# **TRƯỜNG ĐẠI HỌC khoa hỌC TỰ NHIÊN**

**KHOA TOÁN – CƠ – TIN HỌC**

**Báo cáo chuyên đề**

**Một số vấn đề chọn lọc về phát triển phần mềm và ngôn ngữ lập trình**

**GIÁO VIÊN HƯỚNG DẪN**

**PGS-TS. Lê Trọng Vĩnh**

**Nghiên cứu sinh**

**Nguyễn Minh Hiệp**

***Hà Nội, 2014***

1. GIỚI THIỆU ANDROID
   1. Khái niệm về Android

Trước hết Android là nền tảng phần mềm dựa trên mã nguồn mở Linux OS (Kernel 2.6) cho máy di động và những phần mềm trung gian (middleware) để hổ trợ các ứng dụng mà người sử dụng cần đến. Một cách định nghĩa không quá chuyên môn thì có thể coi Android là tên một nền tảng mở cho thiết bị di động của Google (gồm hệ điều hành, middleware và một số ứng dụng cơ bản). Android sẽ đương đầu với một số hệ điều hành (viết tắt là HDH) dành cho thiết bị di dộng khác đang hâm nóng thị trường như Windows Mobile, Symbian và dĩ nhiên là cả OS X (iPhone). Có thể nói một cách nôm na rằng Android là một HDH chạy trên thiết bị di động, cũng giống như Windows, Linux hay Mac chạy trên máy vi tính vậy.

* + 1. Android khác với các hệ điều hành chạy trên thiết bị di động khác

Android đã thu hút được sự chú ý của giới công nghệ khắp toàn cầu khi đứa con của Google sử dụng giấy phép mã nguồn mở. Đó là một sản phẩm kết tinh từ ý tưởng của Khối Liên minh thiết bị cầm tay mở do Google dẫn đầu, gồm 34 thành viên với các công ty hàng đầu về công nghệ và di động toàn cầu như Qualcomm, Intel, Motorola, Texas Instruments và LG Electronics, các nhà mạng như T-Mobile, Sprint Nextel, NTT DoCoMo và China Mobile. Các nhà phát triển có thể sử dụng miễn phí bộ Kit Android Software Development để xây dựng các ứng dụng của mình.

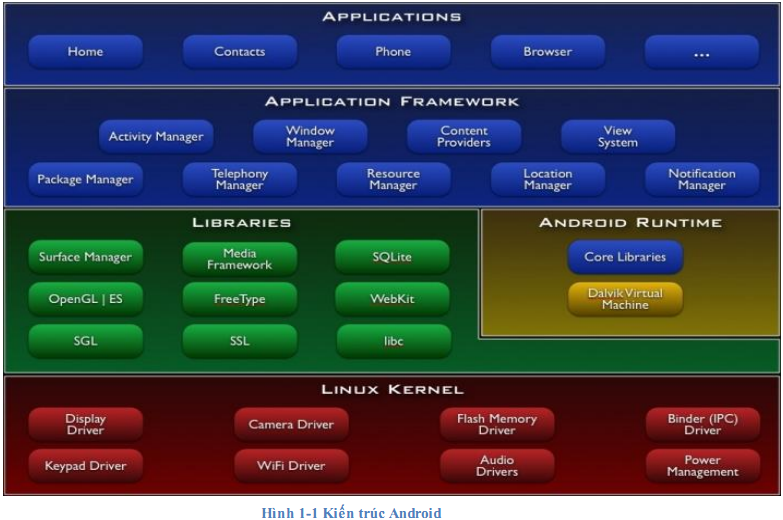
* + 1. Đặc tính mở của Android

Android được xây dựng để cho phép các nhà phát triển để tạo ra các ứng dụng di động hấp dẫn, tận dụng tất cả tính năng một chiếc điện thoại đã cung cấp. Nó được xây dựng để được thực sự mở. Ví dụ, một ứng dụng có thể gọi bất kỳ chức năng lõi của điện thoại như thực hiện cuộc gọi, gửi tin nhắn văn bản, hoặc bằng cách sử dụng máy ảnh, cho phép các nhà phát triển để tạo ra nhiều ứng dụng phong phú hơn cho người dùng (điều này hiện chưa có trên Windows Phone7 của Microsoft). Android được xây dựng trên mã nguồn mở Linux Kernel. Hơn nữa, nó sử dụng một máy ảo tuỳ chỉnh được thiết kế để tối ưu hóa bộ nhớ và tài nguyên phần cứng trong một môi trường di động.

Android không phân biệt giữa các ứng dụng lõi của điện thoại và các ứng dụng của bên thứ ba. Tất cả có thể được xây dựng để có thể truy cập bằng khả năng của một thiết bị di động cung cấp cho người sử dụng với một dải rộng các ứng dụng và dịch vụ. Với các thiết bị xây dựng trên Android, người dùng có thể hoàn toàn thích ứng với điện thoại đến lợi ích của họ. Với Android, một nhà phát triển có thể xây dựng một ứng dụng cho phép người dùng xem vị trí của bạn bè của họ và được cảnh báo khi họ đang có trong vùng phụ cận cho họ một cơ hội để kết nối.

Android cung cấp truy cập đến một loạt các thư viện công cụ hữu ích và có thể được sử dụng để xây dựng các ứng dụng phong phú. Ví dụ, Android cho phép các thiết bị giao tiếp với nhau tạo điều kiện cho đồng đẳng rich-to-peer trong ứng dụng xã hội. Ngoài ra, Android bao gồm một tập hợp đầy đủ công cụ đã được xây dựng công phu, với việc cung cấp nền tảng phát triển, với năng suất cao và cái nhìn sâu vào các ứng dụng .

* 1. KIẾN TRÚC CỦA ANDROID



* + 1. Android Platform

Bao gồm HDH Android đầy đủ tính năng, các ứng dụng và các tầng trung gian để developer có thể mở rộng, tùy chỉnh hoặc thêm vào các component của họ.

Có 4 tầng cơ bản trong HDH Android: Application Framework, Android Runtime, Native Libraries, Linux Kernel ... Mỗi tầng làm việc đều nhờ sự giúp đỡ của tầng bên dưới.

* + 1. Tầng Linux Kernel

Đây là nhân của HDH Android, mọi xử lý của hệ thống đều phải thông qua tầng này. Linux Kernel cung cấp các trình điều khiển thiết bị phần cứng (driver) như: camera, USB, Wifi, Bluetooth, Display, Power Management ... Android dựa trên Linux phiên bản 2.6 lựa chọn các tính năng cốt lõi như bảo mật, quản lý bộ nhớ, quản lý tiến trình, mạng stack và các trình điều khiển phần cứng. Kernel hoạt động như một lớp trừu tượng giữa phần cứng và phần mềm còn lại của hệ thống.

* + 1. Native Libraries
* System C library - có nguồn gốc từ hệ thống thư viện chuẩn C (libc), điều chỉnh các thiết bị nhúng trên Linux.
* Media Libraries - mở rộng từ PacketVideo's OpenCORE; thư viện hỗ trợ playback và recording của nhiều định dạng video và image phổ biến: MPEG4, H.264, MP3, AAC, AMR, JPG, và PNG
* Surface Manager - quản lý việc hiển thị và kết hợp đồ họa 2D và 3D.
* LibWebCore - Android dùng lại webkit engine cho việc render trình duyệt mặc định của HDH Android browser và cho dạng web nhúng (như HTML nhúng)
* SGL - 2D engine
* 3D libraries - Thư viện 3D dựa trên OpenGL ES 1.0 API, có nâng cấp tăng tốc "hardware 3D acceleration"
* FreeType - render bitmap và vector font.
* SQLite - quản lý database của ứng dụng.
  + 1. Tầng Runtime

Mỗi ứng dụng Android chạy trên một proccess riêng của Dalvik VM (máy ảo). Dalvik được viết để chạy nhiều máy ảo cùng một lúc một cách hiệu quả trên cùng một thiết bị.

Máy ảo Dalvik thực thi các file mang định dạng .dex (Dalvik Excutable), định dạng này là định dạng đã được tối ưu hóa để chỉ chiếm một vùng nhớ vừa đủ xài và nhỏ nhất có thể. VM chạy các class (đã được compile trước đó bởi một trình biên dịch ngôn ngữ Java), sở dĩ VM chạy được các class này là nhờ chương trình DX tool đã convert các class sang định dạng .dex.

* + 1. Tầng Application Framework

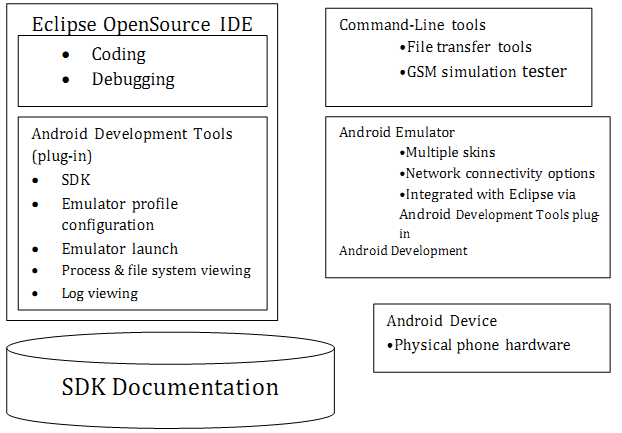
Đây là tầng mà Google xây dựng cho các developer để phát triển các ứng dụng của họ trên Android, chỉ bằng cách gọi các API có sẵn mà Google đã viết để sử dụng các tính năng của phần cứng mà không cần hiểu cấu trúc bên dưới.

Bằng cách cung cấp một nền tảng phát triển mở, Android cho các nhà phát triển khả năng xây dựng các ứng dụng cực kỳ phong phú và sáng tạo. Nhà phát triển được tự do tận dụng các thiết bị phần cứng, thông tin địa điểm truy cập, các dịch vụ chạy nền, thiết lập hệ thống báo thức, thêm các thông báo để các thanh trạng thái, và nhiều, nhiều hơn nữa.

* View UI dùng để xây dựng layout của ứng dụng bao gồm: list view, text field, button, dialog, form ...
* Content Providers cho phép các ứng dụng có thể truy cập dữ liệu từ các ứng dụng khác (như ứng dụng của ta có thể lấy thông tin Contacts của điện thoại Android), hoặc để chia sẻ dữ liệu của riêng ứng dụng.
* Resource Manager cung cấp cách thức truy cập đến non-code resources như các asset, graphic, image, music, video ...
* Notification Manager cho phép tất cả các ứng dụng hiển thị thông báo của mình trên HDH.
* Activity Manager quản lý vòng đời của các ứng dụng.

1. MÔI TRƯỜNG LẬP TRÌNH
   1. GIỚI THIỆU

Trong phần này sẽ giới thiệu các công cụ lập trình cho Android (Android Development Tools). Chúng ta sẽ dần làm quen với Eclipse và Android Development Tool plug-in, thông qua Android SDK và công cụ của nó, chạy một ứng dụng Android trên Emulator (tạm dịch là trình giả lập giao diện của Android trên PC). Với những kĩ năng đó trong tay, chúng ta sẽ tìm hiểu các gói Java packages được cung cấp trong SDK giúp nâng cao khả năng lập trình trên Android.



Hình 2-1. Môi trường lập trình Android

* 1. ANDROID SDK

Android SDK, viết tắt của Android Software Development Kit, có thể gọi là “công cụ phát triển phần mềm” mà cụ thể ở đây là phát triển ứng dụng cho Android OS. Đến đây thì chắc các chúng ta cũng đã hiểu đơn giản Google Adroid SDK là bộ công cụ được chính Google xây dựng và phát hành miễn phí đến giới Developer để họ dễ dàng xây dựng và phát triển các ứng dụng chạy được trên Android OS. Android SDK có các đặc tính sau:

* Được phát triển và cung cấp miễn phí
* Truy cập đến phần cứng Wi-Fi.
* GSM, EDGE và tính năng mạng 3G cho phép việc chuyển dữ liệu, gọi điện hay gửi tin SMS trong mạng di động.
* Gói API toàn diện cho các dịch vụ nền tảng ví dụ như GPS.
* Truy cập toàn bộ phần cứng điều khiển đa phương tiện như chơi nhạc, ghi âm hay sử dụng microphone và camera.
* Chia sẻ dữ liệu trong kho dữ liệu.
* Tích hợp trình duyệt dựa trên bộ WebKit (mã nguồn mở).
* P2P hổ trợ sử dung Google Talk.
* Hỗ trợ tốt đồ họa 3D nhờ sử dụng OpenGLES.
  1. MÁY ẢO DALVIK

Một trong những thành phần quan trọng của Android là máy ảo Dalvik. Thay vì sử dụng máy ảo Java như trước kia Android sử dùng máy ảo của riêng nó được thiết kế để bảo đảm rằng đa ứng dụng có thể chạy mượt mà trên một thiết bị di động.

Máy ảo Dalvik sử dụng lõi Linux để xử lý các chức năng ở mức thấp bao gồm bảo mật, các tiến trình, các luồng và quản lý vùng nhớ. Nó cũng có thể viết bằng ứng dụng C/C++ để chạy trực tiếp lên lõi Linux bên dưới. Giữa phần cứng và các dịch vụ hệ thống được quản lý bi máy ảo Dalvik, nó là một thành phần ở giữa. Bằng cách sử dụng máy ảo này để chạy ứng dụng, các nhà phát triển hoàn toàn không phải bận tâm gì về các phần cứng bên dưới. Khi máy ảo này chạy nó sẽ tạo ra tập tin có đuôi là .dex, tập tin này được SDK tạo ra bởi sự chuyển đổi từ các lớp biên dịch ngôn ngữ Java .

* 1. CÁC GÓI JAVA CẦN THIẾT
* java.lang - gói chứa các lớp lõi của Java
* java.io - xuất nhập.
* java.net - kết nối mạng.
* java.util - chứa các lớp tiện ích. Bao gồm: Log thường được dùng đề viết LogCat.
* java.text - tiện ích sử lý văn bản.
* java.math - các lớp toán học.
* javax.net - các lớp mạng.
* javax.security - các lớp liên quan đến bảo mật.
* javax.xml - các lớp liên quan đến DOM-based XML.
* org.apache.\* - các lớp liên quan đến HTTP-related.
* org.xml - các lớp liên quan đến SAX-based XML .

Các gói dùng cho lập trình Android:

* Android.app - Android application model access
* Android.content - truy xuất dữ liệu trong Android
* Android.net - bao gồm Uri class dùng cho việc truy xuất các nội dung khác
* Android.graphics - đồ họa
* Android.opengl - các lớp OpenGL
* Android.os - truy cập ở mức độ hệ thông đến môi trường Android
* Android.provider - các lớp liên quan đến ContentProvider
* Android.telephony - khả năng truy cập Telephony
* Android.text - Text layout
* Android.util - tập các tiện thao tác trên văn bản,bao gồm XML
* Android.view - thành phần UI (giao diện người dùng)
* Android.webkit - Browser functionality
* Android.widget - thành phần UI mức độ cao hơn
  1. THÀNH PHẦN CHÍNH CỦA ECLIPSE KHI TẠO ỨNG DỤNG TRÊN ANDROID
* Môi trường phát triển Rich Java bao gồm trình biên dịch, class autocompletion, và tích hợp Javadoc
* Bẫy lỗi code
* Quản lý và chạy Android Emulator profile
* The Dalvik Debug Monitoring Service (DDMS)
* Khung nhìn các tiến trình (Thread and heap views)
* Quản lý Hệ thống tập tin giao diện (Emulator filesystem management)
* Điều khiển dữ liệu và hội thoại (Data and voice network control)
* Điều khiển giao diện (Emulator control)
* Ghi lỗi hệ thống và ứng dụng (System and application logging)
  1. THÀNH PHẦN QUAN TRỌNG TRONG MỘT ANDROID PROJECT
* Activity (Android.app.Activity): đây là lớp khởi tạo giao diện ứng dụng nội bộ trên Android tương tư như MIDlet trong J2ME.
* Service (Android.app.Service): cung cấp các dịch vụ liên quan đến client/service. Một Service sẽ chạy ngầm bên dưới, sau đó các client (Activity) sẽ kết nối và truy xuất các hàm trên dịch thông qua Interface class.
* Broadcast receiver (Android.content.BroadcastReceiver): đây là một ứng dụng chạy ngầm dùng để đọc và cập nhật thông tin trên UI, ví dụ như cập nhật sự thay đỗi giờ, pin...
* Content Provider: cung cấp chức năng truy vấn dữ liệu giữa các ứng dụng của Android.
* Intent: nền tảng để truyền tải các thông báo. Intent được sử dụng để gửi các thông báo đi nhằm khởi tạo 1 Activity hay Service để thực hiện công việc mà chúng ta mong muốn.
  1. CHU KỲ SỐNG CỦA ỨNG DỤNG ANDROID

Một tiến trình Linux gói gọn một ứng dụng Android đã được tạo ra cho ứng dụng khi codes cần được xây dựng, khởi chạy và sẽ còn chạy cho đến khi:

* Nó không phụ thuộc.
* Hệ thống cần lấy lại bộ nhớ mà nó chiếm giữ cho các ứng dụng khác.

Một sự khác thường và đặc tính cơ bản của Android là thời gian sống của tiến trình ứng dụng không được điều khiển trực tiếp bởi chính nó. Thay vào đó, nó được xác định bởi hệ thống qua một kết hợp của:

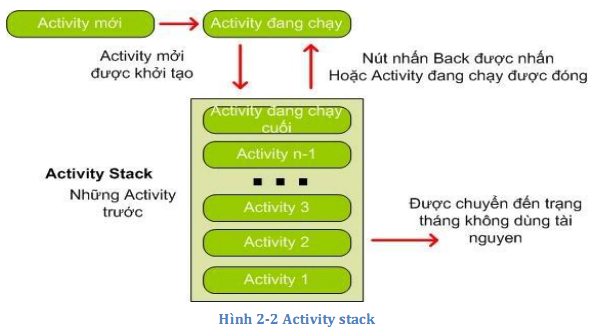
* Những phần của ứng dụng mà hệ thống biết đang chạy.
* Những phần đó quan trọng như thế nào đối với người dùng.
* Bao nhiêu vùng nhớ chiếm lĩnh trong hệ thống.
  + 1. Chu kỳ sống thành phần

Các thành phần ứng dụng có một chu kỳ sống, tức là mỗi thành phần từ lúc bắt đầu khởi tạo và đến thời điểm kết thúc, đôi lúc chúng có thể là active (visible hoặc invisible) hoặc inactive.

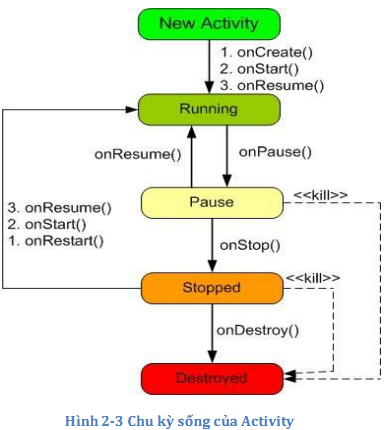
* + 1. Activity Stack

Bên trong hệ thống các activity được quản lý như một activity stack. Khi một Activity mới được start, nó được đặt ở đỉnh của stack và trở thành activity đang chạy activity trước sẽ ở bên dưới activity mới và sẽ không thấy trong suốt quá trình activity mới tồn tại.

Nếu người dùng nhấn nút Back thì activity kết tiếp của stack sẽ di duyển lên và trở thành active.



* + 1. Các trạng thái của chu kỳ sống



Một Activity chủ yếu có 4 chu kỳ chính sau:

* Active hoặc running: Khi Active là được chạy trên màn hình. Activity này tập trung vào những thao tác của người dùng trên ứng dụng.
* Paused: Activity là được tạm dừng (paused) khi mất focus nhưng người dùng vẫn trông thấy. Có nghĩa là một Activity mới ở trên nó nhưng không bao phủ đầy màn hình. Một Activity tạm dừng là còn sống nhưng có thể bị kết thúc bởi hệ thống trong trường hợp thiếu vùng nhớ.
* Stopped: Nếu nó hoàn toàn bao phủ bởi Activity khác. Nó vẫn còn trạng thái và thông tin thành viên trong nó. Người dùng không thấy nó và thường bị loại bỏ trong trường hợp hệ thống cần vùng nhớ cho tác vụ khác.
* Killed: Khi hệ thống bị thiếu bộ nhớ, nó sẽ giải phóng các tiến trình theo nguyên tắc ưu tiên. Các Activity ở trạng thái stop hoặc paused cũng có thể bị giải phóng và khi nó được hiển thị lại thì các Activity này phải khởi động lại hoàn toàn và phục hồi lại trạng thái trước đó.
  + 1. Chu kỳ sống của ứng dụng

Trong một ứng dụng Android có chứa nhiều thành phần và mỗi thành phần đều có một chu trình sống riêng. Và ứng dụng chỉ được gọi là kết thúc khi tất cả các thành phần trong ứng dụng kết thúc. Activity là một thành phần cho phép người dùng giao tiếp với ứng dụng. Tuy nhiên, khi tất cả các Activity kết thúc và người dùng không còn giao tiếp được với ứng dụng nữa nhưng không có nghĩa là ứng dụng đã kết thúc. Bởi vì ngoài Activity là thành phần có khả năng tương tác người dùng thì còn có các thành phần không có khả năng tương tác với người dùng như là Service, Broadcast receiver. Có nghĩa là những thành phần không tương tác người dùng có thể chạy background dưới sự giám sát của hệ điều hành cho đến khi người dùng tự tắt chúng.

* + 1. Các sự kiện trong chu kỳ sống của ứng dụng



Nếu một Activity được tạm dừng hoặc dừng hẳn, hệ thống có thể bỏ thông tin khác của nó từ vùng nhớ bởi việc gọi hàm finish() của nó, hoặc đơn giản giết tiến trình của nó. Khi nó được hiển thị lần nữa với người dùng, nó phải được hoàn toàn restart và phục hồi lại trạng thái trước. Khi một Activity chuyển qua chuyển lại giữa các trạng thái, nó phải báo việc chuyển của nó bằng việc gọi hàm transition.

Tất cả các phương thức là những móc nối mà chúng ta có thể override để làm tương thich công việc trong ứng dụng khi thay đổi trạng thái. Tất cả các Activity bắt buộc phải có *onCreate()* để khởi tạo ứng dụng. Nhiều Activity sẽ cũng hiện thực *onPause()* để xác nhận việc thay đổi dữ liệu và mặt khác chuẩn bị dừng hoạt động với người dùng.

* + 1. Thời gian sống của ứng dụng

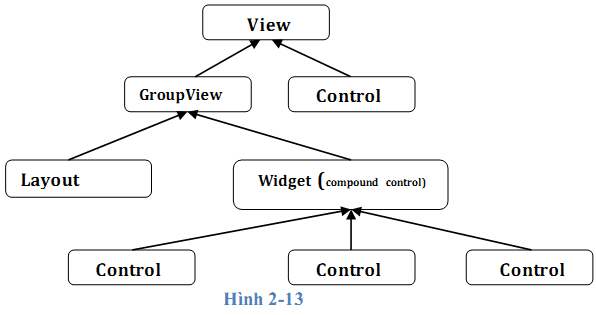
Bảy phương thức chuyển tiếp định nghĩa trong chu kỳ sống của một Activity. Thời gian sống của một Activity diễn ra giữa lần đầu tiên gọi *onCreate()* đến trạng thái cuối cùng gọi *onDestroy().* Một Activity khởi tạo toàn bộ trạng thái toàn cục trong *onCreate(),* và giải phóng các tài nguyên đang tồn tại trong *onDestroy().*

* + 1. Thời gian hiển thị của Activity

Visible lifetime của một activity diễn ra giữa lần gọi một *onStart()* cho đến khi gọi *onStop()*. Trong suốt khoảng thời gian này người dùng có thể thấy activity trên màn hình, có nghĩa là nó không bị foreground hoặc đang tương tác với người dùng. Giữa 2 phương thức người dùng có thể duy trì tài nguyên để hiển thị activity đến người dùng.

* + 1. Các hàm thực thi
* OnCreate(...): hàm này được gọi khi lớp Activity được khởi tạo, dùng để thiết lập giao diện ứng dụng và thực thi những thao tác cơ bản.
* onStart(): hàm này được gọi khi lớp ứng dụng xuất hiện trên màn hình.
* onResume(): hàm được gọi ngay sau OnStart hoặc khi người dùng focus ứng dụng, hàm này sẽ đưa ứng dụng lên top màn hình.
* onPause(): hàm được gọi khi hệ thống đang focus đến một activity trước đó.
* onStop(): hàm được gọi khi một activity khác được khởi động và focus.
* onRestart(): đưọc gọi khi ứng dụng chuyển sang onStop(), nhưng muốn khởi động lại bằng onStart().
  + 1. THÀNH PHẦN GIAO DIỆN ANDROID
       1. View

Trong Android giao diện người dùng được xây dựng từ các đối tượng View. Trong Android Platform, các screen luôn được bố trí theo một kiểu cấu trúc phân cấp như hình dưới. Một screen là một tập hợp các Layout và các widget được bố trí có thứ tự. Để thể hiện một screen thì trong hàm *onCreate* của mỗi Activity cần phải được gọi một hàm là s*etContentView* (R.layout.main); hàm này sẽ load giao diện từ file MAIN.XML lên để phân tích thành mã bytecode.



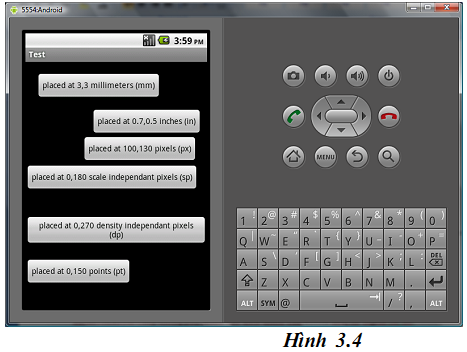
* + - 1. VIEWGROUP

ViewGroup là sự mở rộng của class View hay nói cách khác ViewGroup chính là các WidgetLayout được dùng để bố trí các đối tượng khác trong một screen.Có các loại ViewGroup như sau:

* Linear Layout: LinearLayout được dùng để bố trí các thành phần giao diện theo chiều ngang hoặc chiều dọc nhưng trên một line duy nhất mà không có xuống dòng.
* Frame layout: FrameLayout được dùng để bố trí các đối tượng theo kiểu giống như là các Layer trong Photoshop. Những đối tượng nào thuộc Layer bên dưới thì sẽ bị che khuất bởi các đối tượng thuộc Layer nằm trên. FrameLayer thường được sử dụng khi muốn tạo ra các đối tượng có khung hình bên ngoài chẳng hạn như contact image button.
* Table layout: Layout này được sử dụng khi cần thiết kế một table chứa dữ liệu hoặc cần bố trí các widget theo các row và column. Chẳng hạn như, giao diện của một chiếc máy tính đơn giản hoặc một danh sách dữ liệu.
* AbsoluteLayout: Layout này được sử dụng để bố trí các widget vào một vị trí bất kì trong layout dựa vào 2 thuộc tính toạ độ x, y. Tuy nhiên, kiểu layout này rất ít khi được dùng bởi vì toạ độ của các đối tượng luôn cố định và sẽ không tự điều chỉnh được tỷ lệ khoảng cách giữa các đối tượng. Khi chuyển ứng dụng sang một màn hình có kích thước với màn hình thiết kế ban đầu thì vị trí của các đối tượng sẽ không còn được chính xác như ban đầu.

Dùng câu lệnh *<AbsoluteLayout></ AbsoluteLayout>* để gọi Layout này. Muốn định vị một đối tượng, phải định vị hai giá trị Android:layout\_x="…" và Android:layout\_y="…".





RELATIVE LAYOUT: Layout này cho phép bố trí các widget theo một trục đối xứng ngang hoặc dọc. Để đặt được đúng vị trí thì các widget cần được xác định một mối ràng buộc nào đó với các widget khác. Các ràng buộc này là các ràng buộc trái, phải, trên, dưới so với một widget hoặc so với layout parent. Dựa vào những mối ràng buộc đó mà RetaliveLayout cũng không phụ thuộc vào kích thước của màn hình thiết bị. Ngoài ra, nó còn có ưu điểm là giúp tiết kiệm layout sử dụng nhằm mục đích giảm lượng tài nguyên sử dụng khi load đồng thời đẩy nhanh quá trình xử lý.



* + 1. CÁC CONTROL
       1. BUTTON

Sở dĩ widget button được giới thiệu đầu tiên trong số các widget khác là vì đây là đối tượng có thể nói là được dùng nhiều nhất trong hầu hết các ứng dụng Android.

Để thiết kế giao diện với một button ta có 2 cách như sau:

*<Button*

*Android:layout\_width="wrap\_content"*

*Android:layout\_height="wrap\_content"*

*Android:id="@+id/cmdButton1"*

*Android:text="Touch me!"*

*/>*

Đây là cách làm ra một button đơn giản. Đoạn mã trên sau khi chạy:



Để khai báo một Button trong code ta làm như sau:



* + - 1. LISTVIEW

Được sử dụng để thể hiện một danh sách các thông tin theo từng cell. Mỗi cell thông thường được load lên từ một file XML đã được cố định trên đó số lượng thông tin và loại thông tin cần được thể hiện.

Để thể hiện được một list thông tin lên một screen thì cần phải có 3 yếu tố chính:

* Data Source: Data Source có thể là một ArrayList, HashMap hoặc bất kỳ một cấu trúc dữ liệu kiểu danh sách nào.
* Adapter: Adapter là một class trung gian giúp ánh xạ dữ liệu trong Data Source vào đúng vị trí hiển thị trong ListView. Chẳng hạn, trong Data Source có một trường name và trong ListView cũng có một TextView để thể hiện trường name này. Tuy nhiên, ListView sẽ không thể hiển thị dữ liệu trong Data Source lên được nếu như Adapter không gán dữ liệu vào cho đối tượng hiển thị.
* ListView: ListView là đối tượng để hiển thị các thông tin trong Data Source ra một cách trực quan và người dùng có thể thao tác trực tiếp trên đó.

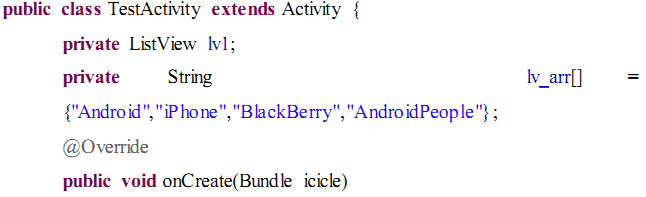
Để gọi đối tượng ListView trong xml. Mỗi ListView phải có ID để gọi trong Activity.

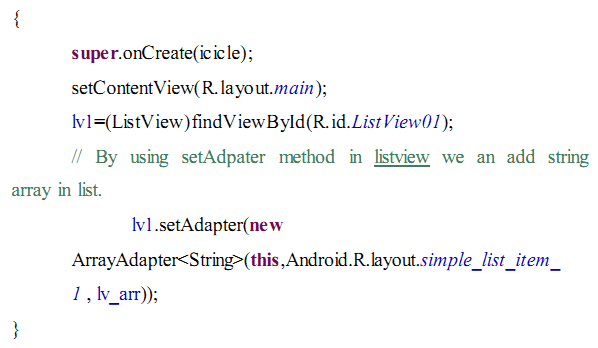
*<ListView Android:id="@+id/ListView01"*

*Android:layout\_width="wrap\_content"*

*Android:layout\_height="wrap\_content" />*

Để tạo các đối tượng trong file Activity thì phải khai báo:

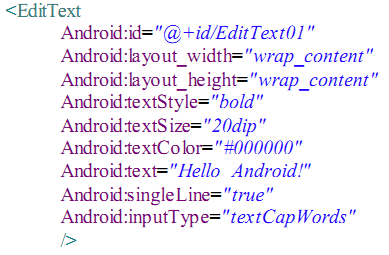


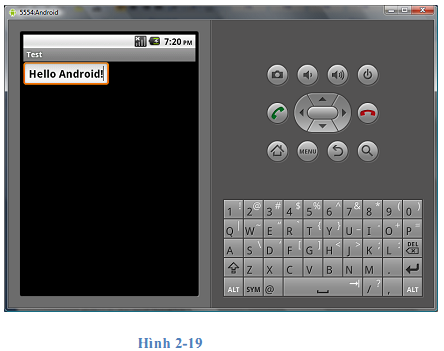




* + - 1. EDITTEXT

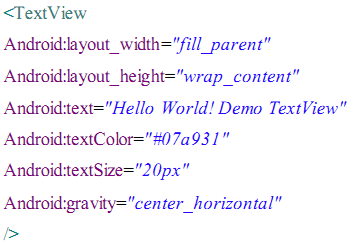
Trong Android đối tượng EditText được sử dụng như một TextField hoặc một TextBox. Giá trị Android:singleLine bằng false, edittext sẽ là một Texbox, ngược lại nó là một Textfield.





* + - 1. TEXTVIEW

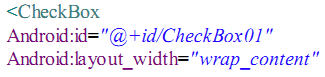
TextView có tác dụng là để hiển thị văn bản.

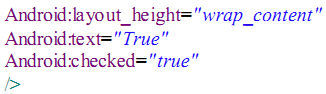


Trong đoạn mã trên, Android:text="" dùng để ghi đoạn text muốn thể hiện, Android:textColor="" để định dạng màu chữ, Android:textSize="" kích cở chữ, Android:gravity="" dùng để canh chỉnh cho đoạn text. Nếu muốn tìm hiểu thêm, hãy gõ Android: sau đó nhấn Crtl+ Space để tìm hiểu thêm về các thuộc tính của nó.

* + - 1. CHECKBOX

Nhận hai giá trị true hoặc false. Đối tượng CheckBox cho phép chọn nhiều item cùng một lúc.

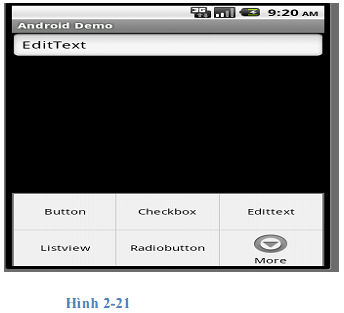






* + - 1. MENUOPTION





* + - 1. CONTEXTMENU

ContextMenu được sử dụng để hiển thị các tuỳ chọn khi người dùng nhấn dài vào một Button trên màn hình. Để tạo một ContextMenu ta cũng có 2 cách giống như tạo MenuOptions ở trên chỉ khác tên phương thức.

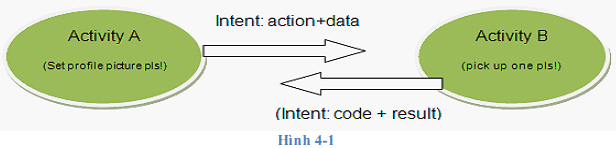
Phương thức: *onCreateContextMenu( ContextMenu menu, View v, ContextMenuInfo menuInfo)* sẽ được gọi và truyền vào 3 tham số là:

* ContextMenu: đối tượng để add các context menu item
* View: Đối tượng nơi mà xảy ra sự kiện
* ContextMenuInfo: Cho biết vị trí xảy ra sự kiện trong ListView.



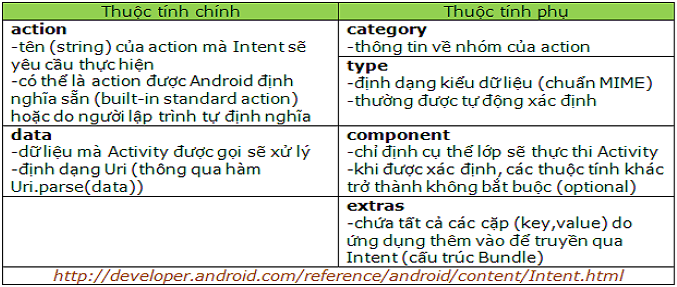
1. LẬP TRÌNH TRÊN ANDROID
   1. GIỚI THIỆU INTENTS

Intent là cầu nối giữa các Activity (ứng dụng Android thường bao gồm nhiều Activity, mỗi Activity hoạt động độc lập với nhau và thực hiện những công việc khác nhau). Intent chính là người đưa thư, giúp chúng ta triệu gọi cũng như truyền các dữ liệu cần thiết để thực hiện một Activity từ một Activity khác.

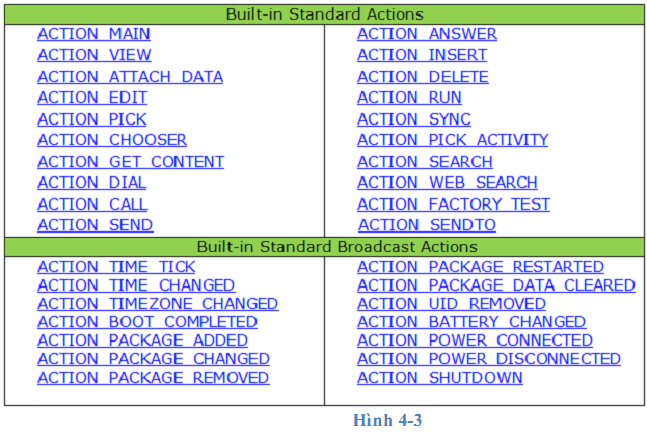


* + 1. Thành phần Intents

Thực ra Intent là một cấu trúc dữ liệu được mô tả trong Android.content.Intent.



Các Action định nghĩa sẵn:



Tự định nghĩa action:

Về nguyên tắc chúng ta có thể đặt tên action của một intent là bất cứ thứ gì theo chuẩn đặt tên thông thường, hay thậm chí dùng luôn hằng action đã định nghĩa sẵn như ACTION\_VIEW (hay “Android.intent.action.VIEW”). Cái tên VIEW thực chất chỉ là một tên gợi tả, chúng ta có thể dùng nó với mục đích thực hiện một activity để … gửi mail! Tuy nhiên điều đó rõ ràng là rất “ngớ ngẩn”. Thay vào đó ta hãy dùng ACTION\_SEND hay ACTION\_SENDTO.

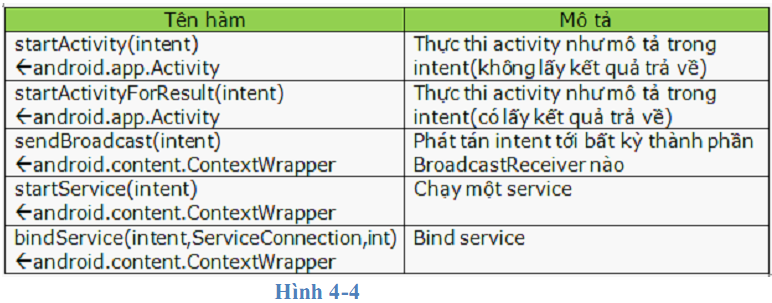
Việc đặt tên action cho intent đúng tên gợi tả còn có một ý nghĩa khác đó là app của chúng ta có thể được triệu gọi từ một app khác. Ví dụ chúng ta viết một app có activity đáp ứng intent ACTION\_SEND và để chia sẻ một bức ảnh lên trang web của chúng ta (giống như ta làm với Facebook, Flickr etc.) Khi đó có thể app của chúng ta sẽ là một lựa chọn chia sẻ ảnh của người dùng điện thoại.

* + 1. Sử dụng Intents khởi động cho Activities

Phổ biến nhất trong Intents là việc liên kết dữ liệu giữa các thành phần ứng dụng, Intents được dùng để bắt đầu, dừng lại và chuyển tiếp giữa các Activities trong ứng dụng.

Để mở một ứng dụng khác trong một ứng dụng có sẵn thì chúng ta gọi startActivity như sau: *startActivity(myIntent);*

Để theo dõi thông tin phản hồi từ phương thức mở ta sử dụng phương thức: *startActivityForResult(Intent)*

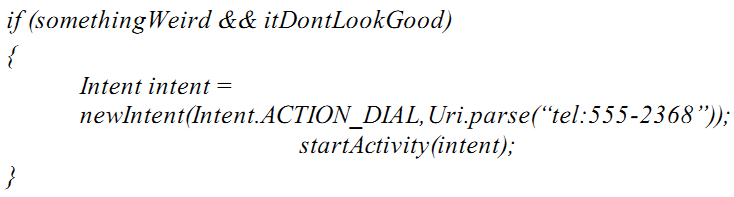


* + 1. Intent không tường minh thực thi Activity

Trong trường hợp này intent không chỉ định một lớp cụ thể mà thay vào đó dùng các dữ liệu khác (action, data, type, etc.) và để hệ thống tự quyết định xem lớp nào (app nào) sẽ thích hợp để đáp ứng intent đó.

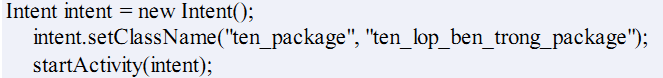
Thông tin action và category của activity trong một app đáp ứng intent đó phải được khai báo trong Manifest của app (AndroidManifest.xml) dưới dạng Intent-filter (tất nhiên nếu chúng ta muốn gọi một built-in action thì ta không cần quan tâm đến vấn đề này).

Ví dụ: chúng ta muốn cho phép người dùng thực hiện cuộc gọi từ một ứng dụng, hơn là thực hiện quay số mới, chúng ta có thể sử dụng một ý định ngầm yêu cầu hành động ("quay số") được thực hiện trên một điện thoại như trong đoạn mã dưới đây:



* + 1. Intent tường minh thực thi Activity

Như đã trình bày ở phần trên, intent có thể dùng thuộc tính phụ component để chỉ định đích danh tên lớp sẽ thực thi Activity. Để thực hiện điều này, lớp Intent cung cấp các hàm đó là *setComponent(ComponentName)* và *setClass(Context, Class)* và *setClassName(Context, String),* *setClassName(String, String)*.



* 1. GIỚI THIỆU ADAPTERS
     1. Một số Adapter

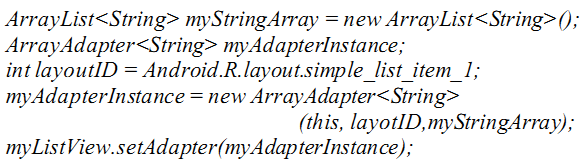
Để sử dụng được Adapter ta phải kế thừa lớp trừu tượng AdapterView, và chúng ta cũng có thể tạo ra một Adapter tùy ý bằng cách này.

ArrayAdapter: là một lớp chung để gắn các Adapter Views vào một mảng đối tượng, theo mặc định thì ArrayAdapter liên kết cá giá trị toString của từng đối tượng ra TextView trong layout, chúng ta có thể thay thế TextView bằng các điều khiển khác phức tạp hơn bằng cách sử dụng kỹ thuật Overriding phương thức getView()

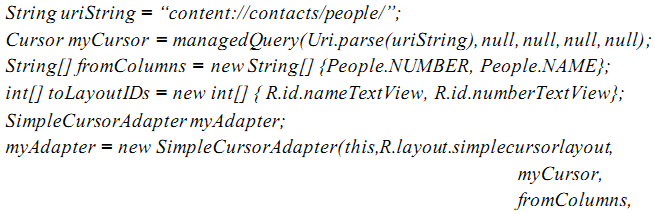
SimpleCursorAdapter : dùng để liên kết các View với con trỏ để trả về câu truy vấn cho Content Provider, nó được định nghĩa trong file XML layout và liên kết các giá trị ràng buộc bên trong mỗi cột kết quả với View trong layout.

* + 1. Sử dụng Adapter hiển thị dữ liệu

Để áp dụng một Adapter đến một class AdapterView ta chỉ cần gọi phương thức setAdapter của View đưa vào trong tập Adapter được minh họa trong đoạn code sau:



SimpleCursorAdapter cho phép chúng ta load dữ liệu từ Cursor đến ListView. SimpleCursorAdapter được tạo ra bằng cách truyền vào các tham số bối cảnh (context) hiện hành, một layout, một Cursor và hai giá trị: tên các cột và một mảng các giá trị chứa dữ liệu cột để hiển thị:



* 1. KỸ THUẬT LƯU TRỮ DỮ LIỆU TRONG ANDROID

Các kỹ thuật duy trì dữ liệu trong Android nhanh chóng, gọn nhẹ, hiệu quả và mạnh mẽ:

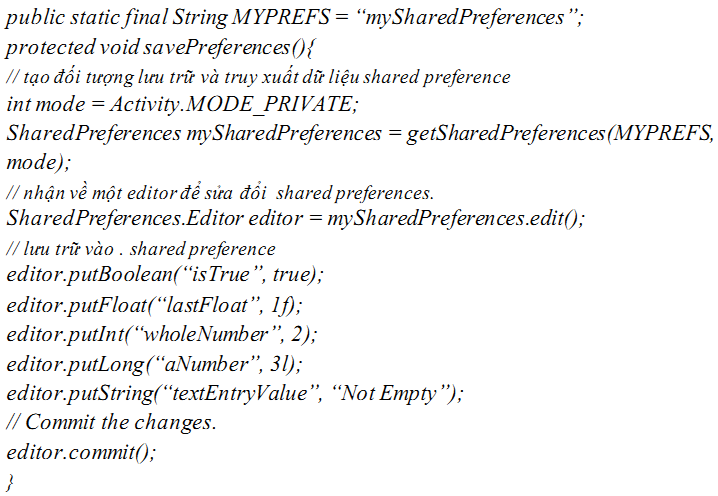
* Shared Preferences: Khi muốn lưu trữ UI state, user preferences hay application setting, chúng ta muốn có một cơ chế gọn nhẹ để lưu trữ một tập hợp các giá trị được biết đến, Shared Preferences cho phép chúng ta lưu trữ nhóm của key/value của dữ liệu gốc.
* Files: Android cho phép tạo và tải các tập tin trên thiết bị di động.
* SQLite Databases: quản lý cấu trúc dữ liệu là cách tốt nhất . Android cung cấp thư viện quan hệ cơ sở dữ liệu SQLite, mỗi ứng dụng có thể tạo ra một cơ sở dữ liệu của mình để nó kiểm soát hoàn toàn trên đó.
* Content Providers: Thay vì theo cơ chế lưu trữ riêng, Content Providers cung cấp nội dung cho phép đưa ra một giao diện để sử dụng và chia sẻ dữ liệu cá nhân. Chúng ta có thể truy cập Content Providers để sử dụng hệ thống được phép.
  1. LƯU TRỮ DỮ LIỆU ỨNG DỤNG MỘT CÁCH ĐƠN GIẢN

Có hai cơ chế để lưu dữ liêu trong Android là lưu theo kiểu name/value hoặc key/value.

* + 1. Tạo và lưu dữ liệu với Share Preferences

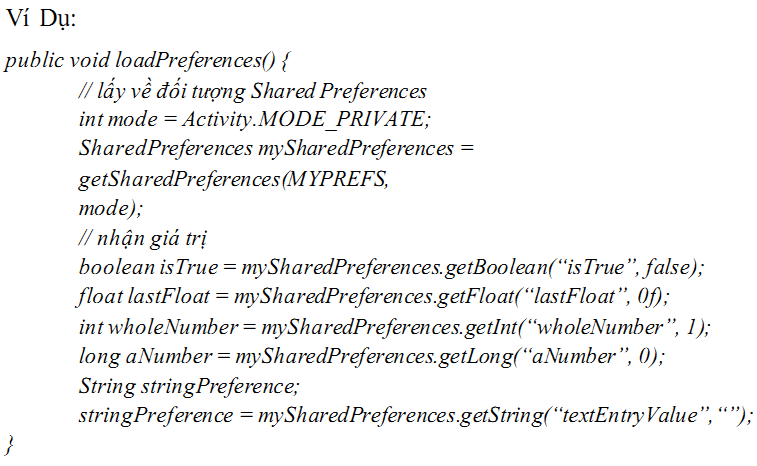
Để tạo hoăc sửa đổi dữ liệu dùng Shared Preferences chúng ta gọi hàm *getSharedPreferences* trong ứng dụng Context, đưa vào tên Shared Preferences muốn thay đổi, Shared Preferences sẽ chia sẻ giữa các thành phần dữ liệu trong ứng dụng hiện sử dụng nó nhưng không có sẵn trong ứng dụng khác.

Để sửa đổi Shared Preferences Chúng ta gọi lớp *Shared Preferences.Editor*, nhận đối tượng Editor bằng cách gọi edit trong đối tượng *ShardePreferences* muốn thay đổi. Để sửa, lưu gọi commit trên Editor, như doạn code sau:



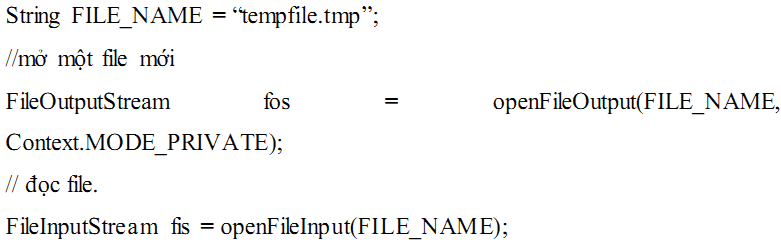
* + 1. Truy xuất Shared Preferences

Để lưu truy cập Shared Preferences thì cũng dùng phương thức getSharedPreferences, đưa vào tên của Shared Preferences mà chúng ta muốn truy cập, dùng phương thức type-safe *get<type>* để trích xuất các giá trị lưu.



* 1. LƯU VÀ ĐỌC CÁC TẬP TIN TRONG ANDROID

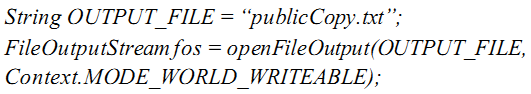
Cũng như các tiêu chuẩn java I/O các lớp và phương thức trong Android cũng cung cấp *openFileInput* và *openFileOutput* để đơn giản hóa việc đọc và viết từ những dòng và từ local file, như đoạn code dưới đây:



Nhưng phương pháp này chỉ hỗ trợ các tập tin trong các thư mục ứng dụng hiện hành.

Nếu tập tin chúng ta chỉ định khi tạo một FileOutputStream không tồn tại, thì Android sẽ tạo nó cho chúng ta, để thêm một tập tin hiện có thì xác định cơ chế Context.MODE\_APPEND.

Theo mặc định, các tập tin được tạo ra bằng cách gọi phương thức openFileOutput để gọi tới một ứng dụng khác, ngoài ra chúng ta có thể chỉ định một Context.MODE\_WORD\_READABLE hoặc Context.MODE\_WORD \_WRITEABLE khi tạo ra một tập tin có sẵn trong các ứng dụng như trong đoạn code sau:



* + 1. Truy xuất các tập tin trong Resources

Nếu ứng dụng của chúng ta đòi hỏi nguồn tài nguyên từ tập tin bên ngoài chúng ta có thể gộp chúng trong gói res/raw trong project của chúng ta.

Để chỉ đọc các tập tin gốc chúng ta gọi phương thức *openRawResource* từ nguồn ứng dụng, để nhận được một InputStream dựa trên các quy định nguồn, đưa vào một filename là tên biến từ lớp R.raw như sau:



* + 1. Các công cụ quản lý tập tin

Android cung cấp một số công cụ quản lý tập tin cơ bản để giúp quản lý hệ thống tập tin.Trong số những tiện ích này nằm trong gói java.io.File .

* deleteFile: xóa tập tin được tạo bởi ứng dụng hiện hành.
* fileList: trả về mảng các tập tin được tạo bởi ứng dụng hiện hành.
  1. CƠ SỞ DỮ LIỆU TRONG ANDROID

Android cung cấp đầy đủ các quan hệ cơ sở dữ liệu thông qua thư viện SQLite mà không áp đặt bất kỳ hạn chế nào. Sử dụng SQLite có thể tạo cơ sở dữ liệu quan hệ độc lập cho mỗi ứng dụng.

Tất cả các cơ sở dữ liệu trong Android được lưu trong thư mục /*data/data/<package\_name>/databases* để chia sẻ cơ sở dữ liệu qua các ứng dụng ta dùng *Context Provider.*

Những phần sau đây sẽ tập trung vào tính thiết thực của việc tạo và quản lý cơ sở dữ liệu SQLite trong Android.

* + 1. Giới thiệu SQLite

SQLite là hệ thống quản lý các quan hệ cơ sở dữ liệu (RDBMS), nó cũng được coi là:

* Mã nguồn mở
* Tiêu chuẩn
* Gọn nhẹ
* Đơn lớp
  + 1. Cursors và Content Values

Content Values là một đối tượng sử dụng để chèn các dòng mới vào bảng cơ sở dữ liệu, mỗi đối tượng Content Values đại diện giá trị cho một dòng.

Truy vấn trong Android được trả về là đối tượng Cursor. Thay vì giải nén và trả lại một bản sao của các giá trị kết quả, Cursors hành động như con trỏ trỏ đến một tập hợp các dữ liệu nằm bên dưới. Cursor quản lý việc kiểm soát vị trí (row) trong tập kết quả truy vấn cơ sở dữ liệu.

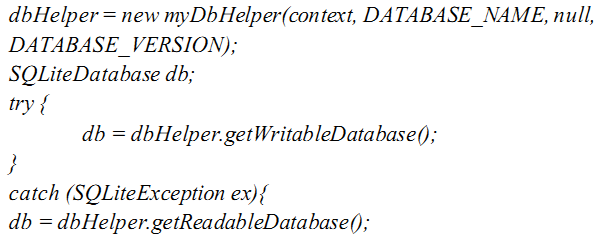
Các lớp con trỏ bao gồm một số chức năng để điều hướng kết quả truy vấn bao gồm, nhưng không giới hạn, những điều sau đây:

* moveToFirst: chuyển con trỏ tới dòng đầu tiên
* moveToNext: chuyển con trỏ tới dòng tiếp theo.
* moveToPrevious: chuyển con trỏ tới dòng duyệt trước đó.
* getCount: trả về số dòng của kết quả truy vấn.
* getColumnIndexOrThrow: trả về chỉ số của cột xác định
* getColumnName: trả về tên của cột có chỉ số truyền vào
* getColumnNames: trả về mảng tên các cột
* moveToPosition: đưa con trỏ đến dòng xác định.
* getPosition: trả về vị trí con trỏ

Khi muốn bắt đầu vòng đời hoạt động của Cursor thì gọi phương thức *:startManagingCursor* vào kết thúc vòng đời con trỏ thì gọi *:stopManagingCursor*.

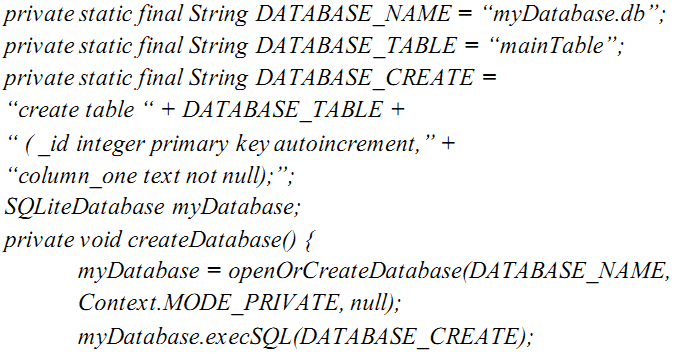
* + 1. Sử dụng SQLiteOpenHelper

SQLiteOpentHelper là một lớp trừu tượng và là môi trường tốt để tạo, truy xuất và cập nhật cở sở dữ liệu. Gọi getReadableDatabases hoặc getWriteableDatabases để mở hoặc trả về dữ liệu trong trường hợp đọc/ghi trong cơ sở dữ liệu như sau:



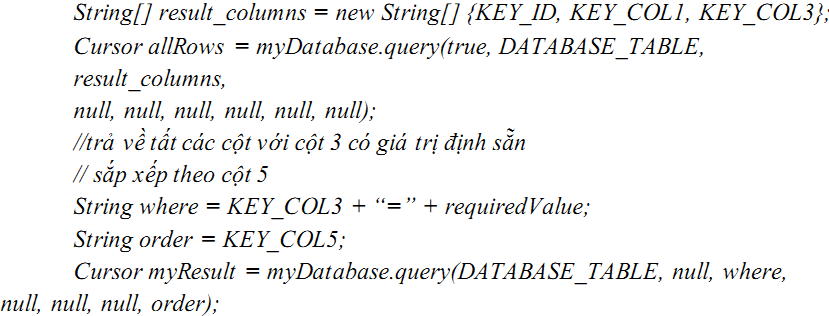
* + 1. Truy xuất và tạo Cơ sở dữ liệu không dùng SQLiteHelper

Chúng ta có thể truy xuất đến cơ sở dữ liệu mà không cần sử dụng lớp SQLiteHelper với phương thức openorCreateDatabases trong ứng dụng Context như sau:



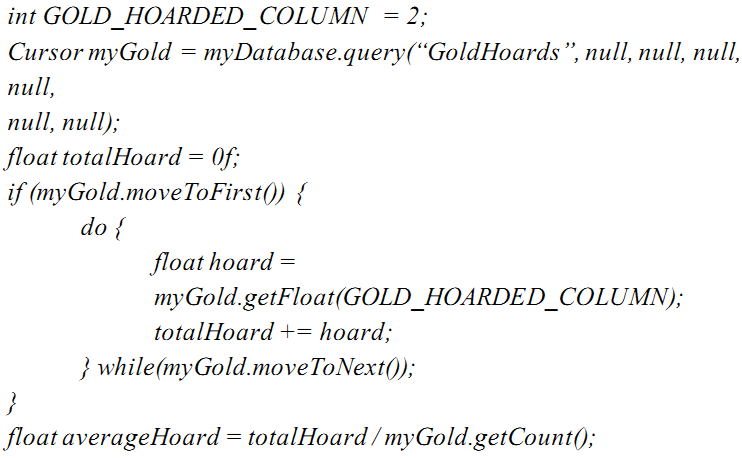
* + 1. Truy vấn cơ sở dữ liệu

Ví Dụ:



* + 1. Lấy kết quả từ Cursors

Ví dụ:

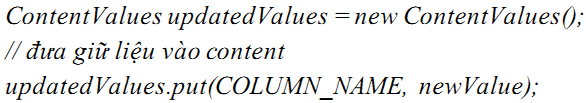


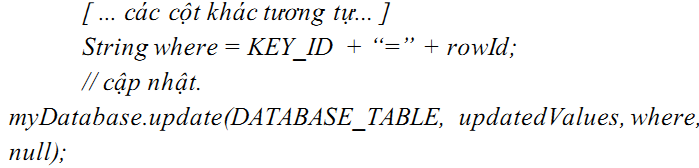
* + 1. Thêm, cập nhật và xóa dòng

Thêm:



Cập nhật:





Xóa dòng:



* + 1. Thao tác trên cơ sở dữ liệu Android

*import Android.content.Context;*

*import Android.database.\*;*

*import Android.database.sqlite.\*;*

*import Android.database.sqlite.SQLiteDatabase.CursorFactory;*

*import Android.util.Log;*

*public class MyDBAdapter {*

*private static final String DATABASE\_NAME = “myDatabase.db”;*

*private static final String DATABASE\_TABLE = “mainTable”;*

*private static final int DATABASE\_VERSION = 1;*

*// khóa chính.*

*public static final String KEY\_ID=”\_id”;*

*// tên và chỉ số cột.*

*public static final String KEY\_NAME=”name”;*

*public static final int NAME\_COLUMN = 1;*

*// TODO: Create public field for each column in your table.*

*// câu SQL tạo bảng*

*private static final String DATABASE\_CREATE = “create table “ + DATABASE\_TABLE + “ (“ + KEY\_ID + “ integer primary key autoincrement, “ + KEY\_NAME + “ text not null);”;*

*// biến Database*

*private SQLiteDatabase db;*

*// áp dụng cho ứng dụng nào*

*private final Context context;*

*//database helper*

*private myDbHelper dbHelper;*

*public MyDBAdapter(Context \_context) {*

*context = \_context;*

*dbHelper = new myDbHelper(context, DATABASE\_NAME, null, DATABASE\_VERSION); }*

*public MyDBAdapter open() throws SQLException {*

*db = dbHelper.getWritableDatabase();*

*return this; }*

*public void close() {*

*db.close(); }*

*public long insertEntry(MyObject \_myObject) {*

*ContentValues contentValues = new ContentValues();*

*return db.insert(DATABASE\_TABLE, null, contentValues); }*

*public boolean removeEntry(long \_rowIndex) {*

*return db.delete(DATABASE\_TABLE, KEY\_ID + “=” + \_rowIndex, null) > 0; }*

*public Cursor getAllEntries () {*

*return db.query(DATABASE\_TABLE, new String[] {KEY\_ID, KEY\_NAME}, null, null, null, null, null); }*

*public MyObject getEntry(long \_rowIndex) {*

*MyObject objectInstance = new MyObject();*

*return objectInstance; }*

*public int updateEntry(long \_rowIndex, MyObject \_myObject) {*

*String where = KEY\_ID + “=” + \_rowIndex;*

*ContentValues contentValues = new ContentValues();*

*return db.update(DATABASE\_TABLE, contentValues, where, null);}*

*private static class myDbHelper extends SQLiteOpenHelper {*

*public myDbHelper(Context context, String name, CursorFactory factory, int version) {*

*super(context, name, factory, version); }*

*// Called when no database exists in*

*@Override*

*public void onCreate(SQLiteDatabase \_db) {*

*\_db.execSQL(DATABASE\_CREATE); }*

*//gọi khi database có phiên bản mới*

*@Override*

*public void onUpgrade(SQLiteDatabase \_db, int \_oldVersion,int \_newVersion) {*

*// Log the version upgrade.*

*Log.w(“TaskDBAdapter”, “Upgrading from version “ + \_oldVersion + “ to “ +*

*\_newVersion + “, which will destroy all old data”);*

*\_db.execSQL(“DROP TABLE IF EXISTS “ + DATABASE\_TABLE);*

*//tạo csdl mới*

*onCreate(\_db);*

*} } }*

* + 1. Giới thiệu Content Providers

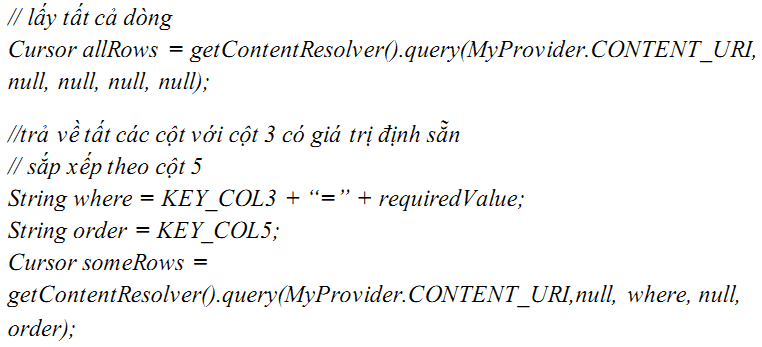
Sử dụng Content Provider bằng lớp ContenResolver.

*ContentResolver cr = getContentResolver();*

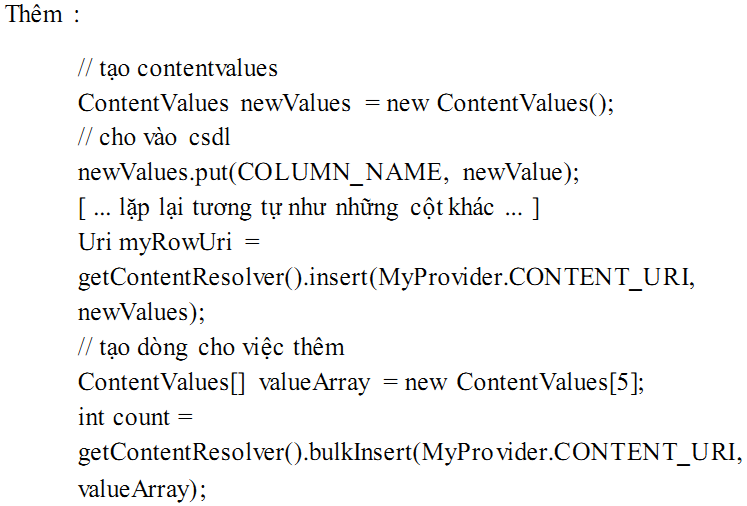
* + - 1. Truy vấn trong Content

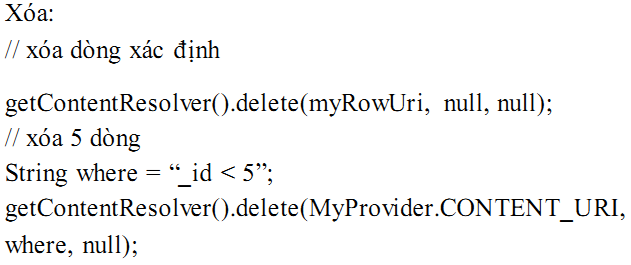
Như trong cơ sở dữ liệu, kết quả tìm kiếm sẽ được trả về Cursors trong một tập kết quả. Chúng ta có thể trích xuất các giá trị từ Cursor bằng cách sử dụng các kỹ thuật được mô tả trước đó trong phần cơ sở dữ liệu trên “Lấy kết quả từ Cursors”:

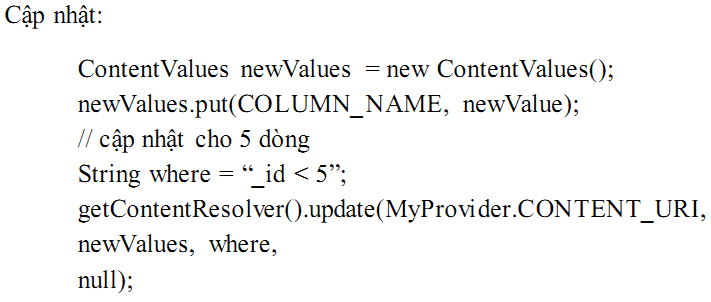
Content Provider có cách truy vấn hình thức rất giống với các truy vấn cơ sở dữ liệu. Sử dụng phương pháp query trên đối tượng ContentResolver.



* + - 1. Thêm, cập nhật và xóa dòng



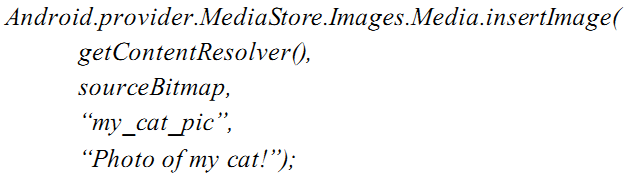




* + - 1. Sử dụng Media Store Provider

Các Android Media Store quản lý tập tin hình ảnh, kho video, âm thanh. Bất cứ khi nào chúng ta thêm một tập tin đa phương tiện mới cho hệ thống tập tin Android là có thể bổ sung vào Media Store để dùng cho các ứng dụng khác.

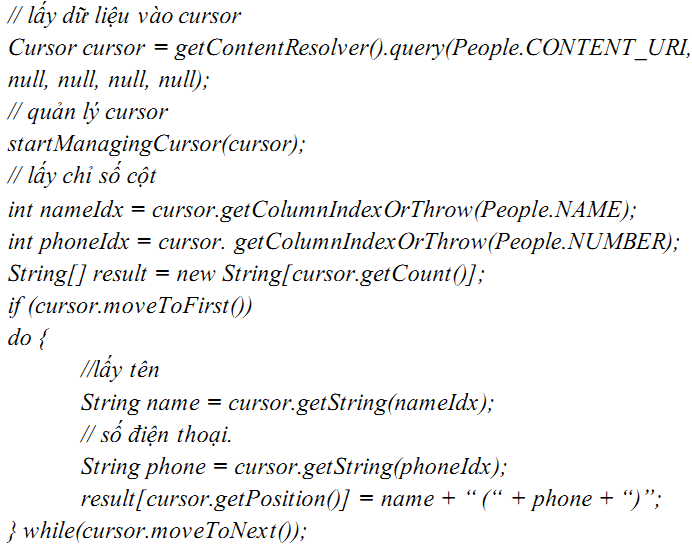
Lớp MediaStore bao gồm một số phương pháp tiện lợi để đơn giản hóa các tập tin chèn vào Media Store. Ví dụ, đoạn mã sau đây cho thấy làm thế nào để chèn một hình ảnh trực tiếp vào Media Store:



* + - 1. Sử dụng Contacts Provider

Truy cập vào Content Provider đặc biệt mạnh mẽ trên một thiết bị truyền thông. Android đã phơi bày tất cả các thông tin có sẵn từ các databases địa chỉ liên hệ đến bất kỳ ứng dụng nào qua READ\_CONTACTS.

Trong ví dụ sau, Activity’s trỏ đến tất cả mọi người trong cơ sở dữ liệu liên lạc, tạo ra một mảng các chuỗi chứa tên của mỗi liên lạc và số điện thoại.



* + - 1. Đăng ký dử dụng Provider

Sử dụng thẻ *authorities* để xác định trong file XML như sau:



* 1. MAPS, GEOCODING, LOCATION BASED SERVICES
     1. Sử dụng dịch vụ định vị

Location Based Services (LBS) là một thuật ngữ chung dùng để mô tả sự khác nhau về kỹ thuật, và dùng để tìm vị trí thiết bị hiện hành. Hai yếu tố chính của LBS là:

* LocationMangager: quản lý kết nối với dịch vụ định vị trong Android.
* LocationProvider: là một lớp trừu tượng cơ bản của các công nghệ tìm kiếm vị trí khác nhau để xác định vị trí thiết bị.

Sử dụng Location Manager ta có thể :

* Xác định vị trí.
* Theo dõi sự chuyển động.
* Đặt trạng thái báo động cho việc phát hiện một sự chuyển động vào và ra khỏi một khu vực xác định.
  + 1. Cài đặt trên môi trường giả lập với Test Providers

Dịch vụ định vị (LBS) phụ thuộc vào các thiết bị phần cứng cho việc tìm kiếm các vị trí hiện tại. Khi phát triển và kiểm thử với các môi trường giả lập, thì phần cứng là ảo và chúng ta có thể ở trong những vị trí khá giống nhau.

Bù lại Android bao gồm các móc nối cho phép chúng ta móc nối với

location Provider để thử nghiệm với ứng dụng location based.

* + 1. Cập nhật vị trí với Emulator Location Providers

Dùng Location Controls có sẵn từ DDMS trong Eclipse như hình dưới đây để thay đổi trực tiếp vị trí trong thử nghiệm GPS\_PROVIDER.



Trong hình trên có các tab Manual, KML và GPX

Sử dung tab Manual chúng ta có thể xác định vĩ độ/ kinh độ.

Sử dụng tab KML và GPX cho phép chúng ta tải các tập tin KML

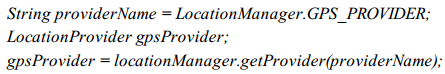
(Keyhole Markup Language) và GPX (GPS Exchange Format) tương ứng.

Lưu ý: vị trí GPS sẽ trả về từ getLastKnownLocation sẽ không thay đổi trừ khi có ít nhất một ứng dụng yêu cầu cập nhật vị trí.

* + 1. Chọn một Location Provider

Tùy thuộc vào thiết bị mà có những kỹ thuật riêng, Android có thể dùng để xác định vị trí hiện tại, mỗi kỹ thuật, mỗi Location Provider sẽ cung cấp những khả năng khác nhau như xác định độ cao, tốc độ, sự chính xác hoặc các thông tin liên quan.

Để có được đối tượng LocationProvider ta gọi *getProvider* và truyền vào tên Provider. Ví dụ dưới đây thì Provider là GPS\_PROVIDER



* + 1. Tìm một Providers có sẵn

Lớp LocationManager bao gồm các chuỗi hằng số và trả về tên của nhà cung cấp cho hai Location Provider phổ biến nhất.



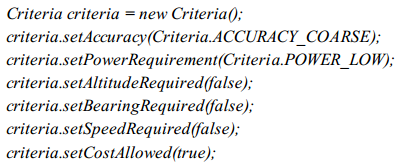
Để có danh sách tất cả tên của nhà cung cấp trên thiết bị ta gọi

getProvider, sử dụng Boolean để biết nếu chúng ta muốn tất cả, hoặc chỉ lấy một Provider đầu tiên.

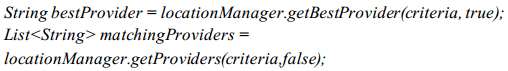


* + 1. Tìm kiếm một Provider theo tiêu chí

Sử dụng lớp Criteria để yêu cầu nhà cung cấp xử lý chính xác những số liệu có sẵn như: vĩ độ và kinh độ, tốc độ, độ cao, chi phí và yêu cầu về năng lượng điện.



Chúng ta có thể dùng getBestProvider là sự kết hợp tốt nhất trả để về cho Location Provider hoặc dùng getProvider, sử dụng getBestProvider nhà cung cấp sẽ trả lại các tiêu chí tốt nhất, sử dụng Boolean cho phép hạn chế những kết quả từ nhà cung cấp đang được hoạt động.

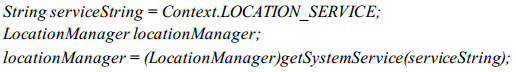


* + 1. Tìm một địa chỉ (Finding Your Location)

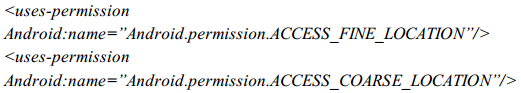
Mục đích của dịch vụ định vị là tìm ra vị trí vật lý của thiết bị.

Để truy cập vào Location Manager thực hiện các yêu cầu

LOCATION\_SERVICES ta sử dụng phương thức getSystemService như sau:



Trước khi sử dụng Location Manager thì cần thêm một hoăc nhiều thẻ uses-permission trong file mainfest để truy cập vào các phần cứng của LBS trong đó bao gồm cả fine và coarse như sau:



Chúng ta có thể tìm thấy vị trí bằng cách xác định một Location Provider sử dụng phương thức getKnowLocation và đưa vào tên của Location Provider.

Ví dụ:



* + 1. Theo dõi sự di chuyển (Tracking Movement)

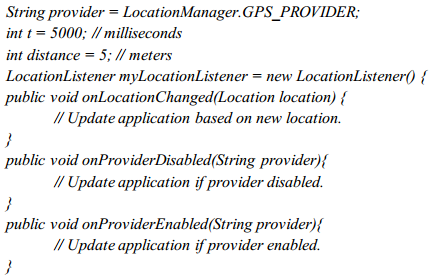
Sử dụng phương thức requestLocationUpdate để cập nhật bất kỳ một vị trí, khi vị trí hiện tại thay đổi chúng ta sử dụng một LocationListener.

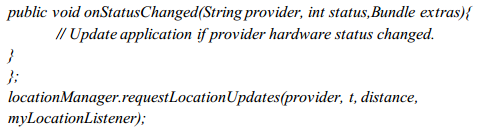
LocationListener thực hiện và đăng ký để nhận thông tin.

LocationListener rất linh hoạt và nhiều tính năng cho phép chọn nhiều loại sự kiện về vị trí dựa trên đặc tính khác nhau.

Phương thức requestLocationUpdate chấp hận một Location Provider hoặc Criteria để xác định nhà cung cấp được sử dụng.

Đoạn code sau sẽ thể hiện một yêu cầu cập nhật thường xuyên dựa trên mức tối thiểu về thời gian và khoảng cách.





* + 1. Sử dụng Goecoder

Geocoding cho phép chuyển giữa các địa chỉ và các mô tả khác vào kinh độ/ vĩ độ của bản đồ tọa độ. Điều này cung cấp cho chúng ta một khung cảnh giúp dễ nhận biết các vị trí và tọa độ được sử dụng trong các dịch vụ dựa trên vị trí và hoạt động dựa trên bản đồ. Để thực hiện việc này lớp Geocoder thực hiện một cuộc gọi mạng (tự động) cho một dịch vụ web.

Lớp Geocoder cung cấp hai chức năng truy cập mã địa lý:

* Forward Geocoding: tìm vĩ độ và kinh độ của đia chỉ.
* Reverse Geocoding: chuyển địa chỉ ra vĩ độ, kinh độ.

Cách tạo một Geocoder :

*Geocoder geocoder = new Geocoder(getApplicationContext(),*

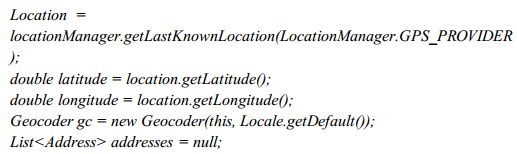
*Locale.getDefault());*

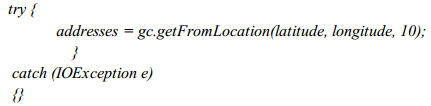
Cả hai chức năng mã hoá địa lý trả về một danh sách các đối tượng Address. Mỗi danh sách có thể chứa nhiều kết quả.

* + 1. Reverse Geocoding

Trả về địa chỉ khi được cung cấp kinh độ, vĩ độ.

Để thực hiện việc tra cứu chuyển đổi chúng ta đưa vào một vĩ độ và kinh độ đến Geocoder và phương thức getFromLocation trả về một danh sách địa chỉ phù hợp, nếu Geocoder không giải quyết được địa chỉ này với các quy định thì nó sẽ trả về null.



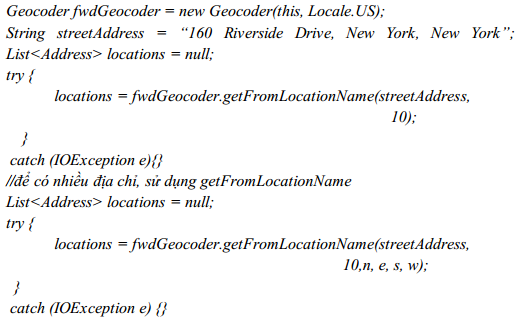


* + 1. Forward Geocoding

Để tra cứu Forward Geocoding, gọi getFromLocationName trên trường Geocoder. Đưa vị trí tạo độ mà chúng ta muốn và số lượng tối đa của các kết quả trả lại, chúng ta sẽ nhận được các địa chỉ thích hợp, như thể hiện trong đoạn code dưới đây:



Locale cung cấp bối cảnh địa lý để giải thích các yêu cầu tìm kiếm của chúng ta như tên của một vị trí có thể tồn tại trong nhiều khu vực. Nếu có thể thì ta xem xét lựa chọn một Locale ở khu vực để tránh sự nhập nhằng tên, như thể hiện trong đoạn code sau:



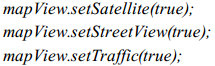
Hình sau cho thấy việc dùng Geocoder để chuyển một địa chỉ kiểu String vào một đối tượng cung cấp vĩ độ và kinh độ.



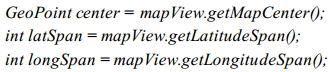
* + 1. Dùng MapView

MapView là một control thu nhỏ của nhiều Google Maps API trong hình thức View cho các ứng dụng Android. MapView hiển thị giao diện của bản đồ, nó có được sử dụng qua mạng như bản đồ là sự di chuyển và thu nhỏ, giống như phiên bản web của Google Maps.

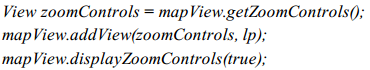
Nhiều khái niệm tiêu chuẩn Google Map API cũng có mặt trong Android thông qua các MapView, ví dụ MapView hỗ trợ chế độ vệ tinh, chế độ xem đường phố, và chế độ giao thông ta khai báo như sau:



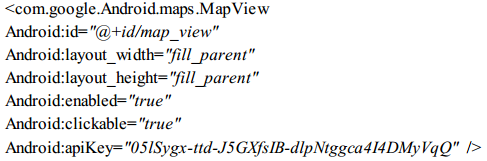
Chúng ta cũng có thể truy ập vào các Map View để tìm ra kinh độ hiện hành và độ phóng đại, cũng như các điểm trung tâm và hiện đang nhìn thấy và khoảng vĩ độ (ở dạng thập phân).



Chúng ta cũng có thể hiển thị bản đồ điều khiển zoom một cách tùy ý.



MapView là một ViewGroup, hãy khai báo nó trong XML.



Chú ý: Android:apiKey ở đây chúng ta phải đăng ký qua Google, nó hoàn toàn miễn phí.

* + 1. Dùng MapActivity

MapActivity là một lớp cơ bản.Lớp MapActivity xử lý vòng đời ứng dụng và quản lý nền tảng cần thiết cho việc hiển thị bản đồ, chúng ta chỉ có thể sử dụng một MapView trong MapAc tivity.

Để sử dụng bản đồ trong ứng dụng chúng ta cần phải tạo một Activity mới kế thừa từ MapActivity và cần thêm vào trong file XML, trong thẻ <Aplication>:

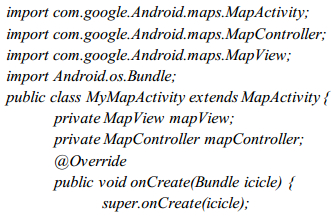
*<uses-library Android:name=”com.google.Android.maps”/>*

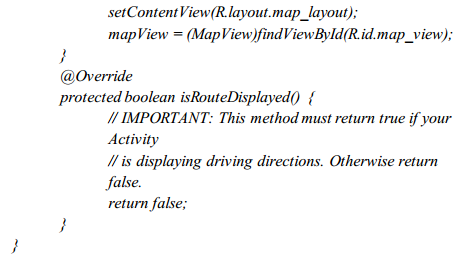
MapView điều khiển chỉ có thể được sử dụng trong một Activity mở

rộng MapActivity. Overriding phương thức onCreate đặt ra các màn hình bao gồm một MapView, và ghi đè lên isRouteDisplayed để trả về true nếu Activity sẽ được hiển thị thông tin định tuyến.

Đoạn code sau sẽ trình bày cách khởi tạo mới lớp bản đồ cơ sở

MyMapActivity :



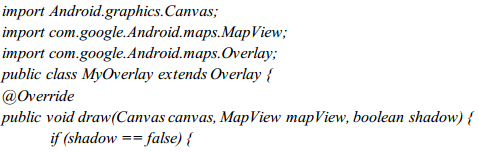


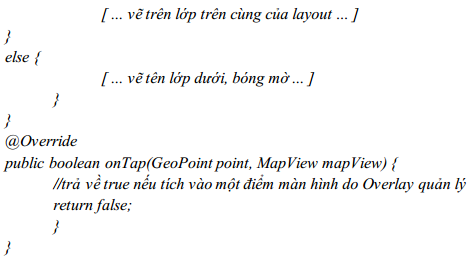


* + 1. Sử dụng Overlay

Overlay là một cách để thêm các chú thích và xử lý nhấn vào MapView.

Để thêm một lớp mới Overlay chúng ta tạo một lớp mới kế thừa từ lớp Overlay, ghi đè các phương thức vẽ để vẽ các chú thích mà chúng ta muốn thêm và ghi đè lên *onTap* để phản lại cho người dùng khi click chuột như đoạn code sau:





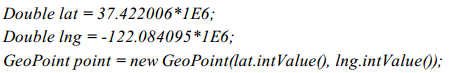
* + 1. Dùng MapController

Dùng MapContrller để quay và phóng to một MapView, dùng getController như đoạn code sau:

*MapController mapController = myMapView.getController();*

Bản đồ các địa điểm trong các lớp bản đồ Android được đại diện bởi đối tượng GeoPoint, trong đó vĩ độ và kinh độ đo bằng microdegrees.

Trước khi chúng ta có thể sử dụng các giá trị vĩ độ và kinh độ lưu trữ trong các đối tượng Location được sử dụng bởi các location-based services, chúng ta cần phải chuyển đổi chúng sang microdegrees và lưu trữ chúng như GeoPoints, như trong đoạn mã sau đây:



Khi muốn biết điểm và phóng to hình trong MapView ta sử dụng phương thức setCenter và setZoom có sẵn trong MapCotroller của MapView như đoạn code dưới đây:

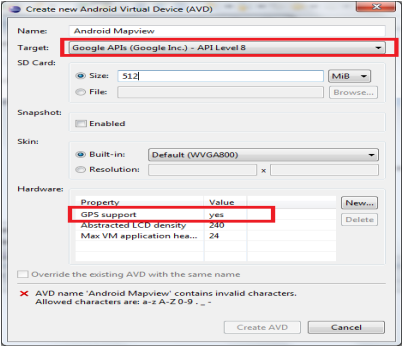


Phương thức setCenter sẽ nhảy tới một vị trí mới ta sử dụng như sau:



* + 1. Chú ý khi tạo một ứng dụng có sử dụng MapView

Không giống như cách tạo và chạy ứng dụng như các chương trước, có một sự khác biệt nho nhỏ khi chạy ứng dụng MapView đó là việc chọn Emulator hay chọn các Emulator có hỗ trợ Google APIs.

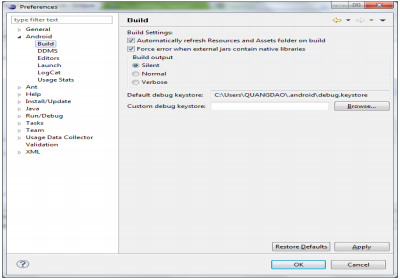


Hình: Tạo emulator hỗ trợ Geocoder và MapView

Một chú ý nữa đó là phải có < Android:apiKey> khi dùng MapView, sau đây là cách lấy key (chúng ta phải kết nối Internet) :

* Lấy debug.keystore

Vào Windows > Prefs > Android > Build để lấy đường dẫn như bên dưới



* Lấy key:

Vào cmd gõ:

*keytool -list -alias Androiddebugkey -keystore <đường dẫn file*

*keystore> -storepass Android -keypass Android*



Vào link http://code.google.com/Android/maps-api-signup.html dán

MD5 fingerprint code vào và click Generate API key, site sẽ tự sinh cho chúng ta 1 key.

* 1. LÀM VIỆC TRÊN BACKGROUND
     1. Giới Thiệu Services

Không giống như các hoạt động khác, Services trình bày một giao diện đồ họa phong phú cho người dùng. Services chạy trên nền cập nhật Content Providers, đóng Intent và kích hoạt thông báo, Services là những cách để thực hiện xử lý hoàn hảo hoặc xử lý sự kiện thường xuyên ngay cả khi ứng dụng của chúng ta không nhìn thấy hoạt động của services, không hoạt động hoặc bị tắt.

Tuy không có giao diện trực quan nhưng Serviecs vẫn có thể được bắt đầu, dừng lại và kiểm soát từ các thành phần ứng dụng khác bao gồm Services, Activities và Broadcast Receivers. Nếu ứng dụng của chúng ta thường xuyên hoặc liên tục thực hiện các hoạt động mà không phụ thuộc trực tiếp vào dữ liệu người dùng đưa vào (input), Sevices có thể là câu trả lời.

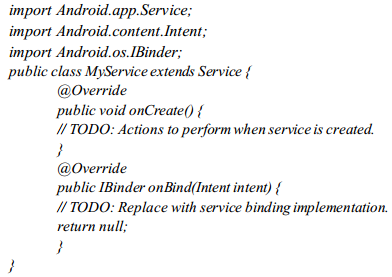
Máy nghe nhạc MP3, giám sát tọa độ là những ví dụ của ứng dụng có thểmchạy và cập nhật mà không có một thành phần tương tác trực quan có thể nhìn thấy.

Các Services được thiết kế để chạy nền, vì vậy cần phải được bắt đầu, dừng lại, và kiểm soát bởi các thành phần ứng dụng khác.

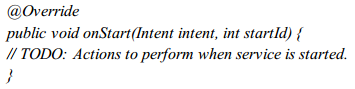
Trong các phần sau đây, chúng ta sẽ học cách để tạo ra một Service mới và làm thế nào để bắt đầu và kết thúc nó bằng cách sử dụng phương thức startServices. Sau đó sẽ học cách để ràng buộc một Services cho một Activity, cung cấp một giao diện phong phú hơn cho tương tác.

* + 1. Tạo Service

Để xác định một Services ta tạo ra một lớp mới kế thừa tử lớp Service cơ sở. Chúng ta cần ghi đè lên *onBind* và *onCreate* như trình bày sau đây:



Trong hầu hết các trường hợp chúng ta nên ghi đè lên **onStart**. Điều này sẽ được gọi bất kỳ khi nào Services được bắt đầu khi gọi đến **startService,** dưới đây là thể hiện phương thức ghi đè lên **onStart** như sau:



Một khi chúng ta đã xây dựng một Service mới chúng ta phải đăng ký trong tập tin mainifest trong tab <service> như sau:



* + 1. Khởi chạy, điều khiển và tương tác với Sercice

Để khởi chạy một Service chúng ta gọi **startService*.***

Nếu Service yêu cầu quyền truy cập mà ứng dụng của chúng ta không có thì lời gọi này sẽ bị chuyển vào *SecurityException* như sau:

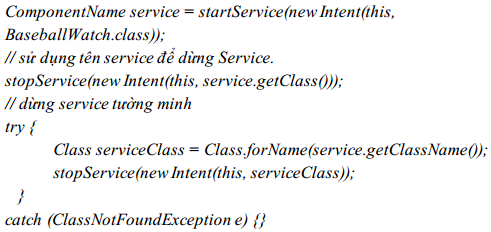
*// khởi tạo không tường minh*

*startService(new Intent(MyService.MY\_ACTION));*

*// khởi tạo tường minh*

*startService(new Inte nt(this, MyService.class));*

Để dừng một Service ta gọi *stopService*, đưa vào một Intent để xác định các Service ngăn chặn.



Nếu **startService** được gọi thì Service đó đang được chạy, phương thức **onStart** của Service được thực hiện một lần nữa. Các cuộc gọi đến **startService** sẽ không lồng lên nhau do đó có một cuộc gọi duy nhất, để chấm dứt thì sử dụng **stopService** không kể là bao nhiêu lần **startService** đã được gọi.

* + 1. Kết nối Activities với Services

Khi một hoạt động liên kết với một Service thì bản thân nó duy trì một tham chiếu đến Service đó, nó cho phép gọi một phương thức trên Service đang chạy.

Liên kết này có sẵn cho các hoạt động, sẽ được thừa hưởng tất cả từ giao diện với một Service. Để hỗ trợ các ràng buộc cho một Service thì thực hiện phương thức **onBind** như trong ví dụ đơn giản sau đây:

*private final IBinder binder = new MyBinder();*

*@Override*

*public IBinder onBind(Intent intent) {*

*return binder;*

*}*

*public class MyBinder extends Binder {*

*MyService getService() {*

*return MyService.this;*

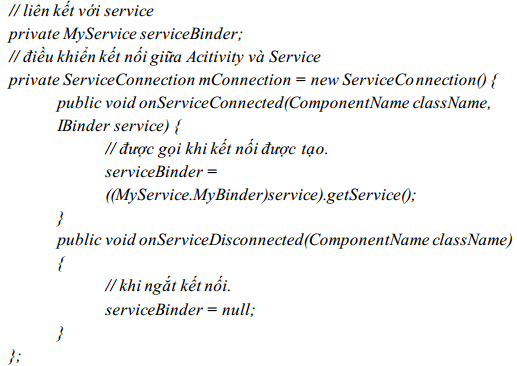
*}*

*Tìm hiểu lập trình trên Android*

*SVTH: Phạm Quang Đạo – Nguyễn Hoàng Nam 73*

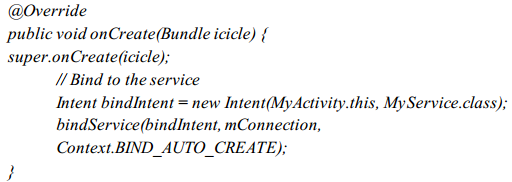
*}*

Sự kết nối giữa các Service và các hoạt động được thể hiện như một **Serviceconnection**. Ta cần phải thực hiện một **ServiceConnection** mới, ghi đè lên phương thức **onServiceConnection** và **onServiceDisconnected** để có thể tham chiếu đến một Service kết nối đã được thành lập.



Để gọi Services cần truyền vào một Intent trong phương thức

*bindService*:



Liên lạc với một Service trong một ứng dụng khác bằng cách sử dụng Broadcast Intent hay biến Bundle trong Intent được sử dụng khi khởi động Service.

* + 1. Giới thiệu Thông báo trong Android

Thông báo là một cách cho các ứng dụng của chúng ta để cảnh báo đối với người sử dụng, mà không cần sử dụng một hành động. Thông báo được xử lý bởi các Manager Notification, nó bao gồm các khả năng:

* Tạo một biểu tượng mới trên thanh trạng thái.
* Hiển thị thêm thông tin trong cửa sổ mở rộng của thanh trạng thái.
* Đèn Flash/LEDs.
* Điện thoại rung.
* Âm thanh báo động (nhạc chuông, âm thanh thông tin lưu trữ).

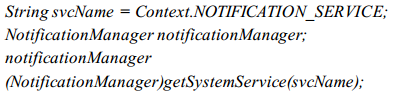
Thông báo có thể tồn tại qua sự lặp lại liên tiếp hoặc bằng các sử dụng một biểu tượng trên thanh trạng thái.Tình trạng các biểu tượng có thể được cập nhật thường xuyên hoặc mở rộng để hiển thị thêm thông tin sử dụng tình trạng cửa sổ như trong hình dưới.



* + 1. Giới thiệu Notification Manager

Notification Manager là một hệ thống dịch vụ được sử dụng để xử lý các thông báo, để tham chiếu đến nó bằng cách sử dụng phương thức:

getSystemService như thể hiện trong đoạn code sau:

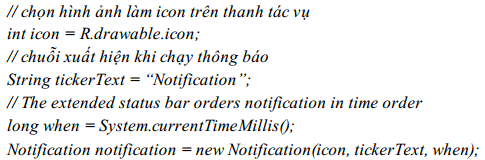


Sử dụng Manager thông báo, chúng ta có thể kích hoạt thông báo mới, sửa đổi những cái hiện có, hoặc loại bỏ những cái không còn cần thiết hoặc không mong muốn.

* + 1. Tạo thông báo

Tạo và cấu hình một thông báo mới được thực hiện trong ba phần:

Trước tiên, tạo một đối tượng thông báo mới đưa vào biểu tượng để hiển thị trên thanh trạng thái cùng với các thanh trạng thái tickertext và thời gian thông báo như trong đoạn code sau:



Các ticker-text sẽ di chuyển dọc theo các thanh trạng thái khi các thông báo được phát ra.

Thứ hai, cấu hình sự xuất hiện của các thông báo trong cửa sổ trạng thái mở rộng sử dụng phương pháp setLatestEventInfo. Điều này mở rộng cửa sổ trạng thái hiển thị biểu tượng và thời gian quy định trong xây dựng và cũng cho thấy tiêu đề và một chuỗi các chi tiết. Thông báo thường xuất hiện cho một yêu cầu một hành động hoặc sự chú ý, vì vậy chúng ta có thể chỉ định một PendingIntent sẽ thoát nếu người dùng nhấp vào mục thông báo.

*Context context = getApplicationContext();*

*// chuỗi hiện khi mở rộngcửa sổ thông báo*

*String expandedText = “Extended status text”;*

*// tiêu đề của*

*String expan dedTitle = “Notification Title”;*

*// Intent để khởi chạy activity khi cửa sổ thông báo mở rộng được click*

*Intent intent = new Intent(this, MyActivity.class);*

*PendingInten tlaunchIntent= PendingIntent.getActivity(context,*

*0,intent, 0);*

*notification.setLatest EventInfo(context,expandedTitle,expandedTe*

*xt,*

*launchIntent);*

Chúng ta cũng có thể sử dụng thuộc tính number để hiển thị số sự kiện cho một biểu tượng trên thanh trạng thái được thể hiện. Thiết lập giá trị này lớn hơn 1, tất cả sự thay đổi của một thông báo chúng ta không cần phải kích hoạt nó để áp dụng sự thay đổi, để loại bỏ các lớp phủ (các thông báo phủ lên nhau), ta thiết lập giá trị là 0 hoặc -1

*notification.number++;*

Cuối cùng, chúng ta có thể cải thiện một Thông báo bằng cách sử dụng các thuộc tính khác nhau trên các đối tượng thông báo chẳng hạn thiết bị đèn LED, rung điện thoại, và âm thanh (nhạc, chuông).

* + 1. Kích hoạt thông báo

Để bật một thông báo và đưa vào trong phương thức notify trong đối

tượng NotificationManager, một số nguyên ID, như trong đoạn code sau:

*int notificationRef = 1;*

*notificationManager.notify(notificationRef, notification);*

Chúng ta có thể sử dụng các ID để hủy bỏ thông báo bằng cách gọi

phương thức *Cancel* trên NotificationManager, như thể hiện sau:

*notificationMa nager.cancel(notificationRef);*

* 1. SỬ DỤNG CÁC MEDIA APIS

Các công nghệ hiện đại ngày nay có thể cạnh tranh với điện thoại di động có mặt ở khắp mọi nơi là máy nghe nhạc và các phương tiện xách tay kỹ thuật số. Kết quả thăm dò thị trường cho thấy thiết bị điện thoại di động là một sự quan tâm đáng kể đối với nhiều người tiêu dùng.

Android đưa ra nhiều thư viện cho ứng dụng và cung cấp các chức năng đa phương tiện gồm cả ghi âm và video, âm thanh và hình ảnh được lưu trữ tại cục bộ trong một ứng dụng.

Android hỗ trợ các định dang đa phương tiện sau đây: JPEG, PNG, OGG, Mpeg 4, 3GPP, MP3, Bitmap

* + 1. Chơi nhạc

Đa phương tiện trong Android được xử lý bởi lớp MediaPlayer. Chúng ta có thể xem lại các thông tin được lưu trữ trong ứng dụng, đia chỉ của tập tin, hoặc từ mạng URI.

Để chơi nhạc, ta tạo mới một Media Player, và gán cho nó đường dẫn đến tập tin nhạc để chơi bằng các sử dụng phương thức sedDataSource. Trước khi chúng ta có thể bắt đầu chơi nhạc, chúng ta cần phải chuẩn bị như trong đoạn mã sau:



Ngoài ra, tạo một phương thức tĩnh (static) làm việc như shortcuts, chấp nhận các đường dẫn đến file nhạc như một tham số như trong đoạn mã sau:

*MediaPlayer mpRes = MediaPlayer.create(context, R.raw.my\_sound);*

Một *Media Player* được gọi để bắt đầu chơi như thể hiện sau đây:

*mpRes.start();*

*mpFile.start();*

Media Player bao gồm các chức năng dừng lại, tạm dừng và phương thức tìm kiếm cũng như phương thức tìm vị trí, thời gian và kích thước hình ảnh của các thông tin liên quan.

Để lặp vòng hoặc phát lặp lại ta sử dụng phương thức setLooping. Một khi chúng ta không sử dụng Media Player, ta gọi release để giải phóng tài nguyên liên quan như trình bày sau:

*mpRes.release();*

*mpFile.release();*

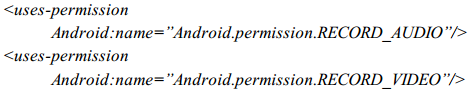
* + 1. Ghi âm

Ghi âm được xử lý bởi lớp *MediaRecorder* để ghi âm thanh hoặc video, tạo một đối tượng *MediaRecorder* như trong đoạn code sau:

*MediaRecorder mediaRecorder = new MediaRecorder()*

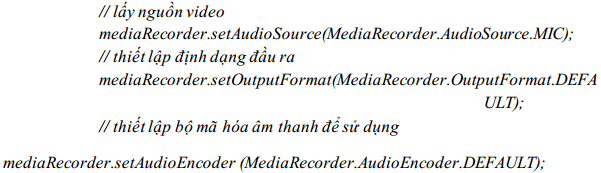
Trước khi chúng ta có thể ghi âm bất kỳ media trong Android, ứng dụng của chúng ta cần được cấp phép RECORD\_AUDIO hoặc RECORD\_VIDEO.

Thêm một thẻ uses-permission trong ứng dụng manifest như sau:



Máy ghi âm (Media Reco rder) có thể sử dụng để cấu hình các video và quay phim (bao gồm cả camera và microphone), định dạng đầu ra, kích thước video, tỷ lệ khung hình, video và thu âm video để sử dụng.

Các đoạn code sau đây cho thấy làm thế nào để cấu hình một máy ghi âm để ghi lại âm thanh từ micro bằng cách sử dụng định dạng mặc định và mã hóa.



Khi chúng ta xác định đầu vào và định dạng đầu ra, chỉ định một tập tin để lưu trữ các media sử dụng phương thức *setOutputFile* như dưới đây:

*mediaRecorder.setOutputFile(“myoutputfile.mp4”);*

Để bắt đầu ghi âm, ta gọi *prepare* bằng phương thức *start* như sau:

*mediaRecorder.prepare();*

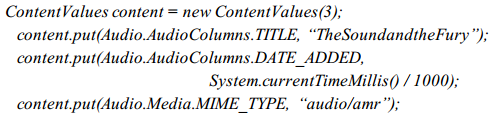
*mediaRecorder.start();*

Khi hoàn thành ta gọi *stop* để kết thúc, tiếp theo là *release* để giải phóng nguồn tài nguyên máy ghi âm.

*mediaRecorder.stop();*

*mediaRecorder.release();*

Sau khi ghi âm một media mới ta tạo ra một đối tượng contentValues mới để thêm một ghi âm mới vào Media Store.Các dữ liệu mà chúng ta chỉ định ở đây có thể bao gồm các chi tiết như tên, thời gian, mã địa lý cho tập tin mới như thể hiện trong đoạn code sau đây:



Chúng ta cũng cần phải chỉ ra đường dẫn cho tâp tin được tạo:

*content.put(MediaStore.Audio.Media.DATA,“myoutputfile.mp4”);*

Có thể truy cập vào *ContentResolver* của ứng dụng, sử dụng để thêm một dòng mới vào *MediaStore* như trong đoạn code sau:

*ContentResolver resolver = getContentResolver();*

*Uri uri = resolver.insert(Audio.Media.EXTERNAL\_CONTENT\_ URI, content);*

Khi thông tin của một tập tin được lưu trữ, chúng ta có thể thông báo nó đã sẵn sàng dùng một Intent để phát như sau:

*sendBroadcast(new*

*Intent(Intent.ACTION\_MEDIA\_SCANNER\_SCAN\_FILE,uri));*

* 1. SỬ DỤNG MÁY ẢNH

Sự phổ biến của máy ảnh kỹ thuật số (đặc biệt là trong điên thoại di động) đã gây ra một sự giảm giá đáng kể. Bây giờ thật là khó, thậm chí tìm một điện thoại đi động mà không có máy ảnh và các thiết ị Android không phải là trường hợp ngoại lệ.

Để truy cập vào phần cứng máy ảnh chúng ta cần phải thêm một CAMERA cho phép trong ứng dụng manifest như trình bày sau đây:

*<uses-permission Android:name=”Android.permission.CAMERA”/>*

Để truy cập vào dịch vụ Camera, sử dụng phương thức tĩnh open trên lớp

Camera. Khi đã hoàn thành với Camera, để kết thúc dịch vụ này ta gọi release sử dụng các mô hình đơn giản thể hiện trong đoạn code sau:

*Camera camera = Camera.open();*

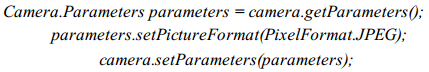
*[ … Do things with the camera … ]*

*camera.release();*

* + 1. Cài đặt điều khiển camera

Đã có sẵn một đối tượng *Camera.Parameters* để thiết lập cho camera hiện hành. Gọi phương thức *getParameters* trên Camera để truy cập vào các thông số hiện hành.

Chúng ta có thể sử dụng phương thức *set\** trên thông số để quay trở lại thiết lập các sửa đổi. Để áp dụng sự thay đổi ta gọi *setParameters* đưa vào các giá trị thay đổi như sau:



* + 1. Sử dụng máy ảnh

Truy cập vào các video của máy ảnh có nghĩa là chúng ta có thể kết hợp trực tiếp các video vào các ứng dụng của chúng ta. Điều thú vị nhất là ứng dụng Android đã sử dụng chức năng này như là lớp cơ sở.

Máy ảnh được hiển thị trong thời gian thực hiện trên một giao diện như thể hiên trong đoạn code sau đây:

*camera.setPreviewDisplay(mySurface);*

*camera.startPreview();*

*[ … ]*

*camera.stopPreview();*

Chúng ta cũng có thể chỉ định một *PreviewCallback* để chọn cho mỗi khung hình xem, cho phép thao tác hoặc hiển thị khung hình xem. Gọi phương thức *setPreviewCallback* trên đối tượng Camera, đưa vào một *PreviewCallback* mới thực hiện ghi đè lên phương thức *onPreviewFrame* như trình bày sau:

*camera.setPreviewCallback(new PreviewCallback() {*

*public void onPreviewFrame(byte[] \_data, Camera \_camera) {*

*// TODO Do something with the preview image.*

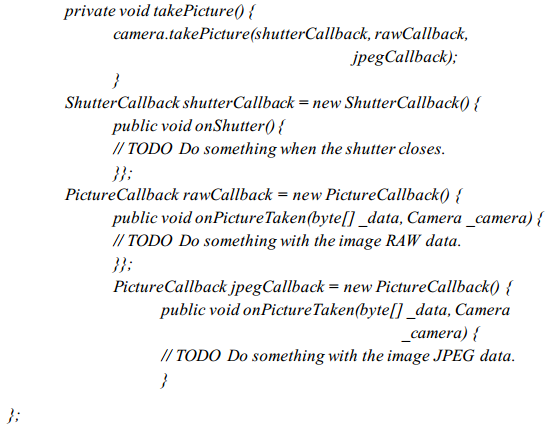
*}*

*});*

* + 1. Chụp ảnh

Chụp ảnh bằng cách gọi *takePicture* trên đối tượng Camera, đưa vào một *ShutterCallback* và *PictureCallback* thực hiện cho hình ảnh RAW và JPEG.

Mỗi lần gọi hình ảnh sẽ nhận được một mảng (byte) đại diện cho hình ảnh trong các định dạng thích hợp, trong khi gọi lại cánh cửa chớp được kích hoạt ngay sau đó cánh cửa chớp được đóng lại.



1. **Ứng dụng**

**4.1. Mô hình chương trình**



**Giảng viên**



**Website trường**

**Chương trình**

**Cơ sở dữ liệu**

**Sinh viên**



***Hình 03****. Mô hình chương trình*

Ứng dụng xem thời khóa biểu trên di động là trung gian của sơ đồ 01. Khi giảng viên hoặc sinh viên yêu cầu lấy thời khóa biểu, ứng dụng sẽ gửi yêu cầu đến website của trường để lấy các thông tin về tiết học trong tuần, sau đó sẽ bóc tách và hiển thị dữ liệu lên. Ứng dụng cho phép người sử dụng lưu thời khóa biểu đã hiển thị xuống cơ sở dữ liệu. Từ ứng dụng này, người dùng còn có thể cập nhật thời khóa biểu đã được lưu lại.

**4.2. Các chức năng của chương trình**

Chương trình được xây dựng với các chức năng chủ yếu như sau:

***Xem thời khóa biểu giáo viên:***

* Lấy dữ liệu thời khóa biểu từ website trường theo tuần và tên của giáo viên.
* Lưu thời khóa biểu đang xem.
* Cập nhật tuần học.
* Cập nhật lại toàn bộ dữ liệu khi bắt đầu năm học mới.

***Xem thời khóa biểu sinh viên***

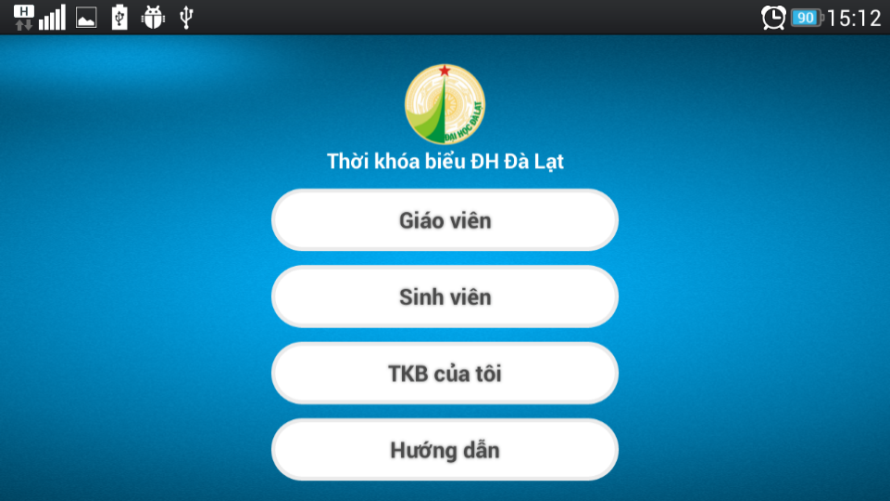
* Lấy dữ liệu thời khóa biểu từ website trường theo tuần và mã lớp.
* Lưu thời khóa biểu đang xem.
* Cập nhật tuần học.
* Cập nhật lại toàn bộ dữ liệu sinh viên khi bắt đầu năm học mới.

***Quản lý thời khóa biểu đã lưu***

* Quản lý danh sách thời khóa biểu đã lưu.
* Xem thời khóa biểu đã lưu.
* Chỉnh sửa thời khóa biểu để quản lý thời gian cá nhân hợp lý hơn.
* Xóa thời khóa biểu đã lưu.

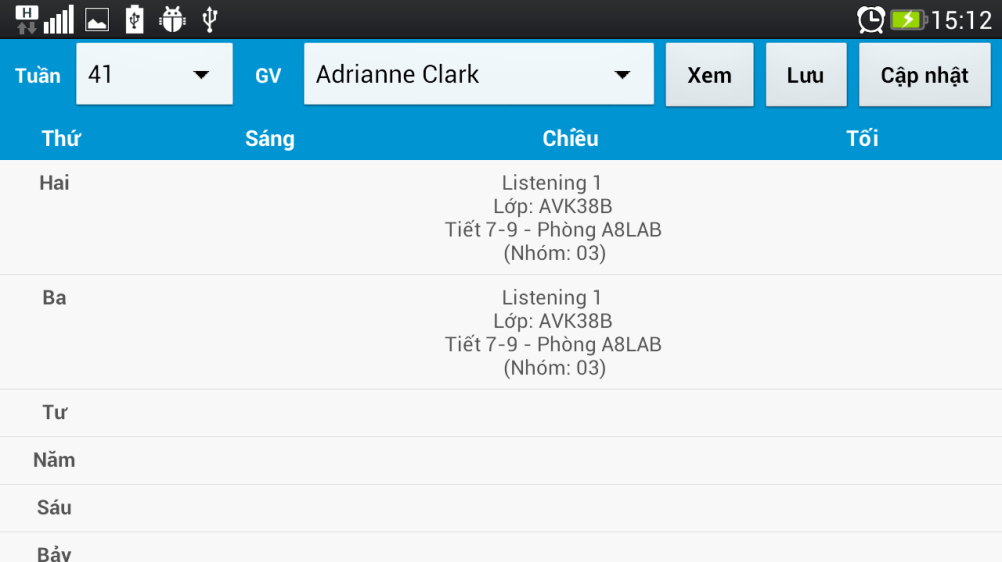
**4.3. Một số hình ảnh giao diện trên Hệ điều hành Android**

*Màn hình chính (dọc và ngang):*

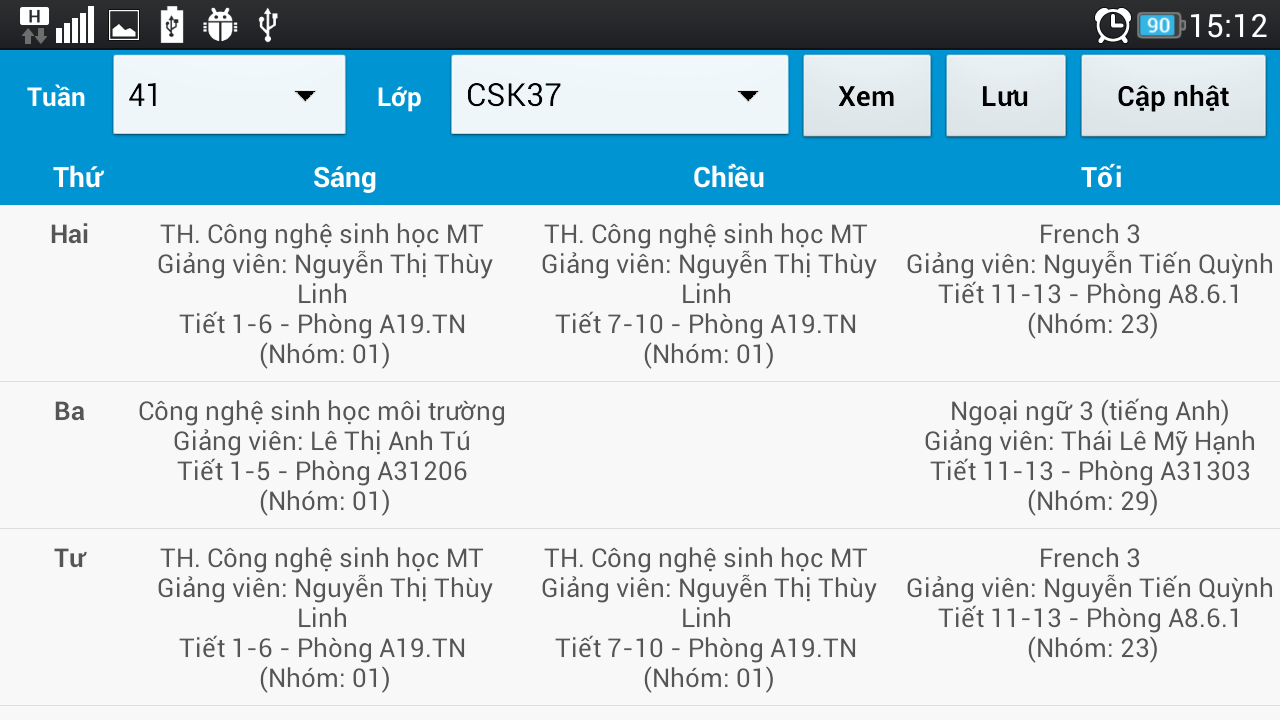
***Hình 04****. Màn hình chính (dọc, ngang)*

*Màn hình xem thời khóa biểu giảng viên:*



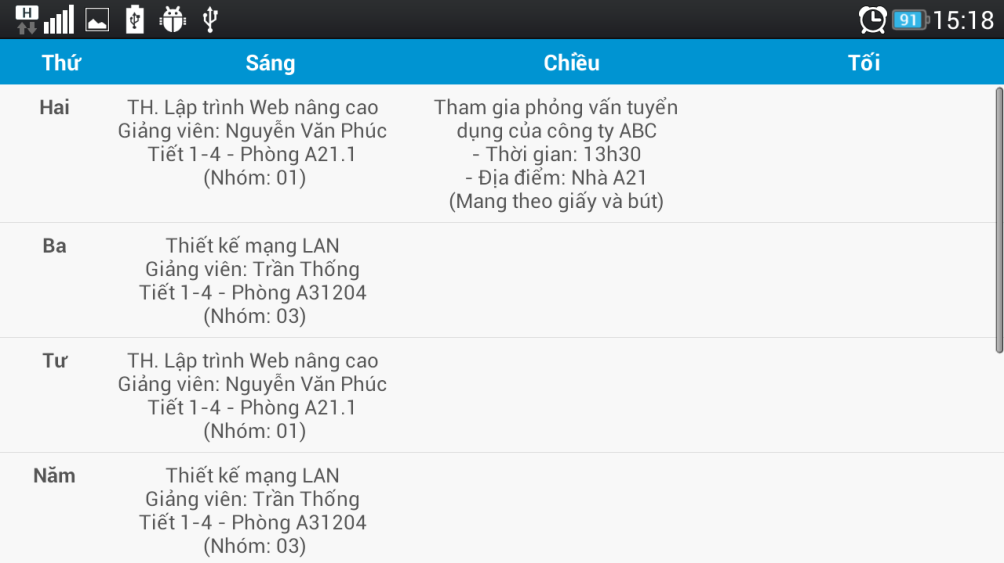
***Hình 05****. Màn hình xem thời khóa biểu giảng viên*

*Xem thời khóa biểu sinh viên*



***Hình 06****. Màn hình xem thời khóa biểu sinh viên*

*Quản lý thời khóa biểu đã lưu*

***Hình 07****. Các màn hình quản lý thời khóa biểu đã lưu*