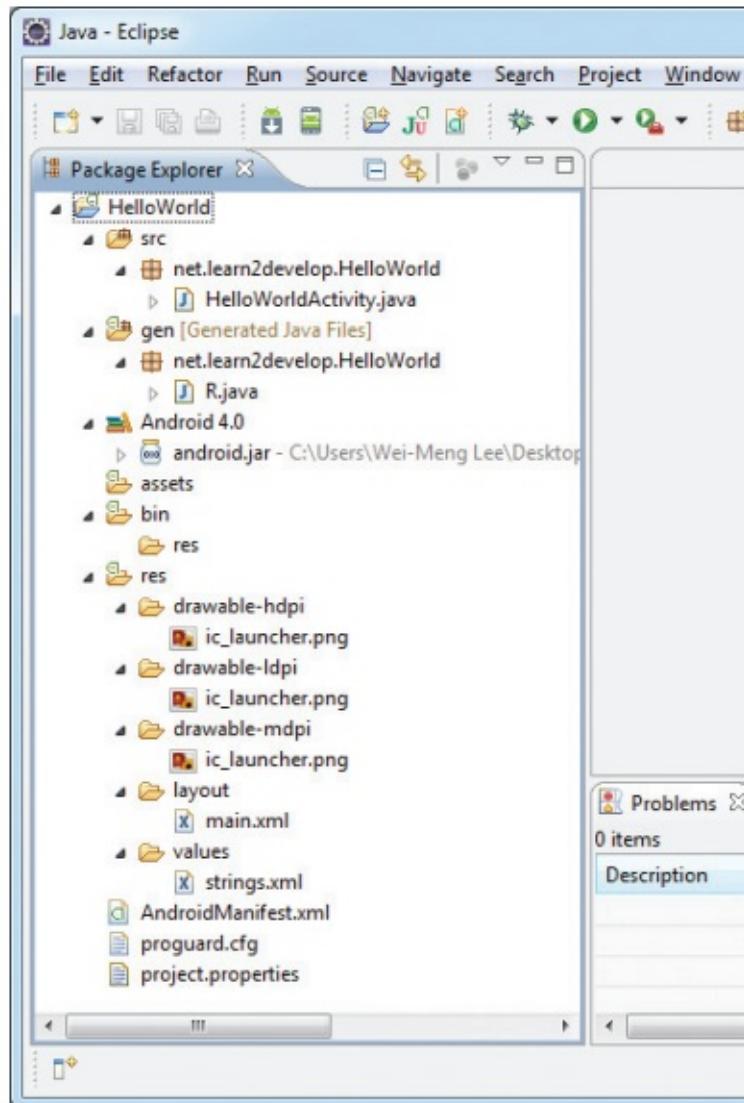
Bước 3. Đặt tên cho dự án là “HelloWorld”

**Bước 4.** Điền thông tin cho ứng dụng cần tạo:

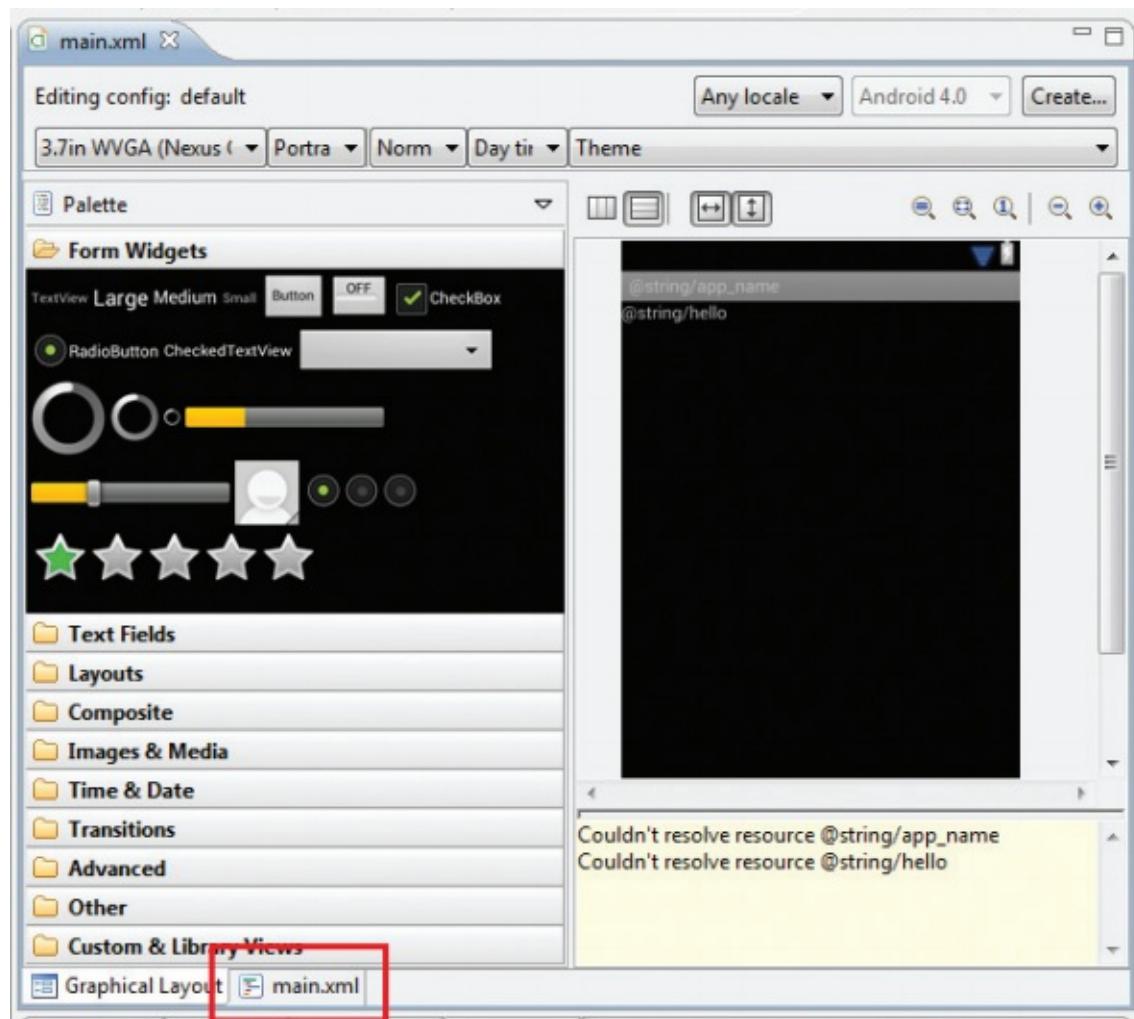
- Tên ứng dụng (sẽ hiển thị trên thiết bị Android)
- Package name: chuỗi định danh duy nhất cho ứng dụng, thường được đặt có định dạng giống với định dạng package của Java
- Chọn "Create Activity" và đặt tên Activity mặc định của ứng dụng. Khái niệm activity trong Android ta sẽ tìm hiểu trong chương tới, ở đây tạm hiểu activity là một "màn hình" hay "form" của ứng dụng
- Chọn mức SDK (tương ứng với phiên bản Android) thấp nhất mà ứng dụng có thể hỗ trợ
- Chọn Finish



Bước 5. Sau khi tạo xong, project mới sẽ xuất hiện trong khung “Package Explorer” của Android như hình dưới:



Bước 6. Trong thư mục `res/layout`, ta thấy có file `main.xml`. Đây là file mô tả giao diện của ứng dụng `HelloWorld` của chúng ta. Kích đúp vào file này, cửa sổ soạn thảo giao diện sẽ được mở ra như hình dưới. Để thay đổi giao diện bằng tay (XML code), ta chuyển sang tab `main.xml`.



Bước 7. Thêm đoạn code dưới đây vào file main.xml:

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

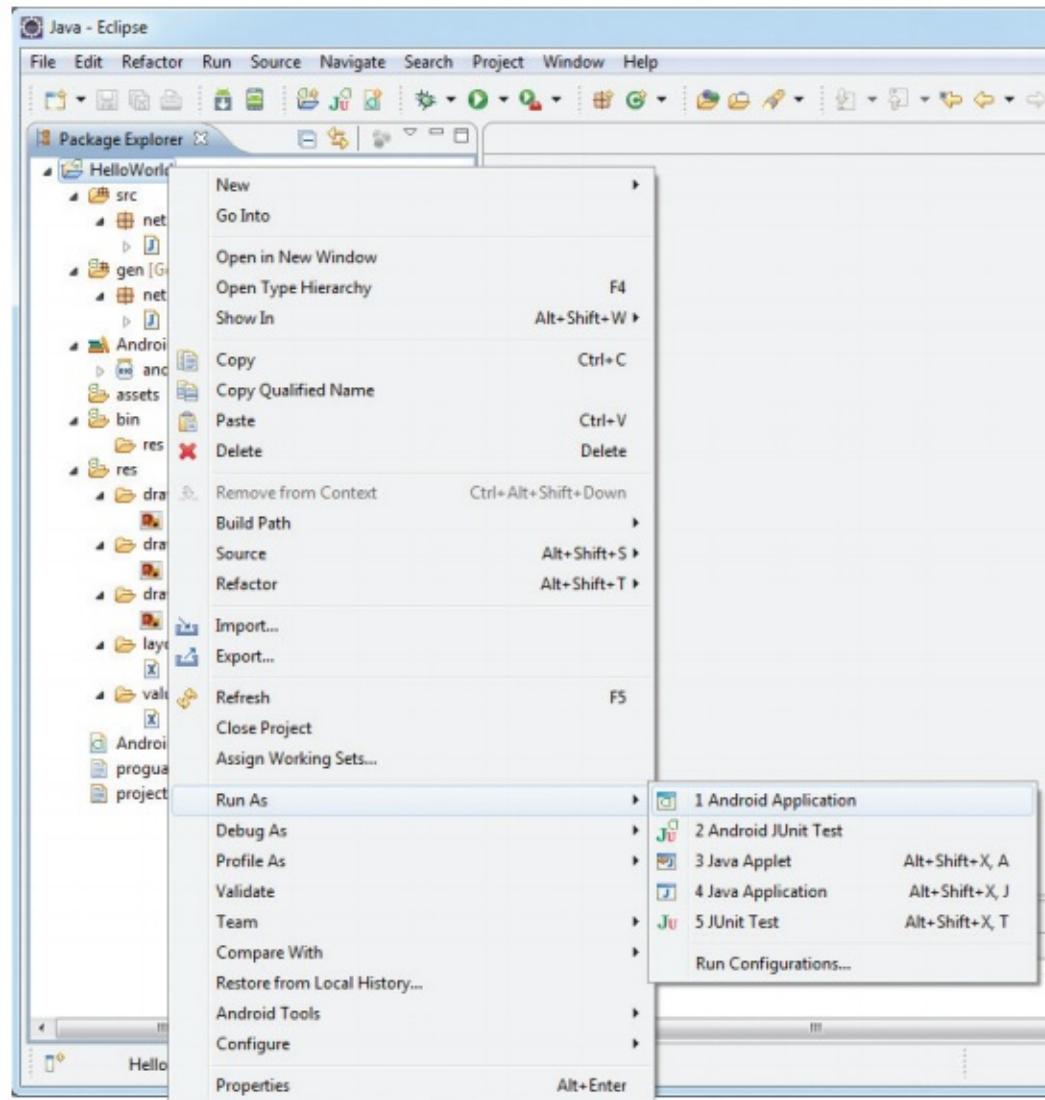
    <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/hello" />

    <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="This is my first Application!" />

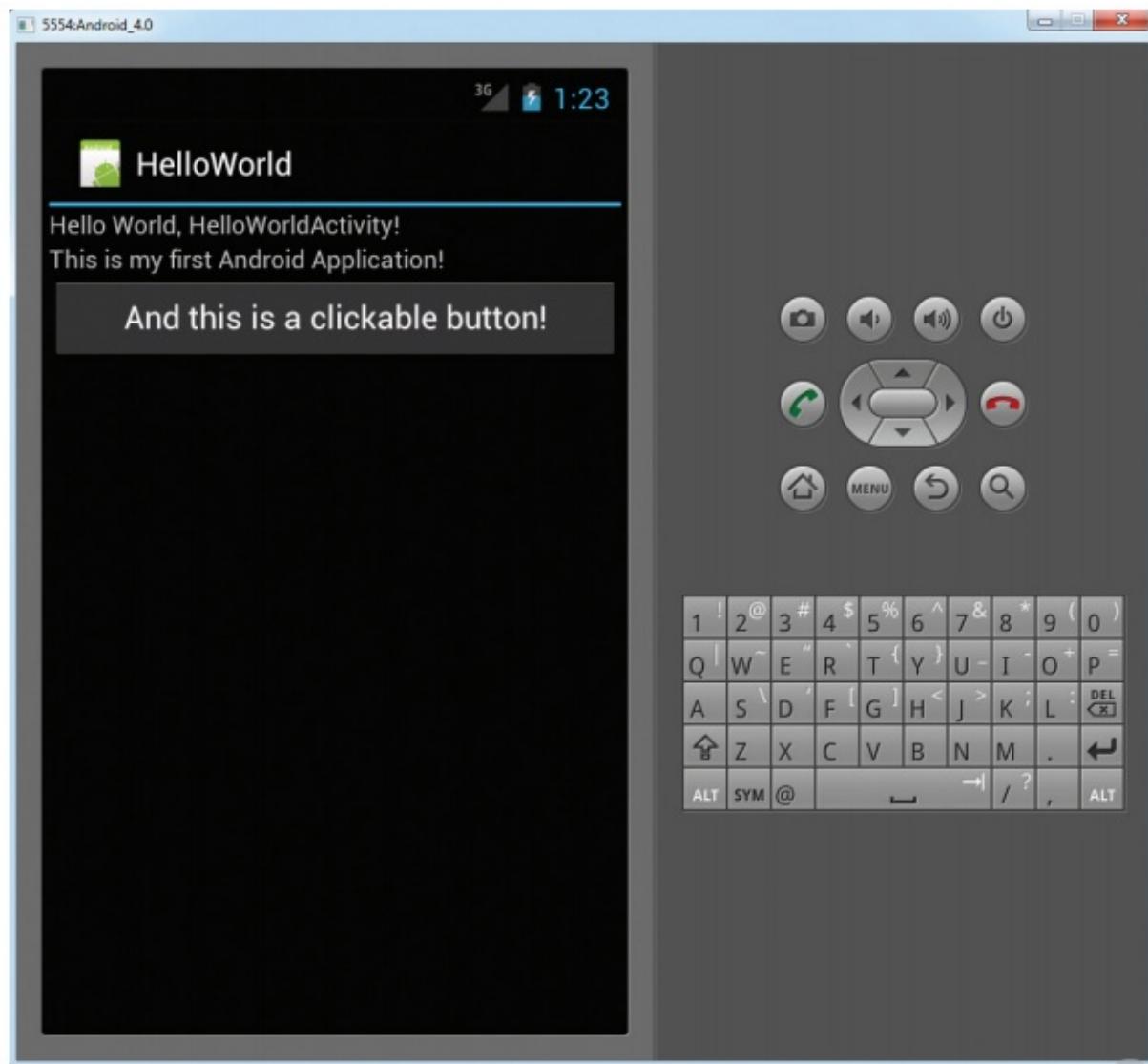
    <Button
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="And this is a clickable button!" />

</LinearLayout>
```

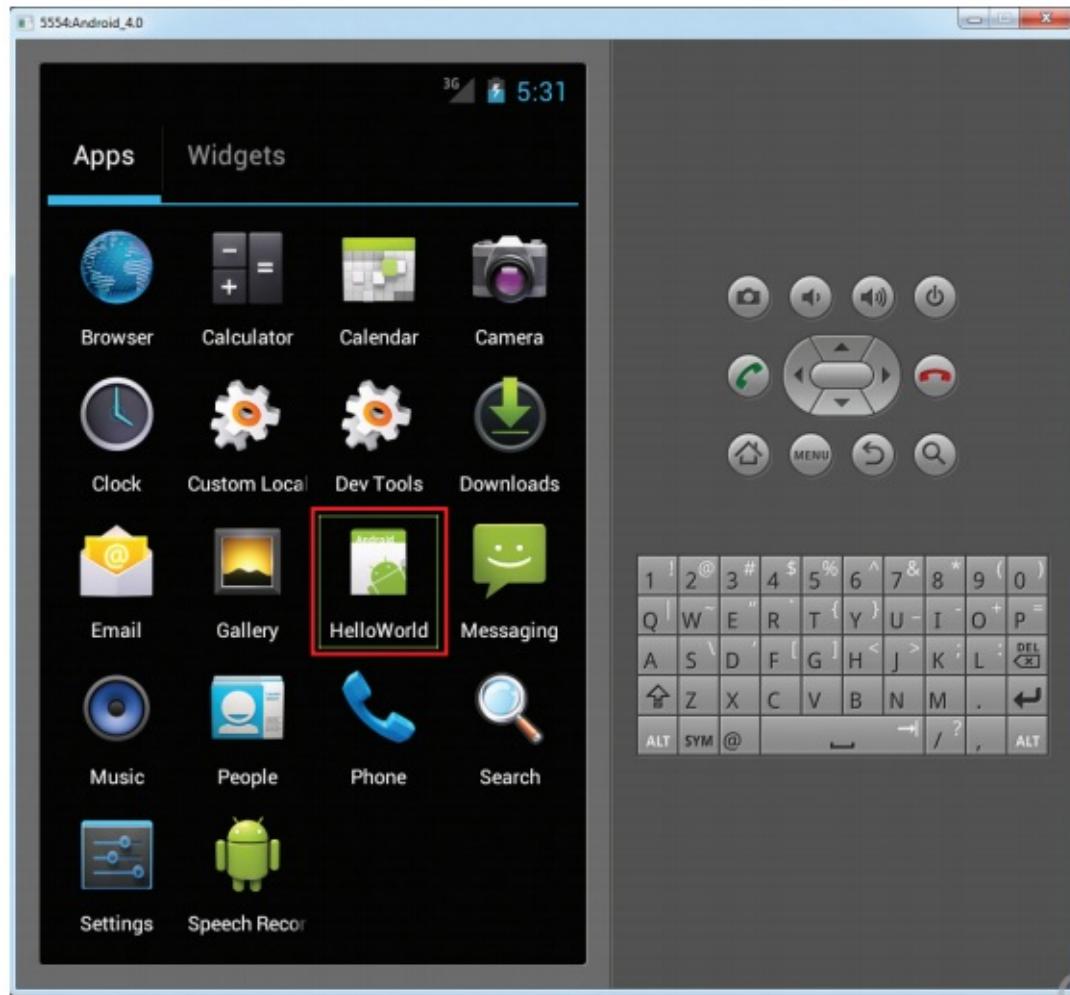
Bước 8. Ghi file (Ctrl+S) và chạy ứng dụng trên thiết bị ảo bằng cách nhấp chuột vào project, chọn Run As > Android Application



Bước 9. Ứng dụng sau khi chạy sẽ có hình như sau:



Bước 10. Click vào nút Home (biểu tượng ngôi nhà trong các phím ảo ở bên phải màn hình emulator) để ẩn ứng dụng và hiển thị màn hình chủ của hệ điều hành Android. Sau đó bấm nút danh sách ứng dụng , ta sẽ thấy ứng dụng HelloWorld của chúng ta xuất hiện trong danh sách các ứng dụng được cài đặt:



Như vậy ứng dụng HelloWorld đã được biên dịch và chạy thành công trên thiết bị ảo (đối với thiết bị thật cũng vậy). Để hiểu được ứng dụng chạy như thế nào, trước tiên ta sẽ tìm hiểu cấu trúc của một ứng dụng Android và các khái niệm cơ bản sử dụng trong dự án này.

Cấu trúc của một dự án Android

Một bộ mã nguồn của ứng dụng Android thường bao gồm các thành phần như sau:

- `src` – thư mục chứa các file mã nguồn Java, các file mã nguồn này được đặt trong các thư mục con của thư mục `src`, tương ứng với package chứa nó. Trong ví dụ ở trên, ta chỉ có 1 file mã nguồn là `HelloWorldActivity.java`, nằm trong package `vn.edu.humg.android_course.HelloWorld`.
- `gen` – thư mục chứa file `R.java`, file này được trình biên dịch tự động sinh ra, chứa tham chiếu đến tất cả các tài nguyên (ảnh, chuỗi, màu sắc, kích thước, layout...) sử dụng trong dự án. Đây là file tự sinh ra, ta không cần tự chỉnh sửa file này trong mọi trường hợp.
- `AndroidManifest.xml` – chứa file `android.jar`, bao gồm tất cả các lớp trong thư viện Android cần thiết cho ứng dụng, file này cũng tự động được gắn vào dự án, tùy thuộc vào mức API bạn chọn trước đó.
- `assets` – thư mục chứa các tài nguyên tĩnh khác được sử dụng trong ứng dụng, như HTML, font, file CSDL...
- `bin` – chứa các file tạm trong quá trình biên dịch, trong đó có file `.apk`, là file nhị phân của ứng dụng Android. File `.apk` là file đã đóng gói toàn bộ mọi thứ cần thiết để chạy ứng dụng. Nội dung thư mục này cũng do trình biên dịch tự động sinh ra, ta không cần can thiệp bằng tay.
- `res` – thư mục chứa tất cả các tài nguyên được sử dụng trong ứng dụng như: hình ảnh (`drawable`), bố cục giao diện (`layout`), các chuỗi, màu sắc, kích thước... (`values`). Chúng ta sẽ xem chi tiết về các loại tài nguyên này trong các chương kế tiếp.
- `AndroidManifest.xml` – là file đặc tả ứng dụng. Mọi ứng dụng Android đều cần có file này. Hệ thống sẽ đọc file này lúc cài đặt để xác định các quyền cần cấp cho ứng dụng, các activity có trong ứng dụng, và nhiều tính năng khác. Chi tiết

về file đặc tả này sẽ được đề cập đến ở phần dưới.

File mã nguồn duy nhất chúng ta sử dụng trong dự án `HelloWorld` là `HelloWorldActivity.java`. Nội dung file này như sau:

```
package vn.edu.humg.android_course.HelloWorld;

import android.app.Activity;
import android.os.Bundle;

public class HelloWorldActivity extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }
}
```

Lớp `HelloWorldActivity` kế thừa từ lớp `android.app.Activity` với cài đặt đè hàm `onCreate` (hàm này được gọi khi Activity được khởi tạo). Trong thân hàm ngoài việc gọi hàm dựng (hàm khởi tạo) của lớp cha, chỉ chứa một dòng duy nhất `setContentView(R.layout.main);` dòng này chỉ ra activity hiện tại sẽ sử dụng giao diện khai báo trong file `main.xml` nằm trong thư mục `res/layout`. File này đã được biên dịch và tham chiếu đến trong file `R.java` dưới tên `R.layout.main`.

Để Activity này có thể được sử dụng và mở ra đầu tiên khi ứng dụng được chạy trên thiết bị, nó cần phải được khai báo trong file `AndroidManifest.xml` như sau:

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

    <uses-sdk android:minSdkVersion="13" />

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

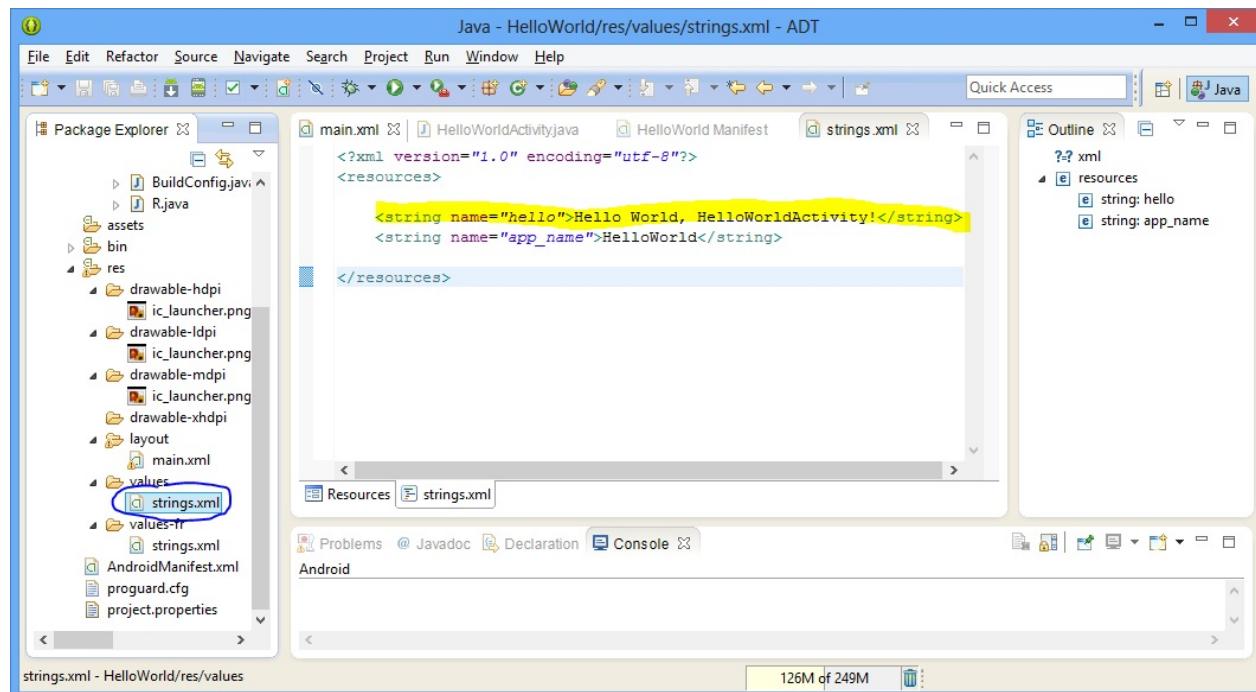
Các thành phần chính của file `AndroidManifest.xml`:

- Khai báo **package name** – định danh duy nhất của ứng dụng trên hệ thống Android (`vn.edu.humg.android_course.HelloWorld`)
- Phiên bản của ứng dụng (tên mã và tên hiển thị của phiên bản): `android:versionCode`, `android:versionName`. Trong đó `versionCode` là số nguyên, được sử dụng cho hệ thống, nhằm xác định khi nào ứng dụng có phiên bản mới hơn, còn `versionName` là chuỗi chứa tên của phiên bản hiển thị lên cho người dùng.
- `android:minSdkVersion` – khai báo mức API thấp nhất mà ứng dụng cần để chạy, thiết bị chạy Android với phiên bản thấp hơn mức API này sẽ không cài đặt được ứng dụng đang phát triển.
- `android:icon="@drawable/ic_launcher"` – khai báo sử dụng file ảnh `ic_launcher.png` trong thư mục `res/drawable` để làm biểu tượng cho ứng dụng
- `android:label="@string/app_name"` – khai báo sử dụng chuỗi có tên là `app_name` trong file `string.xml` làm tên của ứng dụng (chuỗi này có giá trị là `"HelloWorld"`)

- File manifest này cũng khai báo các **activity** có trong ứng dụng, trong ví dụ của chúng ta chỉ có 1 activity là `.HelloWorldActivity`, dấu chấm (.) ở đầu tên Activity chỉ ra rằng activity này nằm trong package cùng tên với package name của ứng dụng (`vn.edu.humg.android_course.HelloWorld.HelloWorldActivity`). Bên trong khai báo Activity này có thêm mục `intent-filter` với `<action android:name= "android.intent.action.MAIN"/>` chỉ ra rằng đây là Activity cần mở ra đầu tiên khi chạy ứng dụng, và `<category android:name= "android.intent.category.LAUNCHER"/>` chỉ ra rằng Activity này sẽ được tạo biểu tượng (shortcut) trên màn hình chủ của Android.

File `main.xml` trong thư mục `res/layout` (như đã mô tả ở trên) chứa mô tả giao diện của activity `HelloWorldActivity`, bao gồm khai báo 2 đoạn chữ (`TextView`) (`@string/hello` và `"This is my first Android Application!"`) và một nút bấm với nội dung `"And this is a clickable button!"`. Ba thành phần này được sắp xếp thứ tự từ trên xuống dưới do chúng được đặt trong một `LinearLayout` với tham số hướng là theo chiều dọc (`android:orientation="vertical"`).

Ta có thể để ý thấy `TextView` thứ 2 được khai báo nội dung dạng chữ một cách tường minh (`"This is my first Android Application!"`), trong khi `TextView` đầu tiên được khai báo bằng tên của hằng số chuỗi nằm trong thư mục tài nguyên của ứng dụng (`res/values/string.xml`) – xem hình dưới. Đây là cách cung cấp tài nguyên phổ biến trong Android, giúp cho việc quản lý các hằng số một cách tập trung và dễ dàng cho việc địa phương hóa (localization) các chuỗi (đa ngôn ngữ). Chúng ta sẽ tìm hiểu thêm về vấn đề này ở các chương sau.



Chúng ta kết thúc dự án `HelloWorld` ở đây. Chương tiếp theo sẽ tìm hiểu sâu hơn về thành phần quan trọng nhất của ứng dụng Android là các Activity và một số khái niệm liên quan.

Các Activity, Fragment và Intent

Trong chương trước, chúng ta đã biết *Activity* là một cửa sổ (hay màn hình, form) chứa giao diện của ứng dụng Android. Đây là thành phần phổ biến và quan trọng nhất trong mỗi ứng dụng Android. Thông thường mỗi ứng dụng có thể có ít nhất một hoặc nhiều *activity*, là nơi người dùng tương tác với ứng dụng (tuy nhiên có những ứng dụng đặc biệt không chứa một *activity* nào, như những ứng dụng chạy ngầm không có giao diện). Trong quá trình thực thi ứng dụng, kể từ khi được khởi tạo đến lúc biến mất khỏi màn hình điện thoại, mỗi *activity* sẽ trải qua một số các trạng thái nhất định, gọi là “vòng đời” của *Activity*. Mỗi lập trình viên Android cần phải nắm vững các trạng thái trong vòng đời của các *Activity* để có những xử lý phù hợp như lưu lại trạng thái dữ liệu trước khi *Activity* biến mất hay khôi phục lại trạng thái dữ liệu khi *Activity* được khôi phục...

Ngoài *Activity*, Android từ phiên bản 3.0 (API mức 11) trở lên có giới thiệu thêm khái niệm “mảnh giao diện”, hay *fragment*. **Fragment** có thể được hiểu như các “mảnh ghép” của giao diện ứng dụng – là một phần của *Activity*. Mỗi *Activity* có thể chứa một hoặc nhiều “mảnh ghép” này trong giao diện của mình và mỗi “mảnh ghép” lại có thể được chèn vào nhiều *Activity* khác nhau. Điều này cho phép giảm thiểu tối đa sự trùng lặp về mã nguồn trong việc tạo giao diện ứng dụng thích nghi được với nhiều loại kích cỡ và hướng màn hình. Các ứng dụng viết cho các phiên bản hệ điều hành thấp hơn cũng có thể sử dụng *Fragment* bằng cách sử dụng “thư viện hỗ trợ” của Android (*Android Support Library*). Thư viện này cung cấp các tính năng của các API mới cho các ứng dụng viết cho API mức thấp hơn, trong đó *Android Support Library v4* cho phép các ứng dụng viết cho API 4 (Android 1.6) trở lên sử dụng được *Fragment* và nhiều tính năng khác của các API mức cao hơn.

Chúng ta sẽ tìm hiểu kỹ hơn về *Activity* và *Fragment* trong phần tiếp theo của chương này. Một khái niệm quan trọng khác của hệ điều hành Android là **Intent**. *Intent* được hiểu một cách nôm na giống như “keo dán” giữa các thành phần của hệ điều hành, giúp các *activity* của cùng một ứng dụng hoặc các ứng dụng khác nhau có thể tương tác với nhau để thực hiện công việc một cách trôi chảy, giống như các *activity* đó cùng thuộc một ứng dụng. Người lập trình cũng có thể sử dụng *Intent* để gọi các ứng dụng có sẵn của hệ điều hành như trình duyệt, trình gọi điện, gửi tin nhắn, bản đồ... Chúng ta sẽ tìm hiểu kỹ hơn về các *Activity* và *Intent* trong phần tiếp theo của giáo trình.

Activity

Vòng đời của Activity

Mỗi activity là một lớp (class) kế thừa từ lớp `android.app.Activity` hoặc các lớp kế thừa từ nó. Ta xem lại Activity trong ví dụ ở chương trước:

```
package vn.edu.humg.android_course.HelloWorld;
import android.app.Activity;
import android.os.Bundle;

public class HelloWorldActivity extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }
}
```

Activity này sử dụng giao diện được khai báo trong file bố cục `main.xml`, trong thư mục `res/layout` bằng cách gọi lệnh:
`setContentView(R.layout.main);`

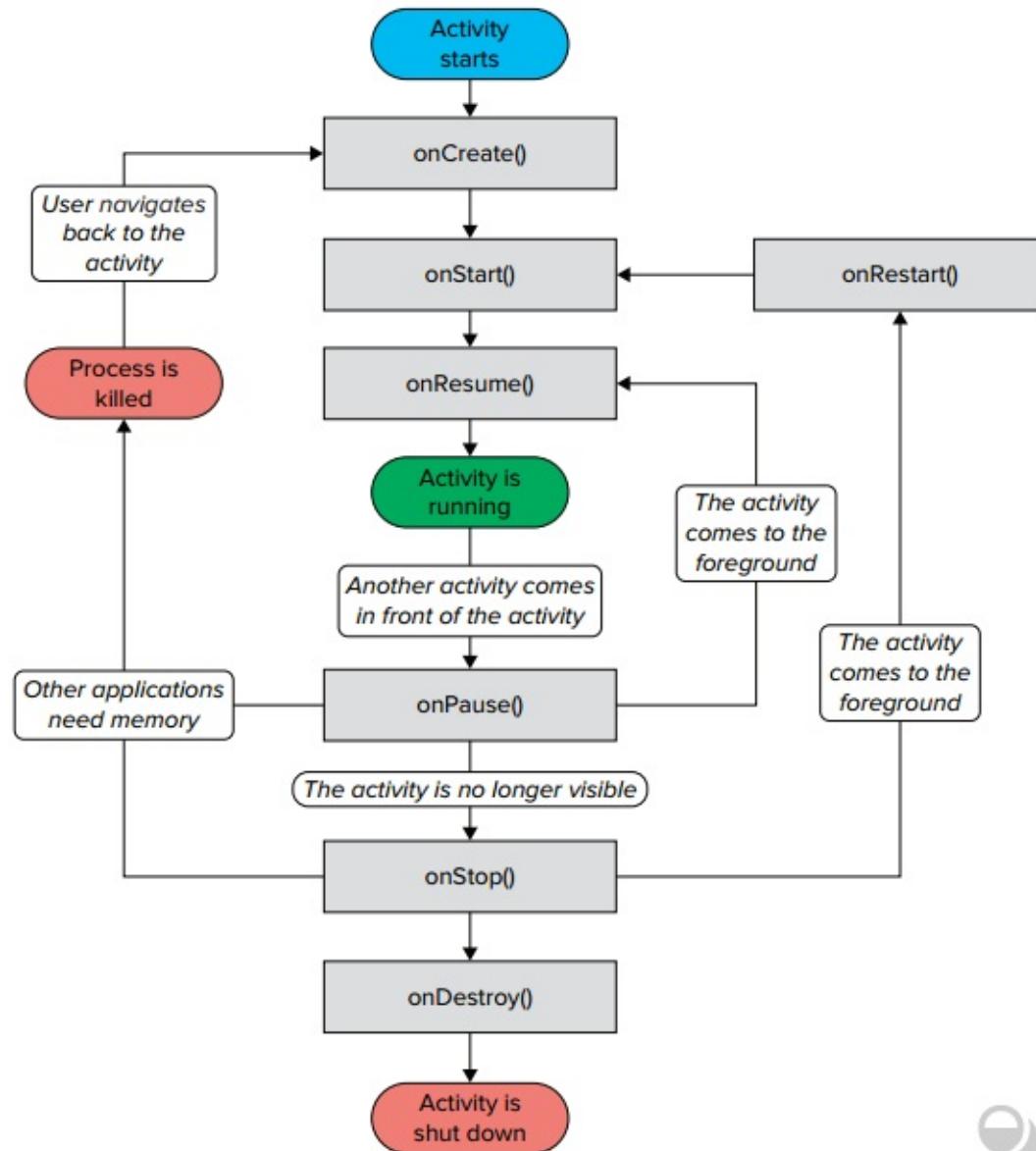
Cũng nhắc lại là mỗi Activity cần được khai báo trong file đặc tả ứng dụng (`AndroidManifest.xml`) như đã đề cập trong chương trước.

Vòng đời của mỗi Activity trải qua các trạng thái nhất định. Tại mỗi trạng thái sẽ có sự kiện tương ứng được gọi đến. Các sự kiện này được khai báo trong lớp cơ sở (`android.app.Activity`). Chúng ta có thể nạp chồng các sự kiện này để theo dõi vòng đời của Activity và có những tác động thích hợp. Các sự kiện như vậy bao gồm:

- `onCreate()` – được gọi khi Activity được khởi tạo
- `onStart()` – được gọi khi Activity bắt đầu hiện ra (chúng ta bắt đầu nhìn thấy giao diện)
- `onResume()` – bắt đầu nhận các tương tác với người dùng
- `onPause()` – được gọi khi activity bị dừng lại để chuyển qua activity khác
- `onStop()` – được gọi khi activity biến mất khỏi màn hình
- `onDestroy()` – được gọi khi activity bị hủy (hủy chủ động hoặc bị hủy bởi hệ thống trong trường hợp hệ điều hành xác nhận thiếu RAM)
- `onRestart()` – được gọi khi activity được khởi động lại sau khi đã bị dừng

Theo mặc định, *Eclipse ADT* sẽ tạo sẵn hàm `onCreate()` cho mỗi Activity mới được tạo ra. Thông thường người lập trình sẽ viết mã nguồn (code) liên quan đến việc khởi tạo giao diện đồ họa trong thân hàm này.

Vòng đời của một Activity được mô tả trong sơ đồ dưới đây:



Hình: Vòng đời của một Activity

Để hiểu rõ hơn vòng đời của một Activity, chúng ta viết một Activity đơn giản, nạp chồng tất cả các hàm sự kiện kể trên và ghi ra log tương ứng:

```

public class Activity101Activity extends Activity {
    String tag = "Lifecycle";

    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        //---hides the title bar---
        requestWindowFeature(Window.FEATURE_NO_TITLE);

        setContentView(R.layout.main);
        Log.d(tag, "In the onCreate() event");
    }

    public void onStart() {
        super.onStart();
        Log.d(tag, "In the onStart() event");
    }

    public void onRestart()

```

```

{
    super.onRestart();
    Log.d(tag, "In the onRestart() event");
}

public void onResume()
{
    super.onResume();
    Log.d(tag, "In the onResume() event");
}

public void onPause()
{
    super.onPause();
    Log.d(tag, "In the onPause() event");
}

public void onStop()
{
    super.onStop();
    Log.d(tag, "In the onStop() event");
}

public void onDestroy()
{
    super.onDestroy();
    Log.d(tag, "In the onDestroy() event");
}
}

```

Chạy ứng dụng này trên thiết bị Android, thực hiện các thao tác bấm phím Home, Back, mở activity khác... và theo dõi thứ tự của các sự kiện trên cửa sổ LogCat sẽ cho ta khái niệm chính xác nhất về vòng đời của các Activity. Phần này coi như bài tập.

Cửa sổ hộp thoại (Dialog)

Trong rất nhiều trường hợp chúng ta cần hiển thị thông báo, hỏi xác nhận của người dùng, hiển thị trạng thái chờ... ở dạng hộp thoại nhanh hiện lên trên Activity hiện tại mà không cần mở ra Activity mới, khi đó ta nạp chồng hàm `onCreateDialog` của lớp cơ sở `android.app.Activity`.

Ví dụ dưới đây mô tả cách tạo một hộp thoại cơ bản:

Bước 1. Tạo project mới trong *Eclipse*, đặt tên là *Dialog*

Bước 2. Thêm một nút bấm vào file `main.xml` với đoạn mã sau:

```

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

    <Button
        android:id="@+id	btn_dialog"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Click to display a dialog"
        android:onClick="onClick" />
</LinearLayout>

```

Bước 3. Để ý tham số `android:onClick="onClick"` của nút bấm `btn_dialog`. Tham số này chỉ ra rằng bạn cần khai vào một phương thức tên là `onClick` trong Activity để xử lý sự kiện khi nút bấm này được chọn.

Bước 4. Thêm đoạn mã sau vào `DialogActivity.java`:

```

public class DialogActivity extends Activity {
    CharSequence[] items = { "Google", "Apple", "Microsoft" };
    boolean[] itemsChecked = new boolean [items.length];

    ProgressDialog progressDialog;

    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }

    public void onClick(View v) {
        showDialog(0);
    }

    @Override
    protected Dialog onCreateDialog(int id) {
        switch (id) {
        case 0:
            return new AlertDialog.Builder(this)
                .setIcon(R.drawable.ic_launcher)
                .setTitle("This is a dialog with some simple text...")
                .setPositiveButton("OK",
                    new DialogInterface.OnClickListener() {
                        public void onClick(DialogInterface dialog, int whichButton) {
                            Toast.makeText(getApplicationContext(), "OK clicked!", Toast.LENGTH_SHORT).show();
                        }
                    })
                .setNegativeButton("Cancel",
                    new DialogInterface.OnClickListener() {
                        public void onClick(DialogInterface dialog, int whichButton) {
                            Toast.makeText(getApplicationContext(), "Cancel clicked!", Toast.LENGTH_SHORT).show();
                        }
                    })
                .setMultiChoiceItems(items, itemsChecked,
                    new DialogInterface.OnMultiChoiceClickListener() {
                        public void onClick(DialogInterface dialog, int which, boolean isChecked) {
                            Toast.makeText(getApplicationContext(), items[which] + (isChecked ? " checked!" : " unchecked!"), Toast.LENGTH_SHORT).show();
                        }
                    }).create();
        }
        return null;
    }
}

```

Bước 5. Chạy ứng dụng và bấm vào nút “Click to display a dialog”. Ta sẽ quan sát thấy hộp thoại tương ứng hiện ra.

Trong ví dụ trên ta đã nạp chồng phương thức `onCreateDialog(int id)` để hiển thị hộp thoại khi có yêu cầu. Trong Activity có thể hiển thị nhiều hộp thoại khác nhau tùy vào tình huống như hộp thoại thông báo, hộp thoại xác nhận, hộp thoại tiến trình..., tham số `id` trong phương thức `onCreateDialog` được dùng để phân biệt hộp thoại nào cần được hiện lên.

Để kích hoạt yêu cầu mở hộp thoại, ta gọi phương thức `showDialog(0)`. Phương thức này được gọi khi ta bấm vào nút “Click to display a dialog” ,

Trong hàm `onCreateDialog`, tùy thuộc vào `id` truyền vào mà ta hiển thị hộp thoại tương ứng, trong ví dụ trên là hộp thoại có `id = 0`. Trong trường hợp này ta sử dụng loại hộp thoại đơn giản nhất là `AlertDialog`. Để tạo ra một hộp thoại loại này, ta tạo ra một đối tượng (`object`) của lớp (`class`) `AlertDialog.Builder` và gọi phương thức `create()` của nó. Đoạn code trên thiết lập 2 nút bấm cho hộp thoại cho hai trường hợp: đồng ý (nút “OK” – `setPositiveButton`) và hủy bỏ (nút “Cancel” – `setNegativeButton`), cũng như thiết lập các hộp tùy chọn (`checkbox`) bằng `setMultiChoiceItems`.

Các hàm xử lý sự kiện của `Dialog` trong ví dụ chỉ đơn giản là hiển thị lên màn hình dòng thông báo dạng text trong một khoảng thời gian ngắn. Dạng thông báo này trong Android gọi là `Toast` - là đoạn chữ hiển thị ở giữa, phía dưới màn hình trong khoảng vài giây. `Toast` thường được dùng để hiển thị các loại thông báo ngắn, ít quan trọng như thông báo SMS đã được gửi, thông báo một tiến trình ngầm đã hoàn thành...

Ta có thể tạo thêm các nút bấm và hiển thị các hộp thoại khác tương ứng bằng cách truyền `id` khác nhau cho hàm `showDialog()`, ví dụ: nút bấm 1 – `showDialog(0)`, nút bấm 2 – `showDialog(1)` ... Có rất nhiều các loại hộp thoại khác nhau trong hệ điều hành Android như hộp thoại chờ (với biểu tượng “quay quay”) hay hộp thoại thể hiện tiến trình (có thanh tiến trình theo %), thậm chí bạn có thể tạo riêng một bố cục (*layout*) bằng file *.xml* cho hộp thoại. Bạn đọc tự tìm hiểu thêm. Phần này coi như bài tập.

Đối tượng Context trong Android

Trong Android, rất nhiều phương thức lấy một trong những tham số là một đối tượng của lớp *Context*. Trong ví dụ trên, hàm dựng của lớp `AlertDialog.Builder` và hàm `Toast.makeText` đều nhận tham số kiểu *Context*. Đối tượng kiểu *Context* này thường dùng để chỉ ra phạm vi, hay ngữ cảnh của phương thức được gọi và thường trả đến ứng dụng hoặc activity hiện tại.

Ở ví dụ trên ta thấy đối tượng của lớp `AlertDialog.Builder` được gọi với tham số `this`: `new AlertDialog.Builder(this)`. Đối tượng `this` ở đây là Activity hiện tại (Activity cũng là lớp con cháu của lớp *Context*), trong khi hàm `Toast.makeText` nhận tham số *Context* là `getBaseContext()` chứ không được dùng con trỏ `this` nữa, bởi vì phương thức này được gọi bên trong đối tượng của lớp `DialogInterface.OnClickListener()`, con trỏ `this` ở đây có kiểu `OnClickListener` chứ không phải Activity, nên không phải là *Context*. Vì vậy, ta dùng hàm `getBaseContext()` để lấy *Context* cơ sở của đối tượng hiện tại. Một cách khác có thể sử dụng là dùng `DialogActivity.this`.

Intent và việc tương tác giữa các Activity

Như đã đề cập ở các phần trên, mỗi ứng dụng Android có thể không có, có một hoặc nhiều Activity. Khi ứng dụng có nhiều hơn một Activity thì việc điều hướng từ Activity này sang Activity khác và ngược lại là việc rất cần thiết và được thực hiện rất thường xuyên. Trong Android, việc điều hướng này được thực hiện thông qua một cơ chế rất đặc thù, gọi là **Intent**.

Sử dụng Intent

Trước tiên ta xem xét một ví dụ đơn giản nhất sử dụng Intent để mở một Activity khác từ Activity hiện tại. Để thực hiện điều này ta tạo một project chứa 2 Activity, trong Activity thứ nhất có một nút bấm, khi bấm vào nút này sẽ mở Activity thứ 2. Các bước cụ thể như sau:

Bước 1. Tạo project *UsingIntent* và tạo 2 Activity: `UsingIntentActivity` (Activity mặc định) và `SecondActivity`, 2 activity này được khai báo trong `AndroidManifest.xml` như sau:

```
<activity
    android:label="@string/app_name"
    android:name=".UsingIntentActivity" >
    <intent-filter >
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</activity>
<activity
    android:label="Second Activity"
    android:name=".SecondActivity" >
    <intent-filter >
        <action android:name="vn.edu.humg.android_course.SecondActivity" />
        <category android:name="android.intent.category.DEFAULT" />
    </intent-filter>
</activity>
```

Bước 2. Tạo layout riêng cho `SecondActivity` bằng cách copy file `main.xml` thành file `secondactivity.xml` và sửa nội dung `TextView` như bên dưới. Sau đó trong hàm `onCreate` của `SecondActivity`, gọi lệnh `setContentView(R.layout.secondactivity);` để khai báo việc sử dụng file layout này cho `SecondActivity`.

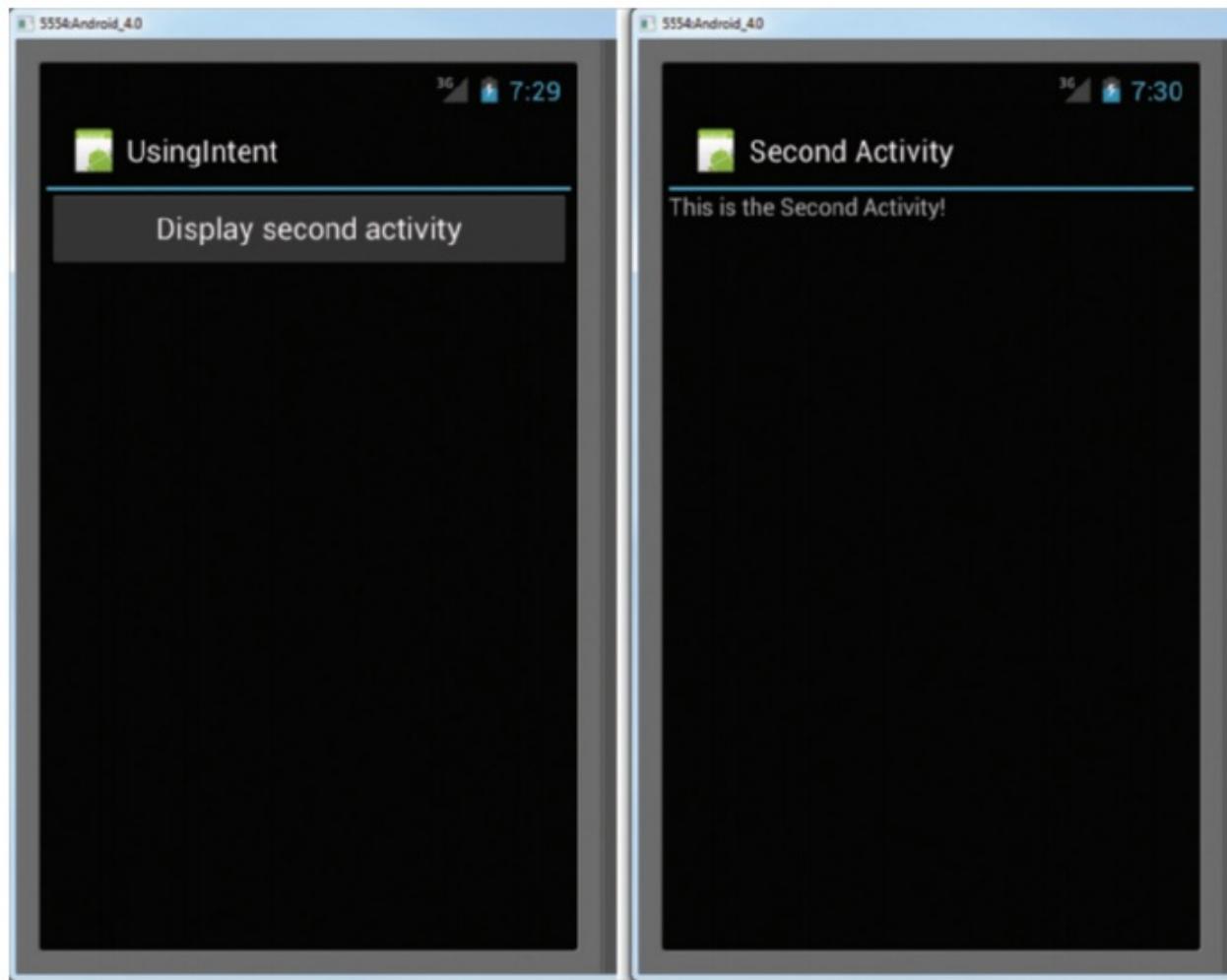
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical" >

    <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="This is the Second Activity!" />
</LinearLayout>
```

Bước 3. Tạo một nút bấm trong `UsingIntentActivity` và trong hàm xử lý sự kiện `onClick` của nút bấm này, ta gọi lệnh mở Activity thứ 2 như sau:

```
startActivity(new Intent("vn.edu.humg.android_course.SecondActivity"));
```

Bước 4. Chạy ứng dụng và bấm vào nút bấm duy nhất trong `UsingIntentActivity`, ta sẽ thấy `SecondActivity` được mở ra:



Trong ví dụ trên, ta thấy dòng lệnh để mở ra Activity thứ 2 là `startActivity(new Intent("vn.edu.humg.android_course.SecondActivity"));`. Trong tham số của hàm dựng `Intent` không hề chỉ ra tên Class của `SecondActivity` mà chỉ ra “**action**” cần thực hiện, khi đó hệ điều hành sẽ quét toàn bộ các Activity của toàn bộ các ứng dụng được cài đặt trên thiết bị và lọc ra Activity có khai báo action tương ứng (`vn.edu.humg.android_course.SecondActivity`) để mở nó. Trong trường hợp của chúng ta, action của `SecondActivity` đã được khai báo trong phần `intent-filter` của activity trong file `AndroidManifest.xml` (`<action android:name="vn.edu.humg.android_course.SecondActivity" />`) nên sẽ được mở ra. Ngoài ra cũng phải chú ý là, để activity có thể được mở ra bằng phương thức `startActivity` như trong ví dụ, thì trong `Manifest` cũng phải khai báo thêm danh mục (`category`) `android.intent.category.DEFAULT` trong phần `intent-filter` (`<category android:name="android.intent.category.DEFAULT" />`). Trong trường hợp Activity cần mở nằm trong cùng ứng dụng với Activity mở nó (như trong ví dụ trên của ta), ta có thể gọi `startActivity` với `Intent` tường minh như sau:

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

Giải quyết “xung đột Intent”

Trong ví dụ trên ta thấy để mở một Activity, ta chỉ cần truyền vào action của Activity đó. Hệ thống sẽ quét tìm kiếm Activity được khai báo với action như vậy để mở ra. Vậy điều gì sẽ xảy ra nếu có nhiều hơn 1 Activity khai báo với cùng một action như vậy?

Chẳng hạn ta khai báo thêm một Activity thứ 3 với cùng `intent-filter` giống như của `SecondActivity`:

```
<activity
    android:label="Third Activity"
    android:name=".ThirdActivity" >
```

```

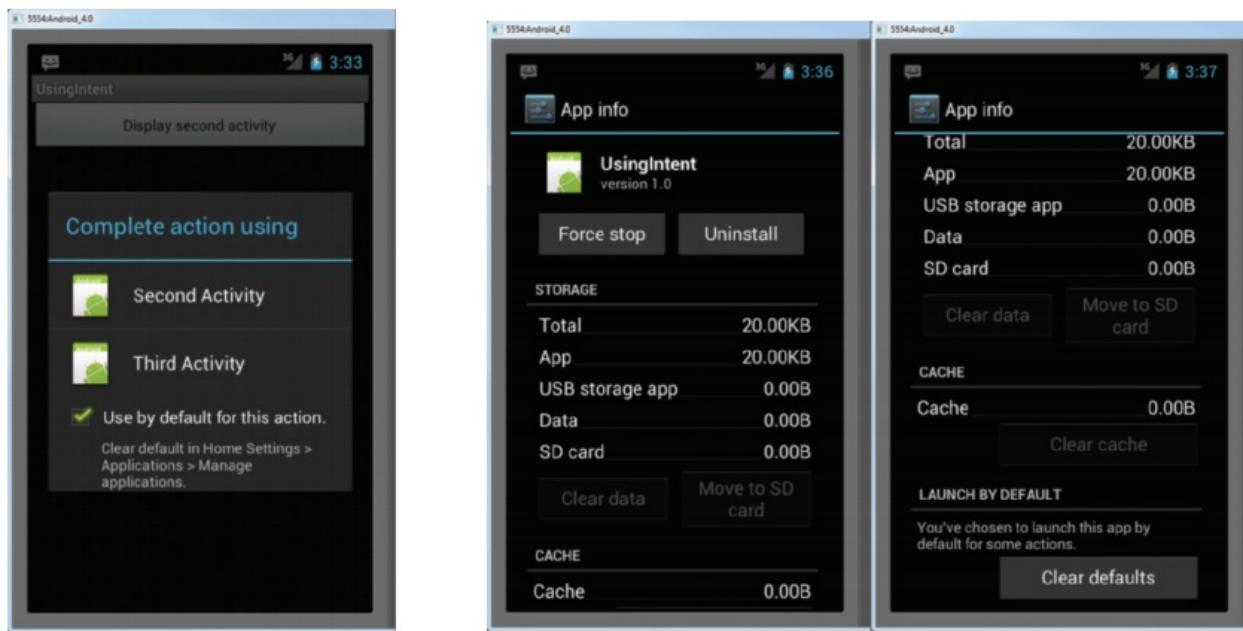
<intent-filter>
    <action android:name="vn.edu.humg.android_course.SecondActivity" />
    <category android:name="android.intent.category.DEFAULT" />
</intent-filter>
</activity>

```

Và giữ nguyên lời gọi Activity như cũ:

```
startActivity(new Intent("vn.edu.humg.android_course.SecondActivity"));
```

Khi được gọi đến, hệ thống sẽ hiển thị danh sách các Activity phù hợp theo bộ lọc Intent (trong trường hợp này là 02 Activity) để người dùng chọn. Kèm theo cửa sổ chọn sẽ là hộp chọn “*Use by default for this action*”. Nếu bạn chọn hộp chọn này thì hệ thống sẽ ghi nhớ lựa chọn của bạn và không hỏi lại cho các lần sau nữa, nếu không cửa sổ chọn Activity cần mở sẽ hiển thị mỗi lần bạn gọi hàm `startActivity` kể trên. Nếu đã lỡ chọn ghi nhớ lựa chọn mặc định này, bạn có thể xóa ghi nhớ này bằng cách vào mục *Settings* của hệ thống, chọn *Apps > Manage Applications*, sau đó chọn ứng dụng đang dùng (*UsingIntent*) và chọn nút “*Clear defaults*” ở phía cuối màn hình. Các hình ảnh dưới đây minh họa các tình huống nói trên:



Các ứng dụng mặc định sẵn có của hệ điều hành Android cũng được mở ra bằng cơ chế Intent với các bộ lọc như vậy. Cơ chế này làm hệ điều hành Android trở nên vô cùng mềm dẻo: mọi thành phần của hệ điều hành đều được đổi xử như nhau và đều có khả năng “thay thế” được. Ví dụ, bạn có thể viết một ứng dụng xử lý tin nhắn *SMS* thay cho ứng dụng *Message* mặc định bằng cách khai báo action cho Activity của bạn trùng với action của ứng dụng tin nhắn mặc định (các intent action này được mô tả rất đầy đủ trong các tài liệu tham khảo của Google). Như vậy, khi có yêu cầu gửi tin nhắn từ một ứng dụng bất kỳ, hệ thống sẽ liệt kê ra các ứng dụng có khả năng xử lý yêu cầu này cho người dùng lựa chọn, bao gồm ứng dụng tin nhắn mặc định và ứng dụng chúng ta mới viết.

Lấy kết quả trả về từ Activity thông qua Intent

Trong rất nhiều trường hợp, ta mở Activity mới với mục đích yêu cầu thêm dữ liệu từ người dùng, như yêu cầu người dùng nhập vào tên truy cập và mật khẩu... Trong trường hợp đó, sau khi người dùng kết thúc nhập liệu và quay trở lại Activity trước đó, dữ liệu người dùng vừa nhập vào cần phải được truyền về Activity ban đầu để xử lý. Để làm được điều này, trong Activity thứ nhất (*UsingIntentActivity* ở trên) thay vì mở Activity thứ 2 (*SecondActivity*) bằng phương thức `startActivity`, ta cần gọi phương thức `startActivityForResult` và khai báo thêm phương thức `onActivityResult` để hứng sự kiện khi *SecondActivity* đóng lại và trả về dữ liệu cho nó. Bên cạnh đó, trong *SecondActivity* cũng phải chứa đoạn mã trả về dữ liệu người dùng nhập vào thông qua Intent. Để minh họa cho quá trình trên, ta thực hiện các bước sau:

Bước 1. Trong SecondActivity, thêm một ô nhập liệu và một nút bấm như sau:

```
<EditText
    android:id="@+id/txt_username"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content" />

<Button
    android:id="@+id	btn_OK"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="OK"
    android:onClick="onClick"/>
```



Bước 2. Trong hàm xử lý sự kiện `click` của nút OK của SecondActivity, ta thêm đoạn mã cho Activity gốc (Activity gọi nó, trong ví dụ là `UsingIntentActivity`) trước khi đóng lại:

```
Intent data = new Intent();
//---get the EditText view---
EditText txt_username = (EditText) findViewById(R.id.txt_username);

//---set the data to pass back---
data.setData(Uri.parse(txt_username.getText().toString()));
setResult(RESULT_OK, data);

//---closes the activity-
finish();
```

Bước 3. Trong `UsingIntentActivity`, thay vì mở SecondActivity bằng phương thức `startActivity`, ta dùng `startActivityForResult` kèm theo tham số thứ hai là một mã yêu cầu (*request code*) tự quy định (chọn bằng 1 chẳng hạn):

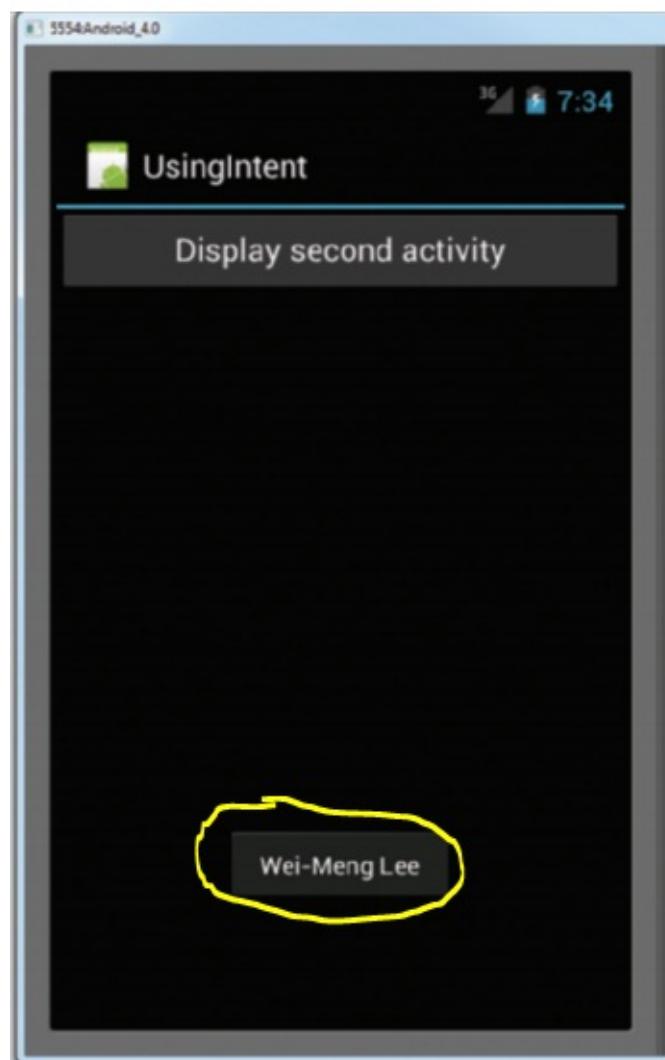
```
startActivityForResult(new Intent("vn.edu.humg.android_course.SecondActivity"), 1);
```

Bước 4. Và khai báo nạp chồng hàm hứng sự kiện Activity trả kết quả về (`onActivityResult`). Khi nhận được kết quả

(dạng chuỗi) từ `SecondActivity`, với mục đích minh họa, ta chỉ hiển thị text này lên màn hình dưới dạng `Toast message`:

```
@Override
public void onActivityResult(int requestCode, int resultCode, Intent data)
{
    if (requestCode == 1) {
        if (resultCode == RESULT_OK) {
            Toast.makeText(this,data.getData().toString(), Toast.LENGTH_SHORT).show();
        }
    }
}
```

Bước 5. Kết quả thu được khi chạy ứng dụng sẽ như hình bên dưới:



Truyền dữ liệu giữa các Activity với Intent

Ngoài việc lấy dữ liệu trả về của Activity, một thao tác phổ biến khác là truyền dữ liệu cho Activity mới được mở ra. Ví dụ trong trường hợp trên, khi gọi mở `SecondActivity` ta có thể truyền chuỗi mặc định cần hiển thị sẵn trong ô nhập liệu của Activity này. Để làm được điều này, trong `UsingIntentActivity`, trước khi gọi `startActivity`, ta gắn dữ liệu cần truyền vào đối tượng Intent dùng để mở `SecondActivity` bằng phương thức `putExtra` của đối tượng `Intent` như sau:

```
Intent i = new Intent("vn.edu.humg.android_course.PassingDataSecondActivity");
//--use putExtra() to add new key/value pairs--
i.putExtra("str1", "This is a string");
```

```
i.putExtra("age1", 25);

//---use a Bundle object to add new key/values pairs---
Bundle extras = new Bundle();
extras.putString("str2", "This is another string");
extras.putInt("age2", 35);

//---attach the Bundle object to the Intent object---
i.putExtras(extras);

//---start the activity
startActivityForResult(i);
```

Khi đó, trong hàm `onCreate` của `SecondActivity`, ta có thể lấy dữ liệu được truyền sang bằng cách lấy `Intent` qua hàm `getIntent()` và lần lượt lấy ra các trường dữ liệu tương ứng bằng các phương thức `getStringExtra` , `getIntExtra` , `getExtras` ... như sau:

```
//---get the data passed in using getStringExtra()---
Toast.makeText(this, getIntent().getStringExtra("str1"), Toast.LENGTH_SHORT).show();

//---get the data passed in using getIntExtra()---
Toast.makeText(this, Integer.toString(getIntent().getIntExtra("age1", 0)), Toast.LENGTH_SHORT).show();

//---get the Bundle object passed in---
Bundle bundle = getIntent().getExtras();

//---get the data using the getString()---
Toast.makeText(this, bundle.getString("str2"), Toast.LENGTH_SHORT).show();

//---get the data using the getInt() method---
Toast.makeText(this, Integer.toString(bundle.getInt("age2")), Toast.LENGTH_SHORT).show();
```

Sử dụng Intent để gọi các ứng dụng sẵn có của hệ điều hành

Như đã nói ở trên, các ứng dụng mặc định sẵn có của hệ điều hành Android cũng được mở ra bằng cơ chế Intent như các ứng dụng thông thường. Từ ứng dụng của mình, chúng ta có thể dùng Intent để mở trình duyệt với một website đã chọn, hay mở ứng dụng gửi tin nhắn với số điện thoại và nội dung tin nhắn chọn sẵn.... Phần này sẽ đưa ra vài ví dụ cụ thể sử dụng Intent trong các việc như vậy.

Mở trình duyệt web:

```
Intent i = new Intent("android.intent.action.VIEW");
i.setData(Uri.parse("http://www.amazon.com"));
startActivity(i);
```

Kết quả:



Mở ứng dụng gọi điện thoại:

```
Intent i = new Intent(android.content.Intent.ACTION_DIAL, Uri.parse("tel:+651234567"));
startActivity(i);
```



Mở ứng dụng bản đồ và điều hướng đến vị trí nhất định:

```
Intent i = new Intent(android.content.Intent.ACTION_VIEW, Uri.parse("geo:37.827500,-122.481670"));
startActivity(i);
```



Đối tượng Intent

Ở các ví dụ trên ta đã xem qua một số cách sử dụng Intent khác nhau, phần này sẽ tổng hợp lại mô tả chi tiết hơn về đối tượng Intent.

Mỗi đối tượng *Intent* (Intent Object) có thể chứa các thông tin sau:

- Action (hành động)
- Data (dữ liệu)
- Type (kiểu dữ liệu)
- Category (loại)

Trong ví dụ trên, ta đã thấy để mở Activity khác, ta cần truyền Action của activity đó trong hàm dựng của đối tượng *Intent* như sau:

```
startActivity(new Intent("vn.edu.humg.android_course.SecondActivity"));
```

Action ở đây ("vn.edu.humg.android_course.SecondActivity") còn được gọi là "component name". Trong trường hợp Activity cần mở nằm cùng dự án (project) với Activity hiện tại, ta có thể gọi như sau:

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

Activity cũng có thể được gọi bằng cách truyền Action và dữ liệu (Data) kèm theo như trong trường hợp mở trình duyệt ở trên:

```
Intent i = new Intent(android.content.Intent.ACTION_VIEW,
Uri.parse("http://www.amazon.com"));
startActivity(i);
```

Phần “hành động” mô tả việc chúng ta cần làm, còn phần “dữ liệu” chứa thông tin cần thiết cho Activity sắp được mở ra xử lý. Trong ví dụ trên, ta tạo Intent với yêu cầu cần hiển thị thông tin, với dữ liệu đi kèm là địa chỉ *url* của một website. Hệ thống Android sẽ lọc trong tất cả các ứng dụng đã được cài đặt trên hệ thống và liệt kê ra các ứng dụng có thể xử lý được Intent này để người dùng lựa chọn.

Intent trên cũng có thể được truyền dữ liệu tường minh hơn như sau:

```
Intent i = new Intent("android.intent.action.VIEW");
i.setData(Uri.parse("http://www.amazon.com"));
```

Một số Intent không cần truyền dữ liệu cụ thể, mà chỉ cần truyền theo kiểu dữ liệu. Ví dụ, để lấy một bản ghi trong danh sách danh bạ, ta có thể dùng Intent như sau:

```
Intent i = new
Intent(android.content.Intent.ACTION_PICK);
i.setType(ContactsContract.Contacts.CONTENT_TYPE);
```

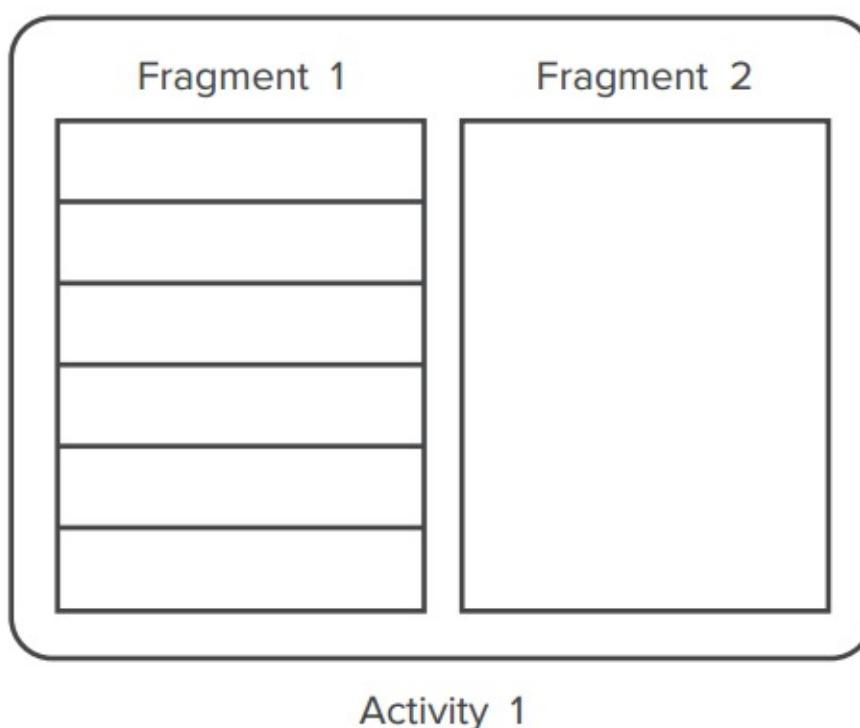
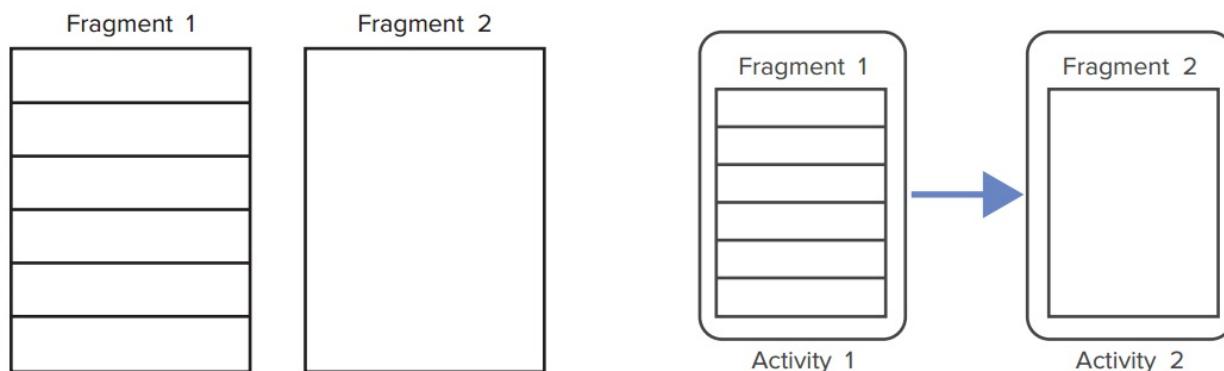
Hàm `setType()` chỉ ra kiểu dữ liệu (*MIME data type*) của thông tin trả về cho activity hiện tại. Kiểu dữ liệu của `ContactsContract.Contacts.CONTENT_TYPE` là xâu `"vnd.android.cursor.dir/contact"`.

Ngoài hành động, dữ liệu và kiểu dữ liệu, mỗi đối tượng *Intent* có thể có thêm tham số “loại” (*category*) dùng để nhóm các Activity lại theo loại để thuận tiện cho việc lọc các Activity theo nhu cầu của ứng dụng. Bạn đọc có thể tìm hiểu kỹ hơn về bộ lọc Activity ở các tài liệu khác. Phần này coi như bài tập.

Fragment

Trong phần trước ta đã tìm hiểu qua về Activity, là giao diện của một “màn hình” ứng dụng, mỗi màn hình chỉ chứa 01 Activity. Tuy nhiên, khi các máy tính bảng ra đời với màn hình lớn hơn rất nhiều so với điện thoại truyền thống, cho phép thiết kế với nhiều loại view khác nhau, và phát sinh nhu cầu dùng lại các view này trên các màn hình khác nhau (điện thoại và máy tính bảng). Khái niệm *fragment* (mảnh) được sinh ra nhằm phục vụ nhu cầu đó. Có thể hiểu fragment như các “tiểu Activity”, chứa tập hợp các view khác bên trong nó. Fragment luôn luôn được chứa trong một Activity hoặc fragment khác, mỗi Activity có thể chứa một hoặc nhiều fragment . Một ví dụ điển hình của việc sử dụng fragment là trường hợp thiết kế “master-detail”, bao gồm 2 view: view chính (master) chứa danh sách các đối tượng (danh sách tiêu đề các bài báo chẳng hạn), và view chi tiết (detail), hiển thị nội dung chi tiết của đối tượng (bài báo) đang được chọn. Mỗi view như vậy được đặt trong 1 fragment. Trên màn hình điện thoại, do kích thước hạn chế, 2 fragment này sẽ nằm trong 2 activity khác nhau, trong khi đối với màn hình máy tính bảng, 2 fragment này nằm trên cùng một Activity (xem hình dưới). Thiết kế này giúp việc dùng lại mã được tối đa. Tầng logic của ứng dụng (nằm trong 2 fragment), được dùng lại cho cả điện thoại và máy tính bảng, còn Activity chỉ là vỏ chứa tối thiểu mã nguồn.

Khái niệm fragment mới được đưa vào từ phiên bản *Android 3.0 HoneyComb*. Tuy nhiên tính năng này cũng được Google bổ sung cho các API thấp hơn (từ Level 4) thông qua thư viện hỗ trợ *Android Support Library v4*.



Sau đây là một ví dụ về một ứng dụng có 2 fragment nói trên.

Bước 1. Tạo project mới, đặt tên là *Fragments*

Bước 2. Tạo file *fragment1.xml* trong *res/layout*, đây là file chứa mô tả giao diện của fragment thứ nhất, với nội dung như sau:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:background="#00FF00"
    >
<TextView
    android:id="@+id/lblFragment1"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="This is fragment #1"
    android:textColor="#000000"
    android:textSize="25sp" />
</LinearLayout>
```

Bước 3. Tạo file *fragment2.xml* trong *res/layout*, đây là file chứa mô tả giao diện của fragment thứ hai, với nội dung như sau:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:background="#FFFFE00"
    >
<TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="This is fragment #2"
    android:textColor="#000000"
    android:textSize="25sp" />
<Button
    android:id="@+id/btnGetText"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Get text in Fragment #1"
    android:textColor="#000000"
    android:onClick="onClick" />
</LinearLayout>
```

Bước 4. Sửa nội dung file *main.xml* như sau:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="horizontal" >
<fragment
    android:name="vn.edu.humg.android_course.Fragments.Fragment1"
    android:id="@+id/fragment1"
    android:layout_weight="1"
    android:layout_width="0px"
    android:layout_height="match_parent" />
<fragment
```

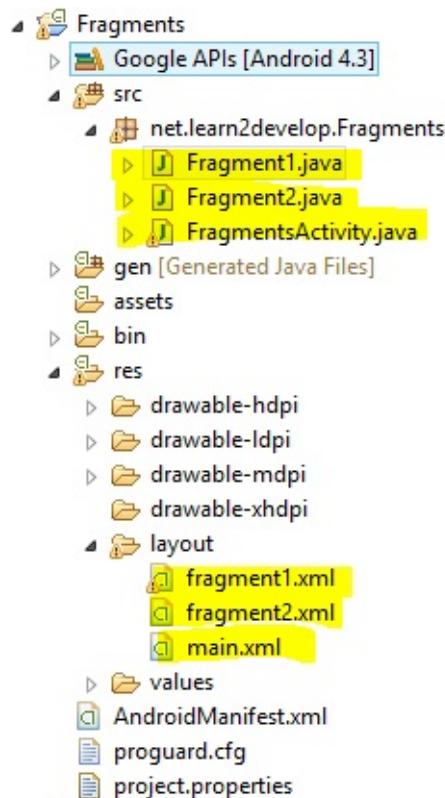
```

    android:name="vn.edu.humg.android_course.Fragments.Fragment2"
    android:id="@+id/fragment2"
    android:layout_weight="1"
    android:layout_width="0px"
    android:layout_height="match_parent" />

</LinearLayout>

```

Bước 5. Tạo file *Fragment1.java* và *Fragment2.java* trong namespace mặc định của ứng dụng:



Bước 6. Nội dung file *Fragment1.java*:

```

public class Fragment1 extends Fragment {
    @Override
    public View onCreateView(LayoutInflater inflater,
    ViewGroup container, Bundle savedInstanceState) {
        Log.d("Fragment 1", "onCreateView");

        //---Inflate the layout for this fragment---
        return inflater.inflate(
            R.layout.fragment1, container, false);
    }
}

```

Bước 7. Nội dung file *Fragment2.java*:

```

public class Fragment2 extends Fragment {
    @Override
    public View onCreateView(LayoutInflater inflater,
    ViewGroup container, Bundle savedInstanceState) {
        //---Inflate the layout for this fragment---
        return inflater.inflate(
            R.layout.fragment2, container, false);
    }

    @Override

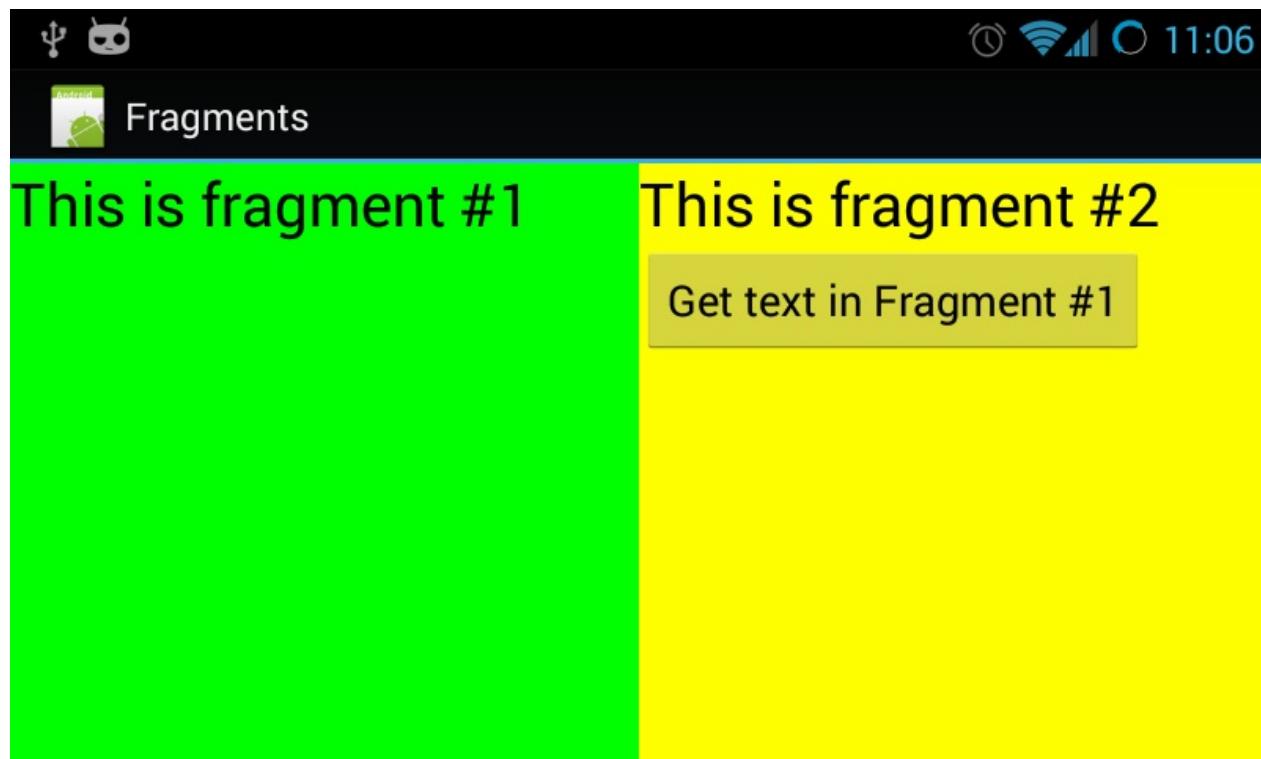
```

```

public void onStart() {
    super.onStart();
    //---Button view---
    Button btnGetText = (Button)
        getActivity().findViewById(R.id.btnGetText);
    btnGetText.setOnClickListener(new View.OnClickListener() {
        public void onClick(View v) {
            TextView lbl = (TextView)
                getActivity().findViewById(R.id.lblFragment1);
            Toast.makeText(getActivity(), lbl.getText(),
                Toast.LENGTH_SHORT).show();
        }
    });
}

```

Bước 8. Sau khi chạy:



Thêm fragment trong thời gian thực thi

Bước 1. Sửa lại file `main.xml`, xóa bỏ khai báo 2 fragment trong này:

```

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

</LinearLayout>

```

Bước 2. Sửa hàm `onCreate` của `FramentActivity` như sau:

```

@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
    FragmentManager fragmentManager = getFragmentManager();
    FragmentTransaction fragmentTransaction =

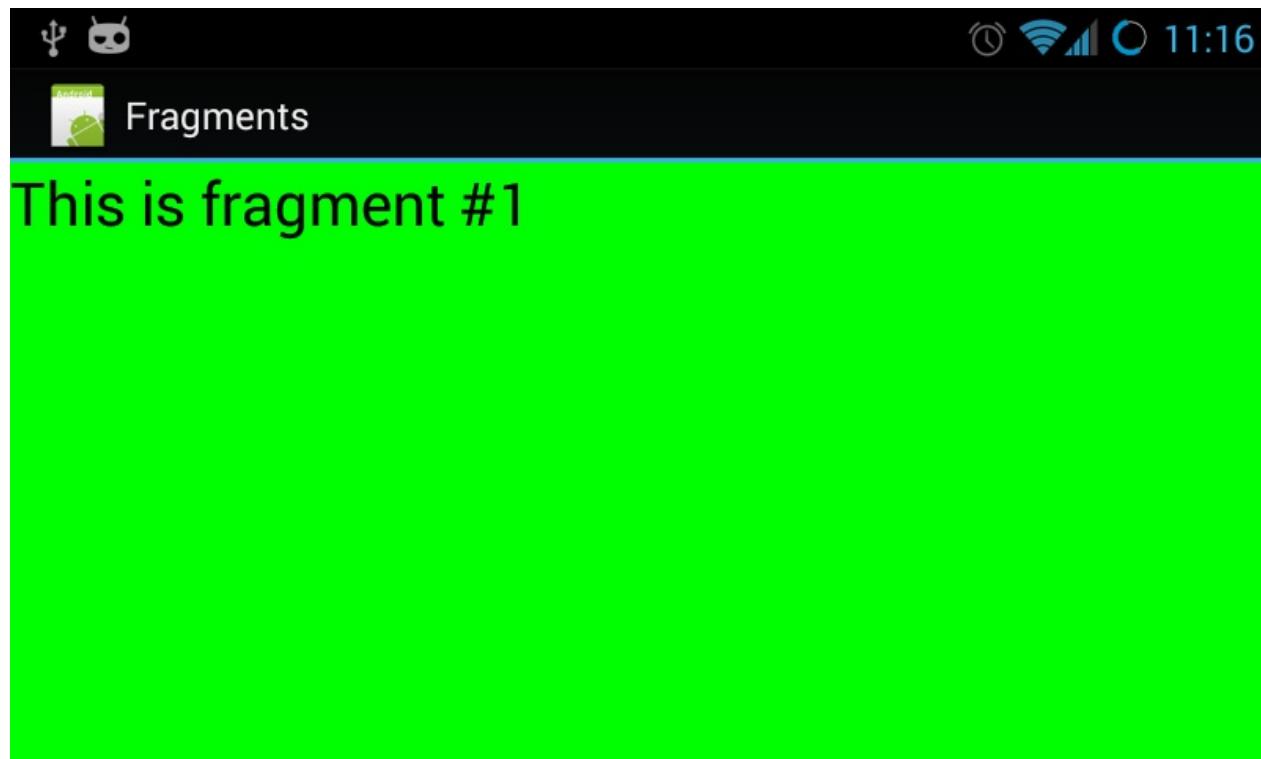
```

```
fragmentManager.beginTransaction();

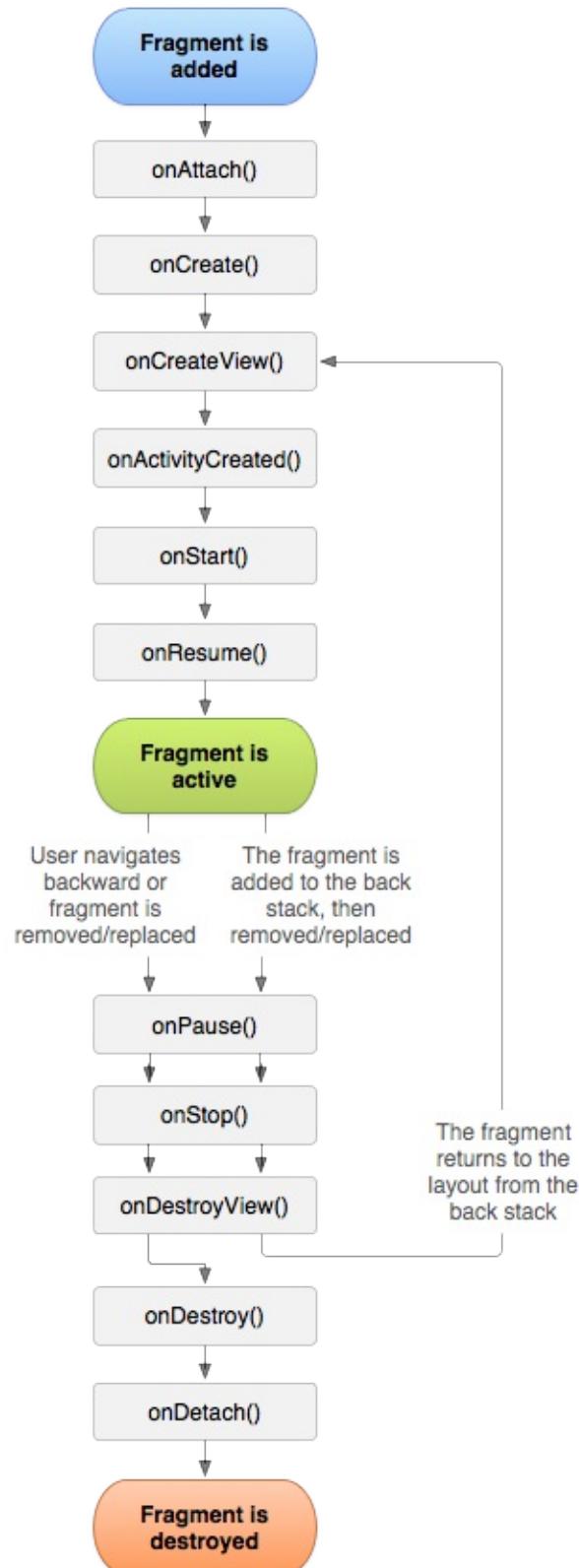
//---get the current display info---
WindowManager wm = getWindowManager();
Display d = wm.getDefaultDisplay();
if (d.getWidth() > d.getHeight())
{
    //---landscape mode---
    Fragment1 fragment1 = new Fragment1();
    // android.R.id.content refers to the content
    // view of the activity
    fragmentTransaction.replace(
        android.R.id.content, fragment1);
}
else
{
    //---portrait mode---
    Fragment2 fragment2 = new Fragment2();
    fragmentTransaction.replace(
        android.R.id.content, fragment2);
}
//---add to the back stack---
fragmentTransaction.addToBackStack(null);
fragmentTransaction.commit();
}
```

Chạy ứng dụng và thay đổi hướng của màn hình (bật *auto-rotate* (tự động xoay) trên thiết bị và quay màn hình theo hướng cần thiết, hoặc bấm Ctrl+F11 nếu đang chạy trên *emulator*), ta sẽ quan sát thấy 2 fragment khác nhau được gắn vào Activity khi thiết bị đang ở 2 hướng khác nhau như hình dưới:





Vòng đời của Fragment



Để hiểu rõ hơn vòng đời của một Fragment, chúng ta nạp chὸng tất cả các hàm sự kiện và ghi ra log (nhật ký):

```

@Override
public void onAttach(Activity activity) {
    super.onAttach(activity);
    Log.d("Fragment 1", "onAttach");
}

@Override

```

```

    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        Log.d("Fragment 1", "onCreate");
    }

    @Override
    public void onActivityCreated(Bundle savedInstanceState) {
        super.onActivityCreated(savedInstanceState);
        Log.d("Fragment 1", "onActivityCreated");
    }

    @Override
    public void onStart() {
        super.onStart();
        Log.d("Fragment 1", "onStart");
    }

    @Override
    public void onResume() {
        super.onResume();
        Log.d("Fragment 1", "onResume");
    }

    @Override
    public void onPause() {
        super.onPause();
        Log.d("Fragment 1", "onPause");
    }

    @Override
    public void onStop() {
        super.onStop();
        Log.d("Fragment 1", "onStop");
    }

    @Override
    public void onDestroyView() {
        super.onDestroyView();
        Log.d("Fragment 1", "onDestroyView");
    }

    @Override
    public void onDestroy() {
        super.onDestroy();
        Log.d("Fragment 1", "onDestroy");
    }

    @Override
    public void onDetach() {
        super.onDetach();
        Log.d("Fragment 1", "onDetach");
    }
}

```

Chạy ứng dụng này trên thiết bị Android, thực hiện các thao tác bấm phím Home, Back, mở ứng dụng khác... và theo dõi thứ tự của các sự kiện trên trong cửa sổ *LogCat* sẽ cho bạn khái niệm chắc chắn nhất về vòng đời của các fragment:

```

11-03 23:24:29.398: D/Fragment 1(11354): onAttach
11-03 23:24:29.398: D/Fragment 1(11354): onCreate
11-03 23:24:29.398: D/Fragment 1(11354): onCreateView
11-03 23:24:29.398: D/Fragment 1(11354): onActivityCreated
11-03 23:24:29.398: D/Fragment 1(11354): onStart
11-03 23:24:29.398: D/Fragment 1(11354): onResume
11-03 23:24:33.835: D/Fragment 1(11354): onPause
11-03 23:24:33.835: D/Fragment 1(11354): onStop
11-03 23:24:33.835: D/Fragment 1(11354): onDestroyView
11-03 23:24:33.835: D/Fragment 1(11354): onDestroy
11-03 23:24:33.835: D/Fragment 1(11354): onDetach
11-03 23:24:33.914: D/Fragment 1(11354): onAttach

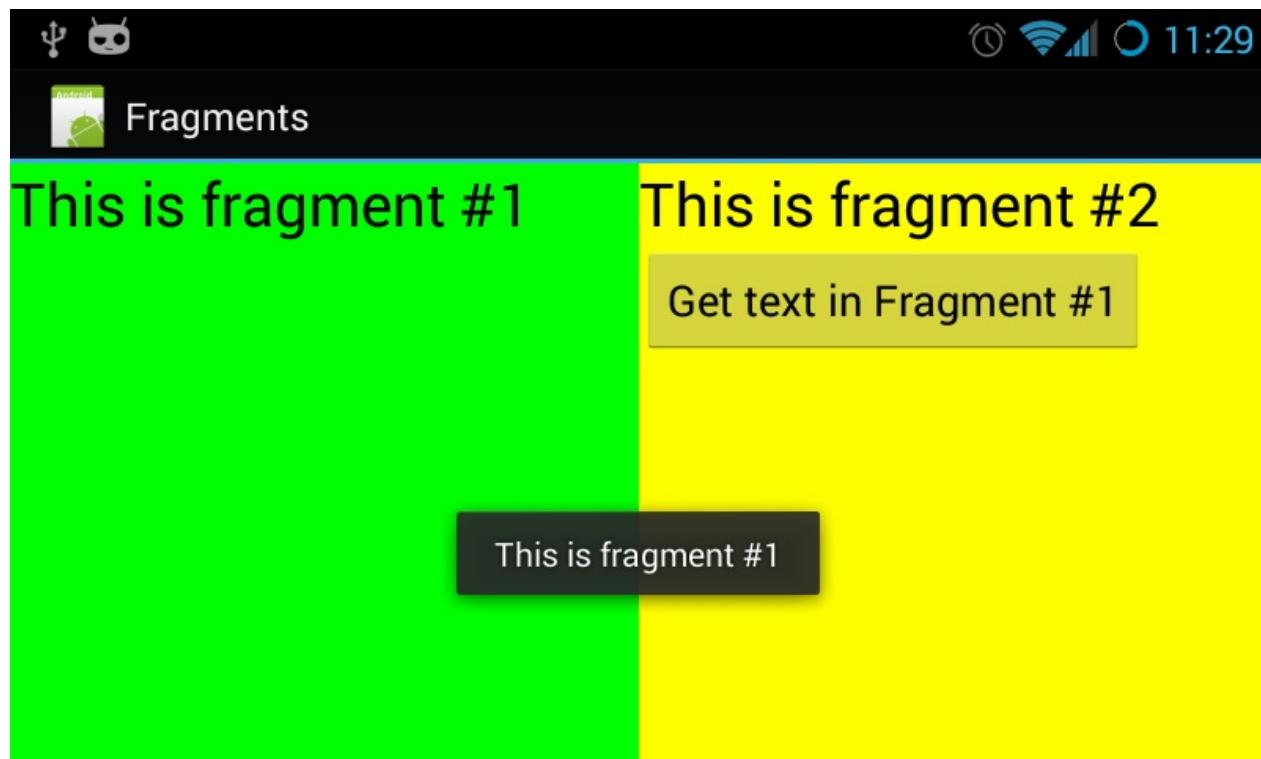
```

Tương tác giữa các fragment

Trong ví dụ trên, khi thực hiện bấm vào nút bấm trong Fragment2, ta tiến hành lấy và hiển thị xâu ký tự chứa trong `TextView` của Fragment1 dưới dạng Toast:

```
@Override  
public void onStart() {  
    super.onStart();  
    //---Button view---  
    Button btnGetText = (Button)  
        getActivity().findViewById(R.id.btnGetText);  
    btnGetText.setOnClickListener(new View.OnClickListener() {  
        public void onClick(View v) {  
            TextView lbl = (TextView)  
                getActivity().findViewById(R.id.lblFragment1);  
            Toast.makeText(getActivity(), lbl.getText(),  
                Toast.LENGTH_SHORT).show();  
        }  
    });  
}
```

Do 2 fragment thuộc cùng một Activity, nên ta có thể truy xuất đến View trong mỗi fragment thông qua Activity chung này. Activity chứa fragment được lấy thông qua phương thức `getActivity()`.



Giao diện người dùng của ứng dụng Android

Trong các chương trước ta đã làm quen với thành phần cơ bản của giao diện Android là Activity và vòng đời của nó. Tuy nhiên, bên thân Activity không phải là thứ chúng ta nhìn thấy trên màn hình điện thoại, thay vào đó Activity cần có các thành phần đồ họa khác bên trong nó, là các View và ViewGroup. Trong chương này chúng ta sẽ tìm hiểu chi tiết hơn về các View và ViewGroup trong Android để tạo nên giao diện đồ họa của ứng dụng, cũng như cách thức tương tác với chúng.

View và ViewGroup

Như đã đề cập ở trên, mỗi **Activity** muốn hiển thị giao diện đồ họa cần chứa các thành phần giao diện khác như nút bấm, các nhãn, các ô nhập liệu, checkbox, radio button... Những thành phần như vậy trong Android được gọi chung là các **View**. Tất cả các **view** đều được kế thừa từ lớp `android.view.View`.

Một hoặc nhiều **view** có thể được nhóm lại với nhau thành một **viewGroup**. Mỗi **viewGroup** cũng là một **view**, được dùng để nhóm các View con bên trong nó và hiển thị chúng theo một thứ tự hay quy luật nào đó. Mọi **viewGroup** đều được kế thừa từ lớp `android.view.ViewGroup`. Các loại **ViewGroup** phổ biến nhất trong Android bao gồm:

- `LinearLayout`
- `AbsoluteLayout`
- `TableLayout`
- `RelativeLayout`
- `FrameLayout`
- `ScrollView`

Các **view** và **viewGroup** tạo thành giao diện của **Activity** và thường được mô tả ngay trong file *layout* của **Activity**, nằm trong thư mục `res/layout` (file `main.xml` trong các ví dụ trước). Ví dụ:

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

    <Button
        android:layout_width="160dp"
        android:layout_height="wrap_content"
        android:text="Button"
        android:layout_gravity="left"
        android:layout_weight="1" />

    <Button
        android:layout_width="160dp"
        android:layout_height="wrap_content"
        android:text="Button"
        android:layout_gravity="center"
        android:layout_weight="2" />

    <Button
        android:layout_width="160dp"
        android:layout_height="wrap_content"
        android:text="Button"
        android:layout_gravity="right"
        android:layout_weight="3" />
</LinearLayout>
```

Một số thuộc tính chung của các **View** và **ViewGroup** được kể ra trong bảng dưới đây:

Thuộc tính	Mô tả
<code>layout_width</code>	Chiều rộng của View/ViewGroup
<code>layout_height</code>	Chiều cao của View/ViewGroup
<code>layout_marginTop</code>	Chiều rộng khoảng trống (lề) phía trên của View
<code>layout_marginBottom</code>	Chiều rộng khoảng trống (lề) phía dưới của View
<code>layout_marginLeft</code>	Chiều rộng khoảng trống (lề) phía bên trái của View

<code>layout_marginRight</code>	Chiều rộng khoảng trống (lề) phía bên phải của View
<code>layout_gravity</code>	Cách xếp đặt View (trái, phải, trên, dưới, giữa theo chiều dọc, giữa theo chiều ngang)

Phần tiếp theo sẽ mô tả chi tiết hơn về một số loại `viewGroup` phổ biến trên. Cần chú ý rằng trong thực tế sử dụng, giao diện đồ họa của ứng dụng thường được tạo thành bởi một tổ hợp phân cấp giữa các loại `viewGroup` khác nhau.

LinearLayout

`LinearLayout` sắp xếp các view con bên trong nó theo một cột (từ trên xuống dưới) hoặc theo một hàng (từ trái qua phải). Các view con được xếp dọc hoặc ngang tùy thuộc vào tham số `android:orientation` của `LinearLayout`, giá trị của tham số này có thể là “ `vertical` ” (dọc) hoặc “ `horizontal` ” (ngang).

Xem ví dụ sau:

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

    <TextView
        android:layout_width="100dp"
        android:layout_height="wrap_content"
        android:text="@string/hello" />

    <Button
        android:layout_width="160dp"
        android:layout_height="wrap_content"
        android:onClick="onClick"
        android:text="Button" />

</LinearLayout>
```

Giao diện trên bao gồm một `LinearLayout` theo chiều dọc, chứa 1 view bên trong là đoạn chữ “Hello world” và nút bấm “Button” bên dưới nó. Chiều rộng và cao của `LinearLayout` là `fill_parent`, tức là nó sẽ chiếm hết chiều rộng, chiều dài của view mẹ (trong trường hợp này là Activity, tức là toàn màn hình). Chiều rộng của `textview` là `100dp` (điểm ảnh không phụ thuộc vào mật độ màn hình, chi tiết xem bên dưới), chiều cao của text này là `wrap_content` tức là bằng đúng chiều cao của nội dung chứa trong nó (phụ thuộc vào số dòng chữ, kích thước chữ, khoảng cách... thực tế).

Các đơn vị đo kích thước trong Android bao gồm:

- dp (hoặc dip) - *Density-independent pixel* (điểm ảnh không phụ thuộc vào mật độ màn hình). Một dp tương đương với một pixel trên màn hình có mật độ 160 dpi (160 điểm ảnh trên mỗi inch màn hình). Đây là đơn vị được khuyến nghị dùng trong hầu hết các trường hợp đặt kích thước của view trong layout. Chi tiết hơn về mật độ màn hình được đề cập ở phần sau của giáo trình.
- sp - *Scale-independent pixel*, đơn vị này tương tự dp, được dùng khi mô tả kích thước font chữ (font size)
- pt - Point. $1 \text{ point} = 1/72 \text{ inch}$, dựa trên kích thước vật lý thật của màn hình.
- px – Pixel – một pixel vật lý trên màn hình, đơn vị này không được khuyên dùng trong thiết kế giao diện ứng dụng vì giao diện sẽ hiển thị không đồng nhất trên các màn hình có độ phân giải khác nhau.

Trong ví dụ ở trên, nút bấm có chiều rộng là `160dp` và `textview` là `100dp`. Để hiểu được kích thước này, trước hết ta xem khái niệm kích thước và mật độ màn hình trong Android. Ta xét trên ví dụ cụ thể: điện thoại *Nexus S* của Google. Thiết bị này có màn hình 4 inch theo đường chéo, 2.04 inch theo chiều ngang, với độ phân giải 480x800 pixel. Chiều rộng 2.04 inch với 480 pixel cho ta mật độ điểm ảnh khoảng 235 dpi (dots per inch – điểm ảnh mỗi inch) – xem hình bên dưới.

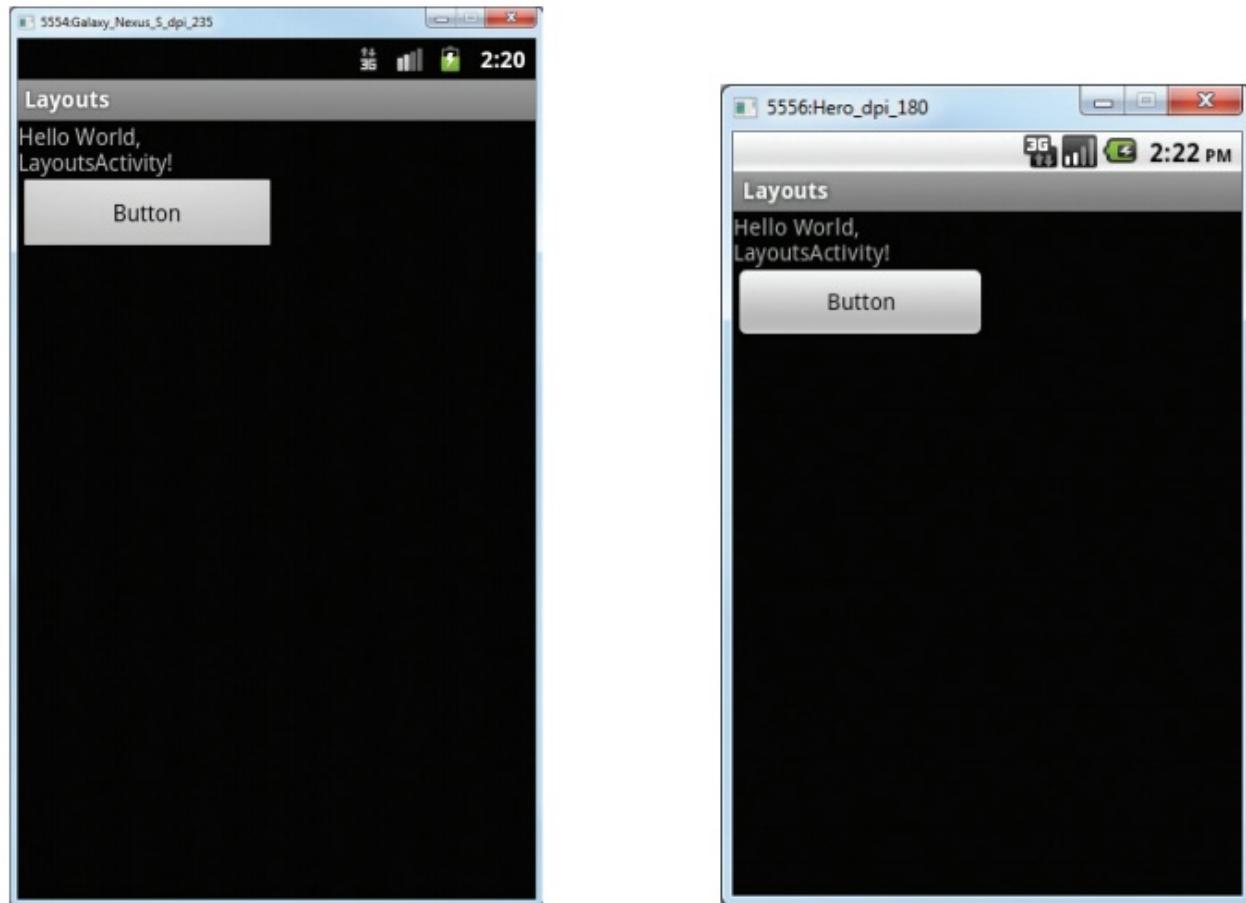


Android định nghĩa 4 loại mật độ màn hình như sau:

- Mật độ thấp: Low density (ldpi) - 120 dpi
- Mật độ trung bình: Medium density (mdpi) - 160 dpi
- Mật độ cao: High density (hdpi) - 240 dpi
- Mật độ rất cao: Extra High density (xhdpi) - 320 dpi

Mỗi thiết bị sẽ được xếp vào một trong các loại mật độ trên. Ví dụ thiết bị Nexus S ở trên sẽ được xếp vào thiết bị mật độ cao, do mật độ màn hình (235dpi) gần nhất với mật độ hdpi – 240dpi. Còn điện thoại The HTC Hero, có màn hình 3.2inch, độ phân giải 320x480, có mật độ 180dpi sẽ được xếp vào điện thoại mật độ trung bình (mdpi) do gần nhất với con số 160dpi.

Dưới đây là hình ảnh của layout trên chạy trên 2 thiết bị có kích thước và độ phân giải khác nhau. Hình bên trái là thiết bị 4 inch, độ phân giải 480x800 (mật độ 235dpi – hdpi), hình bên phải là thiết bị 3.2 inch, độ phân giải 320x480 (mật độ 180dpi).



Có thể thấy mặc dù chạy trên 2 thiết bị có độ phân giải, kích thước và mật độ khác nhau, nhưng nút bấm và text có kích thước rất đồng nhất (nút bấm chiếm khoảng 1/2 chiều ngang màn hình). Kích thước thực tế (tính bằng pixel vật lý) được tính từ kích thước dp như sau: Kích thước pixel thực tế = $dp * (dpi / 160)$, trong đó dpi = 120, 160, 240, hoặc 320 tùy thuộc vào màn hình thiết bị.

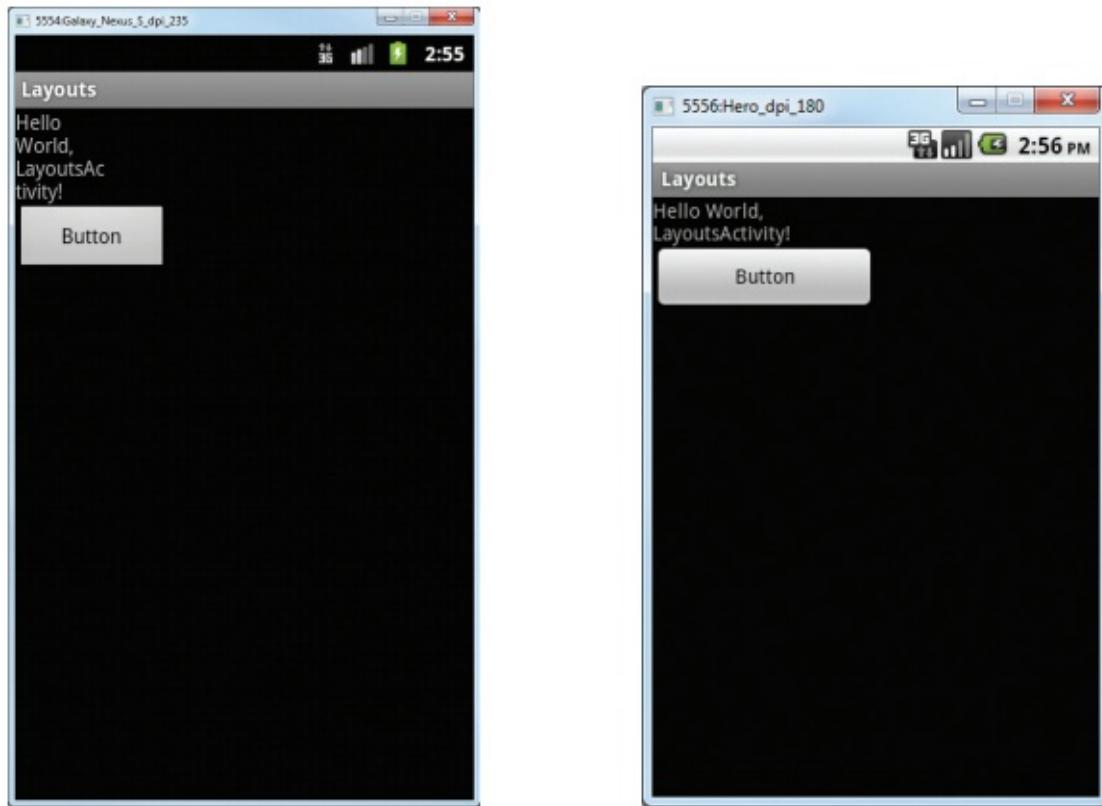
Ở ví dụ trên, nút bấm trên màn hình bên trái sẽ có kích thước thật là: $160 * (240/160) = 240$ pixel, còn trên màn hình bên phải sẽ là $160 * (160/160) = 160$ pixel.

Nếu ta thay đơn vị dp trong khai báo layout ở trên thành đơn vị px như dưới đây:

```
<TextView
    android:layout_width="100px"
    android:layout_height="wrap_content"
    android:text="@string/hello" />

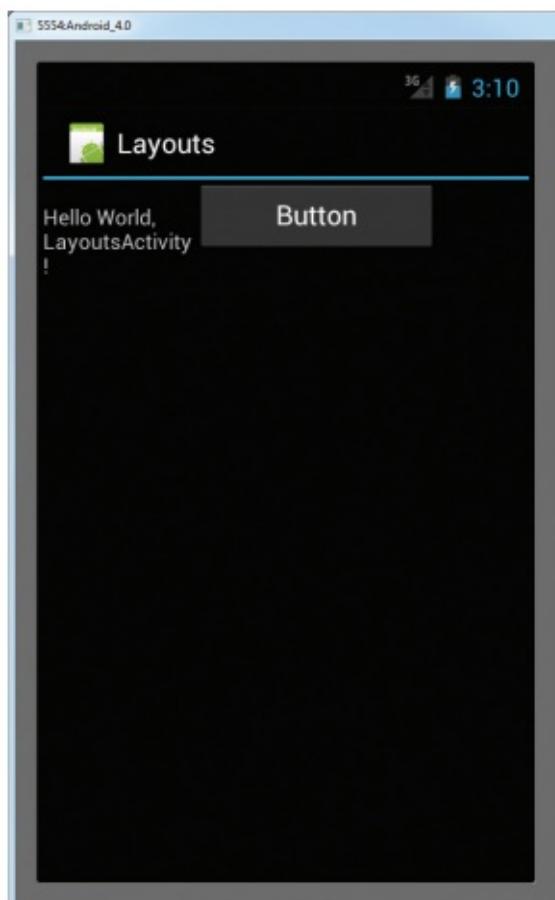
<Button
    android:layout_width="160px"
    android:layout_height="wrap_content"
    android:onClick="onClick"
    android:text="Button" />
```

thì kết quả sẽ là:



Như vậy có thể thấy việc sử dụng đơn vị px trong thiết kế giao diện là không nên, mà nên sử dụng đơn vị dp, đơn vị này đã bao gồm việc thích nghi giao diện với các kích thước màn hình khác nhau. Nếu thay tham số

`android:orientation="vertical"` của `linearlayout` thành `android:orientation="horizontal"`, sẽ thu được:



AbsoluteLayout

AbsoluteLayout cho phép đặt các view con bên trong nó tại vị trí chính xác, cố định, tính theo px hoặc dp. Layout loại này không linh hoạt và không thích nghi được với sự thay đổi của độ phân giải màn hình, vì vậy không được khuyên dùng trong các phiên bản Android gần đây. Vì vậy cho đến thời điểm hiện tại, ta có thể quên đi loại layout này.

TableLayout

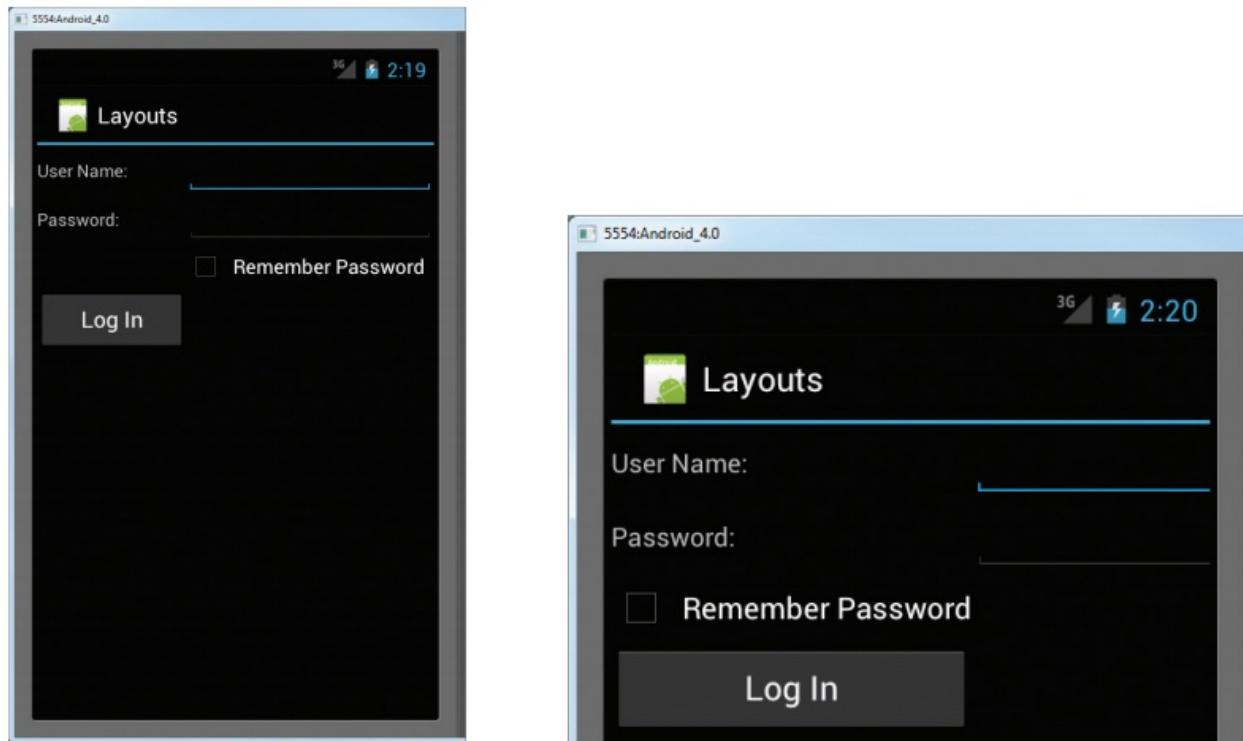
TableLayout cho phép sắp xếp các view con bên trong nó theo dòng và cột. Mỗi dòng được đặt trong thẻ `<TableRow>`, mỗi view con trong `TableRow` được đặt trong một ô của dòng, chiều rộng của mỗi cột được xác định bằng chiều rộng lớn nhất của các ô trong cột đó. Xét ví dụ sau:

```

<TableLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_height="fill_parent"
    android:layout_width="fill_parent"
    >
    <TableRow>
        <TextView
            android:text="User Name:"
            android:width ="120dp"
            />
        <EditText
            android:id="@+id/txtUserName"
            android:width="200dp" />
    </TableRow>
    <TableRow>
        <TextView
            android:text="Password:"
            />
        <EditText
            android:id="@+id/txtPassword"
            android:password="true"
            />
    </TableRow>
    <TableRow>
        <CheckBox android:id="@+id/chkRememberPassword"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:text="Remember Password"
            />
    </TableRow>
    <TableRow>
        <Button
            android:id="@+id/buttonSignIn"
            android:text="Log In" />
    </TableRow>
</TableLayout>

```

Kết quả của layout trên khi chạy trên *emulator* với màn hình dọc và ngang sẽ như sau:



RelativeLayout

RelativeLayout cho phép các view con bên trong được sắp đặt tương đối với nhau và tương đối so với view mẹ.

```
<RelativeLayout
    android:id="@+id/RLayout"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    xmlns:android="http://schemas.android.com/apk/res/android" >

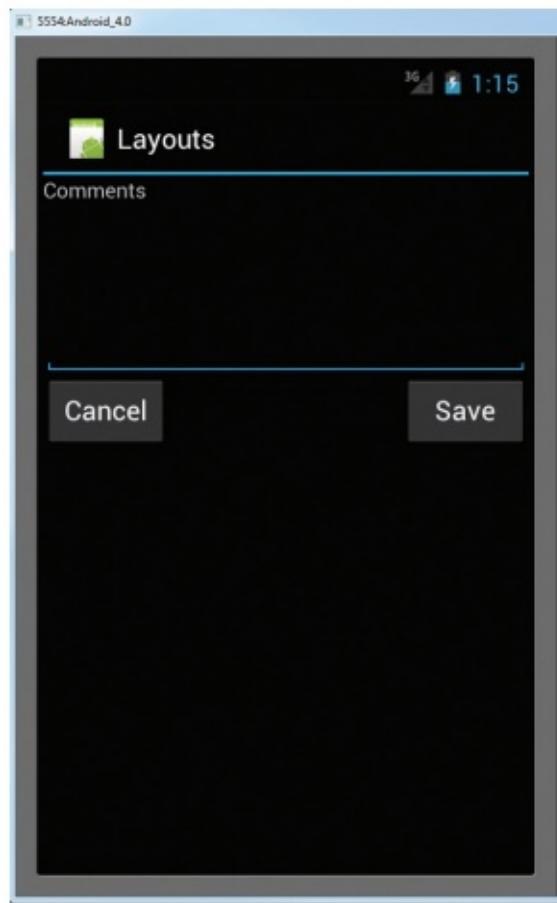
    <TextView
        android:id="@+id/lblComments"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Comments"
        android:layout_alignParentTop="true"
        android:layout_alignParentLeft="true" />

    <EditText
        android:id="@+id/txtComments"
        android:layout_width="fill_parent"
        android:layout_height="170px"
        android:textSize="18sp"
        android:layout_alignLeft="@+id/lblComments"
        android:layout_below="@+id/lblComments"
        android:layout_centerHorizontal="true" />

    <Button
        android:id="@+id/btnSave"
        android:layout_width="125px"
        android:layout_height="wrap_content"
        android:text="Save"
        android:layout_below="@+id/txtComments"
        android:layout_alignRight="@+id/txtComments" />

    <Button
        android:id="@+id btnCancel"
        android:layout_width="124px"
        android:layout_height="wrap_content"
        android:text="Cancel"
        android:layout_below="@+id/txtComments"
        android:layout_alignLeft="@+id/txtComments" />
</RelativeLayout>
```

Kết quả của layout trên như sau:



Mỗi view con trong relativelayout có một số thuộc tính nhất định giúp chúng căn chỉnh theo view mẹ hoặc theo các view khác cùng cấp. Ta có thể thấy các thuộc tính này trong ví dụ ở trên như:

- `layout_alignParentTop` – xếp trên cùng so với layout mẹ
- `layout_alignParentLeft` – căn trái so với layout mẹ
- `layout_alignLeft` – căn trái so với layout khác (có id được chỉ ra trong giá trị của thuộc tính này)
- `layout_alignRight` – căn trái so với layout khác (có id được chỉ ra trong giá trị của thuộc tính này)
- `layout_below` – đặt xuống dưới layout khác (có id được chỉ ra trong giá trị của thuộc tính này)
- `layout_centerHorizontal` – căn giữa theo chiều ngang so với layout mẹ (relativelayout hiện tại)

FrameLayout

Là layout được dùng để hiển thị một view bên trong. View con của Framelayout luôn được căn phía trên, bên trái so với layout mẹ này. Nếu bạn thêm nhiều hơn 1 view vào bên trong FrameLayout, thì các view này sẽ nằm chồng lên nhau như ví dụ dưới đây.

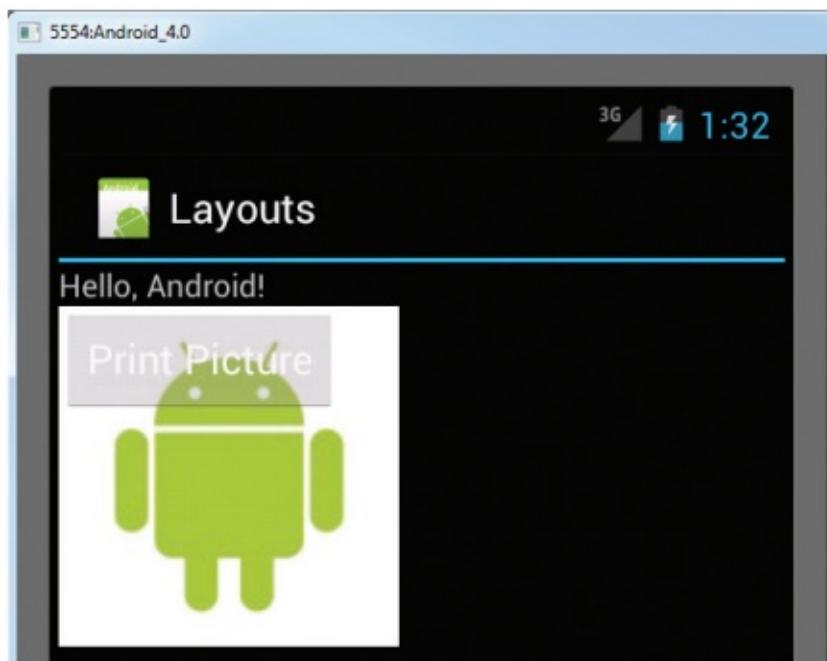
```
<RelativeLayout
    android:id="@+id/RLLayout"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    xmlns:android="http://schemas.android.com/apk/res/android"
    >
    <TextView
        android:id="@+id/lblComments"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello, Android!"
```

```

        android:layout_alignParentTop="true"
        android:layout_alignParentLeft="true"
    />
<FrameLayout
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignLeft="@+id/lblComments"
    android:layout_below="@+id/lblComments"
    android:layout_centerHorizontal="true"
>
<ImageView
    android:src = "@drawable/droid"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" />
<Button
    android:layout_width="124dp"
    android:layout_height="wrap_content"
    android:text="Print Picture" />

</FrameLayout>
</RelativeLayout>

```



ScrollView

ScrollView là một FrameLayout đặc biệt, cho phép người dùng cuộn dọc màn hình khi nội dung bên trong của ScrollView chiếm nhiều diện tích hơn view mẹ.

```

<ScrollView
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    xmlns:android="http://schemas.android.com/apk/res/android" >

    <LinearLayout
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        android:focusable="true"
        android:focusableInTouchMode="true" >
        <Button
            android:id="@+id/button1"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:text="Button 1" />
        <Button
            android:id="@+id/button2"

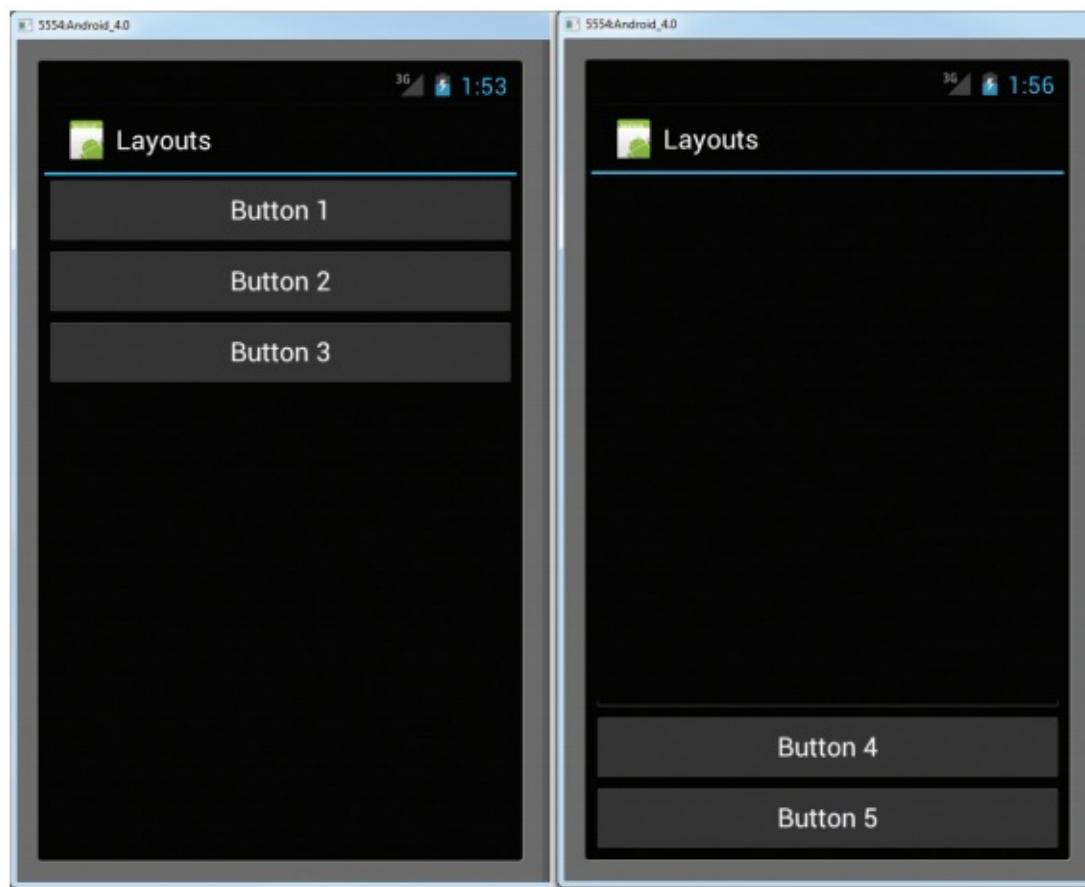
```

```

        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Button 2" />
    <Button
        android:id="@+id/button3"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Button 3" />
    <EditText
        android:id="@+id/txt"
        android:layout_width="fill_parent"
        android:layout_height="600dp" />
    <Button
        android:id="@+id/button4"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Button 4" />
    <Button
        android:id="@+id/buttons5"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Button 5" />
</LinearLayout>
</ScrollView>

```

Trong ví dụ trên, `EditText` ở giữa được đặt chiều cao lên đến 600dp để chắc chắn nó vượt quá chiều cao của màn hình điện thoại, khi đó khi chạy trên điện thoại/emulator, ta sẽ thấy view có thể cuộn lên, cuộn xuống như hình vẽ sau:



Hình: Scroll view đang được cuộn lên (trái) và cuộn xuống (phải)

Thiết kế giao diện thích nghi với hướng màn hình

Một tính năng xuất hiện trên hầu hết các điện thoại thông minh hiện hành là tính năng tự động thay đổi hướng hiển thị của màn hình từ dọc sang ngang hoặc ngược lại khi người dùng xoay điện thoại theo hướng tương ứng. Không ngoại lệ, khi thiết bị Android thay đổi hướng từ dọc sang ngang hoặc ngược lại, activity hiện tại sẽ vẽ lại các view con của mình theo hướng mới, khi đó hàm `onCreate` của Activity sẽ được gọi lại từ đầu. Một khác, do diện tích và tỷ lệ vùng hiển thị theo hướng dọc và hướng ngang khác nhau tương đối nhiều, nên khi thiết kế layout ta cần có biện pháp thích hợp cho sự thay đổi này.

Có 2 phương pháp phổ biến để thiết kế layout thích nghi với sự thay đổi này:

- Neo – neo (xếp cố định) các view con theo các cạnh của màn hình. Đây là cách dễ làm nhất, khi kích thước màn hình thay đổi, vị trí của các layout con sẽ được xếp lại nhưng vẫn căn theo các cạnh gần nhất.
- Thay đổi kích thước và vị trí – cách làm tốn công và linh hoạt hơn và sắp xếp lại toàn bộ layout của các view con khi có sự thay đổi về màn hình. Ngoài, ra còn có thể thêm/bớt view con với những hướng nhất định.

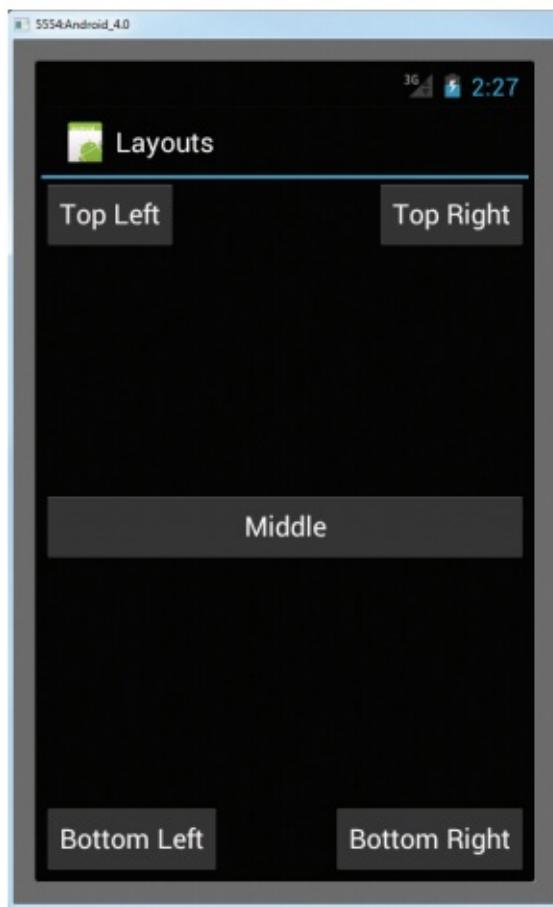
Ta sẽ xem ví dụ cụ thể cho 2 phương án trên.

Neo các view con theo các cạnh màn hình

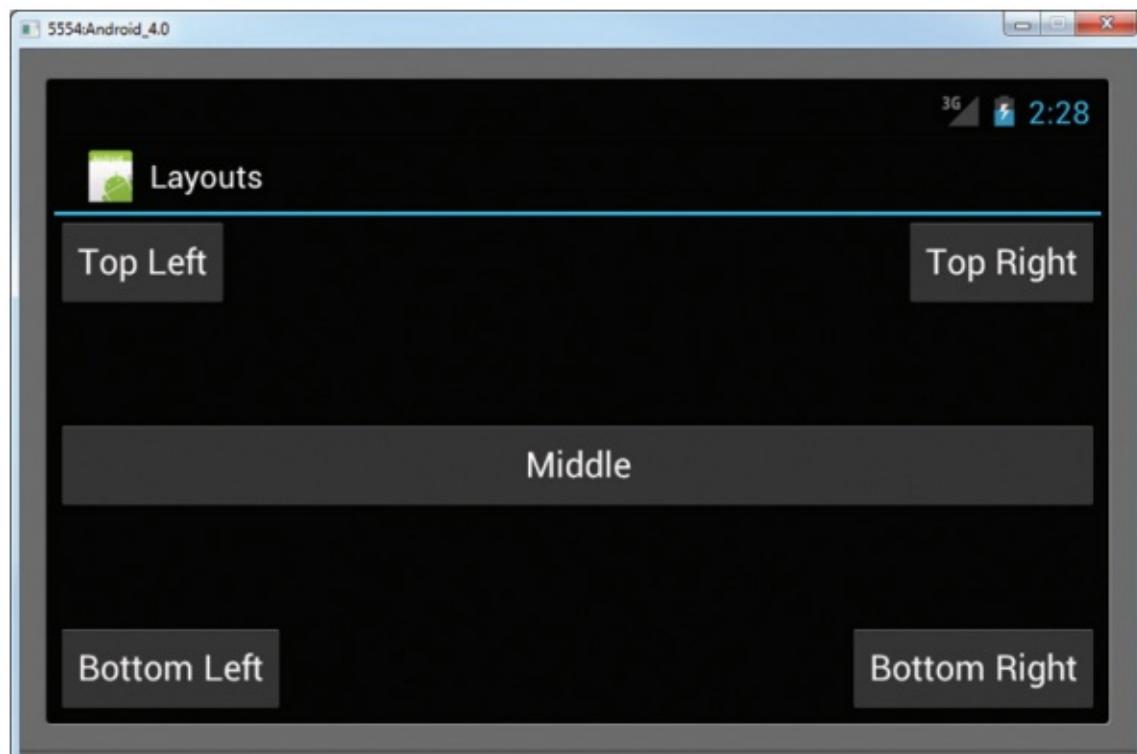
```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    xmlns:android="http://schemas.android.com/apk/res/android">
    <Button
        android:id="@+id/button1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Top Left"
        android:layout_alignParentLeft="true"
        android:layout_alignParentTop="true" />
    <Button
        android:id="@+id/button2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Top Right"
        android:layout_alignParentTop="true"
        android:layout_alignParentRight="true" />
    <Button
        android:id="@+id/button3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Bottom Left"
        android:layout_alignParentLeft="true"
        android:layout_alignParentBottom="true" />
    <Button
        android:id="@+id/button4"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Bottom Right"
        android:layout_alignParentRight="true"
        android:layout_alignParentBottom="true" />
    <Button
        android:id="@+id/button5"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Middle"
        android:layout_centerVertical="true"
        android:layout_centerHorizontal="true" />
</RelativeLayout>
```

Trong ví dụ trên `button1` được neo theo góc trên bên trái, `button2` – góc trên bên phải, `button3` – góc dưới bên trái,

button4 – góc dưới bên phải, button5 – chính giữa màn hình. Layout trên khi hiển thị trên màn hình dọc và ngang sẽ như sau:



Hình: Neo các nút bấm theo cạnh màn hình - màn hình dọc



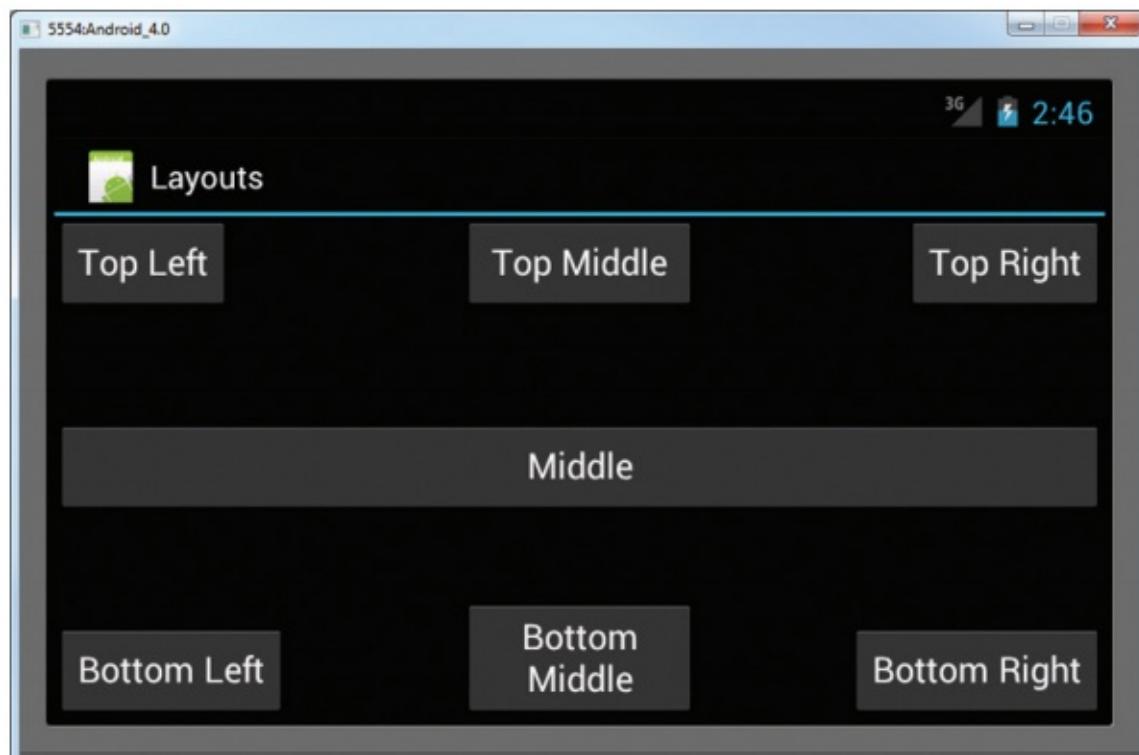
Hình: Neo các nút bấm theo cạnh màn hình - màn hình ngang

Thay đổi kích thước và vị trí

Trong phương pháp này, thông thường mỗi hướng màn hình sẽ có một layout riêng. Layout này có thể được cấu hình trong code, hoặc đặt file layout vào thư mục cấu hình sẵn để hệ thống tự lựa chọn trong quá trình chạy. Để tạo layout riêng cho Activity trên màn hình ngang, ta tạo thư mục *layout-land* trong thư mục *res*, sau đó tạo file *main.xml* (trùng tên với file layout trong thư mục *res/layout*), với nội dung như sau:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    xmlns:android="http://schemas.android.com/apk/res/android">
    <Button
        android:id="@+id/button1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Top Left"
        android:layout_alignParentLeft="true"
        android:layout_alignParentTop="true" />
    <Button
        android:id="@+id/button2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Top Right"
        android:layout_alignParentTop="true"
        android:layout_alignParentRight="true" />
    <Button
        android:id="@+id/button3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Bottom Left"
        android:layout_alignParentLeft="true"
        android:layout_alignParentBottom="true" />
    <Button
        android:id="@+id/button4"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Bottom Right"
        android:layout_alignParentRight="true"
        android:layout_alignParentBottom="true" />
    <Button
        android:id="@+id/button5"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Middle"
        android:layout_centerVertical="true"
        android:layout_centerHorizontal="true" />
    <Button
        android:id="@+id/button6"
        android:layout_width="180px"
        android:layout_height="wrap_content"
        android:text="Top Middle"
        android:layout_centerVertical="true"
        android:layout_centerHorizontal="true"
        android:layout_alignParentTop="true" />
    <Button
        android:id="@+id/button7"
        android:layout_width="180px"
        android:layout_height="wrap_content"
        android:text="Bottom Middle"
        android:layout_centerVertical="true"
        android:layout_centerHorizontal="true"
        android:layout_alignParentBottom="true" />
</RelativeLayout>
```

Để ý ta thấy layout này có 7 nút thay vì 5 nút như layout mặc định (cho màn hình dọc). Kết quả hiển thị cho màn hình ngang sẽ xuất hiện thêm 2 nút bấm ở trên và dưới màn hình, căn giữa theo chiều ngang như hình dưới:



Hình: Giao diện theo màn hình ngang với 2 nút mới xuất hiện (Top Middle và Bottom Middle)

Ngoài ra khi thay đổi hướng màn hình, Activity sẽ được vẽ lại hoàn toàn từ đầu và trạng thái hiện tại sẽ mất, bao gồm giá trị của các trường, nội dung của các view không được đặt tên trong layout... Vì vậy, trong trường hợp cần thiết phải lưu trữ trạng thái hiện tại của Activity, việc ta cần làm là nạp chồng 2 hàm sau:

```
@Override
public void onSaveInstanceState(Bundle outState) {
    //---save whatever you need to persist---
    outState.putString("ID", "1234567890");
    super.onSaveInstanceState(outState);
}

@Override
public void onRestoreInstanceState(Bundle savedInstanceState) {
    super.onRestoreInstanceState(savedInstanceState);
    //---retrieve the information persisted earlier---
    String ID = savedInstanceState.getString("ID");
}
```

Dữ liệu cần lưu trữ được chứa trong một đối tượng `Bundle`, chi tiết về đối tượng này và các loại dữ liệu có thể lưu trữ trong nó bạn đọc tự tìm hiểu và phần này được coi như bài tập. Trong ví dụ trên ta đã tiến hành lưu và lấy lại giá trị của một chuỗi ("0123456789") vào `Bundle` dưới tên là "`ID`".

Điều khiển hướng của màn hình

Trong một số trường hợp, ta cần thiết kế một số Activity chỉ hỗ trợ một loại hướng màn hình cụ thể. Ví dụ, màn hình xem phim có thể được thiết lập chỉ có hướng nằm ngang, hay màn hình gọi điện thoại được thiết lập luôn luôn hiển thị theo chiều dọc.

Để làm việc này, ta có thể khai báo thuộc tính của activity tương ứng trong file `Manifest` hoặc cấu hình trong mã nguồn của hàm `onCreate` tương ứng với Activity cần thiết lập.

Ví dụ về cấu hình hướng màn hình luôn nằm ngang trong mã nguồn:

```
@Override  
public void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.main);  
  
    //---change to landscape mode---  
    setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_LANDSCAPE);  
}
```

Và trong *Manifest*:

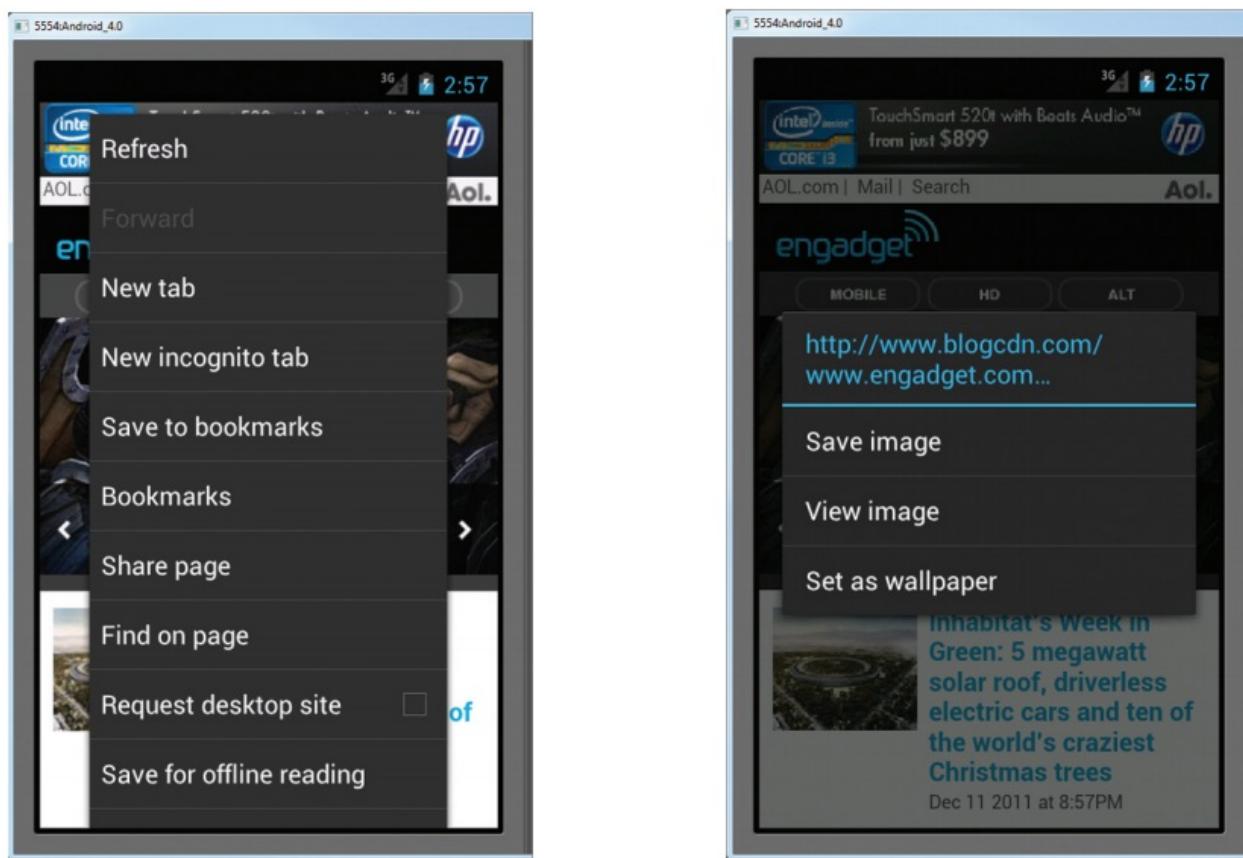
```
<activity  
    android:label="@string/app_name"  
    android:name=".OrientationsActivity"  
    android:screenOrientation="landscape" >  
    <intent-filter >  
        <action android:name="android.intent.action.MAIN" />  
  
        <category android:name="android.intent.category.LAUNCHER" />  
    </intent-filter>  
</activity>
```

Sử dụng trình đơn (Menu)

Trình đơn (bảng chọn, menu) dùng để hiển thị các hành động thường ít dùng hơn và không hiển thị trực tiếp lên màn hình.
Trong Android có 2 loại trình đơn :

- Trình đơn chính (*Options menu*) – hiển thị các hành động liên quan đến toàn bộ Activity hiện tại. Trong Android, để kích hoạt trình đơn này, ta bấm nút Menu của thiết bị (phím cứng hoặc phím ảo trên màn hình)
- Trình đơn ngữ cảnh (*Context Menu*) – hiển thị các hành động liên quan đến một view cụ thể trên màn hình, trình đơn này được kích hoạt bằng cách bấm và giữ ngón tay (*long tap*) trên view tương ứng.

Hình dưới đây minh họa trình đơn chính (bên trái) và trình đơn ngữ cảnh khi bấm và giữ tay trên ảnh (bên phải) của ứng dụng *Browser* (trình duyệt web):



Hình: Trình đơn chính (trái) và trình đơn ngữ cảnh (phải)

Về mặt lập trình, 2 loại menu này sử dụng cùng một lớp (`android.view.Menu`) và chứa các đối tượng giống nhau (`android.view.MenuItem`). Vì vậy ta tạo sẵn 2 phương thức dùng chung cho 2 loại menu này để tạo menu với các mục (item) bên trong và xử lý sự kiện bấm vào từng item như dưới đây:

Hàm `CreateMenu` dùng để thêm 7 mục vào menu có sẵn (được truyền vào dưới ạng tham số):

```
private void CreateMenu(Menu menu)
{
    menu.setQwertyMode(true);
    MenuItem mnu1 = menu.add(0, 0, 0, "Item 1");
    {
        mnu1.setAlphabeticShortcut('a');
        mnu1.setIcon(R.drawable.ic_launcher);
    }
    MenuItem mnu2 = menu.add(0, 1, 1, "Item 2");
}
```

```

    {
        mnu2.setAlphabeticShortcut('b');
        mnu2.setIcon(R.drawable.ic_launcher);
    }
    MenuItem mnu3 = menu.add(0, 2, 2, "Item 3");
    {
        mnu3.setAlphabeticShortcut('c');
        mnu3.setIcon(R.drawable.ic_launcher);
    }
    MenuItem mnu4 = menu.add(0, 3, 3, "Item 4");
    {
        mnu4.setAlphabeticShortcut('d');
    }

    menu.add(0, 4, 4, "Item 5");
    menu.add(0, 5, 5, "Item 6");
    menu.add(0, 6, 6, "Item 7");
}

```

Hàm `MenuChoice` để xử lý sự kiện tương ứng với từng mục được lựa chọn (truyền vào dạng tham số). Với mục đích minh họa, trong ví dụ sau, khi một mục được chọn, ta chỉ đơn giản hiển thị tên của mục đó bằng thông báo dạng `Toast`.

```

private boolean MenuChoice(MenuItem item)
{
    switch (item.getItemId()) {
    case 0:
        Toast.makeText(this, "You clicked on Item 1",
                    Toast.LENGTH_LONG).show();
        return true;
    case 1:
        Toast.makeText(this, "You clicked on Item 2",
                    Toast.LENGTH_LONG).show();
        return true;
    case 2:
        Toast.makeText(this, "You clicked on Item 3",
                    Toast.LENGTH_LONG).show();
        return true;
    case 3:
        Toast.makeText(this, "You clicked on Item 4",
                    Toast.LENGTH_LONG).show();
        return true;
    case 4:
        Toast.makeText(this, "You clicked on Item 5",
                    Toast.LENGTH_LONG).show();
        return true;
    case 5:
        Toast.makeText(this, "You clicked on Item 6",
                    Toast.LENGTH_LONG).show();
        return true;
    case 6:
        Toast.makeText(this, "You clicked on Item 7",
                    Toast.LENGTH_LONG).show();
        return true;
    }
    return false;
}

```

Trình đơn chính

Để hiển thị trình đơn chính, ta nạp chồng hàm `onCreateOptionsMenu(Menu menu)` và thêm các mục vào đối tượng menu (trong tham số của hàm):

```

@Override
public boolean onCreateOptionsMenu(Menu menu) {
    super.onCreateOptionsMenu(menu);
    CreateMenu(menu);
    return true;
}

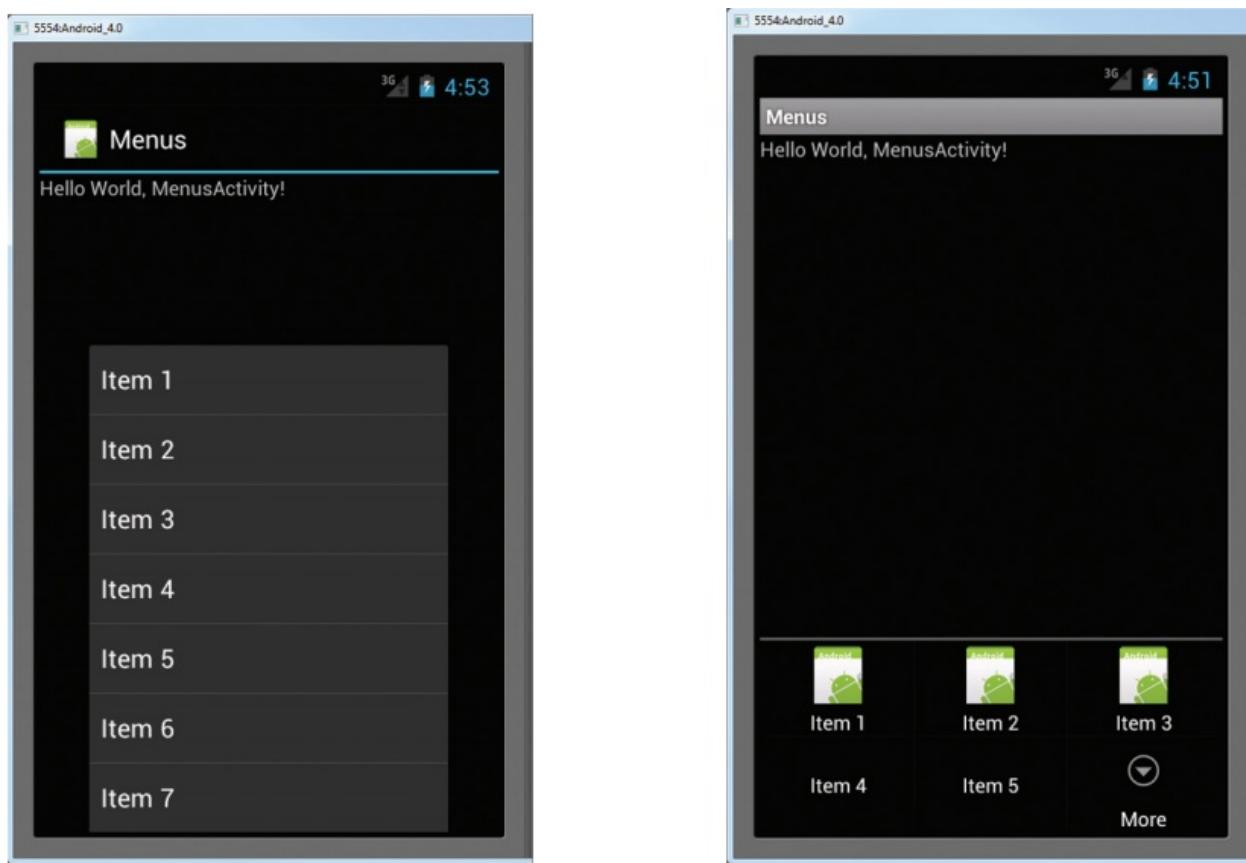
```

Để hiển thị menu này lúc chạy ứng dụng, ta bấm phím MENU của thiết bị.

Để xử lý sự kiện chọn một item trong menu hiện ra, ta cần nạp chồng hàm `onOptionsItemSelected(MenuItem item)`:

```
@Override
public boolean onOptionsItemSelected(MenuItem item)
{
    return MenuChoice(item);
}
```

Hình bên dưới minh họa menu chính được tạo ở trên trong trường hợp thiết lập SDK nhỏ nhất được hỗ trợ trong `AndroidManifest.xml >=10` (`<uses-sdk android:minSdkVersion="10" />` - hình bên trái) và `< 10` (`<uses-sdk android:minSdkVersion="9" />` hình bên phải).



Hình: Trình đơn chính với API mức 10 (trái) và API mức 9 (phải)

Trình đơn ngữ cảnh

Để hiển thị trình đơn ngữ cảnh, ta nạp chồng hàm `onCreateContextMenu(ContextMenu menu, View view, ContextMenuItemInfo menuInfo)` và thêm các mục vào đối tượng menu (là tham số của hàm):

```
@Override
public void onCreateContextMenu(ContextMenu menu, View view,
ContextMenuItemInfo menuInfo)
{
    super.onCreateContextMenu(menu, view, menuInfo);
    CreateMenu(menu);
}
```

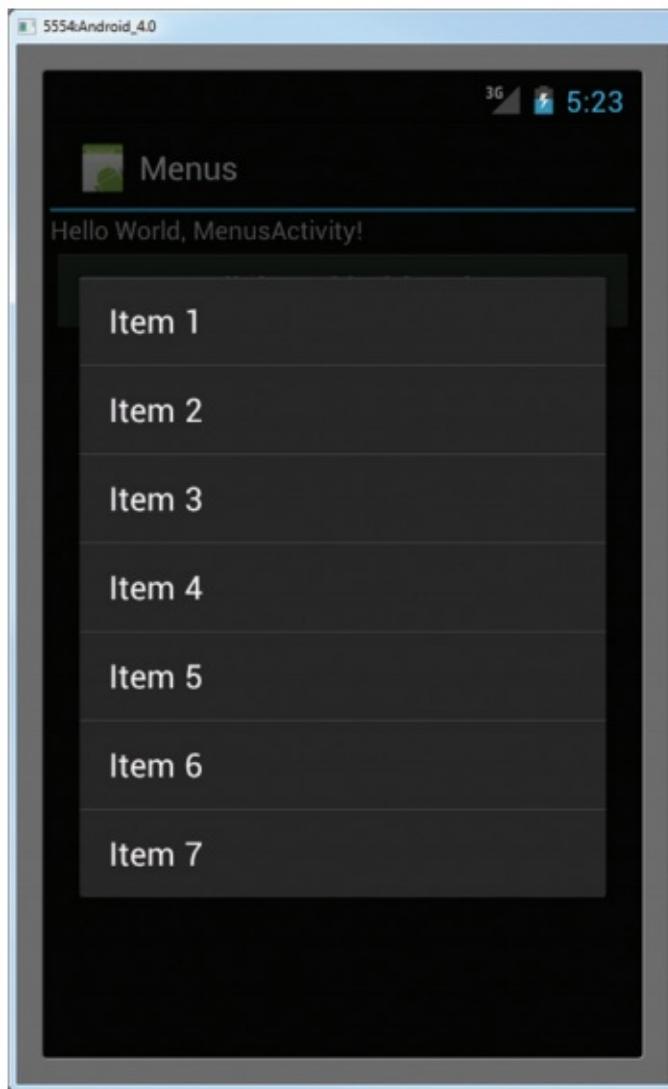
Để xử lý sự kiện chọn một item trong menu hiện ra, ta cần nạp chõng hàm `onContextItemSelected (MenuItem item)`:

```
@Override
public boolean onContextItemSelected (MenuItem item)
{
    return MenuChoice(item);
}
```

Để gắn menu ngữ cảnh này vào nút bấm button1, ta thêm đoạn mã sau vào hàm `onCreate` của activity:

```
Button btn = (Button) findViewById(R.id.button1);
btn.setOnCreateContextMenuListener(this);
```

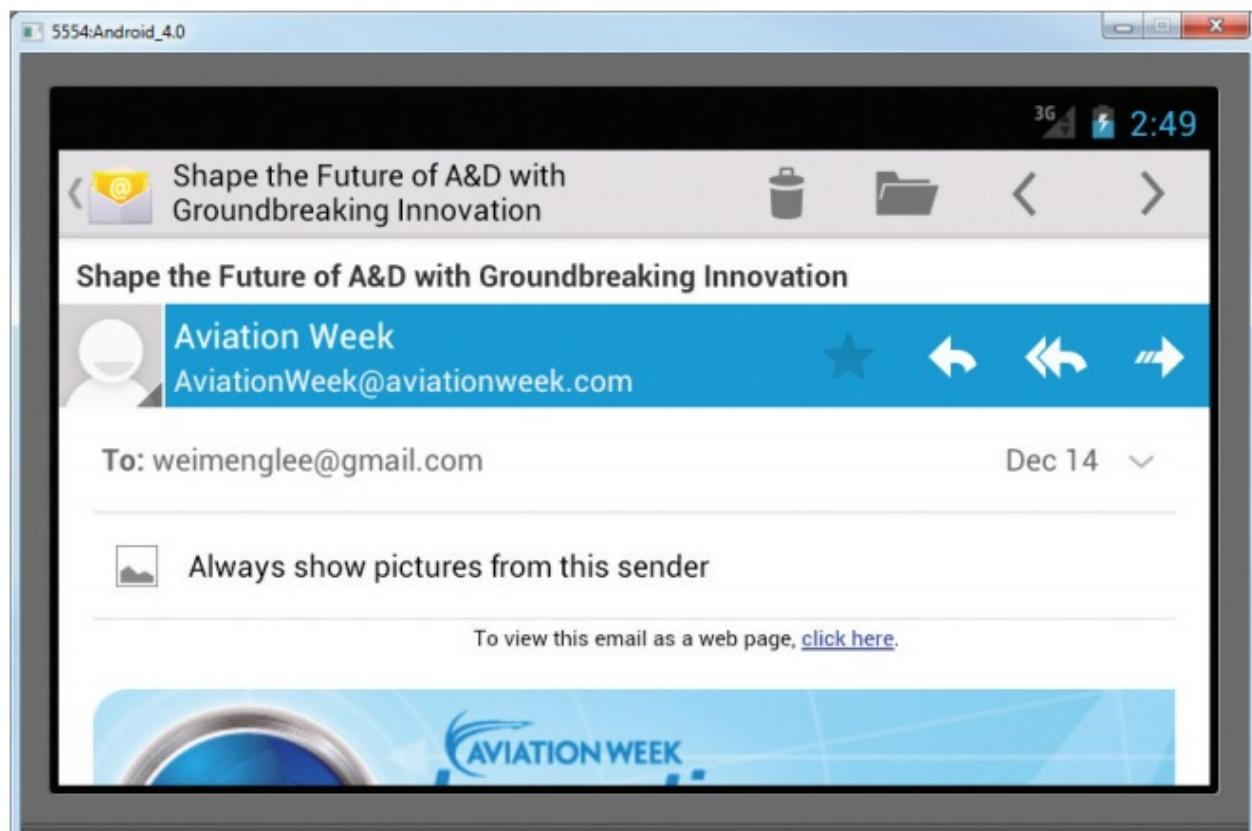
Để hiển thị menu này lúc chạy ứng dụng, ta bấm và giữ ngón tay lên trên nút bấm button1, trình đơn hiện ra như hình bên dưới:



Hình: Trình đơn ngữ cảnh xuất hiện khi bấm và giữ ngón tay trên nút bấm `button1`

Sử dụng thanh tác vụ (ActionBar)

Cùng với Fragment, một tính năng mới được giới thiệu từ *Android 3.0 HoneyComb* là thanh tác vụ bên trong ứng dụng (*Action Bar*) để thay thế cho thanh tiêu đề cũ. Action bar bao gồm icon của ứng dụng và tiêu đề của Activity ở bên trái. Ngoài ra, người dùng có thể đặt ở bên phải các nút bấm đặc biệt, gọi là action item. Hình bên dưới minh họa action bar của ứng dụng *email* trong Android với icon ứng dụng, tiêu đề email và 4 action item ở bên phải:



Hình: ActionBar với tiêu đề và các nút bấm

Action Bar dùng chung menu chính (*option menu*) như mô tả trong phần trước. Để các menu item trong menu chính hiển thị trên Action bar dưới dạng các action item, ta chỉ cần gọi hàm `mnu1.setShowAsAction()` của mỗi menu item.

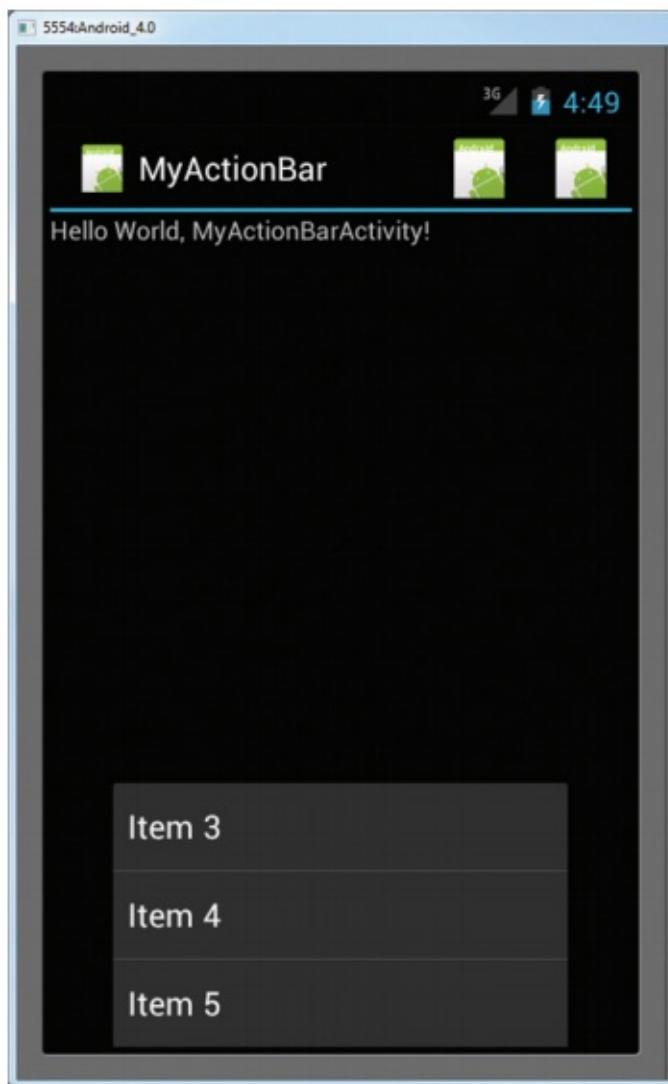
```
private void CreateMenu(Menu menu)
{
    MenuItem mnu1 = menu.add(0, 0, 0, "Item 1");
    {
        mnu1.setIcon(R.drawable.ic_launcher);
        mnu1.setShowAsAction(
            MenuItem.SHOW_AS_ACTION_IF_ROOM |
            MenuItem.SHOW_AS_ACTION_WITH_TEXT);
    }
    MenuItem mnu2 = menu.add(0, 1, 1, "Item 2");
    {
        mnu2.setIcon(R.drawable.ic_launcher);
        mnu2.setShowAsAction(
            MenuItem.SHOW_AS_ACTION_IF_ROOM |
            MenuItem.SHOW_AS_ACTION_WITH_TEXT);
    }
    MenuItem mnu3 = menu.add(0, 2, 2, "Item 3");
    {
        mnu3.setIcon(R.drawable.ic_launcher);
        mnu3.setShowAsAction(
            MenuItem.SHOW_AS_ACTION_IF_ROOM |
            MenuItem.SHOW_AS_ACTION_WITH_TEXT);
    }
}
```

```

    }
    MenuItem mnu4 = menu.add(0, 3, 3, "Item 4");
    {
        mnu4.setShowAsAction(
            MenuItem.SHOW_AS_ACTION_IF_ROOM |
            MenuItem.SHOW_AS_ACTION_WITH_TEXT);
    }
    MenuItem mnu5 = menu.add(0, 4, 4, "Item 5");
    {
        mnu5.setShowAsAction(
            MenuItem.SHOW_AS_ACTION_IF_ROOM |
            MenuItem.SHOW_AS_ACTION_WITH_TEXT);
    }
}

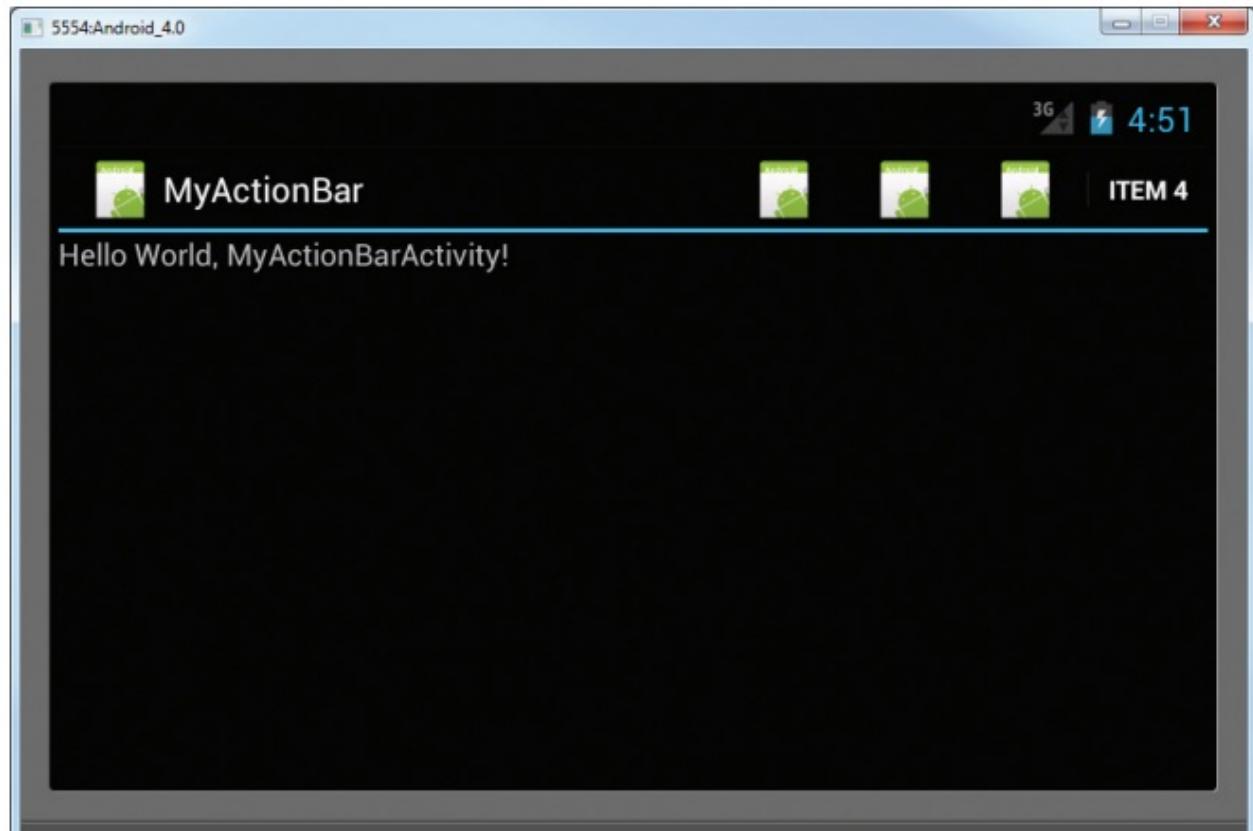
```

Kết quả khi chạy trên thiết bị như hình bên dưới. Đối với màn hình dọc, có ít diện tích để hiển thị action item hơn (2 cái), 3 item còn lại sẽ hiển thị dưới dạng trình đơn chính như bình thường:



Hình: Action Bar với 2 nút bấm khi màn hình dọc

Đối với màn hình ngang, nhiều item được hiển thị dưới dạng action item hơn (4 cái):



Hình: Action Bar với 4 nút bấm khi màn hình ngang

Xử lý sự kiện tương tác với các thành phần đồ họa

Người dùng tương tác với giao diện đồ họa của ứng dụng Android theo 2 mức: mức Activity và mức View. Ở mức Activity, lớp Activity cần nạp chồng các phương thức tương ứng được định nghĩa sẵn. Một số phương thức như vậy có thể kể đến như:

- `onKeyDown` – được gọi khi một phím được nhấn xuống và sự kiện này chưa được xử lý bởi bất cứ view con nào trong activity
- `onKeyUp` – được gọi khi một phím được nhả ra và sự kiện này chưa được xử lý bởi bất cứ view con nào trong activity
- `onMenuItemSelected` – được gọi khi người dùng lựa chọn một item trong menu đang được mở ra
- `onMenuOpened` – được gọi khi một trình đơn được mở ra.

Nạp chồng hàm xử lý sự kiện của Activity

Ví dụ:

```
@Override
public boolean onKeyDown(int keyCode, KeyEvent event)
{
    switch (keyCode)
    {
        case KeyEvent.KEYCODE_DPAD_CENTER:
            Toast.makeText(getApplicationContext(),
                "Center was clicked",
                Toast.LENGTH_LONG).show();
            break;
        case KeyEvent.KEYCODE_DPAD_LEFT:
            Toast.makeText(getApplicationContext(),
                "Left arrow was clicked",
                Toast.LENGTH_LONG).show();
            break;
        case KeyEvent.KEYCODE_DPAD_RIGHT:
            Toast.makeText(getApplicationContext(),
                "Right arrow was clicked",
                Toast.LENGTH_LONG).show();
            break;
        case KeyEvent.KEYCODE_DPAD_UP:
            Toast.makeText(getApplicationContext(),
                "Up arrow was clicked",
                Toast.LENGTH_LONG).show();
            break;
        case KeyEvent.KEYCODE_DPAD_DOWN:
            Toast.makeText(getApplicationContext(),
                "Down arrow was clicked",
                Toast.LENGTH_LONG).show();
            break;
    }
    return false;
}
```

Đăng ký sự kiện cho từng View

Mỗi view trong Android có thể sinh ra các sự kiện khi người dùng tác động lên nó. Ví dụ nút bấm sinh ra sự kiện `onClick` khi người dùng bấm vào nó, EditText sinh ra sự kiện `onFocusChange` khi nhận focus (người dùng bấm vào ô nhập liệu này) hoặc mất focus (khi view khác nhận focus).

Để đăng ký sự kiện tương tác với các view, ta có thể sử dụng “lớp ẩn danh” (*anonymous class*) như ví dụ dưới đây:

```
Button btn1 = (Button)findViewById(R.id.btn1);
btn1.setOnClickListener(btnListener);
```

và:

```
//---create an anonymous class to act as a button click listener---
private OnClickListener btnListener = new OnClickListener()
{
    public void onClick(View v)
    {
        Toast.makeText(getApplicationContext(),
            ((Button) v).getText() + " was clicked",
            Toast.LENGTH_LONG).show();
    }
};
```

Hoặc sử dụng hàm nội bộ ẩn danh (*anonymous inner class*) như sau:

```
//---create an anonymous inner class to act as an onfocus listener---
EditText txt1 = (EditText)findViewById(R.id.txt1);
txt1.setOnFocusChangeListener(new View.OnFocusChangeListener()
{
    @Override
    public void onFocusChange(View v, boolean hasFocus) {
        Toast.makeText(getApplicationContext(),
            ((EditText) v).getId() + " has focus - " + hasFocus,
            Toast.LENGTH_LONG).show();
    }
});
```

Thiết kế giao diện người dùng với các View cơ bản

Trong chương trước, chúng ta đã làm quen với các loại viewgroup trong Android và cách sử dụng chúng để sắp xếp các view con trong activity. Trong chương này, ta sẽ làm quen với các loại view dùng để tạo nên giao diện đồ họa cho ứng dụng: các nút bấm, chữ, hình ảnh, danh sách...

Sử dụng các View cơ bản trong Android

Phần này mô tả một số View cơ bản nhất thường được dùng để tạo nên giao diện đồ họa cho ứng dụng Android như:

- TextView
- EditText
- Button
- ImageButton
- CheckBox
- ToggleButton
- RadioButton
- RadioGroup

TextView

Như tên gọi của nó, TextView dùng để hiển thị đoạn văn bản (dạng chữ) lên màn hình. Đây là view đơn giản nhất và dùng rất nhiều trong Android. Ta đã gặp view này ngay từ dự án *HelloAndroid* trong những chương đầu tiên:

```
<TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="@string/hello" />
```

Ngoài một số thuộc tính chung như `layout_width`, `layout_height`, ta có thể lựa chọn tùy chỉnh kích thước font chữ (thường tính bằng `sp – scalable pixel`), loại font chữ, kiểu chữ (in đậm, in nghiêng), số dòng tối đa để hiển thị đoạn chữ...

Nếu ta muốn cho phép người dùng thay đổi/nhập nội dung chữ thì cần sử dụng một view kế thừa từ `TextView` là `EditText` sẽ được nhắc đến ở phần dưới.

Button và ImageButton

Dùng để hiển thị nút bấm lên màn hình. Sự kiện phổ biến nhất của nút bấm là sự kiện `onClick`, được sinh ra khi người dùng bấm lên nút bấm. Điểm khác nhau giữa `Button` và `ImageButton` là `Button` dùng chữ (text) làm nhãn, còn `ImageButton` dùng ảnh.

EditText

Kế thừa từ lớp `TextView`, cho phép người dùng sửa được nội dung chữ trong nó.

CheckBox

Là một loại nút bấm đặc biệt, có 2 trạng thái: được chọn (`checked`) và không được chọn (`unchecked`)

RadioButton và RadioGroup

`RadioButton` có 2 trạng thái là được chọn và không được chọn, còn `RadioGroup` nhóm các `RadioButton` lại với nhau để đảm bảo cùng một lúc chỉ có tối đa 01 `RadioButton` trong nhóm có trạng thái được chọn.

ToggleButton

Tương tự như `CheckBox`, là một loại nút bấm đặc biệt, có 2 trạng thái là bật (`checked`) và tắt (`unchecked`).

Ta sẽ xem các view cơ bản trên trong cùng một ví dụ dưới đây:

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

    <Button android:id="@+id	btnSave"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/save"
        android:onClick="btnSaved_clicked"/>

    <Button android:id="@+id	btnOpen"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Open" />

    <ImageButton android:id="@+id	btnImg1"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:src="@drawable/ic_launcher" />

    <EditText android:id="@+id	txtName"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content" />

    <CheckBox android:id="@+id/chkAutosave"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Autosave" />

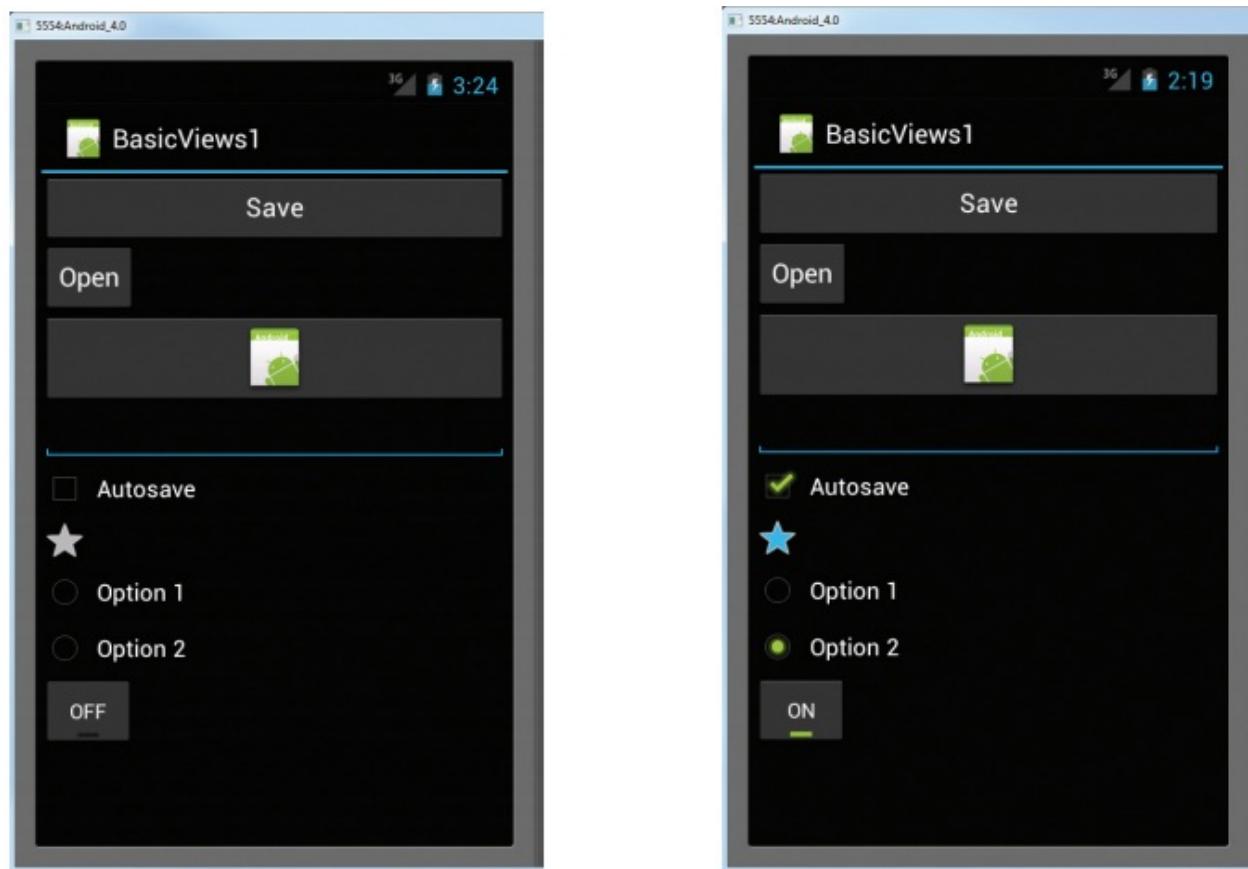
    <CheckBox android:id="@+id/star"
        style="?android:attr/starStyle"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />

    <RadioGroup android:id="@+id/rdbGp1"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal" >
        <RadioButton android:id="@+id/rdb1"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Option 1" />
        <RadioButton android:id="@+id/rdb2"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Option 2" />
    </RadioGroup>

    <ToggleButton android:id="@+id/toggle1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />

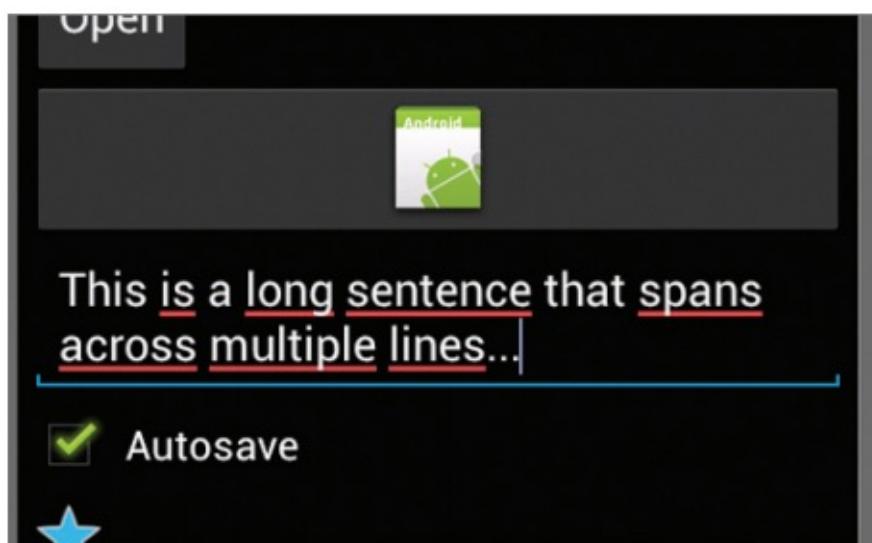
</LinearLayout>
```

Chạy ứng dụng với activity có layout như trên, ta sẽ thấy các view cơ bản này được xếp từ trên xuống dưới theo thứ tự lần lượt (do nằm trong LinearLayout). Thủ thực hiện các thao tác chọn trên các nút `ToggleButton`, `CheckBox` và `RadioButton` ta sẽ thấy trạng thái chọn của chúng như hình bên phải:



Hình: Các view cơ bản và các trạng thái không được chọn (trái) và được chọn (phải)

Ô nhập liệu (EditText) ở trên được thiết lập chiều cao là “wrap_content” nên chiều cao của nó sẽ được tự động điều chỉnh khi nội dung bên trong thay đổi:



View RadioButton có thuộc tính `orientation` tương tự như LinearLayout, cho phép xếp các RadioButton bên trong theo chiều dọc hoặc chiều ngang.

Mỗi view trong ví dụ trên đều được đặt tên (dùng thuộc tính `android:id`), tên này được sử dụng trong mã nguồn để truy cập đến view tương ứng thông qua các hàm `View.findViewById()` hoặc `Activity.findViewById()`.

Sau khi xác định được view này theo id trong mã nguồn, ta có thể thêm các hàm xử lý sự kiện cho chúng. Ta xét ví dụ dưới đây:

```

//---Button view---
Button btnOpen = (Button) findViewById(R.id.btnOpen);
btnOpen.setOnClickListener(new View.OnClickListener() {
    public void onClick(View v) {
        DisplayToast("You have clicked the Open button");
    }
});

//---CheckBox---
CheckBox checkBox = (CheckBox) findViewById(R.id.chkAutosave);
checkBox.setOnClickListener(new View.OnClickListener()
{
    public void onClick(View v) {
        if (((CheckBox)v).isChecked())
            DisplayToast("CheckBox is checked");
        else
            DisplayToast("CheckBox is unchecked");
    }
});

//---RadioButton---
RadioGroup radioGroup = (RadioGroup) findViewById(R.id.rdbGp1);
radioGroup.setOnCheckedChangeListener(new OnCheckedChangeListener()
{
    public void onCheckedChanged(RadioGroup group, int checkedId) {
        RadioButton rb1 = (RadioButton) findViewById(R.id.rdb1);
        if (rb1.isChecked())
            DisplayToast("Option 1 checked!");
        } else {
            DisplayToast("Option 2 checked!");
        }
    }
});

//---ToggleButton---
ToggleButton toggleButton =
    (ToggleButton) findViewById(R.id.toggle1);
toggleButton.setOnClickListener(new View.OnClickListener()
{
    public void onClick(View v) {
        if (((ToggleButton)v).isChecked())
            DisplayToast("Toggle button is On");
        else
            DisplayToast("Toggle button is Off");
    }
});

```

Trong đó, hàm `DisplayToast` đơn giản chỉ hiển thị thông báo dạng `Toast` lên màn hình:

```

private void DisplayToast(String msg)
{
    Toast.makeText(getApplicationContext(), msg,
        Toast.LENGTH_SHORT).show();
}

```

Riêng sự kiện `onClick` cho nút bấm `btnSave` được khai báo ngay trong file layout:

```
android:onClick="btnSaved_clicked"
```

khi đó trong mã nguồn của Activity cần khai báo hàm tương ứng:

```

public void btnSaved_clicked (View view) {
    DisplayToast("You have clicked the Save button1");
}

```

ProgressBar

ProgressBar cho phép hiển thị thanh trạng thái “chờ đợi” khi có một tác vụ gì đó đang chạy, như khi có một tác vụ tải một tập tin chạy ngầm phía dưới. Ví dụ dưới đây minh họa cách sử dụng thanh tiến trình này.

Trong layout của Activity, thêm view ProgressBar như sau:

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

    <ProgressBar android:id="@+id/progressbar"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        />

</LinearLayout>
```

Trong mã nguồn của Activity, ta tạo một Thread riêng để giả lập thao tác dài chạy ngầm và cập nhật lại trạng thái của progressBar sau mỗi bước thực hiện:

```
progress = 0;
progressBar = (ProgressBar) findViewById(R.id.progressbar);
progressBar.setMax(200);

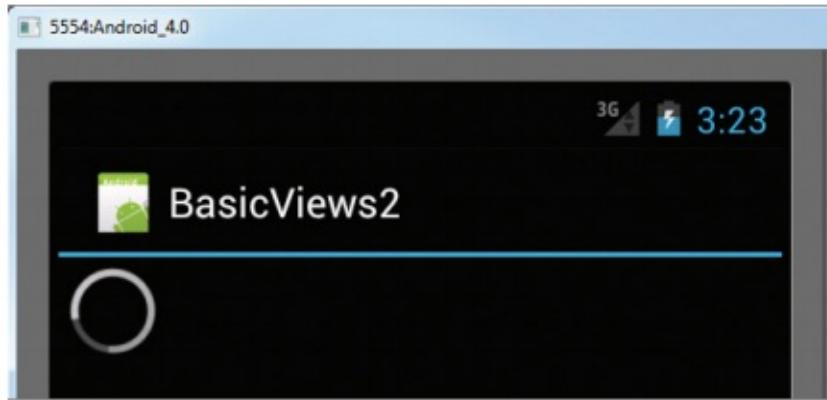
//---do some work in background thread---
new Thread(new Runnable()
{
    public void run()
    {
        //---do some work here---
        while (progressStatus < 100)
        {
            progressStatus = doSomeWork();

            //---Update the progress bar---
            handler.post(new Runnable()
            {
                public void run()
                {
                    progressBar.setProgress(progressStatus);
                }
            });
        }
    }

    //---hides the progress bar---
    handler.post(new Runnable()
    {
        public void run()
        {
            progressBar.setVisibility(View.GONE);
        }
    });
}

//---do some long running work here---
private int doSomeWork()
{
    try {
        //---simulate doing some work---
        Thread.sleep(50);
    } catch (InterruptedException e)
    {
        e.printStackTrace();
    }
    return ++progress;
}
}).start();
```

Chạy ứng dụng với Activity như trên, ta sẽ thấy ProgressBar như hình minh họa bên dưới:

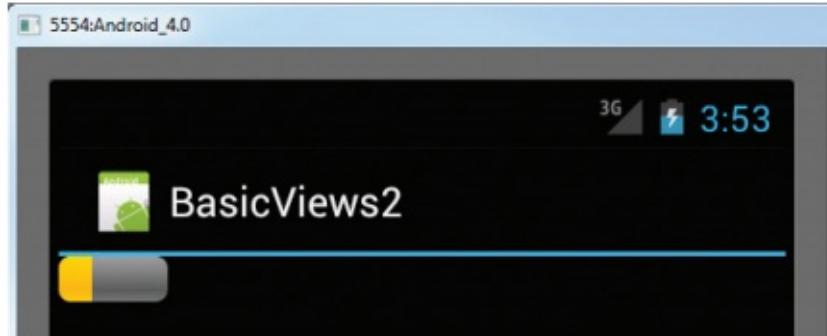


Hình: Progress Bar không xác định tiến trình cụ thể

Mặc định ProgressBar là thanh trạng thái không xác định (không có trạng thái cụ thể) nên ta sẽ chỉ thấy hình tròn xoay xoay.

Ta có thể thay đổi giao diện của thanh trạng thái này theo các cách khác nhau. Ví dụ dưới đây ta thêm thuộc tính `style="@android:style/Widget.ProgressBar.Horizontal"` để biến thanh trạng thái thành dạng thanh chạy ngang có hiển thị phần đã thực hiện và phần còn lại:

```
<ProgressBar android:id="@+id/progressbar"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    style="@android:style/Widget.ProgressBar.Horizontal" />
```



Hình: Progress Bar có xác định tiến trình đã chạy xong

TimePicker và DatePicker

Chọn ngày tháng, giờ là thao tác tương đối phổ biến trong các ứng dụng di động. Android cũng hỗ trợ sẵn thao tác này thông qua các view TimePicker và DatePicker.

TimePicker

TimePicker cho phép người dùng chọn thời gian trong ngày (giờ, phút) theo cả 2 chế độ 24 giờ và 12 giờ với AM/PM.

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

    <TimePicker android:id="@+id/timePicker"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />

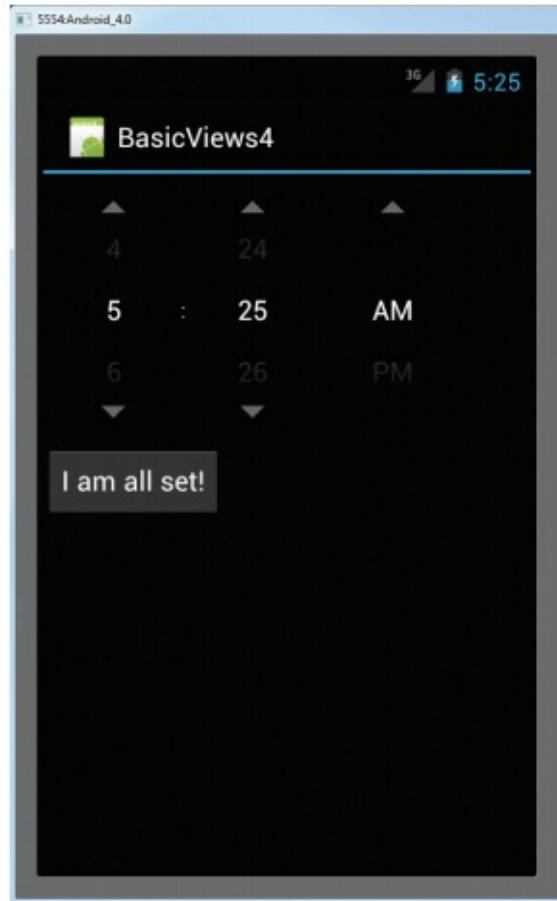
    <Button android:id="@+id	btnSet"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="I am all set!"
        android:onClick="onClick" />

</LinearLayout>
```

Giao diện trên bao gồm view chọn thời gian và nút bấm, trong hàm `onClick` xử lý sự kiện bấm của nút bấm này. Ta sẽ hiển thị thời gian đã được chọn trong TimePicker:

```
public void onClick(View view) {
    Toast.makeText(getApplicationContext(),
        "Time selected:" + timePicker.getCurrentHour() +
        ":" + timePicker.getCurrentMinute(),
        Toast.LENGTH_SHORT).show();
}
```

Giao diện ứng dụng sau khi chạy sẽ như sau:



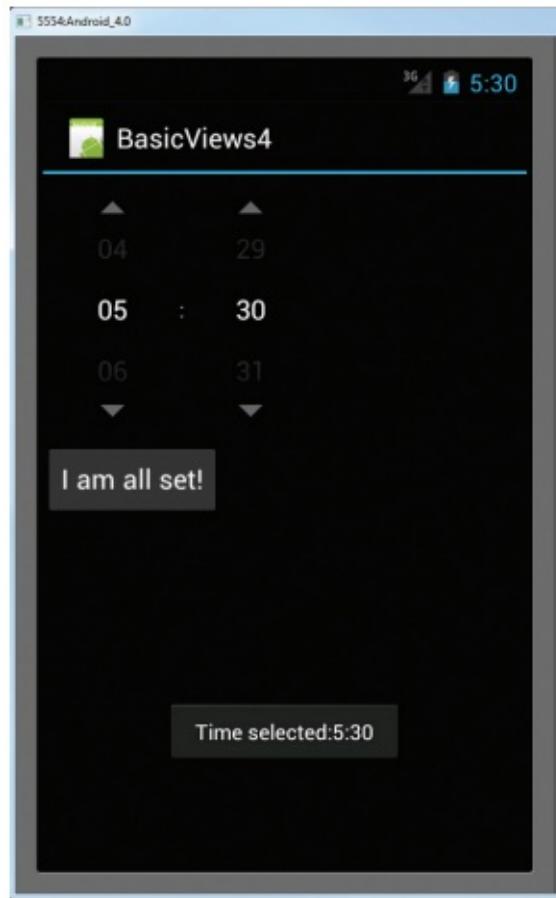
Hình: TimePicker với lựa chọn AM/PM

Bấm nút “I am all set” , thời gian vừa được chọn sẽ được in ra màn hình dưới dạng Toast.

Theo mặc định, TimePicker cho ta chọn thời gian dưới dạng 12 giờ với AM và PM. Để hiển thị thời gian dưới dạng 24 giờ, ta gọi hàm `setIs24HourView` như sau:

```
timePicker = (TimePicker) findViewById(R.id.timePicker);
timePicker.setIs24HourView(true);
```

Kết quả thu được:



Hình: TimePicker với định dạng thời gian theo 24 giờ

Tuy nhiên TimePicker tương đối tốn diện tích màn hình, vì vậy thông thường người ta hay để người dùng lựa chọn thời gian trong một cửa sổ riêng, sau đó cập nhật thời gian vừa lựa chọn vào giao diện chính dưới dạng view khác (TextView chẳng hạn).

Để chọn thời gian trong hộp thoại, ta dùng `TimePickerDialog` cho người dùng chọn thời gian và viết sẵn một hàm lắng nghe sự kiện khi người dùng chọn xong:

```
showDialog(TIME_DIALOG_ID);
```

và:

```

@Override
protected Dialog onCreateDialog(int id)
{
    switch (id) {
    case TIME_DIALOG_ID:
        return new TimePickerDialog(
            this, mTimeSetListener, hour, minute, false);
    }
    return null;
}

private TimePickerDialog.OnTimeSetListener mTimeSetListener =
    new TimePickerDialog.OnTimeSetListener()
{
    public void onTimeSet(
        TimePicker view, int hourOfDay, int minuteOfHour)
    {
        hour = hourOfDay;
        minute = minuteOfHour;

        SimpleDateFormat timeFormat = new SimpleDateFormat("hh:mm aa");
    }
}
```

```

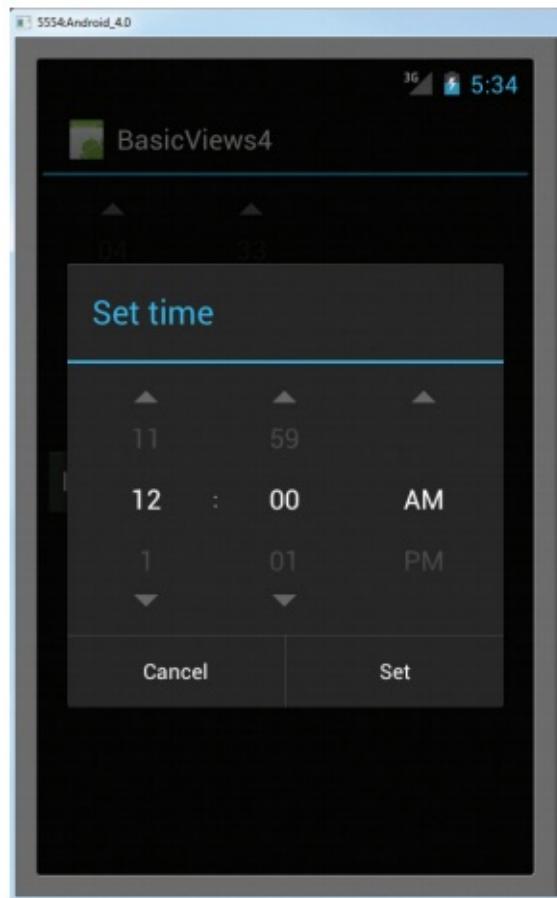
        Date date = new Date(0,0,0, hour, minute);
        String strDate = timeFormat.format(date);

        Toast.makeText(getApplicationContext(),
                "You have selected " + strDate,
                Toast.LENGTH_SHORT).show();
    }
};

}

```

Kết quả ta thu được như sau:



Hình: TimePickerDialog được hiển thị dưới dạng hộp thoại

DatePicker

Tương tự như TimePicker để chọn thời gian, DatePicker được dùng để chọn ngày tháng với cách sử dụng tương tự.

Sử dụng trong layout:

```

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

    <Button android:id="@+id/btnSet"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="I am all set!"
        android:onClick="onClick" />

    <DatePicker android:id="@+id/datePicker"
        android:layout_width="wrap_content"

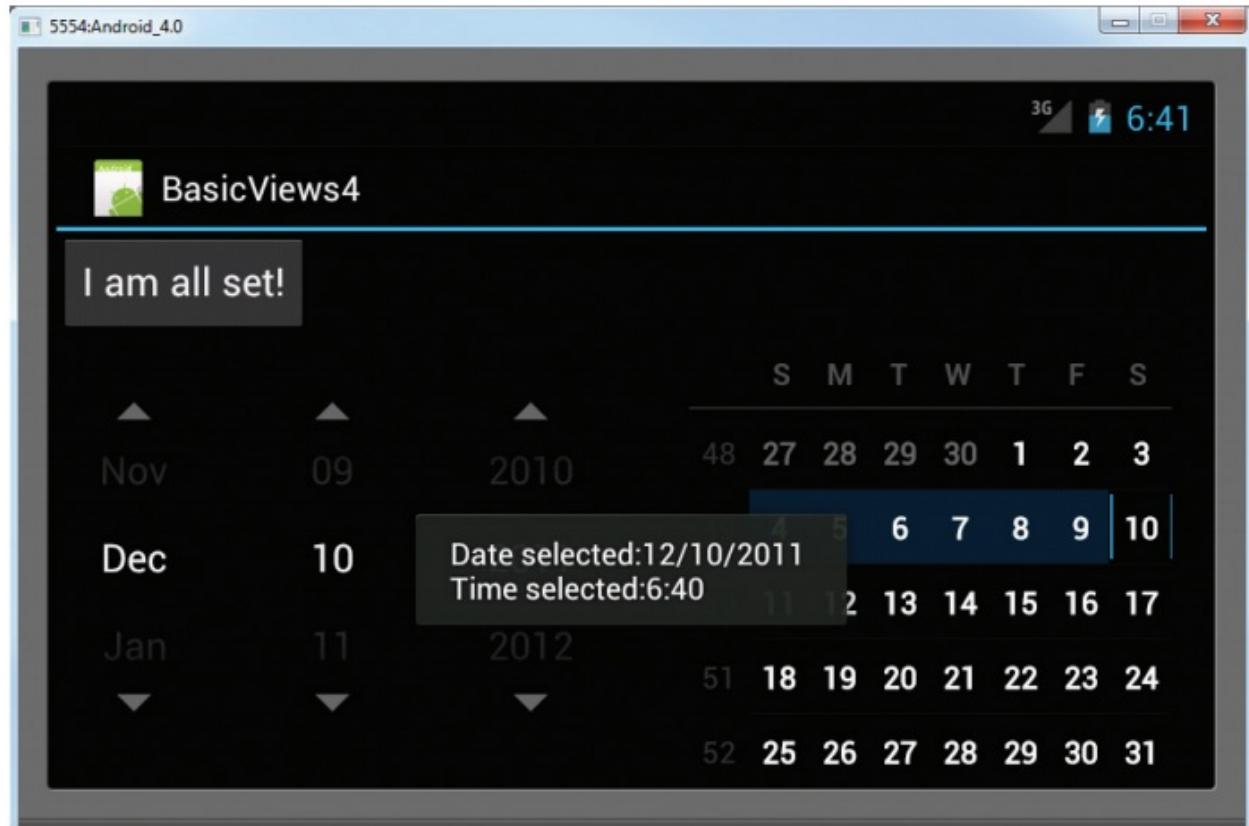
```

```

    android:layout_height="wrap_content" />

<TimePicker android:id="@+id/timePicker"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" />
</LinearLayout>

```



Hình: TimePicker (trái) và DatePicker (phải)

Hoặc sử dụng hộp thoại thông qua

```

DatePickerDialog:
    showDialog(DATE_DIALOG_ID);

```

và:

```

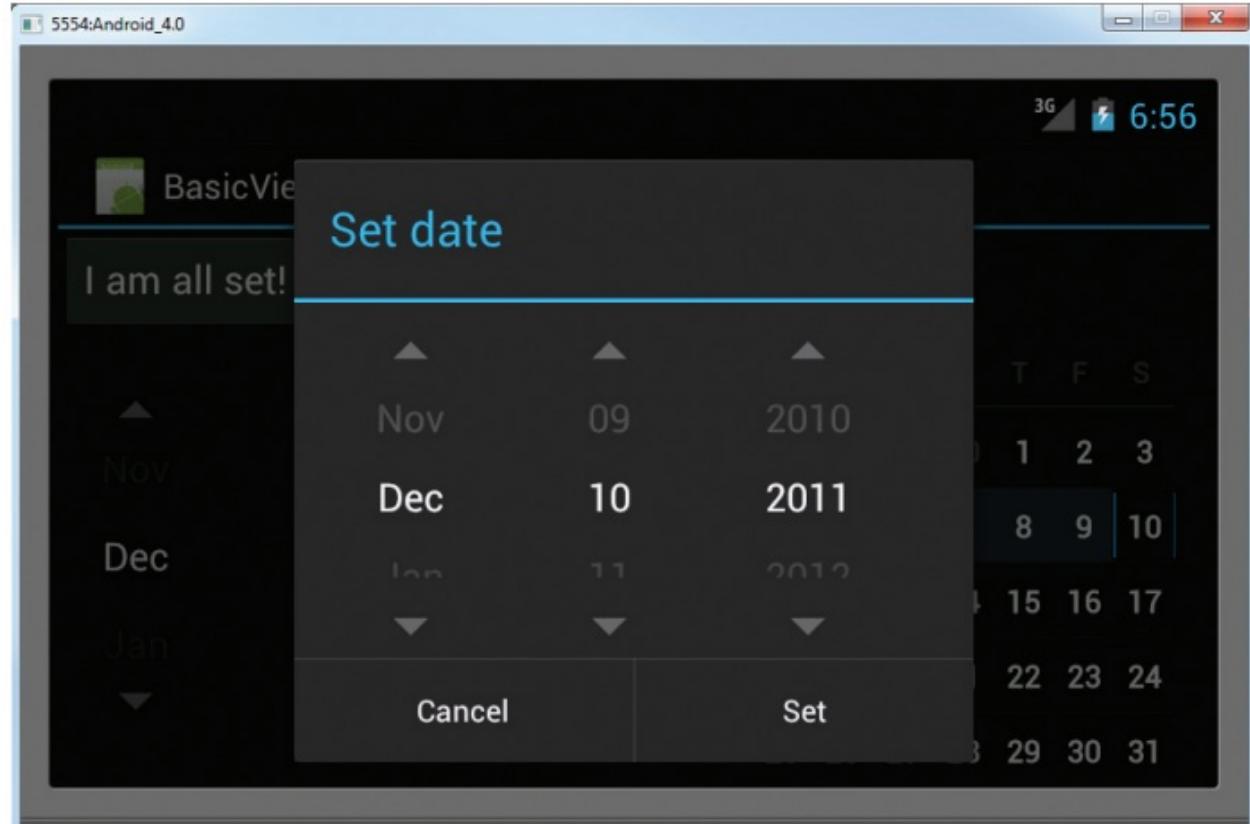
@Override
protected Dialog onCreateDialog(int id)
{
    switch (id) {
    case DATE_DIALOG_ID:
        return new DatePickerDialog(
            this, mDateSetListener, yr, month, day);

    }
    return null;
}

private DatePickerDialog.OnDateSetListener mDateSetListener =
    new DatePickerDialog.OnDateSetListener()
{
    public void onDateSet(
        DatePicker view, int year, int monthOfYear, int dayOfMonth)
    {
        yr = year;
    }
}

```

```
month = monthOfYear;
day = dayOfMonth;
Toast.makeText(getApplicationContext(),
    "You have selected : " + (month + 1) +
    "/" + day + "/" + year,
    Toast.LENGTH_SHORT).show();
}
};
```



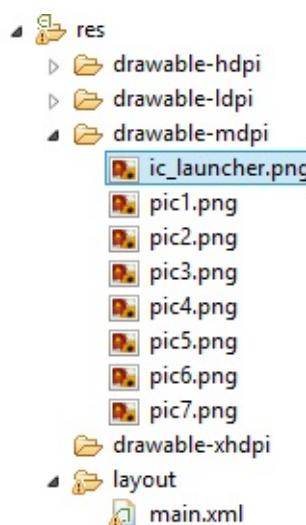
Hình: DatePickerDialog được hiển thị trong hộp thoại

Hiển thị ảnh với ImageView và Gallery

Ảnh là đối tượng được sử dụng rất tích cực trong các ứng dụng hiện đại. Trong phần này ta sẽ tìm hiểu cách hiển thị ảnh trong Android với ImageView và hiển thị danh sách ảnh với Gallery view. Ta sẽ xem xét 2 loại view này trong một ví dụ tương đối điển hình: hiển thị danh sách ảnh cho phép người dùng cuộn và chọn ảnh cần xem. Ảnh được chọn sẽ được hiển thị to hơn ở bên dưới. Trước tiên ta chuẩn bị một số ảnh cho dự án này:



Các ảnh này ta sẽ đặt vào thư mục res/ drawable-mdpi:



Trong mã nguồn, các ảnh này sẽ được truy cập thông qua id của nó trong lớp R (resource):

```
R.drawable.pic1,
R.drawable.pic2,
R.drawable.pic3,
R.drawable.pic4,
R.drawable.pic5,
R.drawable.pic6,
R.drawable.pic7
```

Layout của activity cho ví dụ này như sau:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    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="Images of San Francisco" />

    <Gallery
        android:id="@+id/gallery1"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content" />

    <ImageView
        android:id="@+id/image1"
        android:layout_width="320dp"
        android:layout_height="250dp"
        android:scaleType="fitXY" />

</LinearLayout>
```

Mã nguồn của activity này như sau:

```
public class GalleryActivity extends Activity {
    //---the images to display---
    Integer[] imageIDs = {
        R.drawable.pic1,
        R.drawable.pic2,
        R.drawable.pic3,
        R.drawable.pic4,
        R.drawable.pic5,
        R.drawable.pic6,
        R.drawable.pic7
    };

    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        Gallery gallery = (Gallery) findViewById(R.id.gallery1);

        gallery.setAdapter(new ImageAdapter(this));
        gallery.setOnItemClickListener(new OnItemClickListener()
        {
            public void onItemClick(AdapterView<?> parent, View v,
                int position, long id)
            {
                Toast.makeText(getApplicationContext(),
                    "pic" + (position + 1) + " selected",
                    Toast.LENGTH_SHORT).show();

                //---display the images selected---
                ImageView imageView = (ImageView) findViewById(R.id.image1);
                imageView.setImageResource(imageIDs[position]);
            }
        });
    }

    public class ImageAdapter extends BaseAdapter
    {
        Context context;
        int itemBackground;

        public ImageAdapter(Context c)
        {
```

```

        context = c;
        //---setting the style---
        TypedArray a = obtainStyledAttributes(R.styleable.Gallery1);

        itemBackground = a.getResourceId(
            R.styleable.Gallery1_android_galleryItemBackground, 0);

        a.recycle();
    }

    //---returns the number of images---
    public int getCount() {
        return imageIDs.length;
    }

    //---returns the item---
    public Object getItem(int position) {
        return position;
    }

    //---returns the ID of an item---
    public long getItemId(int position) {
        return position;
    }

    //---returns an ImageView view---
    public View getView(int position, View convertView, ViewGroup parent) {
        ImageView imageView;
        if (convertView == null) {
            imageView = new ImageView(context);
            imageView.setImageResource(imageIDs[position]);
            imageView.setScaleType(ImageView.ScaleType.FIT_XY);
            imageView.setLayoutParams(new Gallery.LayoutParams(150, 120));
        } else {
            imageView = (ImageView) convertView;
        }
        imageView.setBackgroundResource(itemBackground);
        return imageView;
    }
}

```

Sau khi chạy ứng dụng, ta sẽ quan sát thấy danh sách các ảnh nhỏ ở phía trên cùng của activity, ta có thể cuộn ngang danh sách này để chọn ảnh muốn xem, ảnh được chọn sẽ luôn được căn giữa trong *Gallery*. Sau khi được chọn, ảnh đó sẽ được hiển thị bên dưới (xem hình minh họa bên dưới).



Hình: Gallery

Ta cần giải thích thêm một chút về đoạn mã nguồn của Activity phía trên. Đầu tiên, ta lưu danh sách ID của các ảnh cần thêm vào Gallery:

```
Integer[] imageIDs = {
    R.drawable.pic1,
    R.drawable.pic2,
    R.drawable.pic3,
    R.drawable.pic4,
    R.drawable.pic5,
    R.drawable.pic6,
    R.drawable.pic7
};
```

Sau đó, để thêm các ảnh này vào Gallery, ta cần một lớp trung gian, gọi là adapter. Adapter này sẽ cung cấp nguồn nội dung cho các đối tượng gồm nhiều phần tử như Gallery hay các danh sách. Ta sẽ xem chi tiết adapter này ở phía dưới. Để gắn adapter này vào gallery, ta dùng hàm

```
setAdapter:
gallery.setAdapter(new ImageAdapter(this));
```

Khi một ảnh trong gallery được chọn, ta sẽ hiển thị ảnh to phía dưới. Để làm việc này, ta cần thêm hàm xử lý sự kiện

onItemClick của gallery:

```
gallery.setOnItemClickListener(new OnItemClickListener()
{
    public void onItemClick(AdapterView<?> parent, View v,
    int position, long id)
    {
        Toast.makeText(getApplicationContext(),
        "pic" + (position + 1) + " selected",
        Toast.LENGTH_SHORT).show();

        //---display the images selected---
        ImageView imageView = (ImageView) findViewById(R.id.image1);
        imageView.setImageResource(imageIDs[position]);
    }
});
```

Mỗi lớp adapter cho các view nhiều đối tượng (như gallery) cần kế thừa từ lớp `BaseAdapter` và cần nạp chồng các phương thức sau:

```
public class ImageAdapter extends BaseAdapter
{
    public ImageAdapter(Context c){ ... }

    public int getCount(){ ... }

    public Object getItem(int position) { ... }

    public long getItemId(int position) { ... }

    public View getView(int position, View convertView, ViewGroup parent) {...}
}
```

Trong đó hàm `getView` trả về view của đối tượng con (item) trong danh sách, tại vị trí position.

Một số view sử dụng Adapter làm nguồn dữ liệu trong Android có thể kể đến như:

- ListView: danh sách các đối tượng (theo chiều dọc)
- GridView: danh sách các đối tượng dạng bảng (nhiều cột, cuộn theo chiều dọc)
- Spinner: danh sách xổ xuống (giống khái niệm combo box trong lập trình web)
- Gallery: thư viện ảnh như ví dụ ở trên

Android cũng định nghĩa sẵn một số lớp con của lớp `BasicAdapter` như:

- ListAdapter
- ArrayAdapter
- CursorAdapter
- SpinnerAdapter

Chúng ta sẽ tìm hiểu một số lớp trong số đó trong phần còn lại của giáo trình.

Sử dụng ListView để hiển thị danh sách dài

Trong Android để hiển thị tập hợp nhiều phần tử cùng loại, ta dùng danh sách. Có 2 loại danh sách được định nghĩa sẵn là `ListView` và `SpinnerView`. Trong phần này ta sẽ lần lượt xem xét từng loại danh sách này.

ListView

`ListView` hiển thị danh sách các đối tượng con dưới dạng danh sách dọc, có khả năng cuộn khi chiều dài danh sách vượt quá chiều cao của view mẹ. Ta sẽ xem xét `ListView` trong trường hợp đơn giản nhất: hiển thị danh sách các phần tử dạng chữ.

Trước tiên, ta chuẩn bị mảng dữ liệu các chữ cần hiển thị trong danh sách. Ta có thể nhập cứng (hard code) danh sách này trong mã nguồn java của Activity như sau:

```
String[] presidents= {
    "Dwight D. Eisenhower",
    "John F. Kennedy",
    "Lyndon B. Johnson",
    "Richard Nixon",
    "Gerald Ford",
    "Jimmy Carter",
    "Ronald Reagan",
    "George H. W. Bush",
    "Bill Clinton",
    "George W. Bush",
    "Barack Obama"
};
```

Tuy nhiên cách làm này làm cho mã nguồn rối hơn, gây khó khăn cho việc bảo trì, cũng như hạn chế khả năng địa phương hóa (thay đổi ngôn ngữ cho ứng dụng). Vì vậy, trong lập trình Android, phương pháp được khuyên dùng là định nghĩa các dữ liệu tĩnh này trong thư mục "res". Cụ thể, đối với mảng chữ như trên, ta có thể định nghĩa trong file "res/values/string.xml" như sau:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="hello">Hello World, BasicViews5Activity!</string>
    <string name="app_name">BasicViews5</string>
    <string-array name="presidents_array">
        <item>Dwight D. Eisenhower</item>
        <item>John F. Kennedy</item>
        <item>Lyndon B. Johnson</item>
        <item>Richard Nixon</item>
        <item>Gerald Ford</item>
        <item>Jimmy Carter</item>
        <item>Ronald Reagan</item>
        <item>George H. W. Bush</item>
        <item>Bill Clinton</item>
        <item>George W. Bush</item>
        <item>Barack Obama</item>
    </string-array>
</resources>
```

Sau đó, trong mã nguồn Java, ta có thể dễ dàng lấy ra mảng này bằng cách như sau:

```
String[] presidents;
presidents = getResources().getStringArray(R.array.presidents_array);
```

Để tạo một danh sách, trước tiên ta khai báo 1 `ListView` trong file layout của Activity:

```

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

    <ListView
        android:id="@+id/android:list"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />

</LinearLayout>

```

Đối với Activity có một ListView bên trong, Android định nghĩa sẵn lớp con của Activity là ListActivity giúp cho quá trình làm việc với ListView trở đơn giản hơn. Để có thể dụng được ListActivity này, có 2 việc cần phải làm:

- Activity phải kế thừa từ lớp `android.app.ListActivity`
- ListView trong file layout của Activity phải có id là `"@+id/android:list"`

Mã nguồn của Activity sẽ như sau:

```

public class BasicViews5Activity extends ListActivity {

    String[] presidents;

    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        setContentView(R.layout.main);

        presidents =
            getResources().getStringArray(R.array.presidents_array);

        setListAdapter(new ArrayAdapter<String>(this,
            android.R.layout.simple_list_item_checked, presidents));
    }

    @Override
    public void onListItemClick(
        ListView parent, View v, int position, long id)
    {
        Toast.makeText(this,
            "You have selected " + presidents[position],
            Toast.LENGTH_SHORT).show();
    }
}

```

ListView trong Android cũng lấy nội dung từ một Adapter giống như trường hợp của Gallery ta đã làm quen trước đó. Đối với ListActivity, ta không cần lấy tham chiếu đến ListView một cách minh bạch, mà có thể gọi thẳng hàm `setListAdapter` để đặt adapter cho listView trong Activity (được đặt id theo quy định như đã nói ở trên).

Để xử lý sự kiện người dùng bấm chọn một phần tử trong ListView, ta chỉ cần nạp chồng hàm `onListItemClick` như đoạn mã ở trên. Trong ví dụ này ta chỉ đơn thuần in ra chữ của phần tử được chọn.

Activity trên khi chạy trên *emulator* sẽ có dạng như sau:

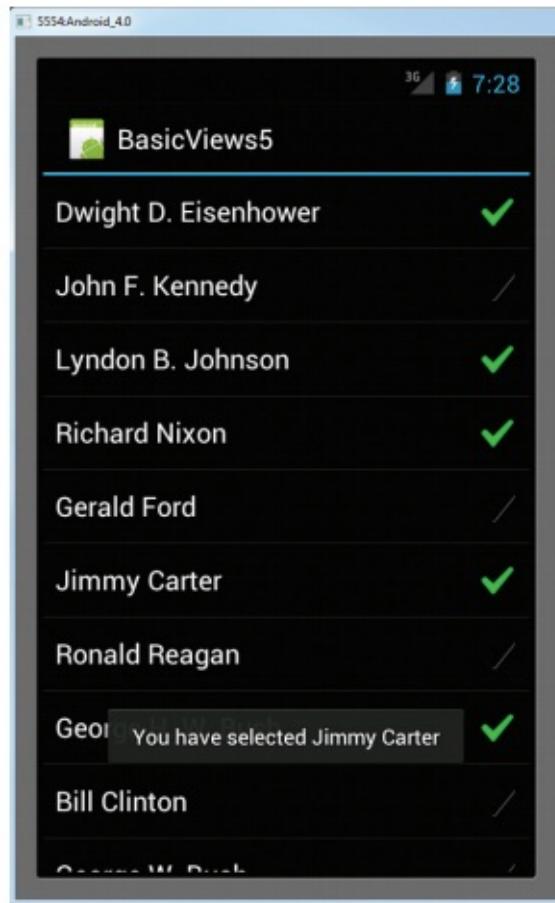


Hình: Hiển thị danh sách với ListView

Ta có thể vuốt lên/xuống màn hình để xem toàn bộ danh sách. Có rất nhiều tùy chọn có thể làm với ListView. Bạn đọc tự tìm hiểu coi như bài tập. Ở đây ta chỉ nói thêm một tính năng của ListView cho phép lựa chọn nhiều phần tử cùng lúc (multi-item selection). Để bật tính năng này, ta chỉ cần gọi hàm `setChoiceMode` của ListView như sau:

```
ListView lstView = getListView();
lstView.setChoiceMode(ListView.CHOICE_MODE_MULTIPLE);
```

ListView với thuộc tính lựa chọn nhiều phần tử được bật sẽ có dạng như sau:



Hình: ListView cho phép lựa chọn nhiều phần tử cùng lúc

SpinnerView

ListView rất tiện dụng cho việc hiển thị danh sách các phần tử đồng dạng. Tuy nhiên, ListView chiếm tương đối nhiều diện tích trên màn hình. Trong thực tế có nhiều trường hợp ta chỉ cần hiển thị phần tử đang chọn của danh sách, khi bấm vào phần tử này, sẽ hiện ra danh sách đầy đủ các phần tử còn lại để ta lựa chọn. Để làm được việc này, ta dùng SpinnerView. Để dễ hình dung, ta có thể nhiều SpinnerView chính là ComboBox trong lập trình web và Windows Form.

Trong ví dụ dưới đây, ta vẫn hiển thị danh sách các phần tử dạng chữ như ví dụ trước, tuy nhiên dùng SpinnerView thay cho ListView.

Trước tiên, ta thêm khai báo một SpinnerView trong file layout của Activity:

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

    <Spinner
        android:id="@+id/spinner1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:drawSelectorOnTop="true" />

</LinearLayout>
```

Sau đó, trong hàm `onCreate` của Activity, ta cần thêm mã nguồn để truy xuất đến SpinnerView này, đặt adapter cho nó và thêm hàm xử lý sự kiện khi ta chọn một phần tử của spinner:

```

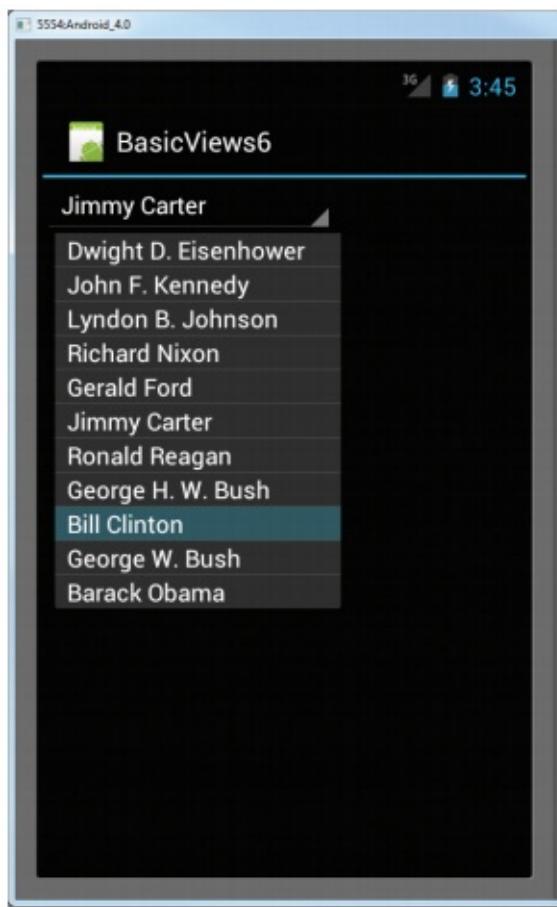
presidents =
    getResources().getStringArray(R.array.presidents_array);
Spinner s1 = (Spinner) findViewById(R.id.spinner1);

ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,
    android.R.layout.simple_spinner_item, presidents);
s1.setAdapter(adapter);
s1.setOnItemSelectedListener(new OnItemSelectedListener()
{
    @Override
    public void onItemSelected(AdapterView<?> arg0,
    View arg1, int arg2, long arg3)
    {
        int index = arg0.getSelectedItemPosition();
        Toast.makeText(getApplicationContext(),
            "You have selected item : " + presidents[index],
            Toast.LENGTH_SHORT).show();
    }

    @Override
    public void onNothingSelected(AdapterView<?> arg0) { }
});

```

Khi một phần tử của SpinnerView được chọn, ta chỉ đơn thuần in lên màn hình thông báo dạng Toast. Chạy ứng dụng vừa tạo lên thiết bị hoặc emulator, spinner view của chúng ta sẽ có dạng như hình dưới đây:



Hình: Spinner View

Hiển thị nội dung trang web với WebView

WebView cho phép chúng ta nhúng trình duyệt web vào bên trong ứng dụng của mình. Sử dụng *web view* rất hữu dụng trong trường hợp chúng ta cần hiển thị nội dung một trang web bên trong ứng dụng, cũng như khi chúng ta muốn thiết kế một phần hoặc thậm chí toàn bộ giao diện bằng ngôn ngữ web quen thuộc (*HTML5, Javascript, CSS*).

Trong ví dụ dưới đây, ta sẽ sử dụng *web view* để hiển thị nội dung của một trang web trên *Internet*, cụ thể ta sẽ hiển thị một biểu đồ từ dịch vụ *Chart* của *Google*.

Trước tiên, ta cần thêm một *WebView* vào file layout của Activity:

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

    <WebView android:id="@+id/webview1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />

</LinearLayout>
```

Sau đó trong hàm `onCreate` của Activity, ta cần thêm đoạn mã để cấu hình *webview* này hiển thị nội dung trang web từ *Internet*:

```
WebView wv = (WebView) findViewById(R.id.webview1);

WebSettings webSettings = wv.getSettings();
webSettings.setBuiltInZoomControls(true);
wv.loadUrl(
    "http://chart.apis.google.com/chart" +
    "?chs=300x225" +
    "&cht=v" +
    "&chco=FF6342,ADDE63,63C6DE" +
    "&chd=t:100,80,60,30,30,30,10" +
    "&chdl=A|B|C");
```

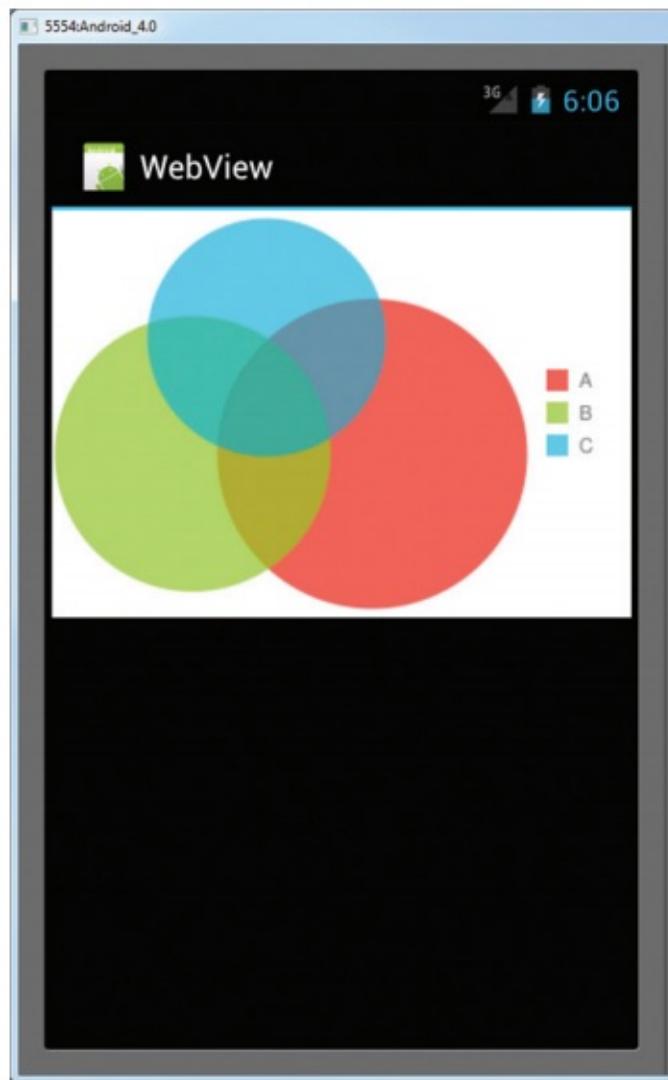
Ngoài ra, do ứng dụng muốn truy cập vào Internet để lấy dữ liệu từ trang web của *Google Chart*, ta cần khai báo quyền truy cập Internet một cách tường minh trong file *AndroidManifest.xml* như sau:

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

    <uses-sdk android:minSdkVersion="14" />
    <uses-permission android:name="android.permission.INTERNET"/>

    <application
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name" >
        ...
    </application>
</manifest>
```

Chạy ứng dụng trên điện thoại hoặc *Emulator*, ta sẽ thấy một đồ thị được vẽ bởi dịch vụ *Google Chart* trên màn hình:



Hình: WebView hiển thị trang Web từ Internet

Ngoài ra, *WebView* cũng có thể được dùng để hiển thị giao diện bằng *HTML* ta tự khai báo trong mã nguồn như ví dụ dưới đây:

```
WebView wv = (WebView) findViewById(R.id.webview1);

WebSettings webSettings = wv.getSettings();
webSettings.setBuiltInZoomControls(true);

final String mimeType = "text/html";
final String encoding = "UTF-8";
String html = "<H1>A simple HTML page</H1><body>" +
    "<p>The quick brown fox jumps over the lazy dog</p></body>";
wv.loadDataWithBaseURL("", html, mimeType, encoding, "");
```

Chạy ứng dụng, ta sẽ thấy giao diện tương tự như minh họa trong hình sau:



Hình: WebView hiển thị trực tiếp mã HTML

Lưu trữ dữ liệu

Trong chương này chúng ta sẽ xem xét cách thức lưu trữ dữ liệu trong Android. Lưu trữ dữ liệu là tính năng quan trọng đối với ứng dụng, giúp cho người dùng có thể dùng lại được những dữ liệu trước đó mà không cần nhập lại. Trong Android có 3 cách để lưu lại dữ liệu:

- Cơ chế “cấu hình chia sẻ” (*shared preferences*) được dùng để lưu những dữ liệu nhỏ dưới dạng *key-value* (tên khóa – giá trị khóa)
- Lưu dữ liệu cố định vào tập tin trong bộ nhớ trong hoặc bộ nhớ ngoài của điện thoại
- Lưu dữ liệu sử dụng cơ sở dữ liệu quan hệ cục bộ *SQLite*

Chúng ta sẽ lần lượt duyệt qua các phương pháp kê trên trong chương này.

Lưu trữ dữ liệu cố định với shared preferences

Android cung cấp sẵn một cơ chế đơn giản giúp chúng ta lưu trữ nhanh các dữ liệu ngắn như cấu hình ứng dụng, tên đăng nhập, email... và lấy lại dữ liệu đã ghi này trong các lần chạy ứng dụng tiếp theo.

Cách thức thuận tiện nhất là mô tả tập trung các cấu hình ta cần lưu lại trong một Activity (thường là màn hình “Settings” của ứng dụng). Android cung cấp sẵn một loại Activity đặc biệt là *PreferenceActivity* giúp đơn giản hóa quá trình này. Ví dụ dưới đây minh họa cách sử dụng Activity này.

Để sử dụng được *PreferenceActivity*, trước hết ta mô tả các thông tin cần lưu lại trong một tài liệu xml nằm trong thư mục “res/xml”. Trong ví dụ này, ta tạo file “res/xml/myapppreferences.xml” với nội dung như sau:

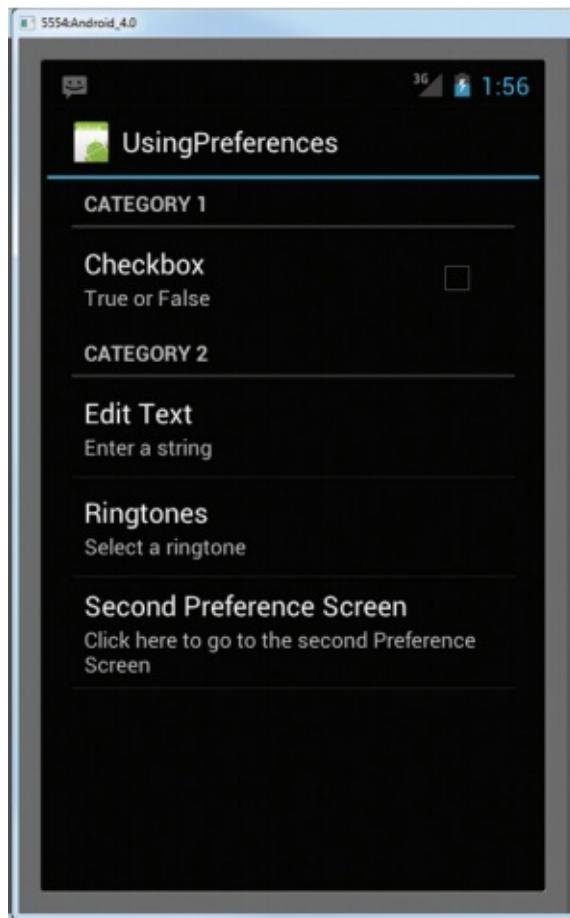
```
<?xml version="1.0" encoding="utf-8"?>
<PreferenceScreen
    xmlns:android="http://schemas.android.com/apk/res/android">
    <PreferenceCategory android:title="Category 1">
        <CheckBoxPreference
            android:title="Checkbox"
            android:defaultValue="false"
            android:summary="True or False"
            android:key="checkboxPref" />
    </PreferenceCategory>
    <PreferenceCategory android:title="Category 2">
        <EditTextPreference
            android:summary="Enter a string"
            android:defaultValue="[Enter a string here]"
            android:title="Edit Text"
            android:key="editTextPref"
            />
        <RingtonePreference
            android:summary="Select a ringtone"
            android:title="Ringtones"
            android:key="ringtonePref"
            />
    <PreferenceScreen
        android:title="Second Preference Screen"
        android:summary=
            "Click here to go to the second Preference Screen"
        android:key="secondPrefScreenPref" >
        <EditTextPreference
            android:summary="Enter a string"
            android:title="Edit Text (second Screen)"
            android:key="secondEditTextPref"
            />
    </PreferenceScreen>
    </PreferenceCategory>
</PreferenceScreen>
```

Sau đó, ta cần tạo một Activity kế thừa từ *PreferenceActivity* với nội dung như sau:

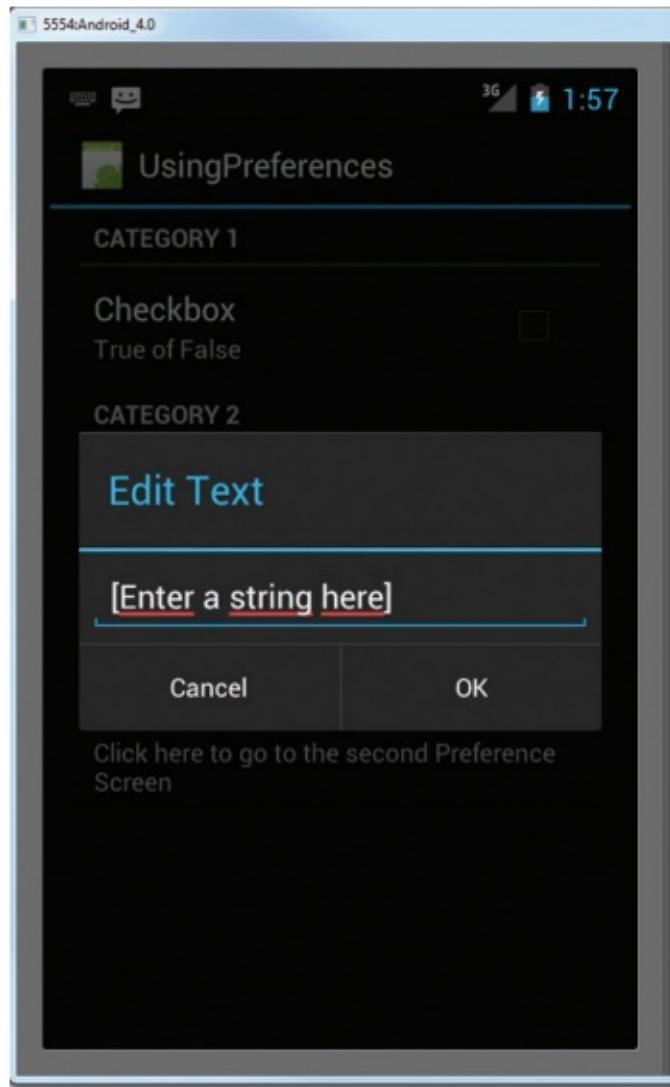
```
import android.os.Bundle;
import android.preference.PreferenceActivity;

public class AppPreferenceActivity extends PreferenceActivity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        PreferenceManager prefMgr = getPreferenceManager();
        prefMgr.setSharedPreferencesName("appPreferences");
        //---load the preferences from an XML file---
        addPreferencesFromResource(R.xml.myapppreferences);
    }
}
```

Chạy ứng dụng với Activity như trên, ta sẽ có ngay một màn hình settings cho ứng dụng:

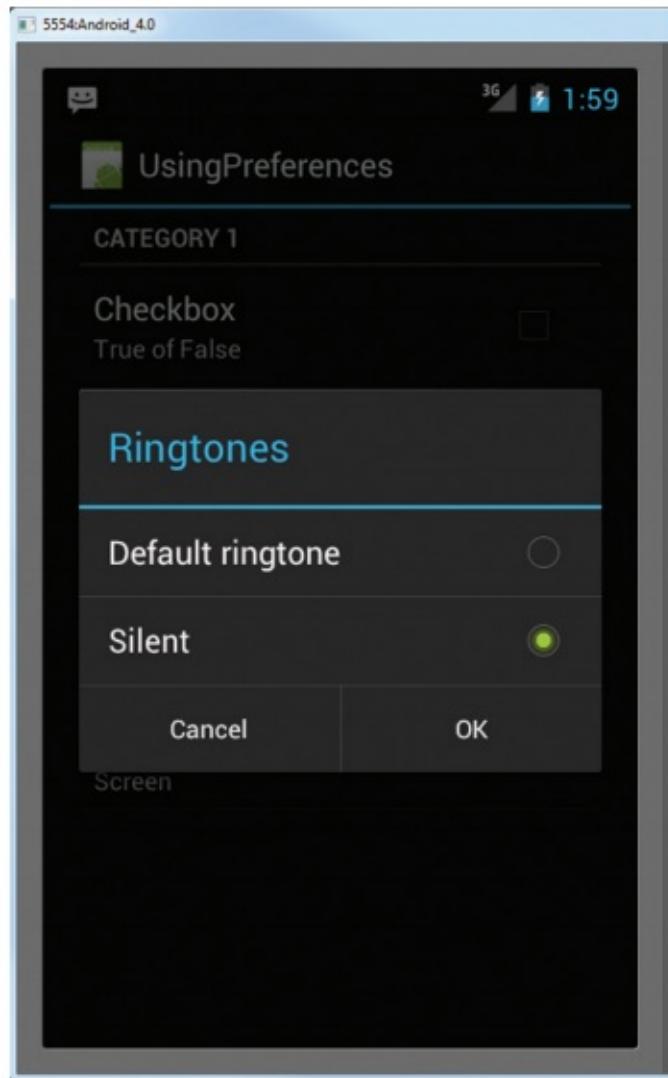


Nội dung của màn hình này được mô tả hoàn toàn trong file *xml* ở trên, trong đó chia các cấu hình này thành 2 danh mục (*category 1* và *category 2*) và một số loại cấu hình khác nhau như *checkbox*, *edittext*, nhạc chuông và một ví dụ minh họa cho việc mở thêm màn hình cấu hình thứ 2 (khi màn hình đầu có nhiều chi tiết). Bấm vào *edittext*, sẽ có cửa sổ *popup* cho bạn nhập giá trị cần lưu lại:



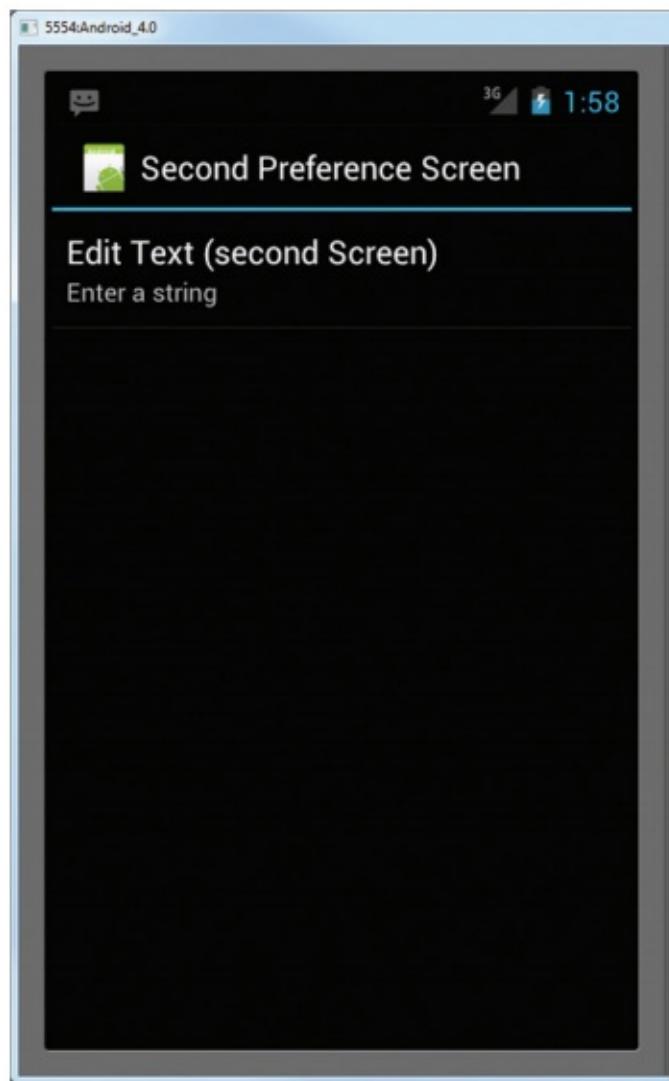
Hình: Cửa sổ popup cho phép nhập cấu hình dạng text (EditText)

Cấu hình nhạc chuông:



Hình: Cửa sổ popup cho phép lựa chọn cấu hình từ danh sách các RadioButton

Màn hình preference thứ 2:



Hình: Màn hình cấu hình mở rộng

Sau khi bạn thay đổi giá trị của các mục cấu hình trong Activity này, hệ thống sẽ tự động lưu lại giá trị của chúng để có thể sử dụng được trong các lần tiếp theo.

Việc lưu trữ dữ liệu này là trong suốt với người dùng. Tuy nhiên với máy ảo Android ta có thể xem được cụ thể các giá trị này. Trên thực tế, chúng được lưu trong 1 file xml nằm trên bộ nhớ trong, trong vùng nhớ chỉ có thể truy cập được bởi ứng dụng tạo ra nó (thư mục `/data/data/{package-name}/shared_prefs/appPreferences.xml`):

Name	Size	Date	Time	Permissions
net.learn2develop.Menus		2013-11-06	22:26	drwxr-x--x
net.learn2develop.UsingPreferences		2013-11-06	22:30	drwxr-x--x
cache		2013-11-06	22:29	drwxrwx--x
lib		2013-11-06	22:29	lrwxrwxrwx
shared_prefs		2013-11-06	22:32	drwxrwx--x
appPreferences.xml	331	2013-11-06	22:32	-rw-rw----

Nội dung file này có dạng như sau:

```
<?xml version='1.0' encoding='utf-8' standalone='yes' ?>
```

```

<map>
    <string name="editTextPref">HUMG - Software engineer</string>
    <string name="secondEditTextPref">HUMG - 2nd screen text</string>
    <string name="ringtonePref">content://settings/system/ringtone</string>
    <boolean name="checkboxPref" value="true" />
</map>

```

Để lấy giá trị của các cấu hình này trong code, ta làm như sau:

```

SharedPreferences appPrefs = getSharedPreferences("appPreferences", MODE_PRIVATE);
Toast.makeText(this, appPrefs.getString("editTextPref", ""), Toast.LENGTH_LONG).show();

```

Trong đó, `appPreferences` là tên của file cấu hình cần mở (được đặt ở hàm `prefMgr.setSharedPreferencesName("appPreferences")` ; phía trên), còn `editTextPref` là tên của thuộc tính cần lấy giá trị (được đặt trong file `res/xml/myapppreferences.xml` ở trên).

Ngoài ra, ta cũng có thể đặt lại giá trị của các cấu hình này bằng tay mà không cần thông qua `PreferenceActivity` bằng cách sử dụng `sharedPreferences.Editor` như sau:

```

SharedPreferences appPrefs =
    getSharedPreferences("appPreferences", MODE_PRIVATE);

SharedPreferences.Editor prefsEditor = appPrefs.edit();
prefsEditor.putString("editTextPref",
    ((EditText) findViewById(R.id.txtString)).getText().toString());
prefsEditor.commit();

```

Sau khi thay đổi giá trị của các thuộc tính cần thay đổi, ta cần gọi phương thức `commit()` của lớp `Editor` này để các thay đổi có hiệu lực (tiến hành ghi vào file xml trong bộ nhớ trong như mô tả ở trên).

Lưu trữ dữ liệu bằng file trên bộ nhớ trong và bộ nhớ ngoài

Trong trường hợp cần lưu lại dữ liệu phức tạp hơn (khó có thể lưu lại dạng *key-value* trong *shared preference*), ta có thể dùng hệ thống file. Trong Android, để làm việc (nhập/xuất) với file, ta có thể dùng các lớp của gói `java.io`. Trong phần này, ta sẽ xem cách làm việc với file trong bộ nhớ trong lẫn bộ nhớ ngoài.

Làm việc với file trong bộ nhớ trong

Ta sẽ tạo một Activity có một ô nhập văn bản (`EditText`) và 2 nút bấm cho phép ghi và đọc văn bản này vào file. Layout của Activity này như sau:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    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="Please enter some text" />

    <EditText
        android:id="@+id/txtText1"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content" />

    <Button
        android:id="@+id/btnSave"
        android:text="Save"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:onClick="onClickSave" />

    <Button
        android:id="@+id/btnLoad"
        android:text="Load"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:onClick="onClickLoad" />
</LinearLayout>
```

Mã nguồn của Activity với 2 hàm đọc (`onClickLoad`) và ghi (`onClickSave`) vào file như sau:

```
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.OutputStreamWriter;

import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.widget.EditText;
import android.widget.Toast;

public class FilesActivity extends Activity {
    EditText textBox;
    static final int READ_BLOCK_SIZE = 100;

    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
```

```

    textBox = (EditText) findViewById(R.id.txtText1);

}

public void onClickSave(View view) {
    String str = textBox.getText().toString();
    try
    {
        FileOutputStream fOut =
            openFileOutput("textfile.txt",
                MODE_PRIVATE);

        OutputStreamWriter osw = new
            OutputStreamWriter(fOut);

        //---write the string to the file---
        osw.write(str);
        osw.flush();
        osw.close();

        //---display file saved message---
        Toast.makeText(getApplicationContext(),
            "File saved successfully!",
            Toast.LENGTH_SHORT).show();

        //---clears the EditText---
        textBox.setText("");
    }
    catch (IOException ioe)
    {
        ioe.printStackTrace();
    }
}

public void onClickLoad(View view) {
    try
    {
        FileInputStream fIn =
            openFileInput("textfile.txt");
        InputStreamReader isr = new
            InputStreamReader(fIn);

        char[] inputBuffer = new char[READ_BLOCK_SIZE];
        String s = "";

        int charRead;
        while ((charRead = isr.read(inputBuffer))>0)
        {
            //---convert the chars to a String---
            String readString =
                String.valueOf(inputBuffer, 0,
                    charRead);
            s += readString;

            inputBuffer = new char[READ_BLOCK_SIZE];
        }
        //---set the EditText to the text that has been
        // read---
        textBox.setText(s);

        Toast.makeText(getApplicationContext(),
            "File loaded successfully!",
            Toast.LENGTH_SHORT).show();
    }
    catch (IOException ioe) {
        ioe.printStackTrace();
    }
}
}

```

Trong đoạn mã trên, ta thấy việc đọc và ghi vào file tương đối đơn giản và quen thuộc. Để ghi dữ liệu vào file, ta tạo một đối tượng `OutputStreamWriter` trên luồng xuất `FileOutputStream` và tiến hành ghi vào qua phương thức `write`. Sau đó, gọi `flush` để đẩy hết dữ liệu trong bộ đệm vào file và đóng luồng lại:

```

FileOutputStream fOut = openFileOutput("textfile.txt", MODE_PRIVATE);

OutputStreamWriter osw = new OutputStreamWriter(fOut);

//---write the string to the file---
osw.write(str);
osw.flush();
osw.close();

```

Để đọc nội dung file này, ta cũng thao tác tương tự. Tạo đối tượng `InputStreamReader` từ luồng nhập liệu (`FileInputStream`) và tiến hành đọc dữ liệu bằng phương thức `read()`. Mỗi lần đọc sẽ đọc `READ_BLOCK_SIZE` byte và lưu vào bộ đệm (mảng byte). Nội dung này sẽ được chuyển thành string và nối thêm vào biến `s`. Quá trình được lặp lại cho đến khi hết nội dung của file:

```

FileInputStream fIn = openFileInput("textfile.txt");
InputStreamReader isr = new InputStreamReader(fIn);

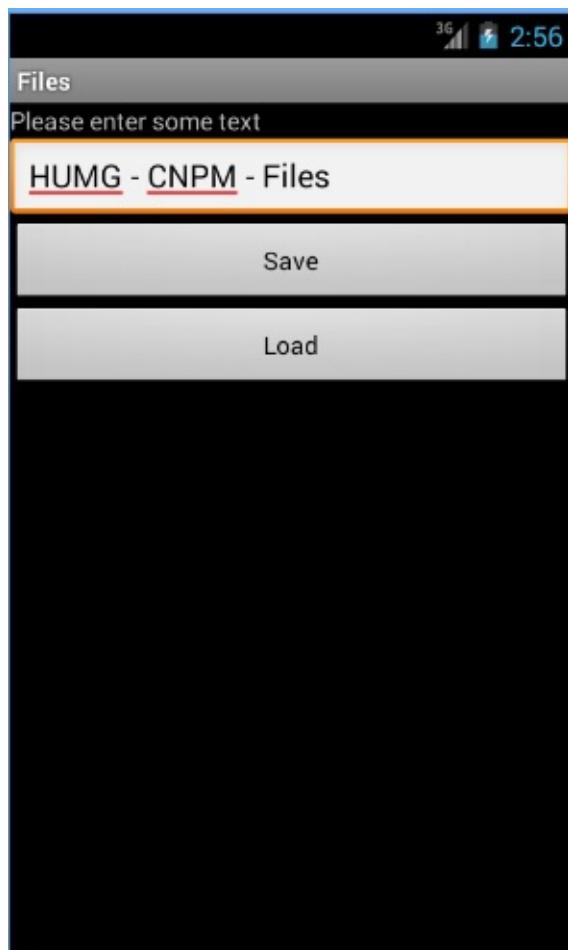
char[] inputBuffer = new char[READ_BLOCK_SIZE];
String s = "";

int charRead;
while ((charRead = isr.read(inputBuffer))>0)
{
    String readString = String.valueOf(inputBuffer, 0, charRead);
    s += readString;
    inputBuffer = new char[READ_BLOCK_SIZE];
}
textBox.setText(s);

```

Một câu hỏi đặt ra là file "textfile.txt" ở trên được tạo ra ở đâu trong cây thư mục. Câu trả lời là nó được tạo ra trong bộ nhớ trong của thiết bị, trong thư mục dành riêng cho ứng dụng (`/data/data/{package-name}/files`)

Chạy ứng dụng, nhập nội dung cho ô nhập liệu và bấm Save, ta sẽ thấy nội dung văn bản này được ghi vào file trong bộ nhớ trong.



Hình: Giao diện nhập/hiển thị dữ liệu đọc và ghi file

Name	Size	Date	Time	Permissions	Info
net.learn2develop.Files		2013-11-07	02:55	drwxr-x--x	
cache		2013-11-07	02:46	drwxrwx--x	
files		2013-11-07	02:55	drwxrwx--x	
textfile.txt	19	2013-11-07	02:55	-rw-rw----	
lib		2013-11-07	02:46	lrwxrwxrwx	-> /data/a...
net.learn2develop.Menus		2013-11-07	02:41	drwxr-x--x	
net.learn2develop.UsingPreferences		2013-11-07	02:41	drwxr-x--x	
dontpanic		2013-11-03	22:37	drwxr-x---	
drm		2013-11-03	22:37	drwxrwx---	
local		2013-11-03	22:37	drwxr-x--x	
lost+found		2013-11-03	22:37	drwxrwx---	

Hình: Vị trí file dữ liệu vừa ghi trong bộ nhớ thiết bị

Kéo file này về máy tính để xem nội dung file, ta sẽ thấy nội dung văn bản ta nhập vào trước đó



Hình: Nội dung file dữ liệu vừa ghi trong bộ nhớ thiết bị

Làm việc với file trong bộ nhớ ngoài

File trong bộ nhớ trong chỉ được truy cập bởi ứng dụng tạo ra nó. Ngoài ra, dung lượng lưu trữ của bộ nhớ trong thường hạn chế hơn bộ nhớ ngoài (*SDCard*). Vì vậy, trong trường hợp ta muốn chia sẻ thông tin lưu trữ với các ứng dụng khác, ta nên sử dụng bộ nhớ ngoài. Làm việc với file trong bộ nhớ ngoài hoàn toàn tương tự với file trong bộ nhớ trong, chỉ khác phần lấy ra `FileInputStream` và `FileOutputStream`:

Thay vì dùng:

```
FileOutputStream fOut = openFileOutput("textfile.txt", MODE_PRIVATE);
```

Ta dùng:

```
File sdCard = Environment.getExternalStorageDirectory();
File directory = new File(sdCard.getAbsolutePath() + "/MyFiles");
directory.mkdirs();
File file = new File(directory, "textfile.txt");
FileOutputStream fOut = new FileOutputStream(file);
```

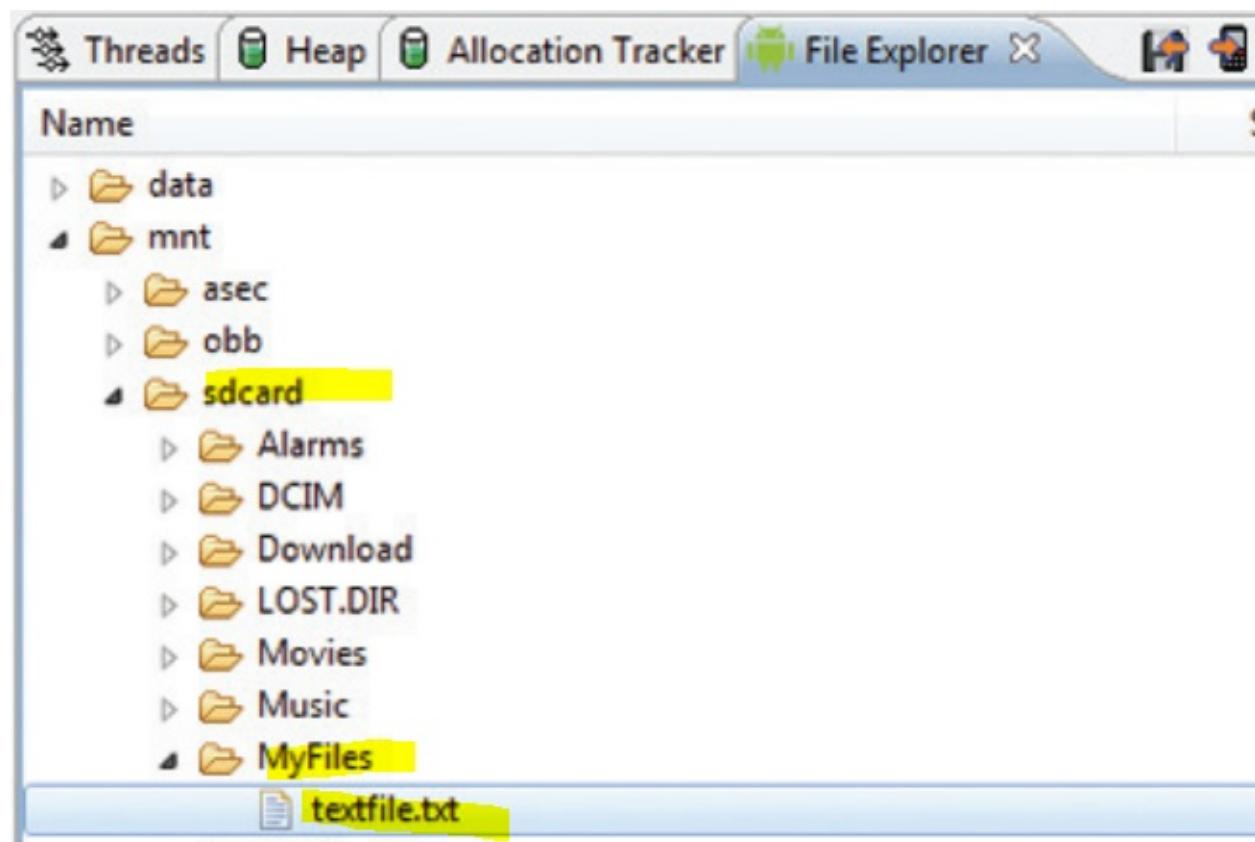
Và thay vì:

```
FileInputStream fIn = openFileInput("textfile.txt");
```

Ta dùng:

```
File sdCard = Environment.getExternalStorageDirectory();
File directory = new File (sdCard.getAbsolutePath() + "/MyFiles");
File file = new File(directory, "textfile.txt");
FileInputStream fIn = new FileInputStream(file);
InputStreamReader isr = new InputStreamReader(fIn);
```

Mọi thao tác vẫn như trường hợp trước, chỉ khác vị trí lưu trữ file trong cây thư mục:



Hình: Vị trí file dữ liệu vừa ghi trên bộ nhớ ngoài của thiết bị

Cơ sở dữ liệu SQLite trong ứng dụng Android

Trong phần trước, ta đã tìm hiểu cách lưu dữ liệu vào file và vào *shared preferences*. Tuy nhiên, với loại dữ liệu quan hệ thì việc sử dụng cơ sở dữ liệu quan hệ sẽ thuận tiện hơn rất nhiều. Ví dụ, ta cần lưu trữ kết quả kiểm tra của các sinh viên trong trường học. Dùng cơ sở dữ liệu sẽ cho phép chúng ta truy vấn kết quả của tập sinh viên nhất định theo các tiêu chí khác nhau. Việc thêm, bớt, thay đổi thông tin thông qua các câu truy vấn SQL cũng dễ dàng hơn nhiều so với việc thao tác trên file. Android sử dụng hệ cơ sở dữ liệu SQLite. CSDL do một ứng dụng tạo ra sẽ chỉ được truy xuất bởi ứng dụng đó, và file CSDL sẽ nằm trong bộ nhớ trong dành riêng cho ứng dụng (*/data/data/{package-name}/databases/*).

Một thói quen tốt thường được các lập trình viên kinh nghiệm sử dụng là tập trung tất cả mã lệnh truy cập đến CSDL vào một lớp riêng để thao tác trên CSDL trở nên trong suốt với môi trường ngoài. Chúng ta sẽ tạo trước một lớp như vậy, gọi là *DBAdapter*.

Tạo lớp DBAdapter

Trong ví dụ này ta sẽ tạo một CSDL tên là *MyDB*, chứa một bảng duy nhất là *contacts*. Bảng này chứa các trường *_id*, *name* và *email*. Lớp DBAdapter có mã nguồn như sau:

```
import android.content.ContentValues;
import android.content.Context;
import android.database.Cursor;
import android.database.SQLException;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;
import android.util.Log;

public class DBAdapter {
    static final String KEY_ROWID = "_id";
    static final String KEY_NAME = "name";
    static final String KEY_EMAIL = "email";
    static final String TAG = "DBAdapter";

    static final String DATABASE_NAME = "MyDB";
    static final String DATABASE_TABLE = "contacts";
    static final int DATABASE_VERSION = 2;

    static final String DATABASE_CREATE =
        "create table contacts (_id integer primary key autoincrement, "
        + "name text not null, email text not null);";

    final Context context;

    DatabaseHelper DBHelper;
    SQLiteDatabase db;

    public DBAdapter(Context ctx)
    {
        this.context = ctx;
        DBHelper = new DatabaseHelper(context);
    }

    private static class DatabaseHelper extends SQLiteOpenHelper
    {
        DatabaseHelper(Context context)
        {
            super(context, DATABASE_NAME, null, DATABASE_VERSION);
        }

        @Override
        public void onCreate(SQLiteDatabase db)
        {
            try {
                db.execSQL(DATABASE_CREATE);
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }
    }
}
```

```

    }

    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion)
    {
        Log.w(TAG, "Upgrading database from version " + oldVersion
            + " to " + newVersion + ", which will destroy all old data");
        db.execSQL("DROP TABLE IF EXISTS contacts");
        onCreate(db);
    }
}

//---opens the database---
public DBAdapter open() throws SQLException
{
    db = DBHelper.getReadableDatabase();
    return this;
}

//---closes the database---
public void close()
{
    DBHelper.close();
}

//---insert a contact into the database---
public long insertContact(String name, String email)
{
    ContentValues initialValues = new ContentValues();
    initialValues.put(KEY_NAME, name);
    initialValues.put(KEY_EMAIL, email);
    return db.insert(DATABASE_TABLE, null, initialValues);
}

//---deletes a particular contact---
public boolean deleteContact(long rowId)
{
    return db.delete(DATABASE_TABLE, KEY_ROWID + "=" + rowId, null) > 0;
}

//---retrieves all the contacts---
public Cursor getAllContacts()
{
    return db.query(DATABASE_TABLE, new String[] {KEY_ROWID, KEY_NAME,
        KEY_EMAIL}, null, null, null, null, null);
}

//---retrieves a particular contact---
public Cursor getContact(long rowId) throws SQLException
{
    Cursor mCursor =
        db.query(true, DATABASE_TABLE, new String[] {KEY_ROWID,
            KEY_NAME, KEY_EMAIL}, KEY_ROWID + "=" + rowId, null,
            null, null, null);
    if (mCursor != null) {
        mCursor.moveToFirst();
    }
    return mCursor;
}

//---updates a contact---
public boolean updateContact(long rowId, String name, String email)
{
    ContentValues args = new ContentValues();
    args.put(KEY_NAME, name);
    args.put(KEY_EMAIL, email);
    return db.update(DATABASE_TABLE, args, KEY_ROWID + "=" + rowId, null) > 0;
}
}

```

Trước tiên ta khai báo các hằng số như: tên CSDL, tên bảng, tên các trường để dễ dàng truy xuất và thay đổi trong quá trình phát triển. Ngoài ra, ta cũng khai báo phiên bản (do ta tự đánh số) của CSDL trong ứng dụng và viết sẵn câu truy vấn dùng để tạo CSDL:

```

static final String KEY_ROWID = "_id";
static final String KEY_NAME = "name";
static final String KEY_EMAIL = "email";
static final String TAG = "DBAdapter";

static final String DATABASE_NAME = "MyDB";
static final String DATABASE_TABLE = "contacts";
static final int DATABASE_VERSION = 2;

static final String DATABASE_CREATE =
    "create table contacts (_id integer primary key autoincrement, "
    + "name text not null, email text not null);";

```

Ta cũng tạo thêm một lớp cục bộ (lớp `DatabaseHelper` ở trên) để trợ giúp cho việc tạo CSDL và nâng cấp cấu trúc khi có sự thay đổi trong các phiên bản tiếp theo. Lớp này kế thừa từ lớp `SQLiteOpenHelper`. Hàm dựng của lớp này sẽ gọi về hàm dựng của lớp mẹ với tên và phiên bản CSDL của ứng dụng:

```
super(context, DATABASE_NAME, null, DATABASE_VERSION);
```

Ngoài ra, trong lớp này ta nạp chồng 2 hàm:

- Hàm `onCreate()` : được gọi để khởi tạo CSDL trong lần đầu tiên chạy ứng dụng. Trong hàm này ta tiến hành thực thi câu lệnh tạo CSDL ở trên (`DATABASE_CREATE`)
- Hàm `onUpgrade()` : được gọi khi ta nâng cấp ứng dụng và thay đổi giá trị của phiên bản CSDL (`DATABASE_VERSION`) ở trên. Trong ví dụ ở trên, khi có thay đổi phiên bản này, ta xóa CSDL cũ và tạo lại cái mới.

Ngoài ra, ta cũng viết thêm các hàm để mở CSDL, tạo mới bản ghi, cập nhật bản ghi, lấy tất cả bản ghi, lấy bản ghi theo id... (xem đoạn mã ở trên).

Sau khi có lớp `DBAdapter` này, việc truy xuất CSDL trở nên tương đối đơn giản, đoạn mã dưới đây minh họa các thao tác thêm, bớt, truy vấn CSDL:

```

DBAdapter db = new DBAdapter(this);

//--- thêm một bản ghi ---
db.open();
long id = db.insertContact("Wei-Meng Lee", "weimenglee@learn2develop.net");
id = db.insertContact("Mary Jackson", "mary@jackson.com");
db.close();

//--- lấy danh sách tất cả bản ghi ---
db.open();
Cursor c = db.getAllContacts();
if (c.moveToFirst())
{
    do {
        DisplayContact(c);
    } while (c.moveToNext());
}
db.close();

//--- lấy một bản ghi theo id ---
db.open();
c = db.getContact(2);
if (c.moveToFirst())
    DisplayContact(c);
else
    Toast.makeText(this, "No contact found", Toast.LENGTH_LONG).show();
db.close();

//--- cập nhật bản ghi ---
db.open();
if (db.updateContact(1, "Wei-Meng Lee", "weimenglee@gmail.com"))
    Toast.makeText(this, "Update successful.", Toast.LENGTH_LONG).show();
else
    Toast.makeText(this, "Update failed.", Toast.LENGTH_LONG).show();
db.close();

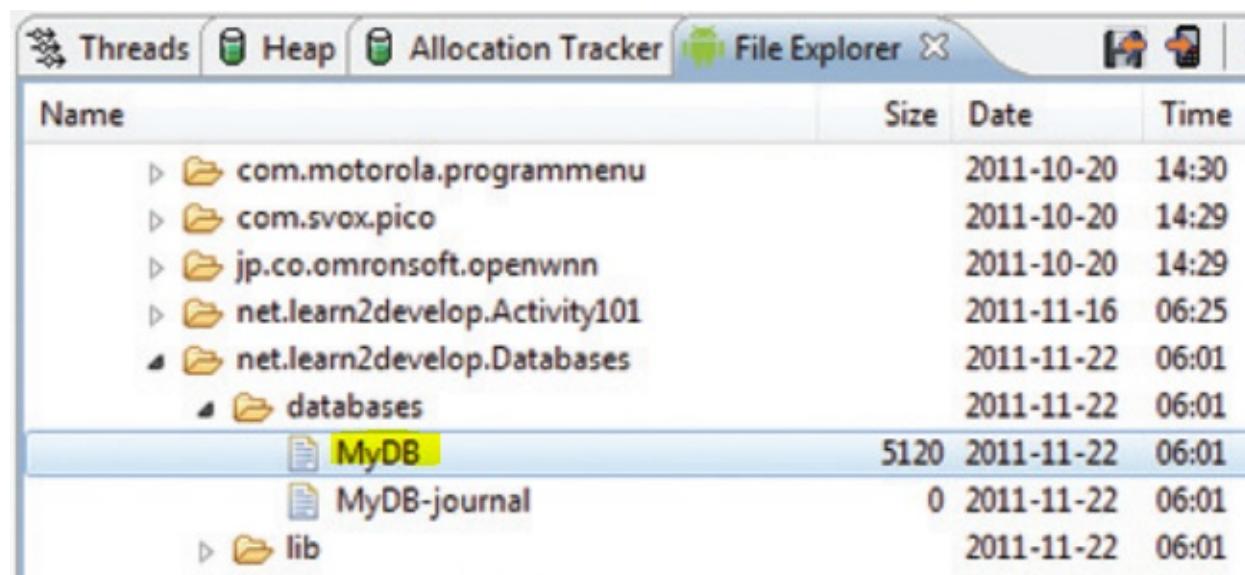
```

```
---- xóa bản ghi ---
db.open();
if (db.deleteContact(1))
    Toast.makeText(this, "Delete successful.", Toast.LENGTH_LONG).show();
else
    Toast.makeText(this, "Delete failed.", Toast.LENGTH_LONG).show();
db.close();
```

Trong đó hàm `DisplayContact(c)` chỉ đơn giản hiển thị lên màn hình nội dung bản ghi dưới dạng `Toast`:

```
public void DisplayContact(Cursor c)
{
    Toast.makeText(this,
        "id: " + c.getString(0) + "\n" +
        "Name: " + c.getString(1) + "\n" +
        "Email: " + c.getString(2),
        Toast.LENGTH_LONG).show();
}
```

Chạy ứng dụng và dùng trình duyệt file của Emulator, ta có thể thấy file CSDL được tạo ra trong thư mục `/data/data/{package-name}/databases/myDB`:



Hình: Vị trí file CSDL trong bộ nhớ của thiết bị

Nếu ta lấy file này về máy tính và đọc nó bằng các phần mềm hỗ trợ CSDL `SQLite` (như `NaviCat`) ta sẽ thấy được nội dung bảng `contacts` bên trong.

Lập trình mạng với Android

Trong các ứng dụng hiện đại, nhu cầu giao tiếp giữa ứng dụng với thế giới bên ngoài (*Internet*) hết sức phổ biến, từ việc lấy và cập nhật nội dung trực tuyến (từ các web site, các web service), tương tác với máy chủ, cho đến lập trình socket để mở kết nối cố định đến ứng dụng trên server.

Trong chương này, ta sẽ tìm hiểu cách thức tải dữ liệu dạng nhị phân cũng như dạng văn bản từ máy chủ web thông qua giao thức *HTTP*, cũng như việc phân tích cú pháp dữ liệu nhận về dạng *XML* hoặc *JSON* để lấy ra thông tin cần thiết.

Việc lập trình socket cần thêm ứng dụng phía server, nằm ngoài phạm vi của giáo trình, nên sẽ không được đề cập ở đây, bạn đọc quan tâm có thể tự tìm hiểu thêm trong các tài liệu khác.

Sử dụng các dịch vụ web services thông qua giao thức HTTP

Cách thức phổ biến nhất để cập nhật dữ liệu trực tuyến là lấy dữ liệu từ trang web trên Internet thông qua giao thức *HTTP*. Sử dụng giao thức này, ta có thể thực hiện rất nhiều việc trao đổi dữ liệu với thế giới bên ngoài, từ việc lấy nội dung trang web, tải dữ liệu nhị phân (file nhị phân, ảnh...), tải dữ liệu dạng văn bản...

Về mặt kỹ thuật, các thao tác này là như nhau cho mọi loại dữ liệu tải về. Vì vậy, trước tiên ta tạo một dự án khung cho các ứng dụng sử dụng tài nguyên mạng theo giao thức *HTTP*.

Trước hết, ta cần yêu cầu quyền truy cập Internet cho ứng dụng một cách tường minh trong file *AndroidManifest.xml* như sau:

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

    <uses-sdk android:minSdkVersion="13" />
    <uses-permission android:name="android.permission.INTERNET"/>

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

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

Sau đó, trong mã nguồn của Activity chính, ta viết thêm một hàm để lấy dữ liệu từ trên mạng về qua giao thức *HTTP*. Kết quả của việc lấy dữ liệu này sẽ cho ta một luồng nhập liệu (*InputStream*) để ta xử lý dữ liệu (việc lấy dữ liệu ra loại gì – nhị phân hay văn bản sẽ do luồng nhập liệu này xử lý).

Mã nguồn của Activity ban đầu sẽ như sau:

```
public class NetworkingActivity extends Activity {

    private InputStream OpenHttpConnection(String urlString)
    throws IOException
    {
        InputStream in = null;
        int response = -1;

        URL url = new URL(urlString);
        URLConnection conn = url.openConnection();

        if (!(conn instanceof HttpURLConnection))
            throw new IOException("Not an HTTP connection");
        try{
            HttpURLConnection httpConn = (HttpURLConnection) conn;
            httpConn.setAllowUserInteraction(false);
            httpConn.setInstanceFollowRedirects(true);
            httpConn.setRequestMethod("GET");
            httpConn.connect();
            response = httpConn.getResponseCode();
            if (response == HttpURLConnection.HTTP_OK) {
                in = httpConn.getInputStream();
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    public void ShowResult() {
        String strOutput = "";
        try {
            BufferedReader rd = new BufferedReader(new InputStreamReader(in));
            String line;
            while ((line = rd.readLine()) != null) {
                strOutput += line;
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
        TextView textView = (TextView) findViewById(R.id.TextView01);
        textView.setText(strOutput);
    }
}
```

```
        }
    }
    catch (Exception ex)
    {
        Log.d("Networking", ex.getLocalizedMessage());
        throw new IOException("Error connecting");
    }
    return in;
}

/** Called when the activity is first created. */
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
}
}
```

Hàm `OpenHttpConnection` ở trên nhận vào đường link đến tài nguyên trên Internet, thực hiện kết nối và trả dữ liệu về vào luồng nhập liệu (`InputStream`). Do việc làm việc với tài nguyên mạng thường xảy ra lỗi (không kết nối được, link không còn tồn tại...), toàn bộ đoạn mã cho việc này nằm trong bộ xử lý nhận ngoại lệ try-catch để bắt các lỗi có thể xảy ra.

Lưu ý là việc kết nối đến máy chủ để lấy tài nguyên có thể kéo dài, tùy thuộc vào kích thước tài nguyên, tốc độ đường truyền, khả năng xử lý của thiết bị... Vì vậy, hàm trên thường được gọi trong thread riêng để tránh việc khóa cứng giao diện người dùng trong quá trình xử lý. Ta sẽ làm việc này trong các ví dụ cụ thể bên dưới với việc tải dữ liệu nhị phân cũng như tải dữ liệu văn bản.

Tải dữ liệu nhị phân thông qua HTTP

Trong phần này, ta sẽ minh họa một trường hợp tải dữ liệu nhị phân từ máy chủ web thông qua giao thức *HTTP*. Ta sẽ dùng ứng dụng khung về sử dụng tài nguyên qua *HTTP* ở trên để tải một ảnh (ảnh là dữ liệu nhị phân) từ trên mạng về và hiển thị bên trong Activity. Trước tiên, ta thêm một đối tượng *ImageView* vào file layout của Activity để hiển thị ảnh khi được tải về:

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

    <ImageView
        android:id="@+id/img"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center" />
</LinearLayout>
```

Như đã nói ở trên, việc tải tài nguyên từ mạng phải nằm trong thread riêng, khác với thread vẽ giao diện ứng dụng (UI thread). Ở đây ta dùng *AsyncTask* cho việc này:

```
private Bitmap DownloadImage(String URL)
{
    Bitmap bitmap = null;
    InputStream in = null;
    try {
        in = OpenHttpConnection(URL);
        bitmap = BitmapFactory.decodeStream(in);
        in.close();
    } catch (IOException e1) {
        Log.d("NetworkingActivity", e1.getLocalizedMessage());
    }
    return bitmap;
}

private class DownloadImageTask extends AsyncTask<String, Void, Bitmap> {
    protected Bitmap doInBackground(String... urls) {
        return DownloadImage(urls[0]);
    }

    protected void onPostExecute(Bitmap result) {
        ImageView img = (ImageView) findViewById(R.id.img);
        img.setImageBitmap(result);
    }
}
```

Hàm *DownloadImage* thực hiện tải ảnh từ trên mạng về và lưu vào một đối tượng *Bitmap* (đối tượng này sẵn sàng để đưa vào *ImageView*). Hàm này sử dụng hàm *OpenHttpConnection* đã viết ở trên để lấy một luồng nhập liệu chứa dữ liệu ảnh này, sau đó dùng hàm tĩnh *decodeStream* của lớp *BitmapFactory* để giải mã luồng nhập liệu này và lưu vào đối tượng *Bitmap*.

Tuy nhiên, hàm *DownloadImage* này cũng sẽ chiếm nhiều thời gian (bằng thời gian tải ảnh về của hàm *openHttpConnection* và thời gian giải mã ảnh). Vì vậy ta cần gọi hàm này bên trong một thread khác. Trong ví dụ trên ta dùng *AsyncTask* cho việc này. Để sử dụng *AsyncTask*, ta cần nạp chõng tối thiểu 2 hàm:

- *doInBackground* - hàm này được thực hiện trong thread riêng. Khi kết thúc hàm này, nó sẽ tự động gọi hàm *onPostExecute* trong UI thread. Đây là nơi ta sẽ thực hiện các thao tác tốn thời gian.
- *onPostExecute* hàm này được gọi trong UI thread, sau khi công việc ngầm ở hàm trên đã kết thúc. Đây là nơi ta cập

nhật lại UI từ dữ liệu được tải về.

Trong ví dụ trên, ta tiến hành tải ảnh trong thread riêng (trong hàm `doInBackground`) và tiến hành hiển thị ảnh mới được tải về trong hàm `onPostExecute`.

Cuối cùng, để tiến hành tải ảnh, ta thêm dòng mã gọi đến `AsyncTask` này trong hàm `onCreate` của Activity:

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

    new DownloadImageTask().execute("http://www.mayoff.com/5-01cablecarDCP01934.jpg");
}
```

Chạy ứng dụng trên điện thoại hoặc Android Emulator, ta thấy ảnh từ mạng sẽ được hiển thị trên màn hình (có thể phải chờ một lúc nếu kết nối mạng không tốt) như hình minh họa bên dưới:



Hình: Hình ảnh được tải từ Internet thông qua HTTP

Tải dữ liệu dạng text thông qua HTTP

Tương tự như việc tải dữ liệu nhị phân ở trên, trong nhiều trường hợp ta cần tải dữ liệu dạng văn bản từ mạng (lấy dữ liệu từ các web service chẳng hạn). Ta cũng sẽ thực hiện thao tác này trong thread riêng, sử dụng AsyncTask và trả về dữ liệu dạng String trước khi hiển thị lên màn hình (dạng Toast).

Đoạn mã tải và hiển dữ liệu văn bản trong thread riêng như sau:

```

private String DownloadText(String URL)
{
    int BUFFER_SIZE = 2000;
    InputStream in = null;
    try {
        in = OpenHttpConnection(URL);
    } catch (IOException e) {
        Log.d("NetworkingActivity", e.getLocalizedMessage());
        return "";
    }

    InputStreamReader isr = new InputStreamReader(in);
    int charRead;
    String str = "";
    char[] inputBuffer = new char[BUFFER_SIZE];
    try {
        while ((charRead = isr.read(inputBuffer)) > 0) {
            //---convert the chars to a String---
            String readString =
                String.valueOf(inputBuffer, 0, charRead);
            str += readString;
            inputBuffer = new char[BUFFER_SIZE];
        }
        in.close();
    } catch (IOException e) {
        Log.d("NetworkingActivity", e.getLocalizedMessage());
        return "";
    }
    return str;
}

private class DownloadTextTask extends AsyncTask<String, Void, String> {
    protected String doInBackground(String... urls) {
        return DownloadText(urls[0]);
    }

    @Override
    protected void onPostExecute(String result) {
        Toast.makeText(getApplicationContext(), result, Toast.LENGTH_LONG).show();
    }
}

```

Đoạn mã trên chỉ khác với việc tải ảnh trong phần trước ở đoạn đọc dữ liệu từ luồng nhập liệu trả về. Trong trường hợp dữ liệu văn bản, ta dùng lớp `InputStreamReader` và hàm `read` của nó để đọc lần lượt dữ liệu dạng char ra, sau đó gắn thêm dần vào string kết quả. Cuối cùng, ta chỉ cần gọi `AsyncTask` mới tạo ra này trong hàm `onCreate` của Activity để xem kết quả:

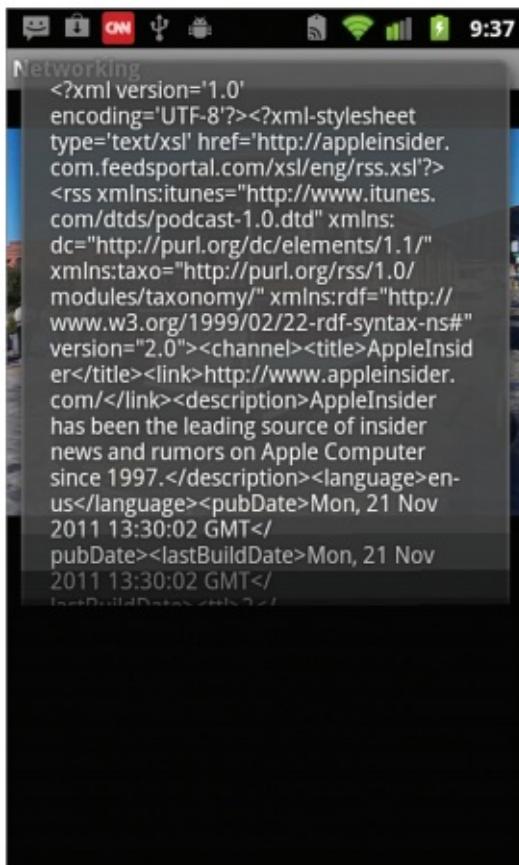
```

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

    //---download text---
    new DownloadTextTask().execute(
        "http://iheartquotes.com/api/v1/random?max_characters=256&max_lines=10");
}

```

Chạy ứng dụng, ta sẽ thấy dữ liệu từ web service trên được in ra màn hình:



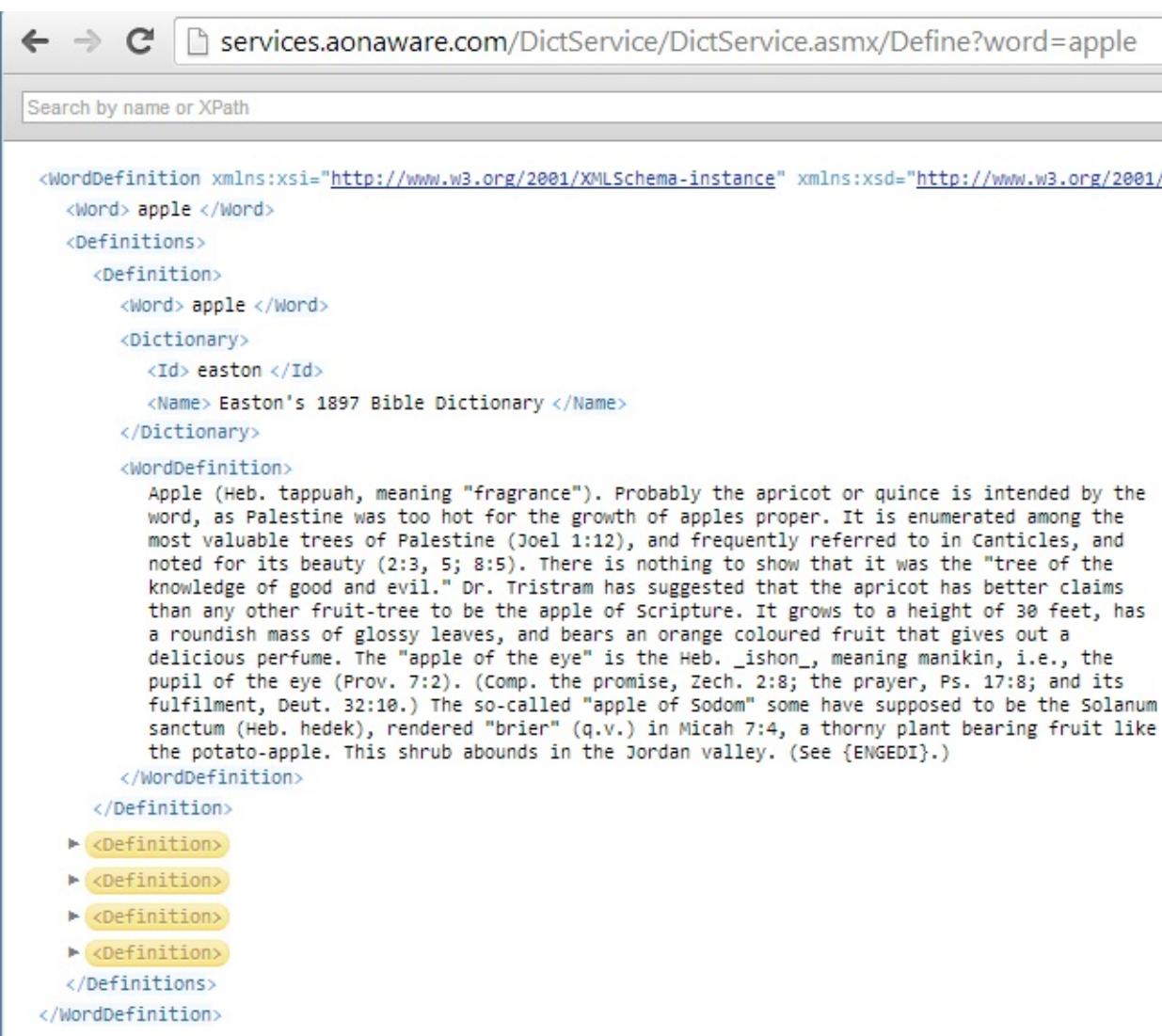
Hình: Dữ liệu dạng văn bản (XML) được tải từ Internet qua HTTP

Web service với dữ liệu XML

Trong phần trên ta đã tiến hành lấy dữ liệu văn bản từ web service thông qua giao thức *HTTP*. Tuy nhiên trong ví dụ trước ta cũng thấy dữ liệu trả về có thể ở dạng *XML*. Đây là định dạng tương đối phổ biến của các web service hiện nay (bên cạnh *JSON*), và để có thể sử dụng được dữ liệu này trong ứng dụng, ta cần phân tích cú pháp (*parsing*) của dữ liệu *XML* này. Trong phần này ta sẽ xem xét cách thức phân tích tài liệu *XML* trong *Android*.

Trong ví dụ này, ta sẽ thực hiện tra nghĩa của một từ tiếng anh trên từ điển trực tuyến của Aonaware qua web service. Để thực hiện tra từ, ta gọi *API* như sau: <http://services.aonaware.com/DictService/DictService.asmx/Define?word={từ-cần-tra}>. Ví dụ để tra nghĩa của từ “apple”, ta cần gọi API như sau:

<http://services.aonaware.com/DictService/DictService.asmx/Define?word=apple> Nếu bạn mở link này vào trình duyệt web, ta sẽ thấy nội dung web service này trả về dưới dạng *XML* như sau:



The screenshot shows a web browser window with the URL <http://services.aonaware.com/DictService/DictService.asmx/Define?word=apple> in the address bar. Below the address bar is a search bar labeled "Search by name or XPath". The main content area displays the XML response:

```

<WordDefinition xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/
<Word> apple </Word>
<Definitions>
  <Definition>
    <Word> apple </Word>
    <Dictionary>
      <Id> easton </Id>
      <Name> Easton's 1897 Bible Dictionary </Name>
    </Dictionary>
    <WordDefinition>
      Apple (Heb. tappuah, meaning "fragrance"). Probably the apricot or quince is intended by the
      word, as Palestine was too hot for the growth of apples proper. It is enumerated among the
      most valuable trees of Palestine (Joel 1:12), and frequently referred to in Canticles, and
      noted for its beauty (2:3, 5; 8:5). There is nothing to show that it was the "tree of the
      knowledge of good and evil." Dr. Tristram has suggested that the apricot has better claims
      than any other fruit-tree to be the apple of Scripture. It grows to a height of 30 feet, has
      a roundish mass of glossy leaves, and bears an orange coloured fruit that gives out a
      delicious perfume. The "apple of the eye" is the Heb. _ishon_, meaning manikin, i.e., the
      pupil of the eye (Prov. 7:2). (Comp. the promise, Zech. 2:8; the prayer, Ps. 17:8; and its
      fulfilment, Deut. 32:10.) The so-called "apple of Sodom" some have supposed to be the Solanum
      sanctum (Heb. hedek), rendered "brier" (q.v.) in Micah 7:4, a thorny plant bearing fruit like
      the potato-apple. This shrub abounds in the Jordan valley. (See {ENGEDI}.)
    </WordDefinition>
  </Definition>
  ▶ <Definition>
  ▶ <Definition>
  ▶ <Definition>
  ▶ <Definition>
</Definitions>
</WordDefinition>

```

Nhiệm vụ của chúng ta là viết đoạn mã phân tích tài liệu *XML* trên để lấy ra phần nghĩa của từ (trong thẻ *WordDefinition*) và hiển thị lên màn hình.

Để thực hiện việc này, ta cũng cần khai báo một *AsyncTask* để tránh khóa cứng *UI thread*:

```

private String WordDefinition(String word) {
    InputStream in = null;
    String strDefinition = "";

```

```

try {
    in = OpenHttpConnection("http://services.aonaware.com/DictService/DictService.asmx/Define?word="
                           + word);
    Document doc = null;
    DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();
    DocumentBuilder db;
    try {
        db = dbf.newDocumentBuilder();
        doc = db.parse(in);
    } catch (ParserConfigurationException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    } catch (Exception e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
    doc.getDocumentElement().normalize();

    // ---retrieve all the <Definition> elements---
    NodeList definitionElements = doc
        .getElementsByTagName("Definition");

    // ---iterate through each <Definition> elements---
    for (int i = 0; i < definitionElements.getLength(); i++) {
        Node itemNode = definitionElements.item(i);
        if (itemNode.getNodeType() == Node.ELEMENT_NODE) {
            // ---convert the Definition node into an Element---
            Element definitionElement = (Element) itemNode;

            // ---get all the <WordDefinition> elements under
            // the <Definition> element---
            NodeList wordDefinitionElements = (definitionElement)
                .getElementsByTagName("WordDefinition");

            strDefinition = "";
            // ---iterate through each <WordDefinition> elements---
            for (int j = 0; j < wordDefinitionElements.getLength(); j++) {
                // ---convert a <WordDefinition> node into an Element---
                Element wordDefinitionElement = (Element) wordDefinitionElements
                    .item(j);

                // ---get all the child nodes under the
                // <WordDefinition> element---
                NodeList textNodes = ((Node) wordDefinitionElement)
                    .getChildNodes();

                strDefinition += ((Node) textNodes.item(0))
                    .getNodeValue() + ". \n";
            }
        }
    }
} catch (IOException e1) {
    Log.d("NetworkingActivity", e1.getLocalizedMessage());
}
// ---return the definitions of the word---
return strDefinition;
}

private class AccessWebServiceTask extends AsyncTask<String, Void, String> {
    protected String doInBackground(String... urls) {
        return WordDefinition(urls[0]);
    }

    protected void onPostExecute(String result) {
        Toast.makeText(getApplicationContext(), result, Toast.LENGTH_LONG).show();
    }
}

```

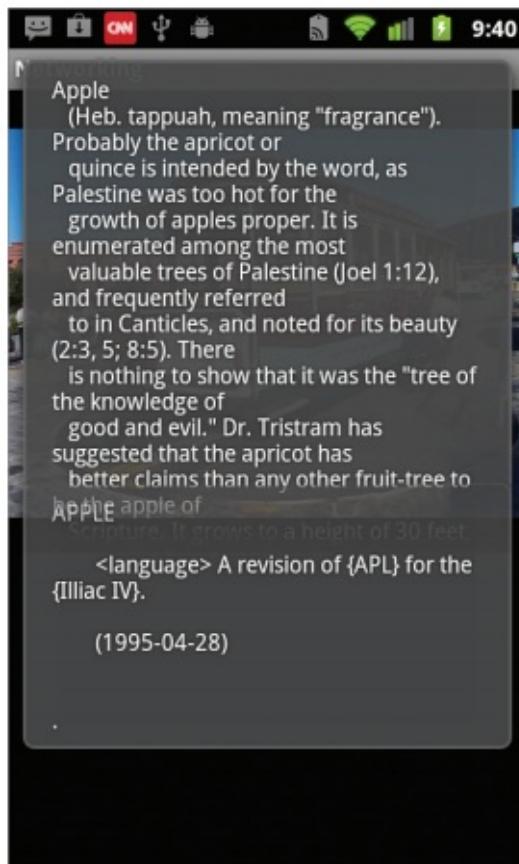
Trong đó hàm `WordDefinition` sẽ tải nội dung của webservice và tiến hành phân tích cú pháp tài liệu XML trả về. Công việc này tốn thời gian nên ta cần làm trong thread riêng, do đó ta cần viết thêm lớp `AccessWebServiceTask` như ở trên.

Việc cuối cùng là gọi `AsyncTask` này trong hàm `onCreate` của Activity:

```
/** Called when the activity is first created. */
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);

    // ---access a Web Service using GET---
    new AccessWebServiceTask().execute("apple");
}
```

Ứng dụng khi chạy sẽ có dạng như minh họa trong hình bên dưới.



Hình: Dữ liệu XML đã được phân tích

Ta sẽ đi chi tiết hơn một chút về hàm `wordDefinition` ở trên. Trước hết, ta vẫn sử dụng hàm `OpenHttpConnection` đã viết ở trên để lấy thông tin từ mạng và đưa vào luồng nhập liệu:

```
in = OpenHttpConnection("http://services.aonaware.com/DictService/DictService.asmx/Define?word=" + word);
```

Sau đó ta dùng lớp `javax.xml.parsers.DocumentBuilder` để phân tích tài liệu XML trong luồng nhập liệu này thành cây đối tượng tài liệu (*DOM – Document object model*):

```
Document doc = null;
DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();
DocumentBuilder db;

db = dbf.newDocumentBuilder();
doc = db.parse(in);
doc.getDocumentElement().normalize();
```

Lưu ý: trong đoạn code trên ta bỏ qua phần bắt ngoại lệ `try-catch` cho dễ quan sát. Sau khi `parsing` như trên, ta thu được một cây đối tượng trong trường “`doc`” (đối tượng của lớp `org.w3c.dom.Document`). Ta có thể duyệt cây tài liệu này để lấy ra các trường mong muốn. Cụ thể: Để lấy danh sách các node “`Definitions`”:

```
NodeList definitionElements = doc.getElementsByTagName("Definition");
```

Sau đó duyệt từng phần tử trong danh sách node trên để lấy ra `WordDefinition` của từng node:

```
NodeList wordDefinitionElements = (definitionElement).getElementsByTagName("WordDefinition");
```

Việc duyệt cây *DOM* để trích xuất ra dữ liệu cần thiết là công việc khá điển hình và phổ biến. Bạn đọc quan tâm có thể tham khảo chi tiết hơn tại các tài liệu khác.

Web service với dữ liệu JSON

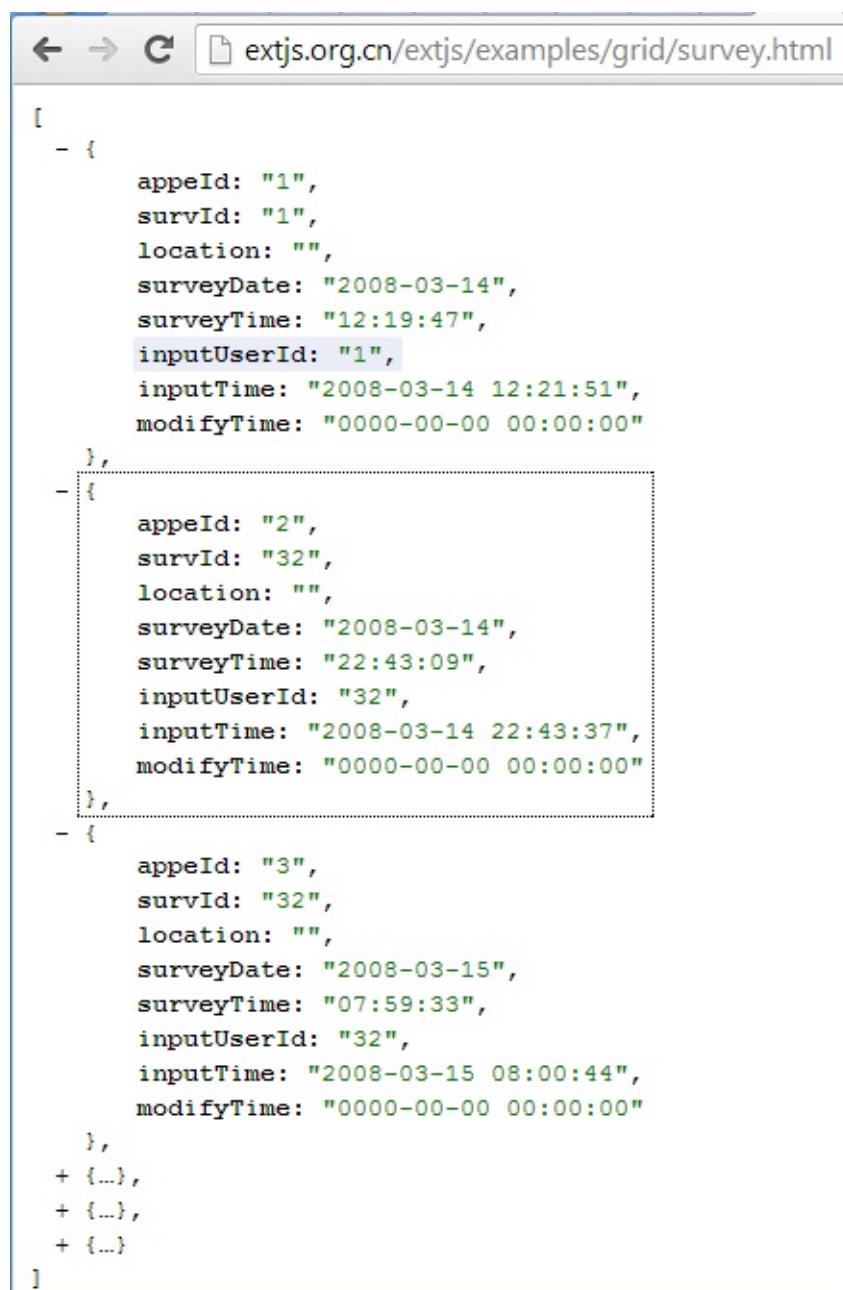
Ngoài XML, một định dạng dữ liệu văn bản được sử dụng rất phổ biến hiện nay trong các web service là JSON (*JavaScript Object Notation*). So với XML, định dạng JSON có một số ưu điểm:

- Json có độ nén dữ liệu tốt hơn: cùng một dữ liệu, XML tốn nhiều dung lượng hơn để đóng gói, do các thẻ (tag) trong XML có độ dài nhất định. Dung lượng lớn ảnh hưởng xấu đến tốc độ truyền tải cũng như khả năng lưu trữ tài liệu.
- Xử lý (phân tích) tài liệu XML tốn kém hơn so với JSON cả về bộ nhớ lẫn tài nguyên CPU.

Ví dụ về một tài liệu JSON ta sẽ thử phân tích trong phần tiếp theo có thể lấy được qua đường link sau:

<http://extjs.org.cn/extjs/examples/grid/survey.html>

Nếu mở link này bằng trình duyệt, ta sẽ quan sát được nội dung như hình bên dưới.



The screenshot shows a browser window with the URL extjs.org.cn/extjs/examples/grid/survey.html. The page displays a JSON array of survey records. One record is highlighted with a dotted rectangle, indicating it is being analyzed or selected.

```
[
  - {
    appId: "1",
    survId: "1",
    location: "",
    surveyDate: "2008-03-14",
    surveyTime: "12:19:47",
    inputUserId: "1",
    inputTime: "2008-03-14 12:21:51",
    modifyTime: "0000-00-00 00:00:00"
  },
  - {
    appId: "2",
    survId: "32",
    location: "",
    surveyDate: "2008-03-14",
    surveyTime: "22:43:09",
    inputUserId: "32",
    inputTime: "2008-03-14 22:43:37",
    modifyTime: "0000-00-00 00:00:00"
  },
  - {
    appId: "3",
    survId: "32",
    location: "",
    surveyDate: "2008-03-15",
    surveyTime: "07:59:33",
    inputUserId: "32",
    inputTime: "2008-03-15 08:00:44",
    modifyTime: "0000-00-00 00:00:00"
  },
  + {...},
  + {...},
  + {...}
]
```

Ví dụ dưới đây sẽ phân tích tài liệu JSON ở trên và in ra màn hình `appId` và `inputTime` cho từng đối tượng trong danh

sách phân tích được.

Toàn bộ mã nguồn của Activity cho ví dụ này như sau:

```

package vn.edu.humg.android_course.JSON;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;

import org.apache.http.HttpEntity;
import org.apache.http.HttpResponse;
import org.apache.http.StatusLine;
import org.apache.http.client.ClientProtocolException;
import org.apache.http.client.HttpClient;
import org.apache.http.client.methods.HttpGet;
import org.apache.http.impl.client.DefaultHttpClient;
import org.json.JSONArray;
import org.json.JSONObject;

import android.app.Activity;
import android.os.AsyncTask;
import android.os.Bundle;
import android.util.Log;
import android.widget.Toast;

public class JSONActivity extends Activity {

    public String readJSONFeed(String URL) {
        StringBuilder stringBuilder = new StringBuilder();
        HttpClient client = new DefaultHttpClient();
        HttpGet httpGet = new HttpGet(URL);
        try {
            HttpResponse response = client.execute(httpGet);
            StatusLine statusLine = response.getStatusLine();
            int statusCode = statusLine.getStatusCode();
            if (statusCode == 200) {
                HttpEntity entity = response.getEntity();
                InputStream content = entity.getContent();
                BufferedReader reader = new BufferedReader(
                    new InputStreamReader(content));
                String line;
                while ((line = reader.readLine()) != null) {
                    stringBuilder.append(line);
                }
            } else {
                Log.e("JSON", "Failed to download file");
            }
        } catch (ClientProtocolException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }
        return stringBuilder.toString();
    }

    private class ReadJSONFeedTask extends AsyncTask<String, Void, String> {
        protected String doInBackground(String... urls) {
            return readJSONFeed(urls[0]);
        }

        protected void onPostExecute(String result) {
            try {
                JSONArray jsonArray = new JSONArray(result);
                Log.i("JSON", "Number of surveys in feed: " +
                    jsonArray.length());

                //---print out the content of the json feed---
                for (int i = 0; i < jsonArray.length(); i++) {
                    JSONObject jsonObject = jsonArray.getJSONObject(i);
                    Toast.makeText(getApplicationContext(), jsonObject.getString("appId") +
                        " - " + jsonObject.getString("inputTime"),
                        Toast.LENGTH_SHORT).show();
                }
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
    }
}

```

```

        }
    } catch (Exception e) {
        e.printStackTrace();
    }
}

/** Called when the activity is first created. */
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);

    new ReadJSONFeedTask().execute("http://extjs.org.cn/extjs/examples/grid/survey.html");
}
}

```

Lưu ý: trong mã nguồn trên, ta dùng lớp `org.apache.http.client.HttpClient` để lấy nội dung web service thông qua *HTTP* thay cho `java.net.HttpURLConnection`. Bạn đọc quan tâm có thể tự tìm hiểu về lớp này ở các tài liệu khác, ở đây ta chỉ quan tâm đến đoạn mã nguồn phân tích tài liệu JSON.

Để phân tích tài liệu JSON và đưa vào một mảng các `JSONObject`, ta chỉ cần gọi một lệnh đơn giản: `JSONArray jsonArray = new JSONArray(result);`

Với `result` là dữ liệu văn bản thô (dạng `String`) lấy được từ *webservice*. Sau đó tiến hành duyệt lần lượt các `JSONObject` thu được bằng hàm `getJSONObject(i)` và lấy nội dung của từng trường dữ liệu trong mỗi đối tượng JSON bằng phương thức `getString` của lớp `JSONObject`.

```

for (int i = 0; i < jsonArray.length(); i++) {
    JSONObject jsonObject = jsonArray.getJSONObject(i);
    Toast.makeText(getApplicationContext(), jsonObject.getString("appleId") +
        " - " + jsonObject.getString("inputTime"),
        Toast.LENGTH_SHORT).show();
}

```

Google Play Store và việc phân phối ứng dụng

Như vậy chúng ta đã học được rất nhiều thứ xuyên suốt giáo trình giúp chúng ta có thể phát triển một ứng dụng Android chất lượng. Tuy nhiên, để ứng dụng có thể được cài đặt lên thiết bị của người dùng, ta cần đóng gói và phân phối. Trong chương này, chúng ta sẽ tìm hiểu cách thức đóng gói ứng dụng Android và phân phối đến máy người dùng qua các cách khác nhau, cũng như tìm hiểu cách phân phối ứng dụng Android lên chợ ứng dụng chính hãng của Google là *Google Play Store*.

Chuẩn bị ứng dụng trước khi phân phối

Google đưa ra quy trình tương đối đơn giản giúp người dùng đóng gói và đưa ứng dụng lên *Google Play Store* để phân phối đến người dùng. Các bước như sau:

- Xuất ứng dụng ra file .apk (*Android Package*)
- Tạo chứng thực chữ ký số và tiến hành ký lên ứng dụng (file apk) trên theo chứng thực mới được tạo ra
- Xuất bản ứng dụng đã được ký này. Có nhiều cách xuất bản như: cài trực tiếp lên thiết bị, đưa lên web site, hoặc phân phối lên các chợ ứng dụng (cả chính hãng lẫn không chính hãng).

Đánh số phiên bản phần mềm

Phiên bản phần mềm được đánh số trong file *AndroidManifest.xml*, dưới hai thuộc tính là `android:versionCode` và `android:versionName`:

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

    <uses-sdk android:minSdkVersion="13" />
    <uses-permission android:name="android.permission.INTERNET"/>

    <application
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name" >
        ...
    </application>
</manifest>
```

Trong đó `versionCode` là số hiệu phiên bản, có kiểu số nguyên, dùng cho hệ thống để phân biệt các phiên bản của ứng dụng được cài đặt. Mỗi khi bạn nâng cấp phiên bản của ứng dụng, bạn cần thay đổi (tăng) số này trước khi phân phối. Còn tham số `versionName` là tên của phiên bản. Tham số này không được dùng bởi hệ thống, mà chỉ là tên phiên bản người dùng sẽ nhìn thấy khi xuất bản lên các chợ ứng dụng. `versionName` có kiểu chuỗi và có thể đặt tùy ý, tuy nhiên định dạng thường được dùng là `<major>. <minor>. <point>`. Ở đó `<major>` là số hiệu phiên bản chính, `<minor>` là phiên bản phụ, `<point>` là số hiệu cập nhật nhỏ trong phiên bản phụ, ví dụ “1.0.1”, “2.1.0”...

Trong nhiều trường hợp, ta cần lấy số hiệu phiên bản ứng dụng hiện tại lúc chạy chương trình. Khi đó, ta dùng lớp `PackageManager` để lấy thông tin như sau:

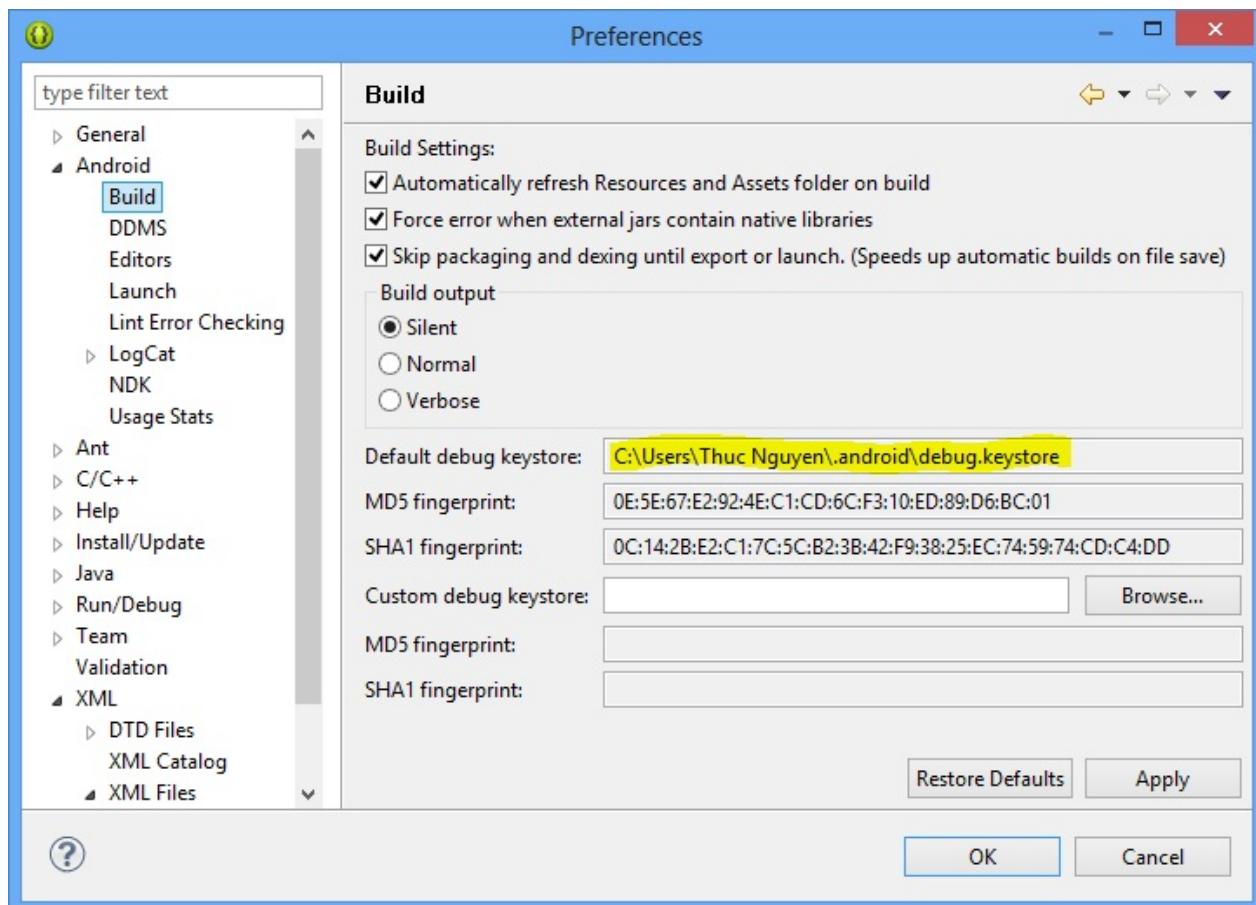
```
PackageManager pm = getPackageManager();
try {
    //---get the package info---
    PackageInfo pi =
        pm.getPackageInfo("vn.edu.humg.android_course.LBS", 0);
    //---display the versioncode---
    Toast.makeText(getApplicationContext(),
        "VersionCode: " + Integer.toString(pi.versionCode),
        Toast.LENGTH_SHORT).show();
} catch (NameNotFoundException e) {
    e.printStackTrace();
}
```

Ngoài ra, để ứng dụng có thể được phân phối trên chợ ứng dụng, bạn cần chỉ ra icon và tiêu đề của ứng dụng để hiển thị trong danh sách ứng dụng. Để làm điều này, ta cần đặt giá trị cho tham số `android:icon` và `android:label` của thẻ `<application>` (xem ví dụ trên).

Chứng thực số cho ứng dụng Android

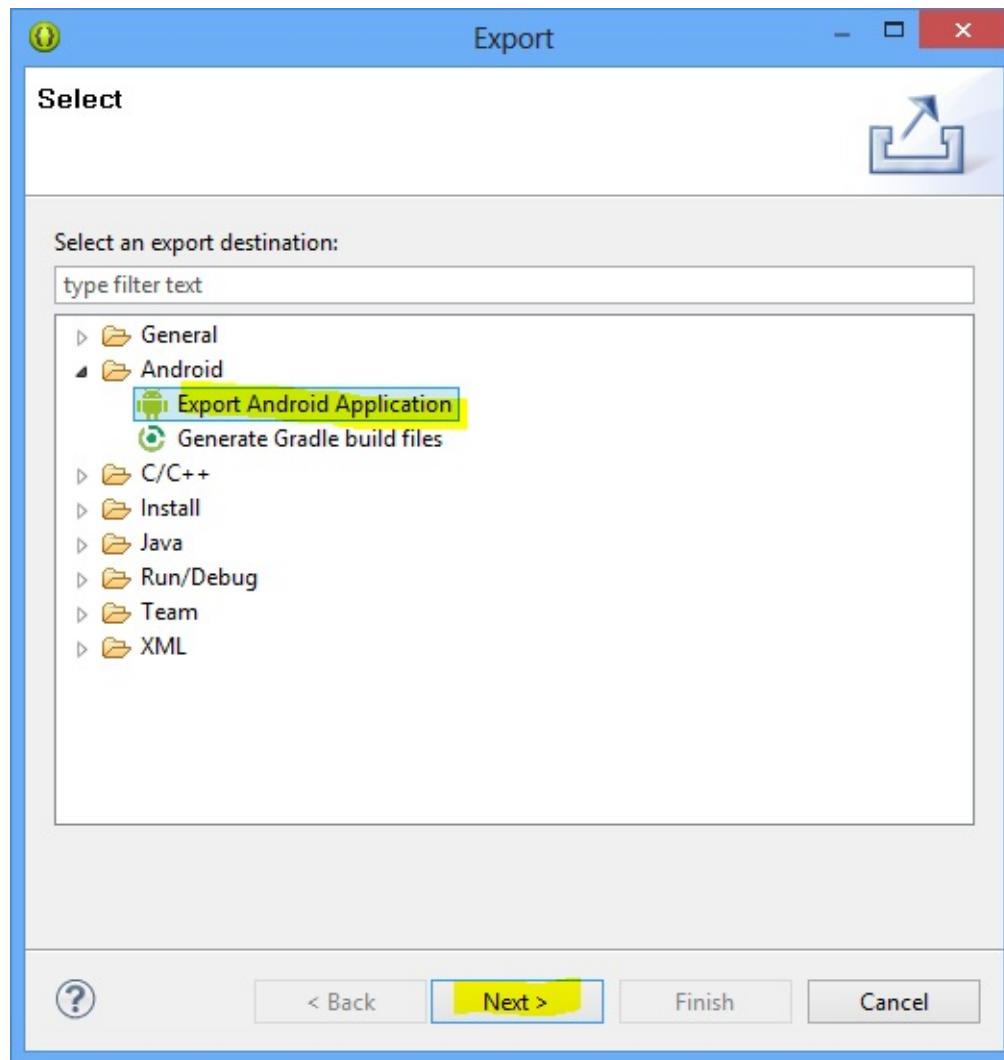
Tất cả ứng dụng Android đều phải được ký theo một chứng thực số trước khi có thể cài đặt lên thiết bị hoặc *Android Emulator*. Không như một số hệ nền khác yêu cầu bạn phải mua chứng chỉ số từ một hãng chuyên cung cấp chứng thực, hệ thống Android cho phép chúng ta tự sinh ra chứng thực số để ký ứng dụng. *Eclipse* và *ADT* cũng cung cấp sẵn công cụ giúp ta tạo ra chứng thực này một cách rất trực quan và dễ dàng.

Trong quá trình phát triển ứng dụng, bạn vẫn có thể chạy ứng dụng của mình trên thiết bị hoặc trình giả lập. Đó là do *Eclipse+ADT* đã ký sẵn ứng dụng của bạn theo một chứng thực mặc định trước khi chuyển ứng dụng lên thiết bị. Chứng thực mặc định này chứa trong file *debug.keystore*:



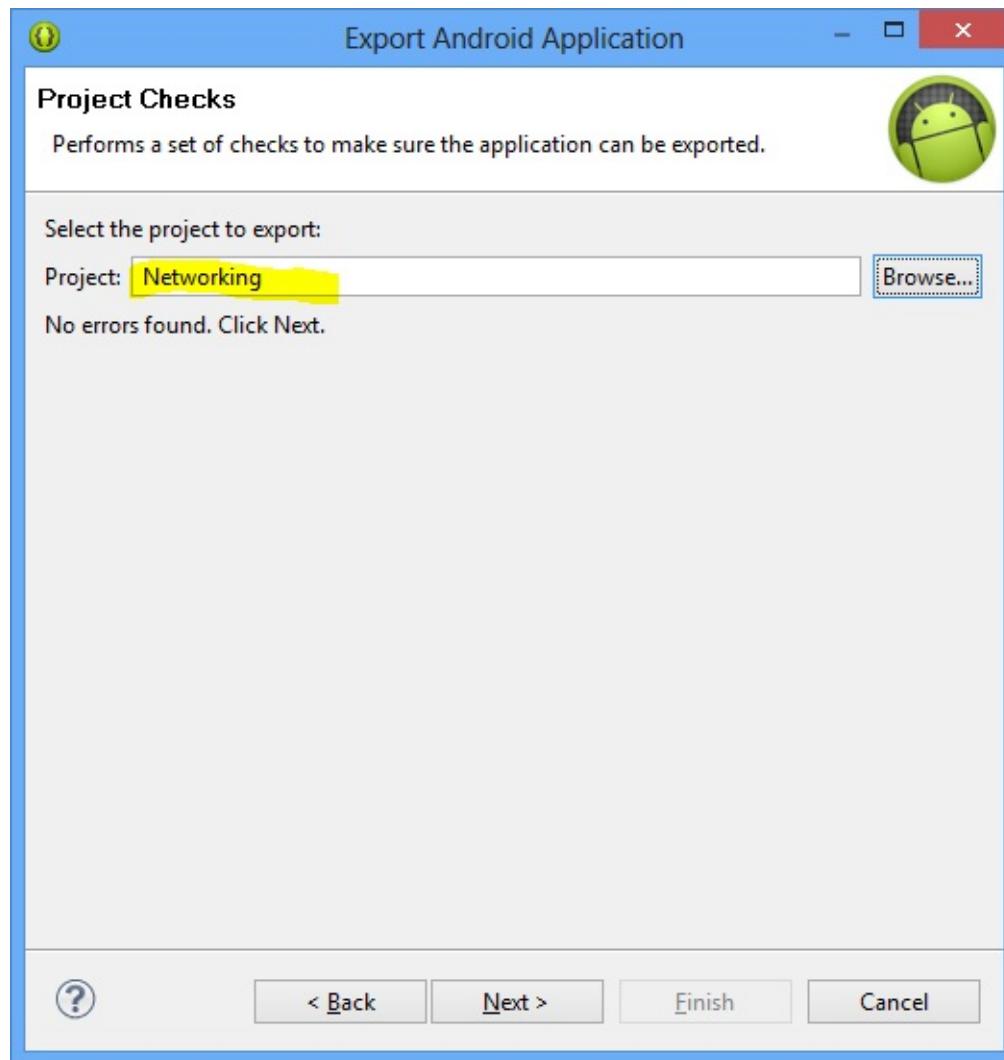
Tuy nhiên, chứng thực này chỉ dùng trong quá trình phát triển. Trước khi bạn xuất bản ứng dụng, bạn cần ký ứng dụng theo một chứng thực khác. Phần dưới đây mô tả quá trình sinh chứng thực số và đóng gói ứng dụng theo chứng thực đó.

Để xuất bản ứng dụng, từ trình đơn File của *Eclipse*, chọn *Export...*, trong cửa sổ mở ra, chọn *Export Android Application*, sau đó bấm *Next*:



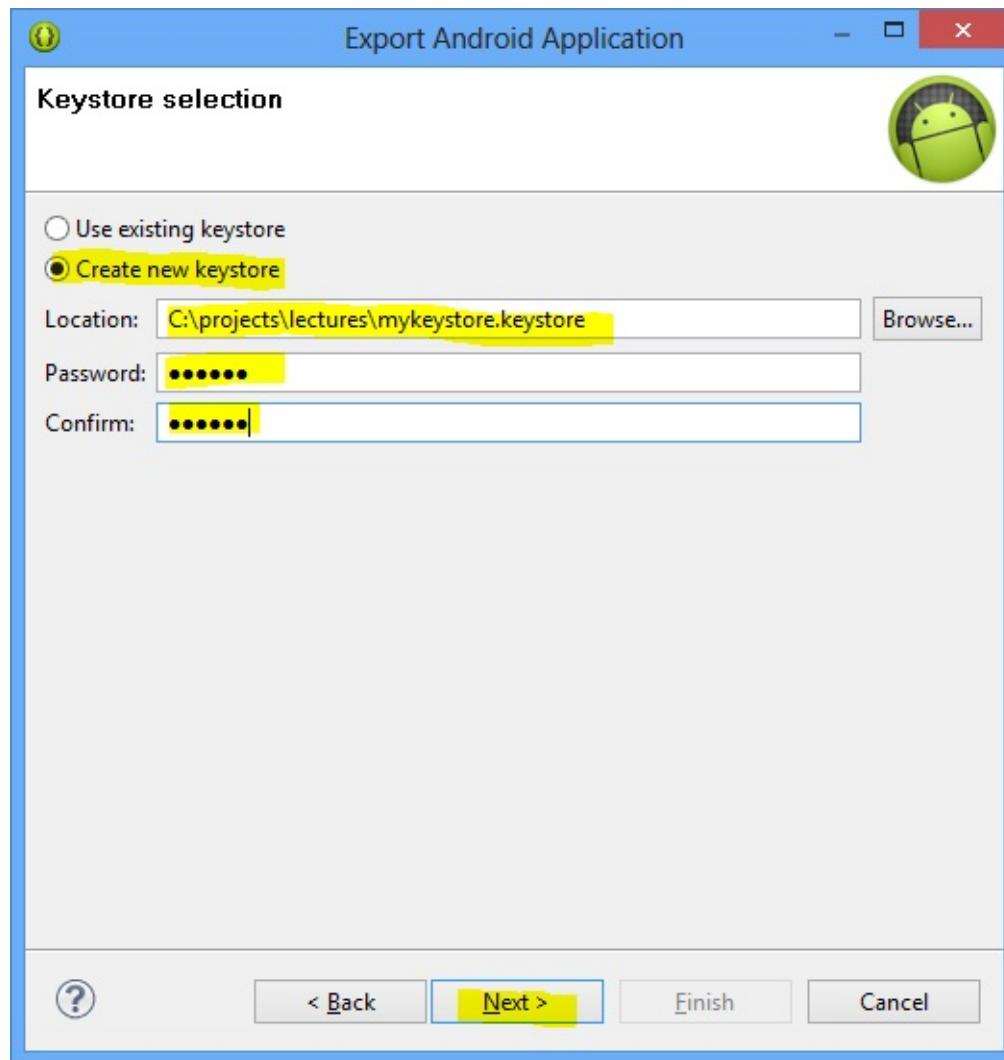
Hình: Xuất bản ứng dụng

Trong màn hình tiếp theo, chọn tên dự án muốn xuất bản:



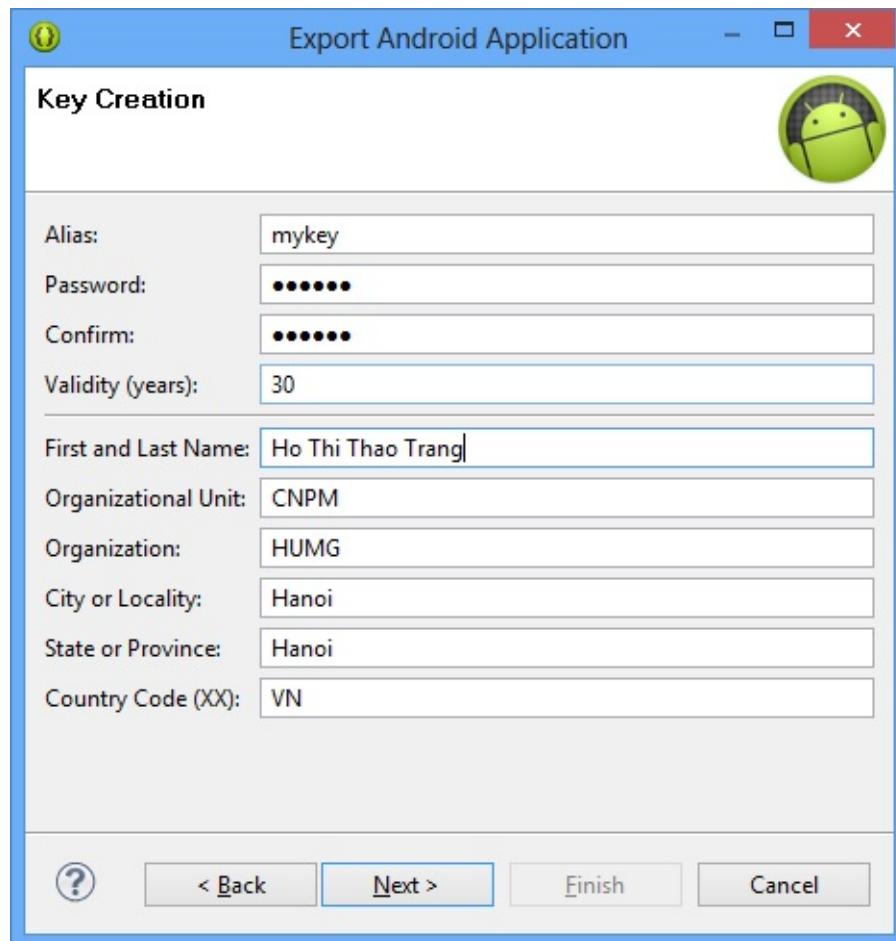
Hình: Chọn dự án cần xuất bản

Tiếp đó ta cần chỉ ra vị trí của file chứa chứng thực số cần dùng để ký ứng dụng và nhập vào mật khẩu của chứng thực này (mật khẩu này ta đặt lúc tạo chứng thực số). Trong trường hợp chưa có chứng thực số, ta chọn “Create new keystore” để tạo mới:



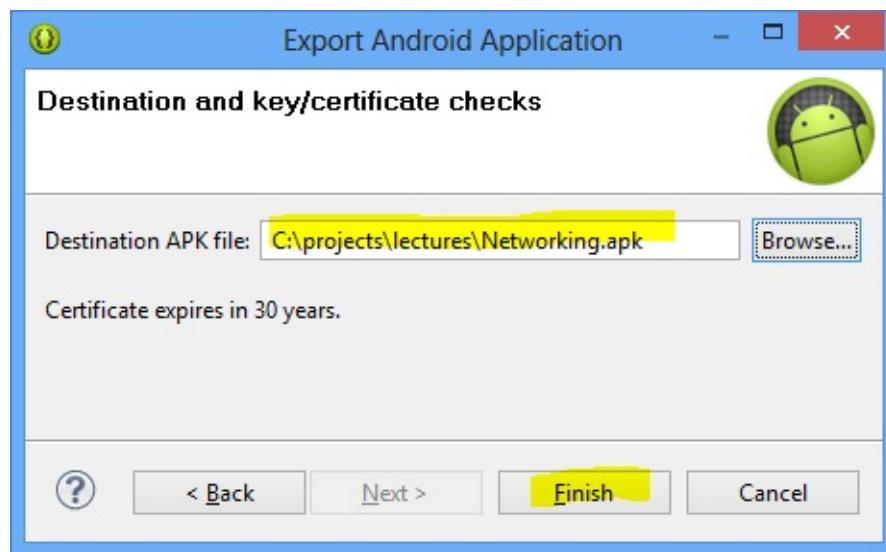
Hình: Cấu hình chữ ký số dùng cho việc xuất bản ứng dụng

Trong trường hợp tạo mới *keystore*, ta cần nhập thông tin để tạo ra một key (dùng để ký ứng dụng), ví dụ ta điền thông tin như bên dưới:



Hình: Thông tin tạo chữ ký số mới

Cuối cùng ta nhập tên file apk cần xuất ra và bấm *Finish*:



Khi quá trình kết thúc, ta sẽ thu được ứng dụng dưới dạng file apk đã đóng gói, sẵn sàng để phân phối.

Phân phối ứng dụng

Sau khi đã đóng gói và ký theo chứng thực số cần thiết, file apk đã sẵn sàng để được phân phối lên các chợ ứng dụng. Có nhiều cách để có thể phân phối ứng dụng đến người dùng như:

- Cài đặt trực tiếp lên thiết bị đầu cuối thông qua công cụ `adb.exe`
- Đặt file cài đặt (.apk) lên web server và phân phối đường link tải xuống cho người dùng
- Phân phối ứng dụng trên các chợ ứng dụng, điển hình là *Google Play Store*

Sử dụng công cụ adb

Công cụ adb có thể được sử dụng để cài trực tiếp ứng dụng lên thiết bị đầu cuối. Để dùng được công cụ này, ta cần:

- Kết nối thiết bị với máy tính thông qua cổng USB
- Cài đặt driver cho thiết bị (nếu chưa có)
- Bật tùy chọn “USB debugging” trong “Developer options” của mục cấu hình (*Settings*) trên thiết bị Android.

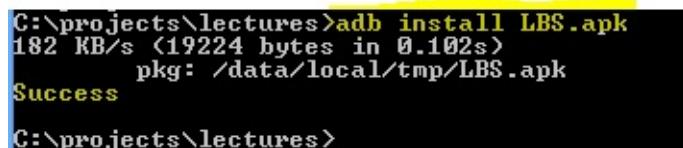
Để kiểm tra thiết bị đã được kết nối thành công với máy tính chưa, ta có thể gõ lệnh “`adb devices`” từ màn hình *console* (*cmd.exe*):



```
C:\>adb devices
List of devices attached
05863c50    device

C:\>
```

Hình trên cho thấy có 01 thiết bị với mã “05863c50” đang được kết nối với máy tính. Lưu ý: công cụ `adb.exe` nằm trong thư mục `platform-tools` của *Android SDK*. Khi thiết bị đã được kết nối, ta có thể dùng lệnh “`adb install <apk file>`” để cài đặt ứng dụng lên thiết bị:



```
C:\projects\lectures>adb install LBS.apk
182 KB/s (19224 bytes in 0.102s)
    pkg: /data/local/tmp/LBS.apk
Success

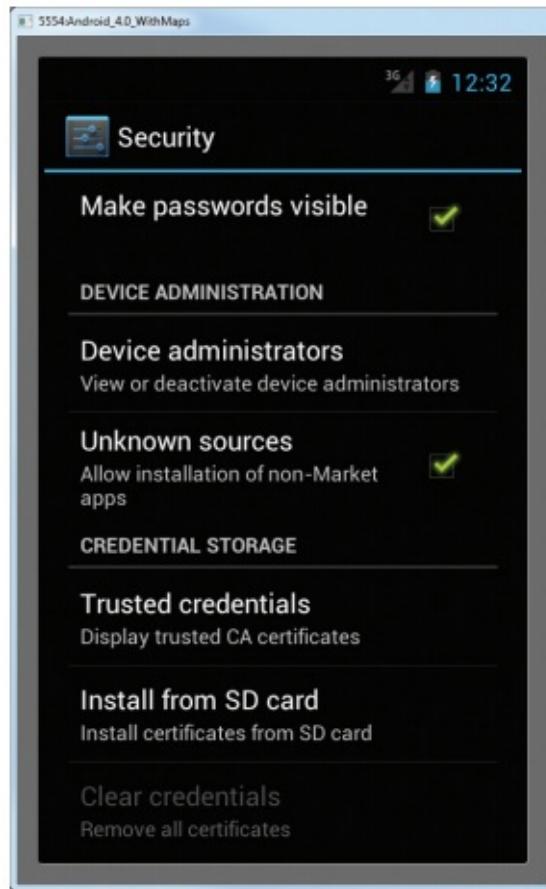
C:\projects\lectures>
```

Chữ “success” báo hiệu quá trình cài đặt đã thành công, ta sẽ thấy ứng dụng xuất hiện trong danh sách ứng dụng của thiết bị và sẵn sàng hoạt động.

Phân phối trên web server

Cách thứ 2 là đưa ứng dụng của bạn lên một web server và phân phối link tải (ví dụ: <http://myserver.com/app/LBS.apk>) đến người dùng. Người dùng chỉ cần bấm vào link này từ thiết bị android của mình, hoặc gõ đường link này vào thanh địa chỉ của trình duyệt web trên thiết bị, ứng dụng sẽ tự động được tải về.

Sau khi tải xong, bấm chọn file vừa tải để thực hiện cài đặt. Tuy nhiên để có thể cài đặt ứng dụng từ nguồn này (không phải từ *Google Play Store*), bạn cần bật tùy chọn “Unknown source” trong phần *Settings > Security* của thiết bị:

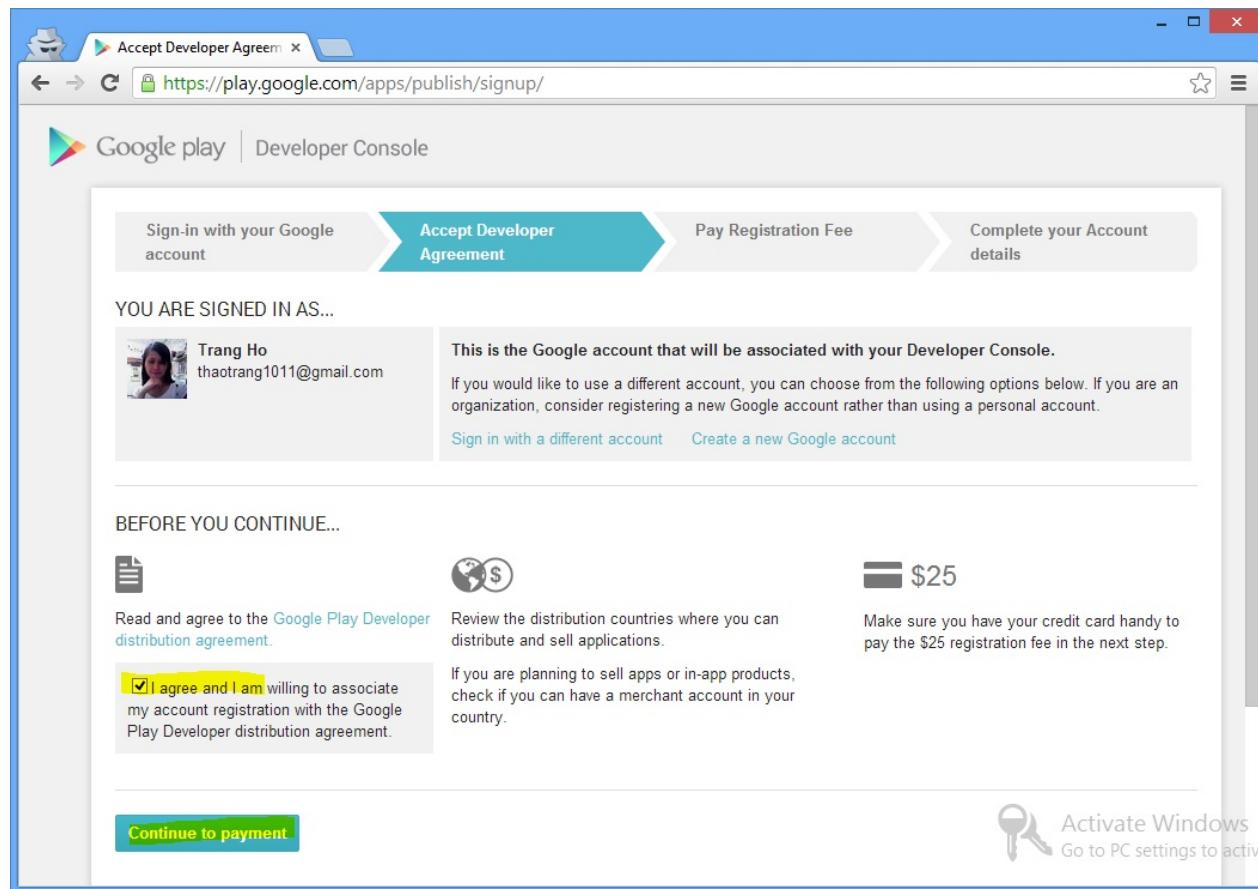


Hình: Bật tùy chọn cho phép cài đặt ứng dụng từ xa không thông qua Google Play

Phân phối trên Google Play Store

Kênh phân phối chính thống và tiềm năng nhất cho ứng dụng Android là chợ ứng dụng *Google Play Store* (cửa hàng Play) của *Google*. Để có thể phân phối ứng dụng trên cửa hàng Play, ta cần đăng ký tài khoản lập trình viên *Google Android*. Lệ phí lập tài khoản này là \$25 trọn đời tài khoản. Bạn có thể thanh toán khoản tiền này bằng thẻ thanh toán hoặc thẻ tín dụng quốc tế (*Visa, Master, America Express, Discovery...*)

Để đăng ký tài khoản lập trình viên, ta truy cập <https://play.google.com/apps/publish/signup/> và đăng nhập bằng tài khoản google của bạn (nếu chưa có tài khoản google, bạn cần tạo mới, miễn phí):



Bấm chấp nhận điều khoản và tiếp tục (*Continue to payment*). Trong màn hình tiếp theo, nhập thông tin thẻ quốc tế để thanh toán và bấm “Đồng ý và tiếp tục”:

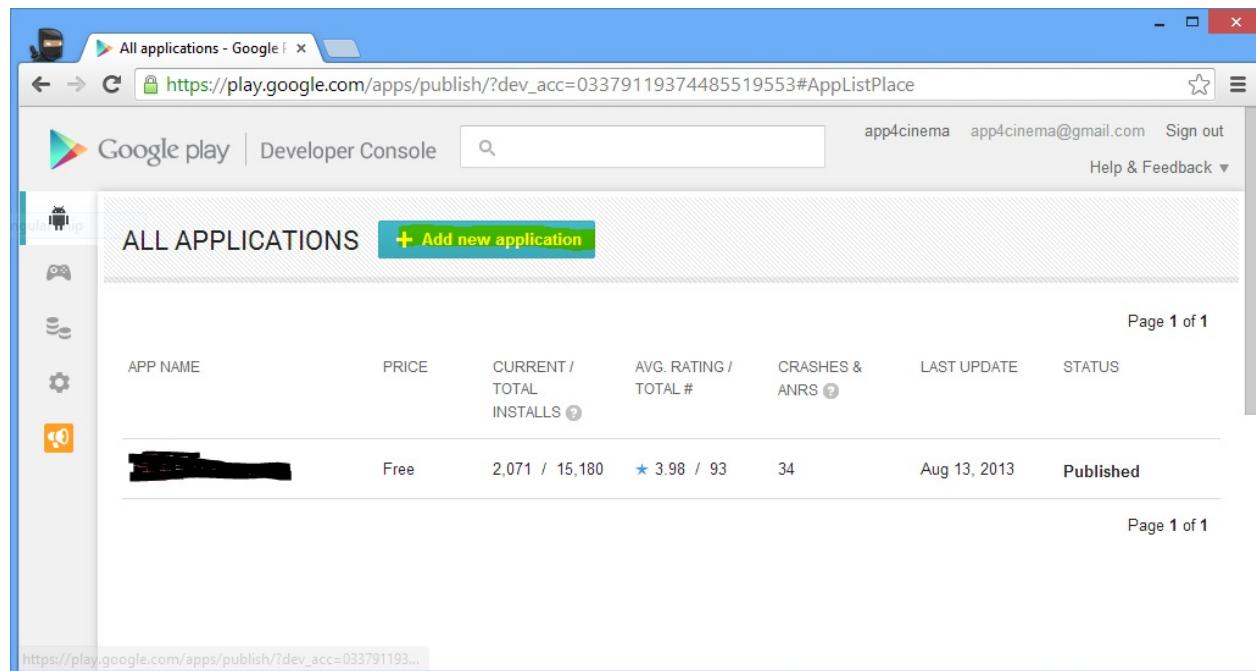
The screenshot shows the 'Thiết lập Google Wallet' (Setup Google Wallet) screen. At the top, it says 'TÊN VÀ VỊ ĐỊA CHỈ RIÊNG' (Name and address). Below this, there are fields for 'Tên' (Name), 'Địa chỉ đường phố' (Street address), 'Thành phố' (City), and a dropdown for 'Tỉnh/Thành phố' (Province/City) set to 'An Giang'. Under 'PHƯƠNG THỨC THANH TOÁN' (Payment method), it says 'Thẻ tín dụng hoặc thẻ ghi nợ' (Credit card or debit card). It shows a placeholder 'Số thẻ' (Card number) and logos for VISA, MasterCard, AMEX, and DISCOVER. Below this, there are fields for 'Ngày hết hạn' (Expiry date) with 'MM / YY' and 'Mã bảo mật' (Security code) with 'CVC' and a question mark icon. A checkbox 'Địa chỉ thanh toán giống với tên và địa chỉ nhà riêng' (Shipping address same as name and address) is checked. Another checkbox 'Gửi cho tôi các ưu đãi đặc biệt, lời mời cung cấp phản hồi về sản phẩm và bản tin của Google Wallet.' (Send me special offers, invites to provide feedback on products and Google Wallet news) is also checked. At the bottom, a note says 'Tôi đồng ý với Điều khoản dịch vụ và Thông báo bảo mật của Google Wallet.' (I agree with the Google Wallet Terms of Service and Privacy Policy). The 'Đồng ý và tiếp tục' (Agree and continue) button is highlighted in yellow.

Hình: Thiết lập thông tin thanh toán

Nếu thẻ của bạn được chấp nhận, Google sẽ đưa tài khoản của bạn vào trạng thái chờ kiểm tra, sau khoảng vài ngày sẽ có phản hồi từ Google chấp nhận tài khoản của bạn hay không. Trong trường hợp không được chấp nhận, Google sẽ yêu cầu bạn gửi thêm tài liệu chứng thực thông tin cá nhân và tài khoản ngân hàng của bạn.

Trong thời gian chờ đợi Google kiểm tra trạng thái thanh toán, bạn vẫn có thể đưa ứng dụng lên server của Google, tuy nhiên chưa thể phân phối đến người dùng.

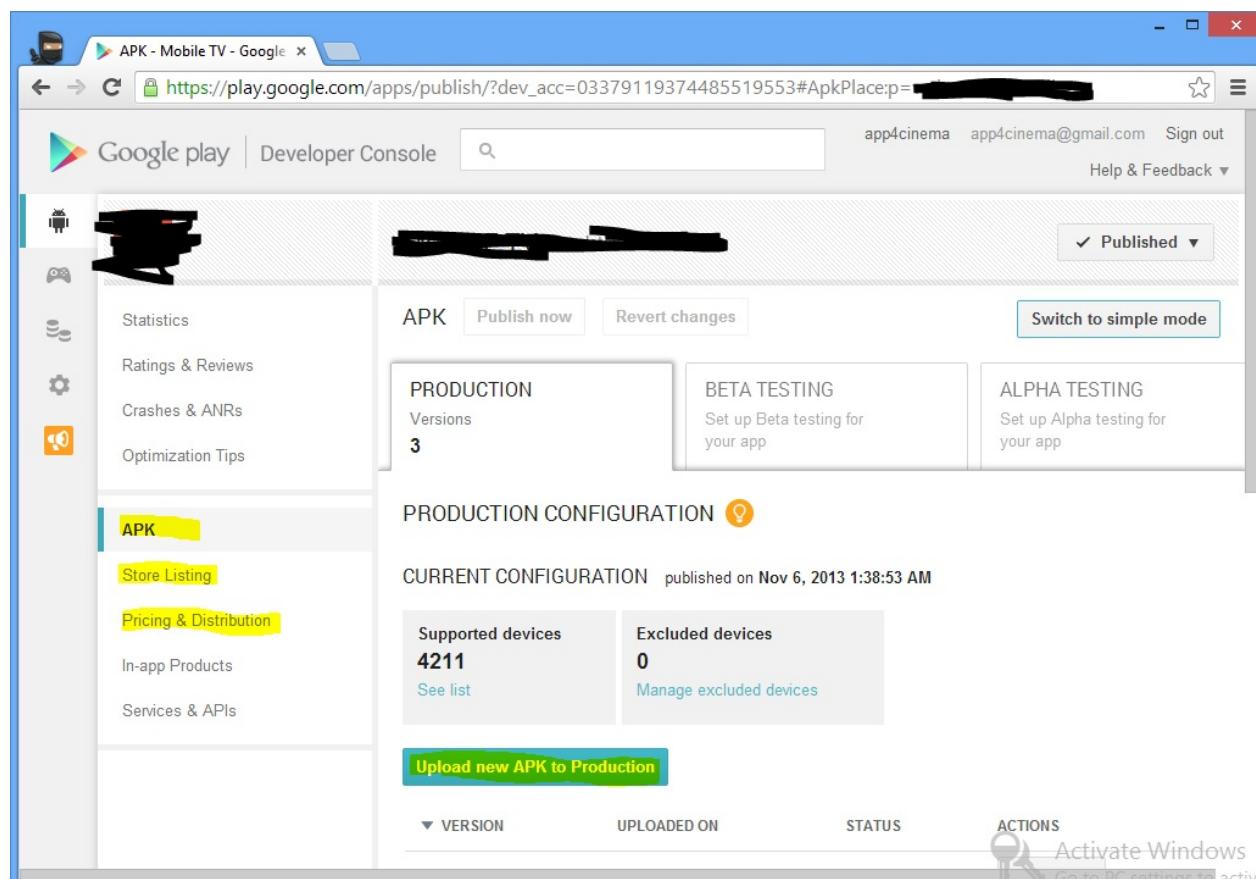
Giao diện của *Developer console* sau khi đăng ký thành công sẽ có dạng như sau:



Hình: Danh sách ứng dụng đang được xuất bản trên Google Play

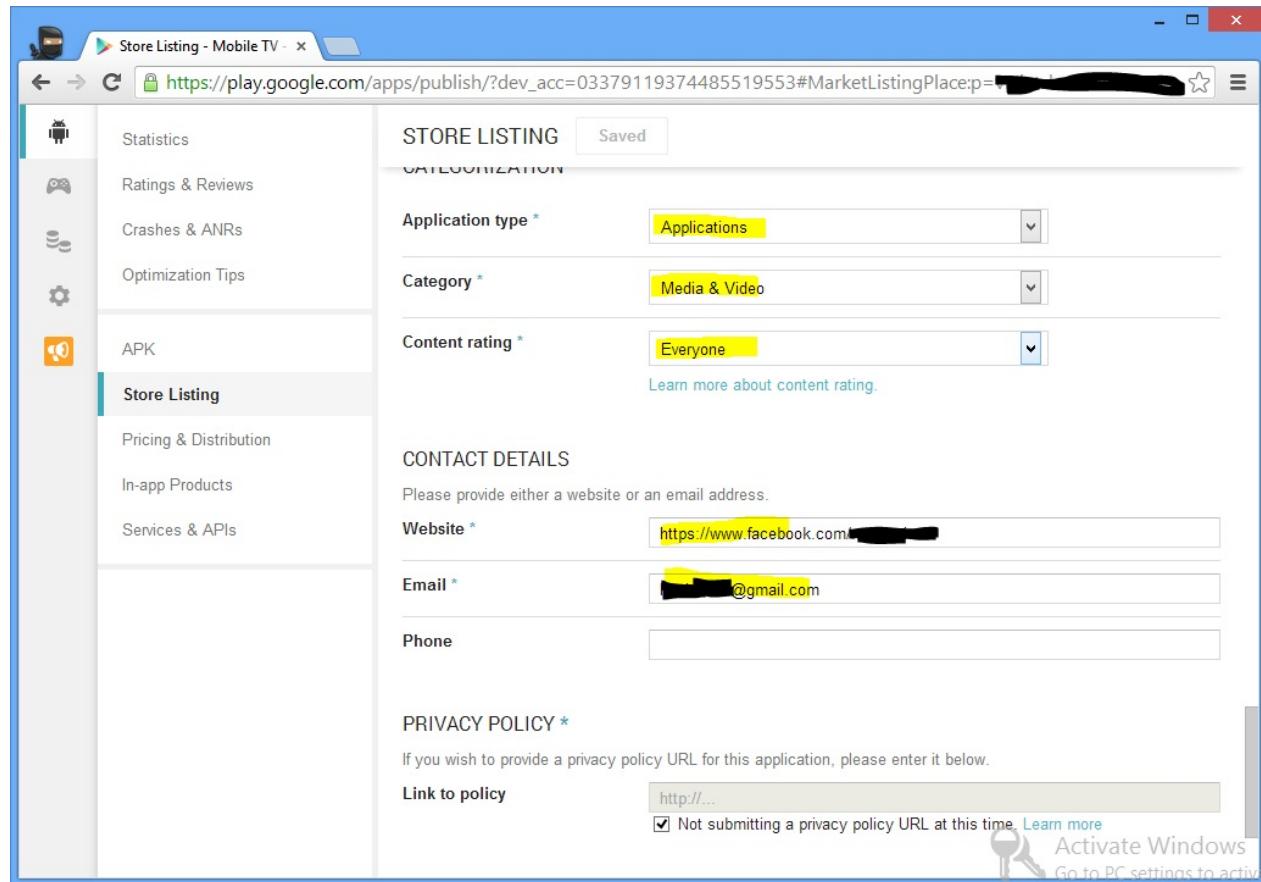
Tài khoản trên đã có một ứng dụng đang được phân phối đến người dùng, ứng dụng này hiện đang có 2071 người dùng trên tổng số 15180 lượt tải, có 93 người bình chọn và điểm trung bình là 3.98/5 sao!!!!

Để phân phối ứng dụng khác, ta bấm vào nút “+ Add new application” phía trên của trang (nút được bôi vàng trên hình). Bạn cần cung cấp đầy đủ thông tin yêu cầu về ứng dụng trước khi có thể phân phối đến người dùng như: tải file ứng dụng (apk) của bạn lên hệ thống, mô tả ứng dụng và chính sách giá.



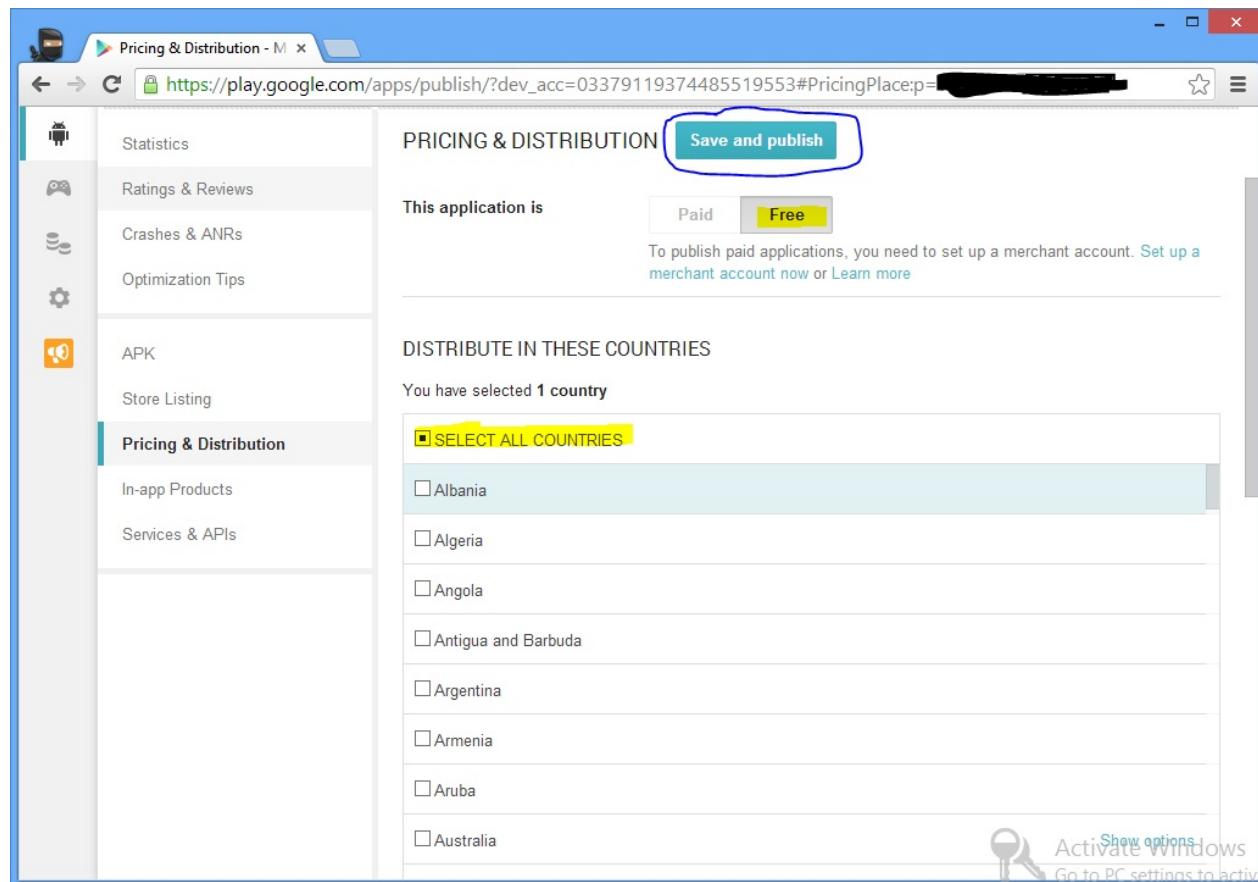
Hình: Tải file ứng dụng đã đóng gói (apk file) lên Google Play

Cung cấp tiêu đề (tên ứng dụng), mô tả, tải lên file icon lớn (512x512px) và ít nhất 02 hình chụp màn hình của ứng dụng, chọn loại ứng dụng, cấu hình website và email hỗ trợ...:



Hình: Thông tin mô tả ứng dụng đã Android trên Google Play

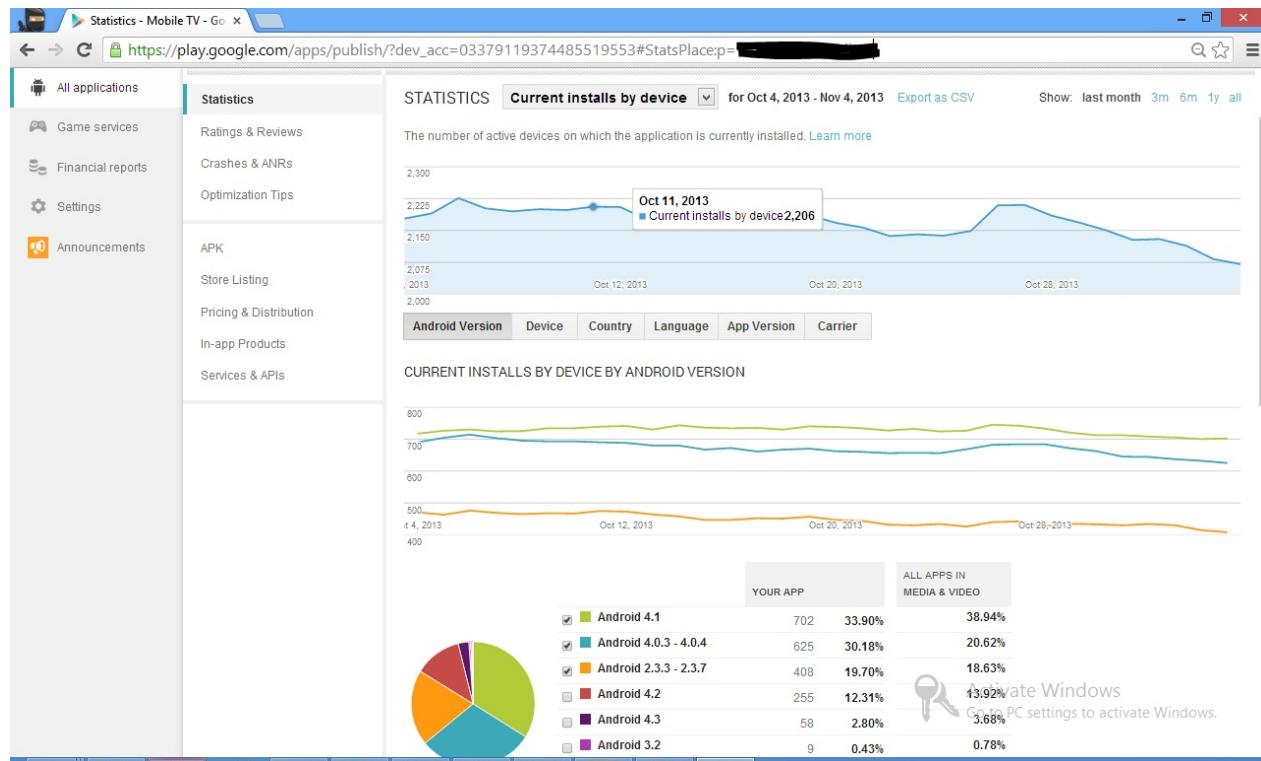
Đặt giá bán ứng dụng (hiện tại ở Việt Nam ta chỉ được phép đăng ứng dụng miễn phí) và các quốc gia muốn phân phối ứng dụng:



Hình: Cấu hình chính sách giá cho ứng dụng trên Google Play

Cuối cùng bấm “ Save and publish ”. Ứng dụng của bạn sẽ bắt đầu được phân phối trên hệ thống Google Play. Chú ý: cần phải mất từ vài giờ đến vài ngày để ứng dụng có thể vượt qua được hệ thống kiểm tra của Google và phân phối khắp mạng CDN của Google.

Sau khi hoàn tất các bước như trên, ta đã hoàn thành việc phát triển một ứng dụng cho hệ điều hành Android và phân phối đến người dùng. Bạn có thể thường xuyên ghé thăm trang dành cho developer này để xem các thống kê khác nhau liên quan đến việc cài đặt và sử dụng ứng dụng của mình như: số lượt cài đặt/gỡ bỏ theo ngày, tỉ lệ các phiên bản Android đang dùng, các lỗi crash ứng dụng, đánh giá, phản hồi của người dùng... Hình dưới đây minh họa một trong những màn hình thống kê như vậy. Phần này cũng kết thúc giáo trình “**Lập trình cho thiết bị di động**” của chúng ta!



Hình: Thống kê các thiết bị đã cài đặt ứng dụng từ Google Play