----------ᔓ 🕮 ᔕ----------

Giáo trình

Lập trình android cơ bản

Lời nói đầu

Đứng trước xu thế toàn cầu hoá, ngành dịch vụ Viễn thông nói chung và dịch vụ điện thoại di động nói riêng ở Việt Nam đã đạt được những thành tựu nhất định. Sơ khai là một ngành với nhũng điều kiện cơ sở vật chất nghèo nàn, dịch vụ viễn thông còn rất lạc hậu. Cho đến nay, ngành Viễn thông Việt nam đã hoà nhập với mạng thông tin toàn cầu, đóng góp vào GDP 0,2% năm 1991 và đến nay đã lên tới con số 10,5%. Đặc biệt số lượng thuê bao dịch vụ điện thoại di động đã thay đổi một cách nhanh chóng, từ 4.060 thuê bao năm 1993 lên tới 1.200.000 thuê bao tính đến hết tháng 3 năm 2005. Trước xu thế hội nhập ngày càng mở rộng, ngành dịch vụ Viễn thông nói chung và dịch vụ điện thoại di động nói chung đang bước vào giai đoạn cạnh tranh rất lớn. Nổi lên trong giao đoạn hiện nay là công nghệ hệ điều hành di động và nổi bật như một ngôi sao mới là hệ điều hành Android của Google.

Chương I: Tìm hiểu về Android_

I.1 Android là gì?

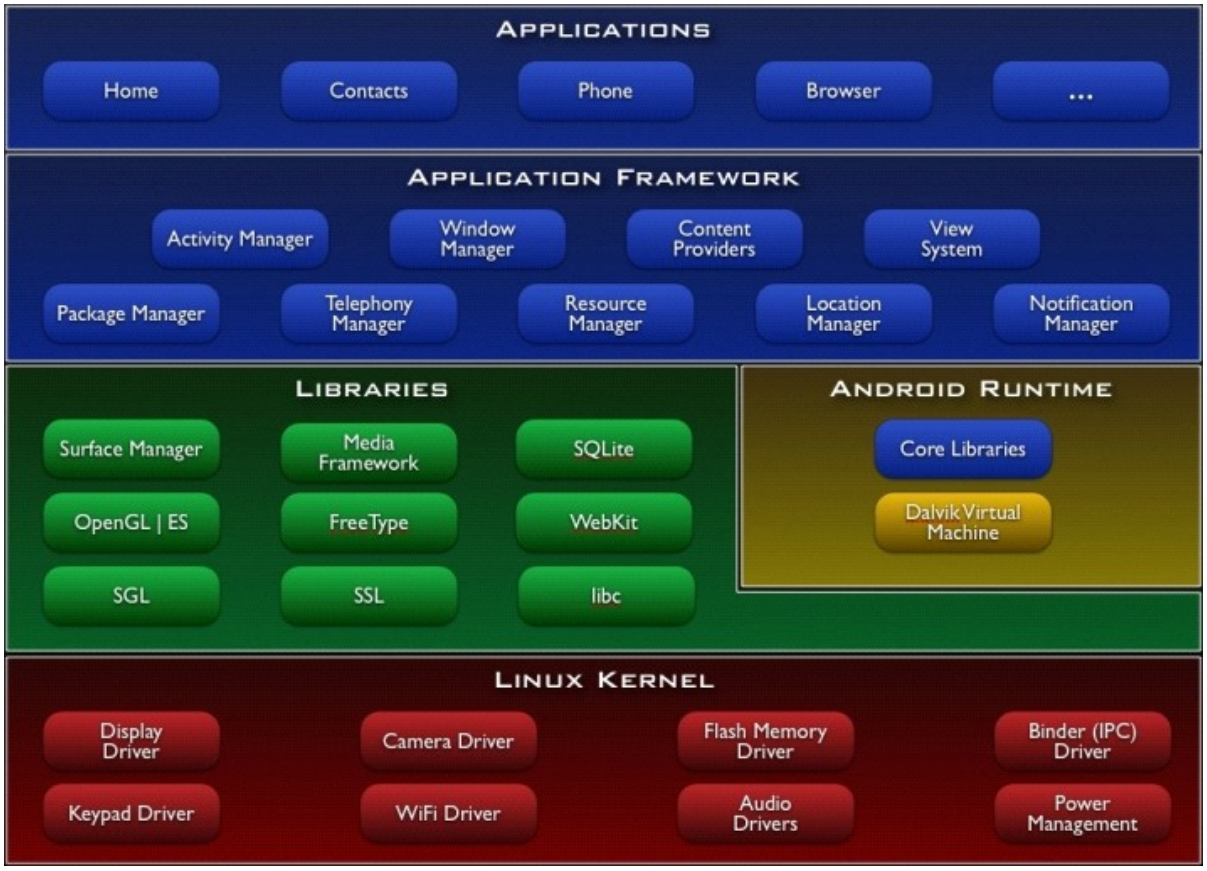
Android là một phần mềm stack cho các thiết bị di động bao gồm một hệ điều hành, middleware và các ứng dụng quan trọng. Android SDK cung cấp các công cụ và API cần thiết để bắt đầu phát triển các ứng dụng trên nền tảng Android bằng cách sử dụng ngôn ngữ lập trình Java.

I.2 Những đặc tính

- Ứng dụng framework cho phép tái sử dụng và thay thế các thành phần
- Dalvik máy ảo được tối ưu hóa cho các thiết bị di động
- Tích hợp trình duyệt dựa trên động cơ WebKit mã nguồn mở
- Tối ưu hóa đồ họa được hỗ trợ bởi một tùy chỉnh đồ họa 2D thư viện; đồ họa 3D dựa trên những đặc điểm kỹ thuật OpenGL ES 1,0 (Tùy chôn tăng tốc phần cứng)
- SQLite cho việc lưu trữ dữ liệu cấu trúc
- Phương tiện truyền thông hỗ trợ cho âm thanh phổ biến, video, và vẫn còn định dạng hình ảnh (MPEG4, H.264, MP3, AAC, AMR, JPG, PNG, GIF)
- GSM điện thoại (phụ thuộc phần cứng)
- Bluetooth, EDGE, 3G và WiFi (phụ thuộc phần cứng)
- Camera, GPS, la bàn, và gia tốc (phụ thuộc phần cứng)
- Môi trường phát triển phong phú bao gồm một mô phỏng thiết bị, công cụ để gỡ lỗi, bộ nhớ và profiling hiệu suất, và một plugin cho IDE Eclipse

I.3 Kiến trúc Android

Sơ đồ dưới đây cho thấy các thành phần chính của hệ điều hành Android. Mỗi phần được mô tả chi tiết hơn dưới đây.



I.3.1 Ứng dụng

Android sẽ ship với một bộ các ứng dụng lõi bao gồm một ứng dụng email, lịch chương trình tin nhắn SMS,, bản đồ, trình duyệt, liên lạc, và những người khác. Tất cả các ứng dụng được viết bằng cách sử dụng ngôn ngữ lập trình Java.

I.3.2 Ứng dụng Framework

Bằng cách cung cấp một nền tảng phát triển mở, Android cung cấp cho các nhà phát triển khả năng để xây dựng các ứng dụng vô cùng phong phú và sáng tạo. Các nhà phát triển được miễn phí để tận dụng lợi thế của các thiết bị phần cứng, thông tin địa điểm truy cập, dịch vụ chạy nền, thiết lập hệ thống báo động, thêm các thông báo đến các thanh trạng thái, và nhiều, nhiều hơn nữa.

Các nhà phát triển có thể truy cập vào các API cùng một khuôn khổ được sử dụng bởi các ứng dụng lõi. Kiến trúc ứng dụng được thiết kế để đơn giản hóa việc tái sử dụng các thành phần; bất kỳ ứng dụng có thể xuất bản các khả năng của mình và ứng dụng nào khác sau đó có thể làm cho việc sử dụng những khả năng (tùy thuộc vào chế bảo đảm thi hành theo khuôn khổ). Cơ chế này cũng cho phép các thành

phần được thay thế bởi người sử dụng.

Nằm bên dưới tất cả các ứng dụng là một tập hợp các dịch vụ và hệ thống, bao gồm:

- Một tập phong phú và mở rộng của xem có thể được sử dụng để xây dựng một ứng dụng, bao gồm các danh sách, lưới, hộp văn bản, các nút, và thậm chí một trình duyệt web nhúng
- Nhà cung cấp nội dung cho phép các ứng dụng để truy cập dữ liệu từ các ứng dụng khác (như Contacts), hoặc chia sẻ dữ liệu của riêng mình
- Một quản lý tài nguyên, cung cấp quyền truy cập vào tài nguyên phi mã như dây bản địa hoá, đồ họa, và bố trí tập tin.
- Một Notification Manager cho phép tất cả các ứng dụng tùy chỉnh để hiển thị cảnh báo trong thanh trạng thái.
- Một Activity Manager quản lý vòng đời của các ứng dụng và cung cấp một backstack phổ biến chuyển hướng.

I.3.3 Thư viện

Android bao gồm một bộ thư viện C/C++, được sử dụng bởi các thành phần khác nhau của hệ thống Android. Những khả năng tiếp xúc với các nhà phát triển thông qua các khuôn khổ ứng dụng Android. Một số các thư viện lõi được liệt kê dưới đây:

- **System C library** - một BSD-có nguồn gốc thực hiện các hệ thống thư viện chuẩn C (LIBC), điều chỉnh cho nhúng dựa trên Linux các thiết bị
- **Media Libraries** - dựa trên OpenCORE PacketVideo's; sự hỗ trợ các thư viện phát lại và ghi âm của âm thanh và phổ biến nhiều định dạng video, cũng như các tập tin hình ảnh tĩnh, bao gồm MPEG4, H.264, MP3, AAC, AMR, JPG, và PNG
- **Surface Manager** - quản lý quyền truy cập vào hệ thống con hiển thị và hoàn toàn phù hợp chất 2D và 3D lớp từ nhiều ứng dụng đồ họa
- **LibWebCore** - một trình duyệt web hiện đại, động cơ có quyền hạn cả hai trình duyệt web của Android và một xem nhúng
- **SGL** - các công cụ đồ họa 2D tiềm ẩn
- **3D libraries** - một việc thực hiện dựa trên OpenGL ES 1,0 API; các thư viện, hoặc sử dụng phần cứng tăng tốc 3D (nếu có) hoặc bao gồm, cao tối ưu rasterizer phần mềm 3D
- **SQLite** - một mạnh mẽ và nhẹ quan hệ cơ sở dữ liệu có sẵn cho tất cả các ứng dụng

I.3.4 Thời gian chạy Android

Android bao gồm một tập các thư viện lõi mà cung cấp hầu hết các chức năng sẵn có trong thư viện cốt lõi của ngôn ngữ lập trình Java.

Mỗi ứng dụng Android chạy trong tiến trình riêng của mình, với trường hợp riêng của các máy ảo Dalvik. Dalvik đã được viết nên một thiết bị có thể chạy nhiều máy ảo hiệu quả. VM Dalvik thực hiện tác phẩm trong các Executable Dalvik (dex). Định dạng được tối ưu hóa cho bộ nhớ tối thiểu. VM là đăng ký trên, và chạy các lớp học biên soạn bởi một trình biên dịch ngôn ngữ Java đã được chuyển thành các định dạng dex. Do dx "bao gồm" công cụ.

VM Dalvik dựa vào hạt nhân Linux cho các chức năng tiềm ẩn như luồng và cấp quản lý bộ nhớ thấp.

Chương II: Sử dụng tài nguyên trong ứng dụng Android

Bạn nên luôn luôn sử dụng nguồn tài nguyên như hình ảnh và chuỗi từ mã ứng dụng của bạn, để bạn có thể duy trì chúng một cách độc lập. Externalizing nguồn lực của bạn cũng cho phép bạn tới cung cấp nguồn tài nguyên thay thế có hỗ trợ cấu hình thiết bị cụ thể như ngôn ngữ khác nhau hoặc kích cỡ màn hình, mà ngày càng trở nên quan trọng như nhiều thiết bị hỗ trợ Android trở nên có sẵn với các cấu hình khác nhau. Để cung cấp khả năng tương thích với cấu hình khác nhau, bạn phải tổ chức các nguồn lực trong thư mục res dự án của bạn, bằng cách sử dụng sub thư mục khác nhau, nhóm các tài nguyên theo loại hình và cấu hình.

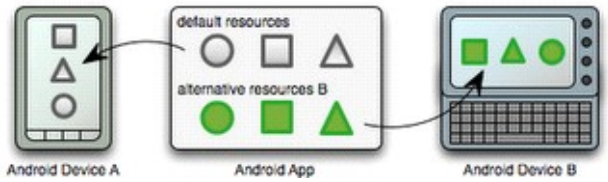
Đối với bất cứ loại tài nguyên, bạn có thể mặc định và thay thế nhiều nguồn lực cho ứng dụng của bạn:

- Mặc định là những tài nguyên được sử dụng không phụ thuộc vào cấu hình thiết bị hoặc khi không có nguồn tài nguyên thay thế phù hợp với cấu hình hiện tại.
- Thay thế các nguồn lực được các mục bạn đã thiết kế để sử dụng với một cấu hình cụ thể. Để xác định nó một nhóm các nguồn lực cho một cấu hình đặc biệt, thêm một vòng loại cấu hình phù hợp với tên thư mục.

Ví dụ, trong khi mặc định layout giao diện của bạn được lưu trong thư mục res/layout/, bạn có thể xác định một layout giao diện khác nhau được sử dụng khi màn hình định hướng phong cảnh, bằng cách lưu nó trong thư mục res/layout-

land/. Android sẽ tự động áp dụng các nguồn lực thích hợp bằng cách kết hợp cấu hình hiện tại của thiết bị tới tên thư mục tài nguyên của bạn.

(Ảnh minh họa từ http://developer.android.com/images/resources/resource_devices_diagram1.png)
Hình 1 thể hiện như thế nào một tập hợp các nguồn tài nguyên mặc định từ một ứng dụng được áp dụng cho hai thiết bị khác nhau khi không có nguồn tài nguyên thay thế có sẵn.



(Ảnh minh họa từ http://developer.android.com/images/resources/resource_devices_diagram2.png)
Hình 2 cho thấy việc áp dụng cùng với một tập hợp các nguồn tài nguyên thay thế nó đủ điều kiện đối với một trong các cấu hình thiết bị, do đó, hai thiết bị sử dụng nguồn tài nguyên khác nhau.

Thông tin trên chỉ là giới thiệu về cách làm việc nguồn lực ứng dụng trên Android. Các tài liệu sau đây cung cấp một hướng dẫn đầy đủ tới làm thế nào bạn có thể tổ chức các nguồn lực ứng dụng của bạn, xác định nguồn tài nguyên thay thế, truy cập chúng trong ứng dụng của bạn, và nhiều hơn nữa:

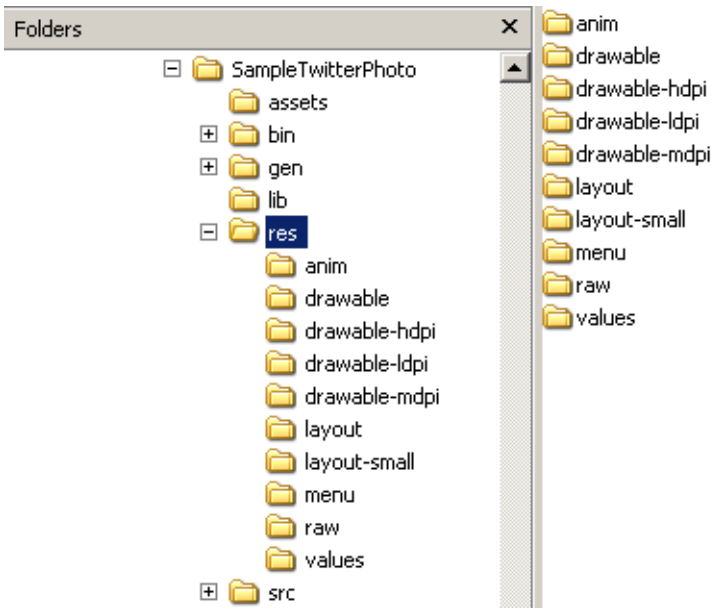
Việc cung cấp tài nguyên : Những loại tài nguyên mà bạn có thể cung cấp trong ứng dụng của bạn, nơi tới lưu lại, và làm thế nào để tạo ra nguồn lực thay thế cho các cấu hình thiết bị cụ thể.
Truy cập các tài nguyên : Làm thế nào tới sử dụng các nguồn lực mà bạn đã cung cấp, hoặc bằng cách tham chiếu đến chúng từ mã ứng dụng của bạn hoặc từ các nguồn khác XML.
Xử lý Thay đổi Runtime : Làm thế nào tới quản lý thay đổi cấu hình xảy ra trong khi Activity của bạn đang chạy.

Localization : Met hướng dẫn từ dưới lên tới địa phương hóa đơn của bạn bằng cách sử dụng nguồn tài nguyên thay thế. Trong khi đây chỉ là một cụ thể sử dụng tài nguyên thay thế, nó là rất quan trọng để tiếp cận người dùng hơn.
Các loại tài nguyên : Một tài liệu tham khảo của các loại tài nguyên khác nhau, bạn có thể cung cấp, mô tả XML của họ yếu tố, thuộc tính, và cú pháp. Ví dụ, tham chiếu này cho thấy bạn làm thế nào để tạo ra một nguồn lực cho các menu ứng dụng, drawables, hình động, và nhiều hơn nữa.

Các loại tài nguyên trong ứng dụng Android

Mỗi của các tài liệu trong phần này mô tả việc sử dụng, định dạng và cú pháp cho một loại tài nguyên ứng dụng mà bạn có thể cung cấp trong thư mục tài nguyên của bạn (res/).

Dưới đây là tóm tắt của từng loại tài nguyên:



Tài nguyên hình ảnh động : Xác định hình ảnh động được xác định trước.

- Tween hình ảnh động được lưu trong **res/Anim/** và truy cập từ các lớp **R.anim.**
- Frame hình ảnh động được lưu trong **res/drawable/** và truy cập từ các lớp **R.drawable.**

Danh sách State tài nguyên màu : Xác định một tài nguyên màu sắc nó thay đổi dựa trên các tiểu bang View.

- Lưu trong **res/color** và truy cập từ các lớp R.color.

Tài nguyên Drawable : Xác định đồ họa khác nhau với bitmap hoặc XML.

- Lưu trong **res/drawable/** và truy cập từ các lớp R.drawable.

Bố trí nguồn lực : Xác định layout cho giao diện người dùng ứng dụng của bạn.

- Lưu trong res/layout/và truy cập từ các lớp **R.layout.**

Menu Resource : Xác định nội dung của các menu ứng dụng của bạn.

- Lưu trong **res/menu/** và truy cập từ các lớp **R.menu.**

Tài nguyên String : Xác định các chuỗi, mảng chuỗi, và số nhiều (và bao gồm các định dạng chuỗi và tạo kiểu tóc).

- Lưu trong **res/value/** và truy cập từ các lớp **R.string, R.array**, và **R.plurals.**

Tài nguyên phong cách : Xác định xem xét và định dạng cho cáccác yếu tố giao diện người dùng .

- Lưu trong **res/value/** và truy cập từ các lớp **R.style.**

Các loại tài nguyên khác : Xác định giá trị như các phép toán luận, số nguyên, kích thước, màu sắc, và các mảng khác.

- Lưu trong **res/value/** nhưng mỗi truy cập từ duy nhất R tiểu học (như **R.bool, R.integer, R.dimen,** vv.)

Quản lý vòng đời của một Service trong Android

Vòng đời của một service đơn giản hơn nhiều so với các activity. Tuy nhiên, nó thậm chí còn quan trọng hơn, bạn có phải quan tâm tới cách service của bạn được tạo ra và hủy diệt, bởi vì một service có thể chạy trong background mà không cho người dùng được biết.

Vòng đời của một Service, từ khi nó được tạo ra cho khi nó bị phá hủy, có thể theo hai con đường khác nhau:

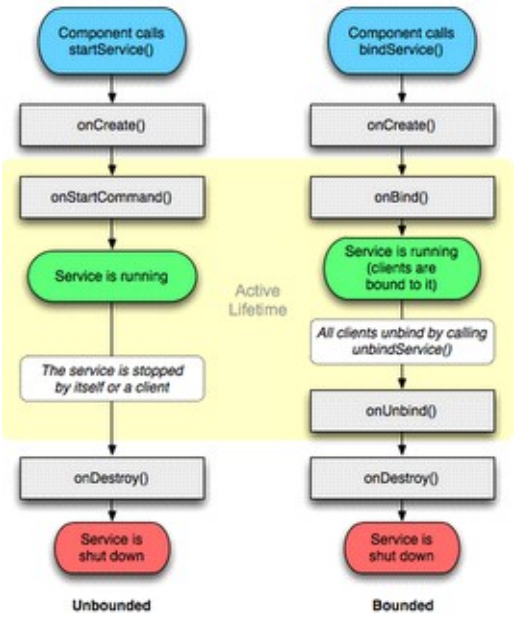
- Một service started : Các service được tạo ra khi component khác gọi **startService().** Các service sau đó nó chạy vô thời hạn và dừng lại bằng cách gọi **stopSelf()**. Một component khác cũng có thể dừng dịch vụ bằng cách gọi **stopService().** Khi dịch vụ này được dừng lại, hệ thống tiêu diệt nó ..
- Một service bound : Các service được tạo ra khi một component (một client) gọi b**indService().** Client sau đó giao tiếp với các service thông qua một giao diện **IBinder.** Các client có thể đóng kết nối bằng cách gọi **unbindService()**. Nhiều khách hàng có thể liên kết với các service tương tự và khi tất cả chúng unbind, hệ thống sẽ phá hủy các service. (chính nó không cần dừng lại.)

Hai con đường không hoàn toàn riêng biệt. Bạn có thể liên kết với một dịch vụ đã được started với **startService().** Ví dụ, một service background âm nhạc có thể được started bằng cách gọi **startService()** với một ý định nhận dạng âm nhạc để

chơi. Sau đó, có thể khi người dùng muốn thực hiện một số kiểm soát đối với các cầu thủ hoặc nhận được thông tin về các bài hát hiện hành, một activity có thể liên kết với các service bằng cách gọi **bindService()**. Trong trường hợp như thế này, **stopService()** hoặc **stopSelf()** không thực sự dừng service cho đến khi khách hàng **unbind** tất cả .

Triển khai thực hiện vòng đời callback

Giống như một activity, một service có vòng đời method callback và bạn có thể triển khai để theo dõi những thay đổi trong trạng thái của service và thực hiện công việc vào thời điểm thích hợp. Các service thể hiện skeleton sau mỗi vòng đời của các method:



(Ảnh minh họa được sử dụng từ http://developer.android.com/images/service_lifecycle.png)

Hình 2. Các vòng đời service. Sơ đồ bên trái cho thấy vòng đời khi dịch vụ được tạo ra với startService() và sơ đồ bên phải cho thấy vòng đời khi dịch vụ được tạo ra với bindService().

Bằng cách thực hiện những phương pháp này, bạn có có thể theo dõi hai vòng lặp lồng nhau trong vòng đời của service:

- Sự sống của một service xảy ra giữa thời gian onCreate() được gọi và onDestroy() thời gian trả về. Giống như một activity, một service không thiết lập ban đầu trong onCreate() cho nó và tất cả các nguồn lực còn lại trong onDestroy(). Ví dụ, một service nghe nhạc có thể tạo ra các chủ đề

mà âm nhạc sẽ được chơi trong onCreate(), sau đó dừng thread trong onDestroy().

- Các onCreate() và onDestroy() method được gọi cho tất cả các dịch vụ, cho dù họ đang tạo ra bởi startService() hoặc bindService().
- Các cuộc đời hoạt động của một service bắt đầu với một cuộc gọi đến hoặc là onStartCommand() hoặc onBind(). Mỗi phương pháp được đưa các ý định đó được thông qua hoặc là startService() hoặc bindService(), tương ứng.

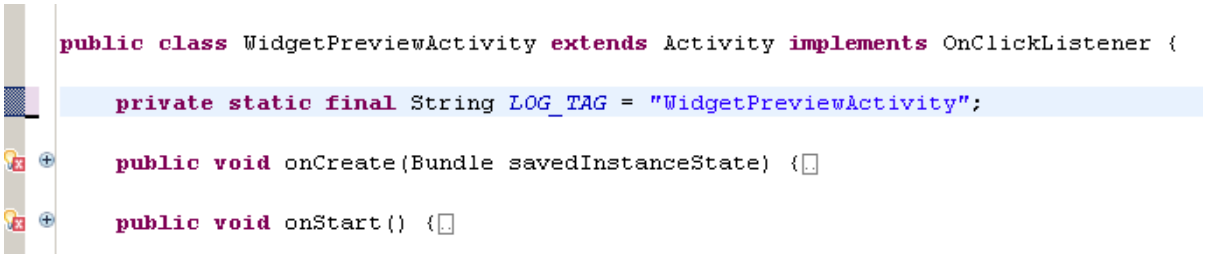
Nếu service được started, các hoạt động kết thúc cuộc đời cùng thời điểm kết thúc toàn bộ cuộc đời (dịch vụ này vẫn còn hoạt động ngay cả sau khi onStartCommand() trả về). Nếu dịch vụ được bound, kết thúc cuộc đời hoạt động trở lại khi onUnbind().

Tìm hiểu về Activity trong Android

Activity Một là một thành phần ứng dụng đó cung cấp một màn hình mà người dùng có thể tương tác để làm một cái gì đó, chẳng hạn như quay số điện thoại, chụp ảnh, gửi email, hoặc xem một bản đồ. Mỗi activity được cho một cửa sổ, trong đó cho vẽ giao diện người dùng của nó. Cửa sổ thường lấp đầy màn hình, nhưng có thể nhỏ hơn so với màn hình và nổi lên trên các cửa sổ khác.

Một ứng dụng thường bao gồm nhiều activity được ràng buộc lỏng lẻo với nhau. Thông thường, một trong những activity trong một ứng dụng được quy định như các activity "chính", được trình bày cho người dùng khi tung ra ứng dụng cho lần đầu tiên. Mỗi activity sau đó có thể bắt đầu activity khác để thực hiện hành động khác nhau. Mỗi lần một activity mới bắt đầu, các activity trước đó được dừng lại, nhưng hệ thống các khu bảo vệ các activity trong một ngăn xếp (các "back stack"). Khi một activity mới bắt đầu, nó được đẩy lên phía sau ngăn xếp và việc chú trọng của người dùng. Sự trở lại ngăn xếp tuân thủ các cơ bản "last in, first out" cơ chế hàng đợi, do đó, khi người dùng được thực thi với các activity hiện tại và nhấn phím BACK, nó là popped khỏi đống (và phá hủy) các hồ sơ activity trước đó. (Sự trở lại ngăn xếp được thảo luận nhiều hơn trong công việc và sắp xếp lại tài liệu.)

Để tạo ra một activity, bạn phải tạo một sub class của Activity. Trong sub class của bạn, bạn cần cho thực thi method gọi hệ thống các cuộc gọi khi chuyển đổi activity giữa các state khác nhau cho vòng đời của nó, chẳng hạn như khi activity đang được tạo ra, dừng lại, nối lại, hoặc bị phá hủy. Hai phương pháp gọi lại quan trọng nhất là:



Create Activity
Bạn phải triển khai method này. Hệ thống các cuộc gọi này khi tạo activity của bạn. Trong thời hạn triển khai thực hiện của bạn, bạn nên khởi tạo các thành phần thiết yếu của activity của bạn. Quan trọng nhất, đây là nơi bạn phải gọi setContentView() để xác định việc layout cho giao diện người dùng của activity.

Start Activity
Khi làm việc trong ứng dụng của riêng bạn, bạn thường sẽ cần cho đơn giản là khởi động một activity được biết đến. Bạn có thể làm như vậy bằng cách tạo ra một intent đó rõ ràng xác định activity bạn muốn bắt đầu, sử dụng tên lớp. Ví dụ, đây là cách một activity bắt đầu activity khác có tên SignInActivity:

Intent intent = new Intent(this, SignInActivity.class);
startActivity(intent);

Đóng một Activity
Bạn có thể đóng một activity bằng cách gọi methos **finish()** của nó. Bạn cũng có thể đóng cửa một activity riêng biệt trước đó bạn đã bắt đầu bằng cách gọi **finishActivity().**

Ngoài ra trong Activity còn nhiều method khác,

```java
public class ExampleActivity extends Activity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        // The activity is being created.
    }
    @Override
    protected void onStart() {
        super.onStart();
        // The activity is about to become visible.
    }
    @Override
    protected void onResume() {
        super.onResume();
        // The activity has become visible (it is now "resumed").
    }
    @Override
    protected void onPause() {
        super.onPause();
        // Another activity is taking focus (this activity is about to be "paused").
    }
    @Override
    protected void onStop() {
        super.onStop();
        // The activity is no longer visible (it is now "stopped")
    }
    @Override
    protected void onDestroy() {
        super.onDestroy();
        // The activity is about to be destroyed.
    }
}
```

Quản lý vòng đời Activity

Quản lý vòng đời Activity của bạn bằng cách thực hiện gọi method là rất quan trọng cho phát triển một ứng dụng mạnh mẽ và linh hoạt. Vòng đời của một activity trực tiếp bị ảnh hưởng bởi kết hợp nhiệm vụ của nó và back stack với các activity khác.

Một hoạt động có thể tồn tại trong ba trạng thái cơ bản: Resumed, Paused và Stopped

Nếu một activity bị tạm dừng hoặc dừng lại, hệ thống có thể drop nó từ bộ nhớ hoặc là bằng cách yêu cầu cho nó kết thúc (gọi method finish() của nó), hoặc đơn giản là giết chết quá trình của nó. Khi hoạt động này được mở lại (sau khi được finished hoặc chết), nó phải được tạo ra như trên.

Tìm hiểu về Service trong Android

Service là một ứng dụng component có thể thực hiện các hoạt động long-running trong background và không cung cấp một giao diện người dùng. Một ứng dụng component có thể bắt đầu một service và nó sẽ tiếp tục chạy trong background

thậm chí nếu người dùng chuyển cho ứng dụng khác. Ngoài ra, một component có thể liên kết cho một dịch vụ tương tác với nó và thậm chí thực hiện giao tiếp InterProcess (IPC). Ví dụ, một dịch vụ có thể xử lý các giao dịch mạng, nghe nhạc, thực hiện các tập tin I/O, hoặc tương tác với một provider content, tất cả từ background.

Một service cơ bản có thể có hai hình thức:

Started
service là "started" khi một component ứng dụng (như một activity) bắt đầu nó bằng cách gọi startService (). Khi bắt đầu, một service có thể chạy trên background vô thời hạn, ngay cả khi các thành phần đó bắt đầu nó bị phá hủy. Thông thường, một service bắt đầu thực hiện một hoạt động đơn lẻ và không trả lại kết quả cho người gọi. Ví dụ, nó có thể tải về hoặc tải lên một tập tin qua mạng. Khi hoạt động được thực hiện, các service nên dừng lại bản thân.

Bound
service là "bound" khi một component ứng dụng liên kết cho nó bằng cách gọi bindService (). Một service ràng buộc cung cấp một giao diện client-server cho phép các component tương tác với các dịch vụ, gửi các yêu cầu, có được kết quả, và thậm chí làm như vậy qua quá trình giao tiếp InterProcess (IPC). Một service bound chỉ chạy miễn là ứng dụng component khác bị ràng buộc vào nó. Nhiều thành phần có thể liên kết cho dịch vụ cùng một lúc, nhưng khi tất cả chúng unbind, service này bị phá hủy.

Các khái niệm cơ bản :

Để tạo ra một service, bạn phải tạo một lớp dịch vụ (hoặc một trong các subclasses hiện tại của nó). Trong implementation của bạn, bạn cần cho ghi đè lên một số method gọi lại có thể xử lý các khía cạnh quan trọng của vòng đời service và cung cấp một cơ chế cho các component cho gắn kết với dịch vụ, nếu thích hợp. Các method quan trọng nhất bạn có nên ghi đè lên gọi là:

```java
/**
 * @author dangcongthanhtrung@gmail.com
 */
public class SimpleService extends Service {

    int mStartMode;       // indicates how to behave if the service is killed
    IBinder mBinder;      // interface for clients that bind
    boolean mAllowRebind; // indicates whether onRebind should be used

    @Override
    public IBinder onBind(Intent arg0) {
        // A client is binding to the service with bindService()
        return mBinder;
    }

    @Override
    public void onCreate() {
        super.onCreate();
        Toast.makeText(this,"Service created ...", Toast.LENGTH_LONG).show();
    }

    @Override
    public void onDestroy() {
        super.onDestroy();
        Toast.makeText(this, "Service destroyed ...", Toast.LENGTH_LONG).show();
    }

    @Override
    public int onStartCommand(Intent intent, int flags, int startId) {
        return mStartMode;
    }

    @Override
    public boolean onUnbind(Intent intent) {
        return mAllowRebind;
    }
```

onStartCommand()

Hệ thống gọi method này khi component như là một activity, yêu cầu các service được started, bằng cách gọi startService (). Sau khi thực hiện phương pháp này, các service được khởi động và có thể chạy trong background vô thời hạn. Nếu bạn có triển khai điều này, đó là trách nhiệm của bạn để dừng dịch vụ khi công việc của mình được thực hiện, bởi stopSelf gọi điện thoại () hoặc stopService (). (Nếu bạn có chỉ muốn cung cấp ràng buộc, bạn không cần phải triển khai method này.)

onBind()

Hệ thống gọi method này khi component muốn liên kết với các service (chẳng hạn như cho thực hiện RPC), bằng cách gọi bindService(). Trong implementation của bạn của phương pháp này, bạn có phải cung cấp một giao diện đó khách hàng sử dụng để giao tiếp với dịch vụ, bằng cách trả lại một IBinder. Bạn luôn luôn phải

triển khai phương pháp này, nhưng nếu bạn không muốn cho phép liên kết, sau đó bạn nên trở về null.

onCreate()
Hệ thống gọi method này khi service đầu tiên tạo ra, để thực hiện các thủ tục thiết lập một lần (trước khi cuộc gọi hoặc là onStartCommand() hoặc onBind()). Nếu service đang chạy, phương pháp này không được gọi.

onDestroy()
Hệ thống gọi method này khi service không còn được sử dụng và đang được bị phá hủy. service của bạn nên triển khai điều này cho làm sạch bất cứ nguồn tài nguyên như chủ đề, người nghe đã đăng ký, nhận, vv Đây là các cuộc gọi qua dịch vụ này nhận được.

Giao diện người dùng trong Android

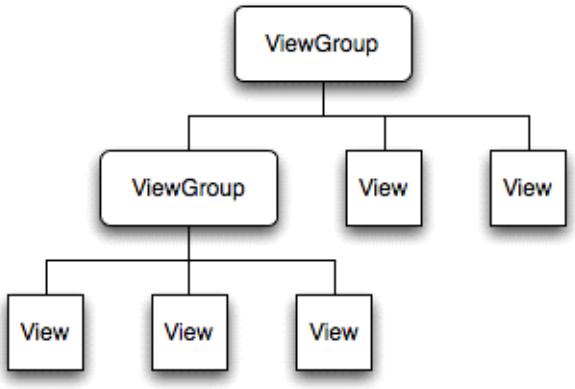
Trong một ứng dụng Android, giao diện người dùng được xây dựng bằng cách sử dụng View và ViewGroup đối tượng. Có nhiều loại quan điểm và các nhóm view, mỗi một trong số đó là hậu duệ của lớp View. View objects là các đơn vị cơ bản của biểu hiện giao diện người dùng trên nền tảng Android. Các class xem như là cơ sở phục vụ cho class con được gọi là "widget", trong đó cung cấp đầy đủ các đối tượng thực hiện giao diện, giống như các lĩnh vực văn bản và nút. Class ViewGroup phục vụ như là cơ sở cho lớp con được gọi là " layouts", cung cấp các loại khác nhau của kiến trúc bố trí, như linear, tabular và relative.

- Một View object là một cấu trúc dữ liệu có đặc tính lưu trữ các thông số bố trí và nội dung cho một khu vực cụ thể hình chữ nhật của màn hình.
- Một View object xử lý đo lường riêng của mình, bố trí, bản vẽ thay đổi tập trung,, di chuyển, và phím/tương tác cử chỉ cho khu vực hình chữ nhật của màn hình.

Là một object trong giao diện người dùng, view cũng là một điểm tương tác cho người sử dụng và nhận các sự kiện tương tác.

Xem Hierarchy

Trên nền tảng Android, bạn xác định một hoạt động của giao diện người dùng bằng cách sử dụng một hệ thống phân cấp của View và ViewGroup, như trong biểu đồ dưới đây.



Cây này có thể được phân cấp đơn giản hay phức tạp như bạn cần nó được, và bạn có thể xây dựng nó lên bằng cách sử dụng thiết lập Android của widgets và layouts định sẵn, hoặc với Views tùy chỉnh mà bạn tạo ra cho mình.

Để đính kèm với cây phân cấp xem màn hình cho rendering, Hoạt động của bạn phải gọi setContentView() Phương pháp và thông qua một tham chiếu đến đối tượng button gốc.

Hệ thống Android nhận được tập tin này và sử dụng nó để làm mất hiệu lực, đo lường, và vẽ cây.

Nút gốc của các yêu cầu phân cấp cho nó vẽ các nút con - lần lượt, mỗi nút view là nhóm chịu trách nhiệm kêu gọi mỗi lần view nút con riêng của mình để vẽ có thể yêu cầu một kích thước và vị trí trong nút gốc., Nhưng đối tượng viewgroup có quyết định cuối cùng về nơi làm thế nào, có thể được cho mỗi nút con.

Android parses các yếu tố của cách bố trí của bạn trong thứ tự (từ phía trên cùng của cây phân cấp), instantiating việc xem và thêm chúng vào parent(s).

Bởi vì đây là những trích ra trong trật tự, nếu có các yếu tố đó chồng chéo nhau các vị trí, một lần cuối để được rút ra sẽ nằm trên đầu trang của những người khác trước đây để rút ra không gian đó.

Giao diện

Cách phổ biến nhất để xác định bố trí của bạn và thể hiện sự phân cấp view là với một tập tin XML layout. XML cung cấp một cơ cấu có thể đọc được cho bố trí, giống như HTML. Mỗi phần tử trong XML là cả một View hoặc đối tượng ViewGroup (hoặc hậu duệ đó). Các đối tượng View là lá trong cây, ViewGroup đối tượng là các nhánh trong cây.
Tên của một phần tử XML là tương ứng với lớp học Java mà nó đại diện. Vì vậy, một yếu tố **<textview>** tạo ra một TextView trong UI của bạn, và một phần tử

<linearlayout> tạo ra một LinearLayout viewgroup. Khi bạn tải một layout resource, hệ thống Android khởi chạy thời gian các đối tượng, tương ứng với các yếu tố trong cách bố trí của bạn. Ví dụ, một bố trí dọc đơn giản với một lần xem văn bản và một nút sẽ như thế 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="vertical">
        <TextView android:id="@+id/text"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="Hello, I am a TextView"/>
        <Button android:id="@+id/button"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="Hello, I am a Button" />
</LinearLayout>
```

Chú ý rằng các phần tử LinearLayout chứa cả TextView và Button. Bạn có thể làm khác LinearLayout (hoặc các loại hình xem nhóm) bên trong ở đây, để kéo dài sự phân cấp xem và tạo ra một bố cục phức tạp hơn. Để biết thêm về việc xây dựng một bố cục UI, đọc Giao diện kê khai.
Có rất nhiều cách mà bạn có thể xem cách bố trí của bạn. Sử dụng nhiều hơn và các loại khác nhau của các view group, bạn có thể cấu trúc views con và view groups trong vô số cách . Xác định các nhóm xem được cung cấp bởi Android (gọi là layouts) bao gồm LinearLayout, RelativeLayout, TableLayout, GridLayout và khác. Mỗi cung cấp một bộ duy nhất của các thông số bố trí được sử dụng để xác định vị trí của views con và cơ cấu layout Để tìm hiểu về một số các loại khác nhau của các view group được sử dụng cho một layout, đọc Giao diện đối tượng thường gặp.

Widgets

Widget là một object View phục vụ như một giao diện để tương tác với người dùng. Android cung cấp một tập các widgets thực hiện đầy đủ, giống như các button, Checkbox, và text-entry , do đó bạn có thể nhanh chóng xây dựng giao diện người dùng của bạn. Một số widgets được cung cấp bởi Android phức tạp hơn, giống như một date picker, clock, và zoom controls. Nhưng nó không giới

hạn trong các loại widgets được cung cấp bởi các nền tảng Android.

Nếu bạn muốn làm một cái gì thêm cho tùy biến và tạo ra các yếu tố của hành động của bạn, bạn có thể, bằng cách xác định object view của riêng bạn hoặc bằng cách mở rộng và kết hợp các Widget hiện có. Đọc tiếp tại Building Custom Components. Để có một danh sách các vật dụng được cung cấp bởi Android, xem gói android.widget

UI Sự kiện

Một khi bạn đã thêm một số Views/widgets đến giao diện, có thể bạn muốn biết về sự tương tác của người dùng với họ, vì vậy bạn có thể thực hiện hành động.

Để được thông báo về UI events người dùng, bạn cần phải làm một trong hai điều:

- Xác định một sự kiện nghe và đăng ký nó với các View. Khác thường hơn không, đây là cách bạn sẽ lắng nghe cho các sự kiện. Các class View có chứa một tập hợp các giao diện lồng nhau đặt tên **On<something>Listener,** đều có một phương pháp gọi lại được gọi là **On<something>()** Ví dụ. **View.OnClickListener** (để xử lý "nhấp chuột" trên một View), **View.OnTouchListener** (để xử lý các sự kiện màn hình cảm ứng trong một View), và **View.OnKeyListener** (để xử lý thiết bị ép quan trọng trong một View). Vì vậy nếu bạn muốn View của bạn được thông báo khi nó là " clicked" (chẳng hạn như khi một nút được chọn), thực hiện và xác định **OnClickListener** của nó gọi method **onClick()** (nơi bạn thực hiện các hành động sau khi nhấp chuột), và đăng ký nó vào Xem với **setOnClickListener()**
- Ghi đè một callback method hiện cho View. Đây là những gì bạn nên làm gì khi bạn đã thực hiện lớp View của riêng bạn và muốn lắng nghe cho các sự kiện cụ thể xảy ra trong nó. Ví dụ về các sự kiện bạn có thể xử lý bao gồm màn hình là touched **onTouchEvent()** khi trackball là di chuyển **onTrackballEvent()** hoặc khi một phím trên thiết bị được nhấn **onKeyDown()**. Điều này cho phép bạn xác định các hành vi mặc định cho từng sự kiện bên trong tuỳ chỉnh View của bạn và xác định xem sự kiện này cần được thông qua ngày để View con khác. Một lần nữa, đây là những callbacks View class, do đó, cơ hội duy nhất của bạn để xác định đó là khi bạn xây dựng một phần tùy chỉnh.

Menus

Menu đơn có một phần quan trọng của giao diện người dùng trong một ứng dụng. Menus cung cấp một giao diện đáng tin cậy cho thấy rằng các chức năng ứng dụng và cài đặt. Trong trình đơn ứng dụng phổ biến nhất là tiết lộ bằng cách bấm phím MENU trên thiết bị. Tuy nhiên, bạn cũng có thể thêm Context Menus, có thể hiển

thị khi người sử dụng máy nhấn và nắm giữ phím trên một mục. Thực đơn cũng được hệ thống phân cấp cấu trúc bằng cách sử dụng một xem, nhưng bạn không xác định cấu trúc này cho mình.

Thay vào đó, bạn xác định **onCreateOptionsMenu()** hoặc **onCreateContextMenu()** gọi method cho hoạt động của bạn và tuyên bố các mục mà bạn muốn bao gồm trong menu của bạn. Trong một thời gian thích hợp, Android sẽ tự động tạo ra hệ thống View phân cấp cần thiết cho menu, và rút ra mỗi trong mỗi menu items đó.

Menus cũng xử lý các sự kiện riêng của nó, do đó không cần phải đăng ký sự kiện listeners trên các item trong menu của bạn. Khi một item trong menu của bạn được chọn, **onOptionsItemSelected()** hoặc **onOptionsItemSelected()** **onContextItemSelected()** method **onContextItemSelected()** sẽ được gọi bằng framework. Và cũng giống như layout của bạn, bạn có tùy chọn để khai báo các menu item cho bạn trong một tệp tin XML. Đọc Tạo Menus để tìm hiểu thêm.

Adapters

Thỉnh thoảng bạn sẽ muốn populate một view group với một số thông tin mà không thể hard-coded được, thay vào đó, bạn muốn bind để xem một nguồn dữ liệu bên ngoài. Để làm điều này, bạn sử dụng một AdapterView xem như là view group của bạn và View con được khởi tạo và populated với dữ liệu từ Adapter. Các đối tượng AdapterView là một implementation của ViewGroup xác định những view con của nó dựa trên một đối tượng Adapter nhất định. Adapter các hành vi như là chuyển phát nhanh giữa các nguồn dữ liệu của bạn (có lẽ là một mảng của chuỗi bên ngoài) và AdapterView, hiển thị nó trong đó. Có một số hiện thực của class Adapter, cho nhiệm vụ cụ thể, chẳng hạn như CursorAdapter việc đọc dữ liệu cơ sở dữ liệu từ một Cursor, hoặc một ArrayAdapter đọc từ một mảng tùy ý. Để tìm hiểu thêm về cách sử dụng một adapter cho populate views của bạn, hãy đọc đóng vào dữ liệu với AdapterView.

Styles and Themes

Có lẽ bạn không hài lòng với dáng vẻ của các widgets tiêu chuẩn. Để sửa đổi chúng, bạn có thể tạo một số style riêng và chủ đề của bạn.

- Một Style là một tập hợp của một hay nhiều thuộc tính định dạng mà bạn có thể áp dụng như một đơn vị đến các yếu tố cá nhân trong layout của bạn. Ví dụ, bạn có thể xác định một Style chỉ định một văn bản kích thước và màu sắc nhất định, sau đó áp dụng nó để chỉ các yếu tố View cụ thể.
- Một Theme là một tập hợp của một hay nhiều thuộc tính định dạng mà bạn có thể áp dụng như một đơn vị đến tất cả các hoạt động trong một ứng

dụng, hoặc chỉ hoạt động đơn lẻ. Ví dụ, bạn có thể định nghĩa một theme mà bộ màu sắc cụ thể cho khung cửa sổ và nền bảng, và đặt kích cỡ chữ và màu sắc cho các menu. Theme này sau đó có thể được áp dụng cho các hoạt động cụ thể hoặc ứng dụng toàn bộ. Styles and themes là nguồn tài nguyên.

Android cung cấp một số kiểu mặc định và style and themes mà bạn có thể sử dụng, hoặc bạn có thể phát triển riêng tài nguyên style and theme của bạn. Learn more about using styles and themes in the Applying Styles and Themes document. Tìm hiểu thêm về cách sử dụng phong cách và chủ đề trong các ứng dụng Styles và tài liệu đề.

Kỹ thuật xử lý bộ nhớ trong Android Mobile

Android ứng dụng, trên T-Mobile G1, được giới hạn ít nhất 16 MB của heap. Đó là bộ nhớ cho điện thoại và rất ít cho những gì một số nhà phát triển muốn đạt được. Thậm chí nếu bạn không kế hoạch sử dụng tất cả các bộ nhớ này, bạn nên sử dụng ít nhất có thể cho các ứng dụng khác chạy mà không phải đào thải chúng.
Các ứng dụng khác Android có thể giữ trong bộ nhớ, nhanh hơn, cho người sử dụng để chuyển đổi giữa các ứng dụng của mình.

Là một phần của công việc của tôi, tôi chạy vào các vấn đề rò rỉ bộ nhớ trong các ứng dụng Android và chúng tôi thấy được phần lớn thời gian do các sai lầm: giữ quá lâu cho tham chiếu đến một Context. Trên Android, một Context được sử dụng cho nhiều hoạt động, nhưng chủ yếu là để tải và truy cập tài nguyên. Đây là lý do tại sao tất cả các widgets nhận được một số Context trong xây dựng của họ. Trong một ứng dụng bình thường của Android, bạn thường có hai loại Context Activity và Application. Nó thường là một trong những cái đầu tiên mà nhà phát triển đi đến các classes và methods cần một Context :

```
@Override
protected void onCreate(Bundle state) {
   super.onCreate(state);

    TextView label = new TextView(this);
    label.setText("Leaks are bad");

     setContentView(label);
}
```

Điều này có nghĩa là Views có một tham chiếu đến toàn bộ hoạt động và do đó bất cứ điều gì đến Activity đang nắm giữ; thường được View toàn bộ hệ thống phân cấp và tất cả các nguồn tài nguyên của nó. Vì vậy, nếu bạn bị rò rỉ các Context ("leak" có nghĩa là như vậy bạn giữ một tham chiếu đến nó, ngăn chặn các GC từ

thu thập), bạn bị rò rỉ rất nhiều bộ nhớ. Rò rỉ toàn bộ hoạt động có thể thực sự dễ dàng nếu bạn không cẩn thận. Khi định hướng sẽ thay đổi hệ thống, theo mặc định, tiêu huỷ activity hiện tại và tạo ra một cái mới trong khi giữ trạng thái. Trong khi làm điều đó, Android sẽ tải lại giao diện của ứng dụng từ các nguồn tài nguyên. Bây giờ hãy tưởng tượng bạn đã viết một ứng dụng với một bitmap lớn mà bạn không muốn thay phiên tất cả. Cách đơn giản nhất để giữ cho nó xung quanh và không phải tải xoay vòng lại mỗi ngày nên để nó trong static field:

```
private static Drawable sBackground;

@Override
protected void onCreate(Bundle state) {
    super.onCreate(state);

    TextView label = new TextView(this);
    label.setText("Leaks are bad");

    if (sBackground == null) {
        sBackground = getDrawable(R.drawable.large_bitmap);
    }
    label.setBackgroundDrawable(sBackground);

    setContentView(label);
}
```

Mã này là rất nhanh và cũng rất sai, nó tạo ra leak các hoạt động đầu tiên khi thay đổi định hướng. Khi một Drawable được đính kèm để View, view được thiết lập như là một callback trên drawable. Trong đoạn mã trên, điều này có nghĩa là drawable có một tham chiếu đến TextView mà tự nó có một tham chiếu đến các hoạt động (trong Context mà lần lượt có tham chiếu khá nhiều đến bất cứ điều gì (tùy thuộc vào mã của bạn.)

Ví dụ này là một trong những trường hợp đơn giản nhất leaking của các Context và bạn có thể xem thế nào, chúng tôi đã làm việc xung quanh nó trong mã nguồn của các màn hình chủ (tìm unbindDrawables() method) bằng cách thiết lập các callbacks drawables lưu trữ để null khi Activity này phá hủy. Điều thú vị, có những trường hợp bạn có thể tạo ra một chuỗi các hoàn cảnh bị rò rỉ, và khó khăn. Nó làm cho bạn hết bộ nhớ khá nhanh chóng.

Có hai cách dễ dàng để tránh rò rỉ bộ nhớ ngữ cảnh có liên quan. The most obvious one is to avoid escaping the context outside of its own scope. Một rõ ràng nhất là để tránh bối cảnh bên ngoài phạm vi của chính mình.

Ví dụ trên cho thấy trường hợp của một tham chiếu tĩnh nhưng bên trong classe và tiềm ẩn reference cho class bên ngoài có thể nguy hiểm như nhau. The second solution is to use the Application context. Giải pháp thứ hai là sử dụng ngữ cảnh Application.

Context này sẽ không sao miễn là ứng dụng của bạn vẫn hoạt động và không phụ thuộc vào chu kỳ hoạt động của activities. Nếu bạn có kế hoạch duy trì lâu dài sống các đối tượng mà cần một bối cảnh đó, nhớ đối tượng áp dụng. Bạn có thể có được nó một cách dễ dàng bằng cách gọi Context.getApplicationContext() hoặc Activity.getApplication()

Tóm lại, để tránh những context có liên quan rò rỉ bộ nhớ, xin hãy nhớ:

- Không giữ tham chiếu đến context activity quá lâu(một tham chiếu đến một activity nên có chu kỳ sống tương tự như các hoạt động chính nó)
- Hãy thử sử dụng các context của application thay vì context của activity
- Tránh non-static bên trong các classes của một activity nếu bạn không kiểm soát vòng đời của nó, sử dụng một class static bên trong và thực hiện một tham chiếu yếu đến activity bên trong. Các giải pháp cho vấn đề này là sử dụng một static class bên trong với một WeakReference cho class bên ngoài, như thực hiện trong ViewRoot và lớp đó của nó chẳng hạn
- Một bộ thu rác không phải là một bảo hiểm chống rò rỉ bộ nhớ.

Khai báo Layout trong Android

Layout của bạn là kiến trúc cho các giao diện người dùng trong một Activity. Nó xác định cơ cấu và nắm giữ các yếu tố xuất hiện cho người dùng thấy. Bạn có thể khai báo layout của bạn theo hai cách:

- **Khai báo phần tử UI trong XML.** Android cung cấp một vốn từ vựng đơn giản XML tương ứng với các lớp View và subclasses, chẳng hạn như widget và layout.
- **Khởi tạo các yếu tố layout trong thời gian chạy.** Ứng dụng của bạn thể tạo ra đối tượng View và ViewGroup (và thao tác các properties của nó) theo chương trình.

Viết XML

Sử dụng vốn từ vựng của Android XML, bạn có hể nhanh chóng thiết kế UI layout và các thành phần chứa chúng, giống như cách bạn tạo các trang web trong HTML - với một loạt các phần tử lồng nhau.

Mỗi tập tin layout phải bao gồm phần tử gốc một cách chính xác, và phải là một đối tượng View hoặc ViewGroup. Một khi bạn đã xác định các phần tử gốc, bạn có thể thêm đối tượng layout bổ sung hoặc các widget như là các phần tử con cho

từng bước xây dựng một hệ thống View định nghĩa layout của bạn. Ví dụ, đây là có layout XML có sử dụng một LinearLayout dọc cho tổ chức một TextView và một Button:

```
<?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:id="@+id/text"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello, I am a TextView" />
    <Button android:id="@+id/button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello, I am a Button" />
</LinearLayout>
```

Sau khi bạn đã khai báo layout của bạn trong XML, lưu tập tin với phần mở rộng .xml, trong thư mục res/layout/ dự án Android của bạn, do đó, để đúng cách biên dịch.

Nạp tài nguyên XML

Khi bạn biên dịch ứng dụng của bạn, mỗi tập tin XML layout là biên dịch vào một nguồn tài nguyên View. Bạn cần phải tải các nguồn tài nguyên layout từ mã ứng dụng của bạn, trong **Activity.onCreate() callback implementation** của bạn. Làm như vậy bằng cách gọi **setContentView(),** qua nó tham chiếu cho tài nguyên layout của bạn trong các hình thức: **R.layout.layout_file_name**

Ví dụ : nếu XML layout của bạn được lưu như **main_layout.xml**, bạn sẽ tải nó cho Activity của bạn như sau:

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

Thuộc tính của Layout khai báo trong Android

Mỗi đối tượng View và ViewGroup hỗ trợ đa dạng của riêng nó các thuộc tính XML. Một số thuộc tính được cụ thể cho một đối tượng View (ví dụ, TextView hỗ trợ thuộc tính textSize), nhưng các thuộc tính này cũng được kế thừa bởi bất cứ đối tượng View có thể mở rộng lớp này. Một số được sử dụng chung cho tất cả đối tượng View, bởi vì nó được thừa kế từ lớp View gốc(giống như các thuộc tính id). Và, các thuộc tính khác được coi là "tham số layout", mà được thuộc tính mô tả định hướng layout nhất định của đối tượng View, theo định nghĩa của đối tượng đó là đối tượng ViewGroup parent.

ID

Bất kỳ đối tượng View có thể có ID số nguyên liên kết với nó, để nhận diện ra các View bên trong cây. Khi ứng dụng biên dịch, ID này là tham chiếu như là một số nguyên, nhưng ID thường được giao trong layout các file XML như một chuỗi, trong thuộc tính id. Đây là thuộc tính chung cho tất cả đối tượng View (được định nghĩa bởi lớp View) và bạn sẽ sử dụng nó rất thường xuyên. Cú pháp của có ID, bên trong thẻ XML là:

android:id="@+id/my_button"

Tại biểu tượng (@) tại đầu của chuỗi cho thấy rằng cú pháp XML cần phân tích và mở rộng phần còn lại của chuỗi ID và xác định nó như là một nguồn tài nguyên ID. Các biểu tượng cộng (+) có nghĩa rằng đây là một tên tài nguyên mới phải được tạo ra và thêm để nguồn tài nguyên của chúng tôi (trong file R.java). Có một số tài nguyên ID khác được cung cấp bởi các khuôn khổ Android.

android:id="@android:id/empty"

Để tạo ra views và tham khảo chúng từ ứng dụng, một mô hình phổ biến là:

- Xác định group/widget trong tập tin layout và gán cho nó một mã số duy nhất:

```
<Button android:id="@+id/my_button"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/my_button_text"/>
```

- Sau đó tạo một instance của đối tượng group và nắm bắt nó từ layout (thông thường dùng trong method onCreate()):

```
Button myButton = (Button) findViewById(R.id.my_button);
```

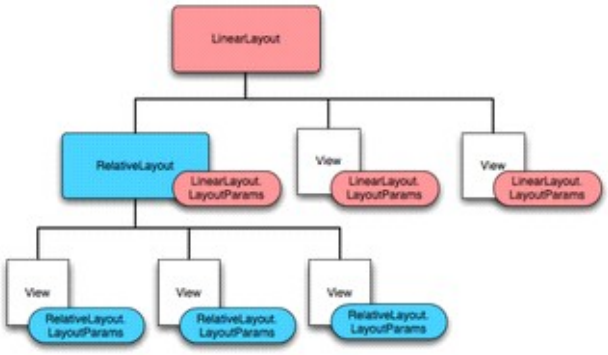
Xác định ID cho các đối tượng group là quan trọng khi tạo ra một RelativeLayout. Trong cách bố trí tương đối, views sibling có thể xác định layout của mình tương đối khác group sibling, đó là tham chiếu bởi ID duy nhất.

Một ID không cần phải là duy nhất trong suốt toàn bộ cây, nhưng nó cần là duy nhất trong một phần của cây bạn đang tìm kiếm (thường có thể là toàn bộ cây, vì vậy tốt nhất cho là hoàn toàn độc nhất khi có thể).

Layout Parameters

layout các thuộc tính có tên là XML định nghĩa các thông số layout_something layout cho các View thích hợp cho các ViewGroup trong đó cư trú.

Mỗi lớp ViewGroup thực hiện mở rộng một lớp ViewGroup.LayoutParams lồng nhau. Lớp này có chứa các loại properties xác định kích thước và vị trí cho từng group child, như thích hợp cho nhóm group. Như bạn nhìn thấy ở hình 1, các nhóm parent group thông số xác định layout cho từng group con(bao gồm cả nhóm group child).



(Ảnh minh họa được sử dụng từ
http://developer.android.com/images/layoutparams.png)

Lưu ý đó tất cả các lớp con LayoutParams có cú pháp của riêng mình để thiết lập các giá trị. Mỗi phần tử child phải xác định LayoutParams thích hợp cho parent của nó, mặc dù nó cũng có thể định nghĩa LayoutParams khác nhau cho con của nó.

Tất cả các view group bao gồm chiều rộng và chiều cao (layout_width và layout_height), và mỗi group bắt buộc định nghĩa cho nó. Rất nhiều LayoutParams cũng bao gồm margins và borders.

Kỷ thuật sử dụng phiên bản API cũ và mới trên Android Mobile

Một loạt các thiết bị Android được hỗ trợ đang sẵn có cho người tiêu dùng trên thế giới. Trên các thiết bị, một loạt các phiên bản nền tảng Android đang sử dụng, một số chạy phiên bản mới nhất, những người khác đang chạy phiên bản cũ. Là một nhà phát triển, bạn cần phải xem xét các cách tiếp cận tương thích trong ứng dụng

của bạn - bạn có muốn cho phép các ứng dụng của bạn chạy trên tất cả các thiết bị, hoặc chỉ những người đang chạy phần mềm mới nhất? Trong một số trường hợp, nó sẽ hữu ích để sử dụng các API mới hơn trên các thiết bị có hỗ trợ nó, trong khi tiếp tục hỗ trợ thiết bị cũ.

Đặt minSdkVersion

Nếu việc sử dụng một API mới được tách rời vào ứng dụng - có lẽ bạn cần để ghi lại bằng cách sử dụng một API được giới thiệu trong Android 1,5 (API Level 3) - bạn nên thêm một <android:minSdkVersion> đến ứng dụng, để đảm bảo ứng dụng của bạn sẽ không được cài đặt trên các thiết bị cũ.
Ví dụ, nếu ứng dụng của bạn phụ thuộc vào một API được giới thiệu trong API Level 3, bạn sẽ chỉ định "3" là giá trị của các phiên bản SDK tối thiểu:

```
<manifest>
    ... ...
    <uses-sdk android:minSdkVersion="3" />
    ... ...
</manifest>
```

Tuy nhiên, nếu bạn muốn thêm một tính năng hữu ích nhưng không cần thiết, chẳng hạn như popping lập một bàn phím trên màn hình ngay cả khi có một bàn phím phần cứng có sẵn, bạn có thể viết chương trình của bạn một cách cho phép nó sử dụng các tính năng mới hơn mà không thất bại trên thiết bị cũ.

Sử dụng reflection

Giả sử có một simple mới gọi bạn muốn sử dụng, như android.os.Debug.dumpHprofData(String filename). Các lớp debug đã tồn tại từ Android 1.0, nhưng là phương pháp mới trong Anroid 1.5 (API Level 3). Nếu bạn cố gắng gọi nó trực tiếp, ứng dụng của bạn sẽ không chạy trên các thiết bị chạy Android 1.0 hoặc sớm hơn.
Cách đơn giản để gọi method là thông qua reflection. Điều này đòi hỏi phải thực hiện một thời gian tra cứu và kết quả là bộ nhớ đệm trong một đối tượng Method. Sử dụng phương pháp là một vấn đề của gọi Method.invoke và kết quả un-boxing. Xem xét sau đây:

```
public class Reflect {
    private static Method mDebug_dumpHprofData;

    static {
        initCompatibility();
    };
```

```
    private static void initCompatibility() {
          try {
                mDebug_dumpHprofData = Debug.class.getMethod(
                "dumpHprofData", new Class[] { String.class }  );
          /* success, this is a newer device */
          } catch (NoSuchMethodException nsme) {
                /* failure, must be older device */
          }
    }

    private static void dumpHprofData(String fileName) throws IOException
{
          try {
               mDebug_dumpHprofData.invoke(null, fileName);
          } catch (InvocationTargetException ite) {
               /* unpack original exception when possible */
               Throwable cause = ite.getCause();
               if (cause instanceof IOException) {
                    throw (IOException) cause;
               } else if (cause instanceof RuntimeException) {
                    throw (RuntimeException) cause;
               } else if (cause instanceof Error) {
                    throw (Error) cause;
               } else {
                   /* unexpected checked exception; wrap and re-throw */
                   throw new RuntimeException(ite);
               }
          } catch (IllegalAccessException ie) {
                System.err.println("unexpected " + ie);
          }
    }

    public void fiddle() {
         if (mDebug_dumpHprofData != null) {
            /* feature is supported */
            try {
                 dumpHprofData("/sdcard/dump.hprof");
            } catch (IOException ie) {
                 System.err.println("dump failed!");
            }
         } else {
             /* feature not supported, do something else */
```

```
            System.out.println("dump not supported");
        }
    }
}
```

Mã này sử dụng một initializer tĩnh để gọi initCompatibility mà không tra cứu các phương pháp. Nếu điều đó thành công, nó sử dụng một phương pháp riêng với ngữ nghĩa giống như (lập luận ban đầu, giá trị trả về, kiểm tra trường hợp ngoại lệ) để thực hiện cuộc gọi. Giá trị trả lại (nếu nó có một) và trường hợp ngoại lệ được mở gói và trả lại một cách bắt chước bản gốc. Các phương pháp fiddle minh chứng cách logic ứng dụng sẽ chọn để gọi API mới hoặc làm điều gì đó khác nhau dựa trên sự hiện diện của phương pháp mới.

Đối với mỗi phương pháp bổ sung mà bạn muốn gọi, bạn sẽ thêm một private Method field, initializer field, và wrapper gọi class.

Cách tiếp cận này trở nên phức tạp hơn một chút khi phương pháp này là declared trong một undefined class trước đây. Nó cũng chậm hơn để gọi Method.invoke() hơn là để gọi phương thức trực tiếp. Những vấn đề này có thể được giảm nhẹ bằng cách sử dụng một lớp wrapper.

Sử dụng một lớp wrapper

Ý tưởng là để tạo ra một class kết thúc tốt đẹp mà tất cả các API mới tiếp xúc bởi một class mới hay class hiện có. Mỗi phương thức trong lớp wrapper chỉ các cuộc gọi thông qua method thực tế tương ứng và trả về kết quả tương tự.

Nếu target class và method tồn tại, bạn sẽ có được những hành vi đó và bạn sẽ nhận được bằng cách gọi các class trực tiếp, với một lượng nhỏ trên không từ cuộc gọi phương thức bổ sung. Nếu target class hoặc method không tồn tại, việc khởi tạo của class wrapper thất bại, và ứng dụng của bạn biết rằng nó nên tránh sử dụng các cuộc gọi mới hơn. Giả sử class này mới được thêm vào:

```
public class NewClass {
    private static int mDiv = 1;

    private int mMult;

    public static void setGlobalDiv(int div) {
        mDiv = div;
    }

    public NewClass(int mult) {
```

```
        mMult = mult;
    }

    public int doStuff(int val) {
        return (val * mMult) / mDiv;
    }
}
```

Chúng ta sẽ tạo ra một lớp wrapper cho nó:

```
class WrapNewClass {
    private NewClass mInstance;

    /* class initialization fails when this throws an exception */
    static {
        try {
            Class.forName("NewClass");
        } catch (Exception ex) {
            throw new RuntimeException(ex);
        }
    }

    /* calling here forces class initialization */
    public static void checkAvailable() {}

    public static void setGlobalDiv(int div) {
        NewClass.setGlobalDiv(div);
    }

    public WrapNewClass(int mult) {
        mInstance = new NewClass(mult);
    }

    public int doStuff(int val) {
        return mInstance.doStuff(val);
    }
}
```

Điều này có một method cho mỗi constructor và method trong initializer, cộng với static initializer mà tests cho sự hiện diện của class mới. Nếu lớp mới không có, initialization của WrapNewClass không thành công, đảm bảo rằng các class wrapper không thể được sử dụng vô tình. The checkAvailable method is used as a simple way to force class initialization. Các phương pháp checkAvailable được sử

dụng như là một cách đơn giản đến class initialization. Chúng tôi sử dụng nó như thế này:

```
public class MyApp {
    private static boolean mNewClassAvailable;

     /* establish whether the "new" class is available to us */
    static {
        try {
            WrapNewClass.checkAvailable();
            mNewClassAvailable = true;
         } catch (Throwable t) {
            mNewClassAvailable = false;
        }
    }

     public void diddle() {
         if (mNewClassAvailable) {
              WrapNewClass.setGlobalDiv(4);
              WrapNewClass wnc = new WrapNewClass(40);
                System.out.println("newer API is available - " +
wnc.doStuff(10));
         } else {
             System.out.println("newer API not available");
         }
    }
}
```

Nếu cuộc gọi đến checkAvailable thành công, chúng ta biết những class mới là một phần của hệ thống. Nếu không, chúng tôi biết những lớp học không phải, và sữa chữa cho phù hợp. Cần lưu ý rằng các cuộc gọi đến checkAvailable sẽ không thành công trước khi nó bắt đầu ngay cả nếu bytecode quyết định rằng nó không muốn chấp nhận một class mà có tham chiếu đến một class không tồn tại. Cách mã này có cấu trúc, kết quả cuối cùng là trường hợp ngoại lệ như nhau cho dù đến từ các xác hoặc từ các cuộc gọi đến Class.forName

Khi wrapping một class bây giờ có một phương pháp mới, bạn chỉ cần đưa các phương pháp mới vào lớp wrapper. Gọi trực tiếp các method cũ. Các initializer static trong WrapNewClass sẽ được tăng cường để làm một thời gian kiểm tra với reflection.

Thử nghiệm là chìa khóa

Bạn phải thử nghiệm ứng dụng của bạn trên tất cả các phiên bản của Android framework mà dự kiến sẽ hỗ trợ nó. Theo định nghĩa, trên mỗi hành vi ứng dụng của bạn sẽ khác nhau. Bạn nên nhớ rằng: nếu bạn không thử nó, nó không hoạt động.

Bạn có thể kiểm tra tính tương thích ngược bằng cách chạy ứng dụng của bạn trong một mô phỏng sử dụng một phiên bản cũ hơn của nền tảng này. Android SDK cho phép bạn làm điều này một cách dễ dàng bằng cách tạo ra "Thiết bị Android ảo" với API mức độ khác nhau. Một khi bạn tạo AVDs, bạn có thể thử nghiệm ứng dụng của bạn với phiên bản cũ và mới của hệ thống, có lẽ nó chạy side-by-side để xem sự khác biệt. Thông tin thêm về AVDs giả lập có thể được tìm thấy trong tài liệu AVD và từ emulator -help-virtual-device

Tìm hiểu quản lý thiết bị trong Android

Android 2,2 giới thiệu hỗ trợ cho các ứng dụng doanh nghiệp bằng cách cung cấp các thiết bị Android chính API. Các thiết bị chính API cung cấp các tính năng quản lý thiết bị ở cấp độ hệ thống. Những API này cho phép bạn tạo các ứng dụng bảo mật biết rằng có ích trong các thiết lập doanh nghiệp, trong đó các chuyên gia CNTT cần kiểm soát nhiều hơn các thiết bị nhân viên. Ví dụ, xây dựng trong ứng dụng Email Android đã thừa hưởng các API mới để cải thiện hỗ trợ Exchange. Thông qua ứng dụng Email, Exchange quản trị có thể thực thi các chính sách mật khẩu - bao gồm cả chữ và số mật khẩu hoặc số PIN - qua nhiều thiết bị. Các quản trị viên cũng có thể xóa sạch từ xa (có nghĩa là, khôi phục lại mặc định nhà máy trên) thiết bị cầm tay bị mất hoặc bị đánh cắp. Exchange người dùng có thể đồng bộ hóa email của họ và dữ liệu lịch.

Tài liệu này được thiết kế để phát triển những người muốn phát triển các giải pháp doanh nghiệp cho các thiết bị hỗ trợ Android. Nó bàn về các tính năng khác nhau được cung cấp bởi chính thiết bị API để cung cấp bảo mật mạnh mẽ hơn cho các thiết bị nhân viên được trang bị Android.

Tổng quan API quản lý thiết bị

Dưới đây là ví dụ về các loại ứng dụng mà có thể sử dụng các API thiết bị chính :

- Email khách hàng.
- An ninh ứng dụng mà xóa từ xa.
- Thiết bị dịch vụ quản lý và ứng dụng.

Nó làm việc như thế nào ?

Bạn sử dụng các thiết bị API chính để viết các ứng dụng thiết bị admin mà người dùng cài đặt trên thiết bị của họ. Các ứng dụng thiết bị admin thực thi các chính sách mong muốn. Đây là cách nó hoạt động:

- Một viên quản trị hệ thống viết một ứng dụng thiết bị admin mà thực thi từ remote/local chính sách thiết bị an ninh. Những chính sách này có thể được mã hóa cứng vào ứng dụng, hoặc các ứng dụng tự động có thể lấy chính sách từ một máy chủ của bên thứ ba.
- Các ứng dụng được cài đặt trên thiết bị của người dùng. Android hiện tại không có một giải pháp cung cấp tự động. Một số trong những cách một quản trị hệ thống có thể phân phối các ứng dụng cho người dùng như sau:
    - Android Market.
    - Cho phép cài đặt phi thị trường.
    - Phân phối các ứng dụng thông qua các phương tiện khác, chẳng hạn như email hoặc các trang web.
- Hệ thống nhắc nhở người dùng để kích hoạt các thiết bị ứng dụng admin. Làm thế nào và khi điều này xảy ra phụ thuộc vào ứng dụng được thực hiện.
- Một khi người dùng kích hoạt các ứng dụng thiết bị admin, họ có thể chính sách của mình. Tuân thủ các chính sách đó thường trao lợi ích, chẳng hạn như truy cập vào các hệ thống nhạy cảm và dữ liệu.

Nếu người dùng không cho phép các ứng dụng thiết bị admin, nó vẫn còn trên thiết bị, nhưng trong một trạng thái không hoạt động. Người dùng sẽ không bị áp dụng các chính sách của mình, và họ sẽ ngược lại không nhận được bất kỳ của các ứng dụng của các lợi ích, ví dụ, họ có thể không có khả năng đồng bộ hóa dữ liệu.

Nếu người dùng không tuân thủ các chính sách (ví dụ, nếu người dùng đặt mật khẩu mà vi phạm các hướng dẫn), đó là vào các ứng dụng để quyết định cách xử lý này. Tuy nhiên, thông thường điều này sẽ dẫn đến việc người dùng không thể đồng bộ hóa dữ liệu.

Nếu thiết bị của một nỗ lực để kết nối đến một máy chủ mà không đòi hỏi các chính sách hỗ trợ trong API quản lý thiết bị, kết nối sẽ không được phép. Các API thiết bị chính hiện tại không cho phép cung cấp một phần. Nói cách khác, nếu một thiết bị (ví dụ, một thiết bị di sản) không hỗ trợ tất cả các chính sách quy định, không có cách nào để cho phép các thiết bị để kết nối.

Nếu một thiết bị chứa nhiều kích hoạt các ứng dụng quản trị, các chính sách khắt khe được thi hành. Không có cách nào để nhắm mục tiêu một ứng dụng cụ thể admin.

Để gỡ bỏ cài đặt một ứng dụng thiết bị hiện có admin, người dùng cần phải đầu tiên đăng ký của các ứng dụng như là một quản trị viên.

Chính sách
Trong các thiết lập doanh nghiệp, đó là trường hợp thường thấy các thiết bị nhân viên phải tuân thủ nghiêm ngặt của một bộ chính sách cai trị sử dụng của thiết bị. Các thiết bị chính API hỗ trợ các chính sách được liệt kê trong Bảng 1. Lưu ý rằng các thiết bị hiện hành API chỉ hỗ trợ mật khẩu để khóa màn hình:

- **Mật khẩu được kích hoạt:** Yêu cầu các thiết bị yêu cầu mã PIN hay mật khẩu.
- **Chiều dài mật khẩu tối thiểu:** Thiết lập số lượng ký tự cho yêu cầu của mật khẩu. Ví dụ, bạn có thể yêu cầu mã PIN hoặc mật khẩu có ít nhất sáu ký tự.
- **Chữ số mật khẩu yêu cầu** Yêu cầu có mật khẩu có kết hợp các chữ cái và số. Họ có thể bao gồm các nhân vật tượng trưng.
- **Tối đa nỗ lực không thành công mật khẩu** Chỉ định bao nhiêu lần một người sử dụng có thể nhập mật khẩu sai trước khi thiết bị lau dữ liệu của nó. Các thiết bị chính API cũng cho phép quản trị từ xa đặt lại thiết bị để mặc định của nhà máy. Điều này đảm bảo dữ liệu trong trường hợp điện thoại bị mất hoặc bị đánh cắp.
- **Thiết lập thời gian** tối đa không hoạt động khóa chiều dài của thời gian kể từ khi người sử dụng cuối cùng chạm vào màn hình hoặc ấn một nút trước khi thiết bị khóa màn hình. Khi điều này xảy ra, người dùng cần phải nhập mã PIN hay mật khẩu của mình một lần nữa trước khi họ có thể sử dụng thiết bị của họ và truy cập dữ liệu. Giá trị có thể được từ 1 đến 60 phút.

Các tính năng khác
Ngoài việc hỗ trợ các chính sách được liệt kê trong bảng trên, các thiết bị chính API cho phép bạn làm như sau:

- Nhắc nhở người dùng để thiết lập một mật khẩu mới.
- Khóa thiết bị ngay lập tức.
- Xóa các dữ liệu của thiết bị (có nghĩa là, khôi phục lại các thiết bị về mặc định của nó).

Phát triển Android trong các IDE khác

Khuyến nghị nên phát triển một ứng dụng Android là sử dụng Eclipse với plugin ADT .. Các plugin ADT cung cấp chỉnh sửa, xây dựng, gỡ lỗi, và đóng gói apk và chức năng ký quyền tích hợp vào IDE.

Tuy nhiên, nếu bạn thích phát triển các ứng dụng của bạn trong một IDE, như IntelliJ, hoặc trong một trình soạn thảo cơ bản, chẳng hạn như Emacs, bạn có thể làm điều đó thay thế. SDK này bao gồm tất cả các công cụ bạn cần thiết lập một dự án Android, xây dựng nó, gỡ lỗi nó và sau đó đóng gói để phân phối. This document is your guide to using these tools. Tài liệu này là hướng dẫn của bạn để sử dụng những công cụ này.

Công cụ cần thiết
Khi phát triển trong các IDE hoặc biên tập viên khác so với Eclipse, bạn sẽ yêu cầu sự quen thuộc với công cụ SDK Android sau đây:

Android
Để tạo / cập nhật các dự án Android và tạo / di chuyển / xóa AVDs.
Android Emulator
Để chạy các ứng dụng Android của bạn trên một nền tảng Android mô phỏng.
Android Debug Bridge
Để giao tiếp với giả lập của bạn hoặc kết nối thiết bị (cài đặt các ứng dụng, vỏ các thiết bị, ban hành lệnh, vv.)

Ngoài các công cụ trên, bao gồm trong các SDK, bạn sẽ sử dụng mã nguồn mở sau và bên thứ ba công cụ:

Ant :

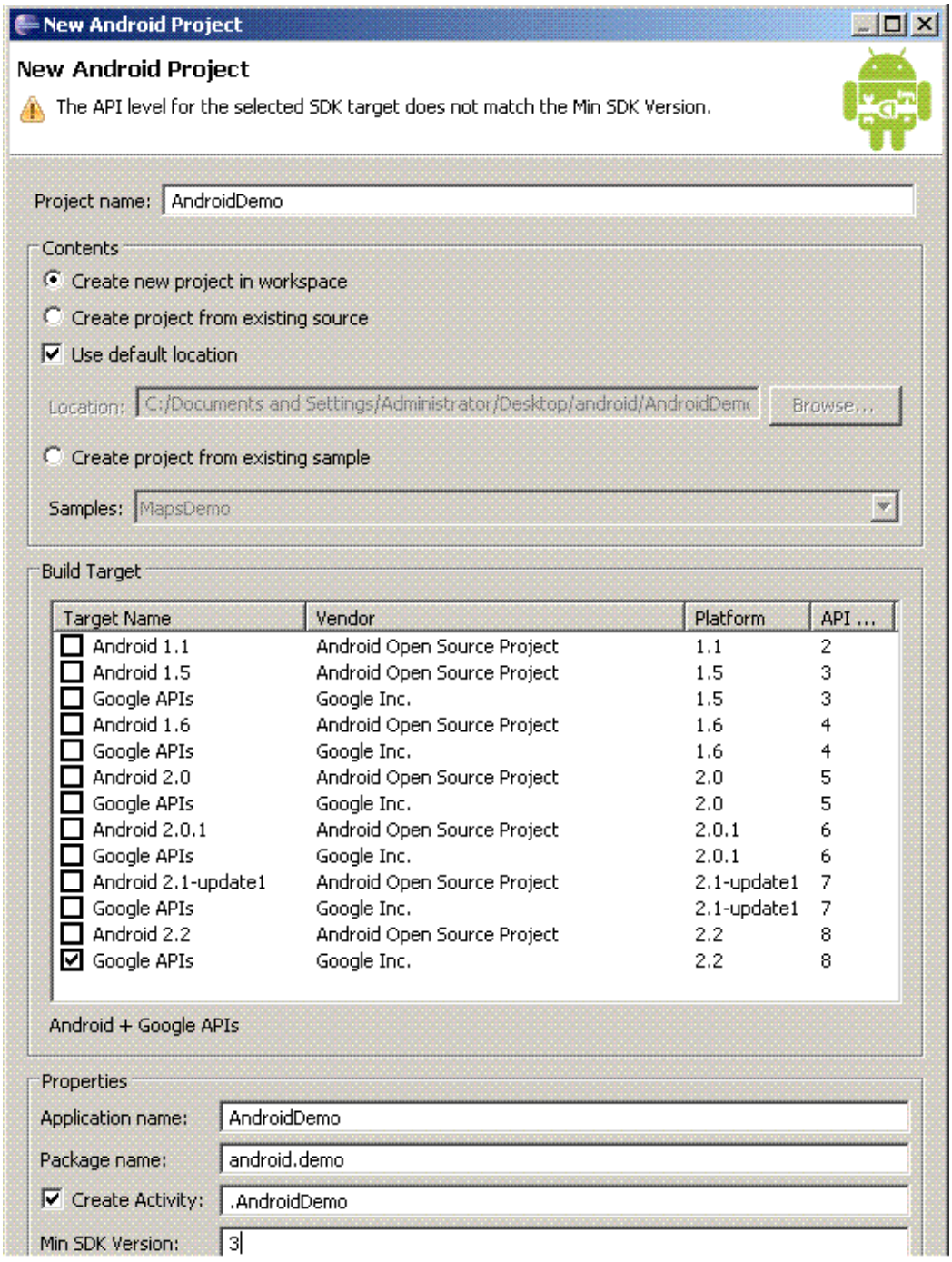
- Để biên dịch và xây dựng dự án Android của bạn thành một file apk cài đặt..

Keytool Keytool :

- Để tạo một khoá keystore và tư nhân, sử dụng để đăng file apk. Của bạn. Jarsigner (hoặc tương tự như công cụ chữ ký) .
- Để đăng file apk. Của bạn với một khóa riêng được tạo ra bởi keytool.

Trong các chủ đề tiếp theo, bạn sẽ được giới thiệu với mỗi công cụ này là cần thiết,. Để nâng cao hơn nữa các hoạt động xin vui lòng đọc tài liệu tương ứng cho mỗi công cụ.

Tạo một dự án Android



Để tạo một dự án Android, bạn phải sử dụng android công cụ. Khi bạn tạo một dự án mới với android , nó sẽ tạo ra một thư mục dự án với một số tập tin ứng dụng mặc định, các tập tin còn sơ khai, các file cấu hình và xây dựng một tập tin.

Tạo một dự án mới

Nếu bạn đang bắt đầu một dự án mới, sử dụng android create project lệnh để tạo ra tất cả các file cần thiết và thư mục.

Để tạo một dự án Android mới, mở ra một dòng lệnh, điều hướng đến các tools/ thư mục của bạn chạy SDK và:

```
android create project \
  --target <target_ID> \
  --name <your_project_name> \
  --path path/to/your/project \
  --activity <your_activity_name> \
  --package <your_package_namespace>
```

- **target** là "xây dựng mục tiêu" cho ứng dụng của bạn. Nó tương ứng với một thư viện nền tảng Android (bao gồm bất kỳ tiện ích, chẳng hạn như Google API) mà bạn muốn xây dựng các dự án của bạn chống lại. Để xem danh sách các mục tiêu có sẵn và các ID tương ứng của họ, thực hiện: android list targets .
- **name** là tên cho dự án của bạn. Nếu cung cấp, tên này sẽ được sử dụng cho apk. Bạn tên tập tin khi bạn xây dựng ứng dụng của bạn.
- **path** là vị trí của thư mục dự án của bạn. Nếu các thư mục không tồn tại, nó sẽ được tạo ra cho bạn.
- **activity** là tên mặc định của bạn Activity lớp. Đây lớp tập tin sẽ được tạo ra cho bạn bên trong /src/ / . Điều này cũng sẽ được sử dụng cho apk. bạn tên tập tin, trừ khi bạn cung cấp một các name .
- **package** là các không gian tên gói cho dự án của bạn, theo các quy tắc tương tự như đối với các gói ngôn ngữ lập trình Java.

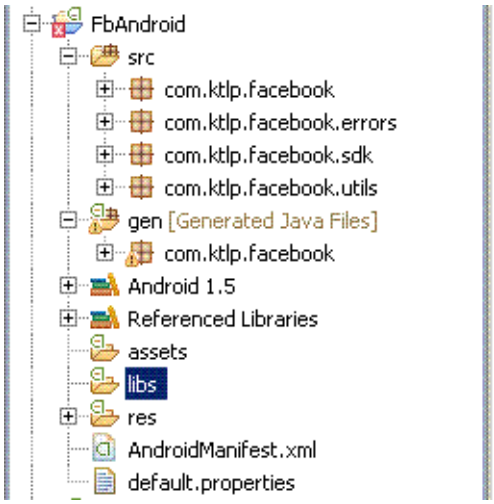
Dưới đây là ví dụ:

```
android create project \
   --target 1 \
   --name MyAndroidApp \
   --path ./MyAndroidAppProject \
   --activity MyAndroidAppActivity \
   --package com.example.myandroid
```

Công cụ này tạo ra các tập tin và thư mục sau đây:

- **AndroidManifest.xml** - Các biểu hiện tập tin ứng dụng, đồng bộ hóa với lớp cuối quy định cho dự án.
- **build.xml** - Xây dựng các tập tin cho Ant.
- **default.properties** - cho xây dựng. hệ thống properties không sửa đổi tập tin này.

- **build.properties** - tính tùy biến cho việc xây dựng hệ thống. Bạn có thể chỉnh sửa file này để ghi đè lên các thiết lập mặc định xây dựng được sử dụng bởi Ant và cung cấp một con trỏ tới keystore của bạn và bí danh chủ chốt để xây dựng các công cụ có thể đăng ký ứng dụng của bạn khi được xây dựng trong chế độ phát hành.
- **src** /your/package/namespace/ActivityName .java - The Activity class you specified during project creation. src /your/package/namespace/ActivityName .java - Các lớp cuối bạn chỉ định trong việc tạo ra dự án.
- **bin/** - Output directory for the build script. bin/
- **gen/** - Giữ Ant -tạo ra các tập tin, chẳng hạn như R.java .
- **libs/** - Giữ tin thư viện.
- **res/** - Giữ nguồn lực dự án.
- **src/** - Giữ mã nguồn.
- **tests/** - Giữ một bản sao của tất cả các-of-the trên, cho mục đích thử nghiệm.



Một khi bạn đã tạo ra dự án của bạn, bạn đã sẵn sàng để bắt đầu phát triển. Bạn có thể di chuyển thư mục dự án của bạn bất cứ nơi nào bạn muốn phát triển, nhưng hãy nhớ rằng bạn phải sử dụng Android Debug Bridge (adb) - nằm trong SDK tools/ thư mục - để gửi đơn của bạn để trình giả lập (được thảo luận sau). Vì vậy, bạn cần truy cập giữa các giải pháp dự án của bạn và các tools/ thư mục.

Chú ý: Bạn nên hạn chế di chuyển vị trí của thư mục SDK, vì điều này sẽ phá vỡ xây dựng kịch bản. (Họ sẽ cần phải tự cập nhật để phản ánh vị trí SDK mới trước khi họ sẽ làm việc một lần nữa.)

Thay đổi tiêu đề cửa sổ trong Android

Thay đổi tiêu đề cửa sổ trong một hoạt động hay trên trên toàn bộ ứng dụng của bạn có vẻ đơn giản, nhưng có rất ít tài liệu viết về nó và để làm cho nó xảy ra. Bạn cần phải xác định kiểu tùy biến và áp dụng những phong cách cho một chủ đề tùy chỉnh.

Điều này được đặt trong một tài liệu xml trong thư mục "value". Trong bản demo của nó được gọi là custom_styles.xml. Tên gọi này là tùy ý bạn có thể thay đổi.


```
<!-- Sets the text styles -->
<?xml version="1.0" encoding="utf-8"?>
<!-- Sets the text styles -->
<resources>
            <style         name="CustomWindowTitleText"
parent="android:TextAppearance.WindowTitle">
          <item name="android:textSize">20dip</item>
          <item name="android:textColor">#5599FF</item>
               <item      name="android:textStyle">bold|
italic</item>
     </style>
     <!-- Changes the background color of the title bar
-->
     <style name="CustomWindowTitleBackground">
                                                  <item
name="android:background">#222222</item>
     </style>

     <!-- Set the theme for the window title -->
     <!-- NOTE: setting android:textAppearence to style
defined above -->
               <style           name="CustomWindowTitle"
parent="android:WindowTitle">
                                                  <item
name="android:textAppearance">@style/CustomWindowTitleT
ext</item>
          <item name="android:shadowDx">0</item>
          <item name="android:shadowDy">0</item>
          <item name="android:shadowRadius">5</item>
```

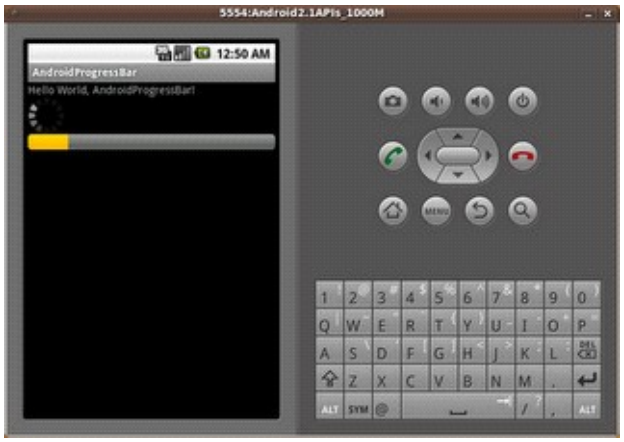
```
                                                    <item
name="android:shadowColor">#1155CC</item>
      </style>
      <!-- Override properties in the default theme -->
        <!--   NOTE:   you   must   explicitly   the
windowTitleSize property, the title bar will not re-
size automatically, text will be clipped -->
      <style name="CustomTheme" parent="android:Theme">
                                                    <item
name="android:windowTitleSize">40dip</item>
                                                    <item
name="android:windowTitleStyle">@style/CustomWindowTitl
e</item>
                                                    <item
name="android:windowTitleBackgroundStyle">@style/Custom
WindowTitleBackground</item>
      </style>
</resources>
```

Lastly, to make the change take affect you need to set the theme in the manifest file.

```
<application                android:icon="@drawable/icon"
android:label="@string/app_name"
android:theme="@style/CustomTheme">
```

Làm thế nào sử dụng ProgressBar trong Android?

ProgressBar là một chỉ báo trực quan mô tả một hành động. Nó hiển thị cho người dùng các hành động đang tiến triển, các ứng dụng có thể thay đổi chỉ số tăng giảm khi nó di chuyển về phía trước hoặc phía sau.



ví dụ này tôi dùng implemenets một ProgressBar vòng tròn mặc định và một ProgressBar ngang. Đối với các ProgressBar ngang, một chủ đề Runnable được thực hiện để gửi tin nhắn đến một xử lý để tăng tăng hoặc giảm các chỉ số của ProgressBar.

Thiết lập thông số cho main.xml :

```
< LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
android:orientation="vertical"
android:layout_width="fill_parent"
android:layout_height="fill_parent" >
< TextView android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:text="@string/hello" />
< ProgressBar android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:id="@+id/progressbar_default" />
< ProgressBar android:layout_width="fill_parent"
android:layout_height="wrap_content"
style="?android:attr/progressBarStyleHorizontal"
android:id="@+id/progressbar_Horizontal"
android:max="100" />
```

Code cho Activity :

```
package com.android.sample.AndroidProgressBar;


import android.app.Activity;
import android.os.Bundle;
import android.os.Handler;
import android.os.Message;
import android.widget.ProgressBar;
```

```
public class AndroidProgressBar extends Activity {
  ProgressBar myProgressBar;
int myProgress = 0;


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

myProgressBar=(ProgressBar)findViewById(R.id.progressbar_Horizontal);
    new Thread(myThread).start();
}


private Runnable myThread = new Runnable(){
    @Override
    public void run() {
        // TODO Auto-generated method stub
        while (myProgress<100){
        try{
              myHandle.sendMessage(myHandle.obtainMessage());
              Thread.sleep(1000);
        }
        catch(Throwable t){   }
    }
}


Handler myHandle = new Handler(){
      @Override
       public void handleMessage(Message msg) {
            // TODO Auto-generated method stub
            myProgress++;
            myProgressBar.setProgress(myProgress);
       }
```

```
};
};
}
```

Code đã hoàn tất, bạn có thể run và xem kết quả thế nào.

Làm thế nào sử dụng SeekBar trong Android

SeekBar là một mở rộng của ProgressBar mà Android đã hỗ trợ có thể tùy chỉnh trực tiếp. Người dùng có thể kéo sang trái hoặc phải để thiết lập mức độ chỉ số hiện tại hoặc sử dụng các phím mũi tên.

Thiết lập thông số cho Main.xml :

```
< LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
android:orientation="vertical" android:layout_width="fill_parent"
android:layout_height="fill_parent" >
< TextView android:layout_width="fill_parent"
android:layout_height="wrap_content" android:text="@string/hello" />
< SeekBar android:layout_width="fill_parent"
android:layout_height="wrap_content" android:layout_margin="10px"
android:id="@+id/seekbar" android:max="100" android:progress="50" />
< TextView android:layout_width="fill_parent"
android:layout_height="wrap_content" android:id="@+id/seekbarvalue"
android:text="50" />
```

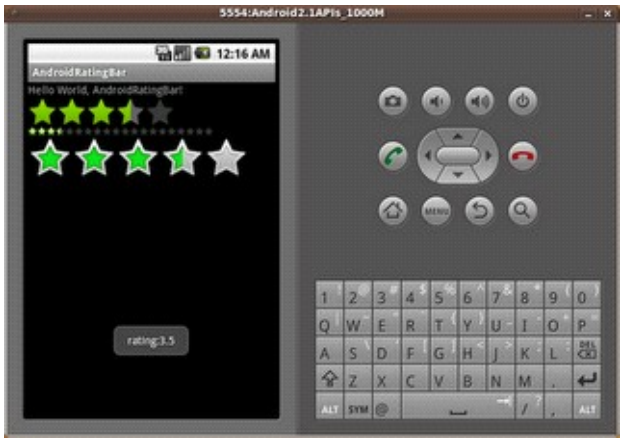
Code cho Activity :

```
package com.AndroidSeekBar;


import android.app.Activity;
import android.os.Bundle;
import android.widget.SeekBar;
import android.widget.TextView;


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


SeekBar seekBar = (SeekBar)findViewById(R.id.seekbar);
final TextView seekBarValue = (TextView)findViewById(R.id.seekbarvalue);



seekBar.setOnSeekBarChangeListener(new
SeekBar.OnSeekBarChangeListener(){


@Override
public void onProgressChanged(SeekBar seekBar, int progress,
boolean fromUser) {
// TODO Auto-generated method stub
seekBarValue.setText(String.valueOf(progress));
}


@Override
public void onStartTrackingTouch(SeekBar seekBar) {
// TODO Auto-generated method stub
```

```
}


@Override
public void onStopTrackingTouch(SeekBar seekBar) {
// TODO Auto-generated method stub
}
});
}
}
```

Làm thế nào sử dụng RatingBar trong Android?

Một RatingBar là một mở rộng của SeekBar và ProgressBar thể hiện một đánh giá trong các ngôi sao. Người dùng có thể chạm/kéo hoặc sử dụng các phím mũi tên để thiết lập các đánh giá khi sử dụng RatingBar kích thước mặc định. Các RatingBar nhỏ hơn theo phong cách (ratingBarStyleSmall) và phong cách chỉ thị, chỉ lớn hơn (ratingBarStyleIndicator) không hỗ trợ tương tác người dùng và chỉ nên được sử dụng như là các chỉ số.



Thiết lập thông số cho Main.xml :

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

```
< RatingBar android:layout_width="wrap_content"
android:layout_height="wrap_content" style="?
android:attr/ratingBarStyleSmall" android:id="@+id/ratingbar_Small"
android:numStars="20" />
< RatingBar android:layout_width="wrap_content"
android:layout_height="wrap_content" style="?
android:attr/ratingBarStyle" android:id="@+id/ratingbar_default" />
```

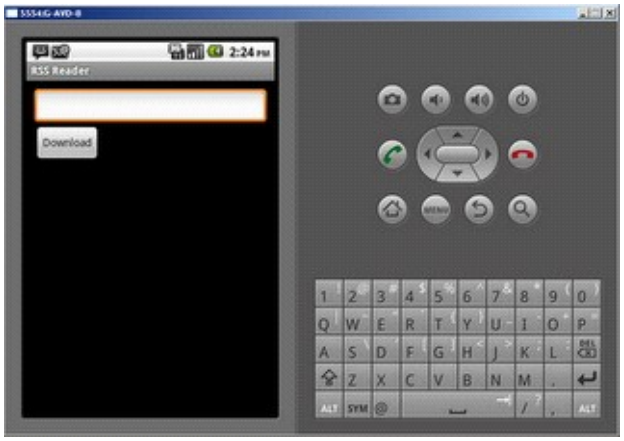
Code Activity :

```
package com.android.sample.AndroidRatingBar;

import android.app.Activity;
import android.os.Bundle;
import android.widget.RatingBar;
import android.widget.Toast;

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

final RatingBar ratingBar_Small =
(RatingBar)findViewById(R.id.ratingbar_Small);
final RatingBar ratingBar_Indicator =
(RatingBar)findViewById(R.id.ratingbar_Indicator);
final RatingBar ratingBar_default =
(RatingBar)findViewById(R.id.ratingbar_default);

ratingBar_default.setOnRatingBarChangeListener(new
RatingBar.OnRatingBarChangeListener(){

@Override
public void onRatingChanged(RatingBar ratingBar, float rating,
```

boolean fromUser) {
// TODO Auto-generated method stub
ratingBar_Small.setRating(rating);
ratingBar_Indicator.setRating(rating);
Toast.makeText(AndroidRatingBar.this, "rating:"+String.valueOf(rating),
Toast.LENGTH_LONG).show();
}});
}
}

Làm thể nào tải dữ liệu RSS trong Android

Ví dụ này tôi muốn cho các bạn biết làm thế nào? để tải dữ liệu từ rss trong Android. Đây chỉ là phần cơ bản trong xử lí dữ liệu động, bạn có thể dùng nhiều dữ liệu khác nhau.



Khi demo được run, màn hình đầu tiên tôi có như trên.

Bây giờ tôi bắt đầu code cho demo, trước tiên tôi thiết lập một số thông số cho một vài xml:

Đối với AndroidManifest.xml :

<?xml version="1.0" encoding="utf-8"?>

<!-- Declare the contents of this Android application. The namespace attribute
brings in the Android platform namespace, and the package supplies a unique
name for the application. When writing your own application, the package

**name must be changed from "com.example.*" to come from a domain that you
own or have control over. -->**

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
package="com.android.sample.rss" android:versionCode="1"
android:versionName="1.0">

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

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

Trong folder layout tôi rename main.xml = rss_layout.xml và thông số 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:padding="10dip"
android:orientation="vertical">

<EditText android:id="@+id/urltext"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:textSize="12sp"
android:autoText="false"
```

```
android:capitalize="none"
android:text="@string/rss_layout_urltext_text" />


<Button android:id="@+id/download"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="@string/rss_layout_download_text" />


<TextView android:id="@+id/statustext"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:autoText="false"
android:capitalize="none"
android:textSize="12sp"
android:text="@string/rss_layout_statustext_text" />


<ListView android:id="@android:id/list"
android:layout_width="fill_parent"
android:layout_height="0dip"
android:layout_weight="1"
android:drawSelectorOnTop="false" />
</LinearLayout>
```

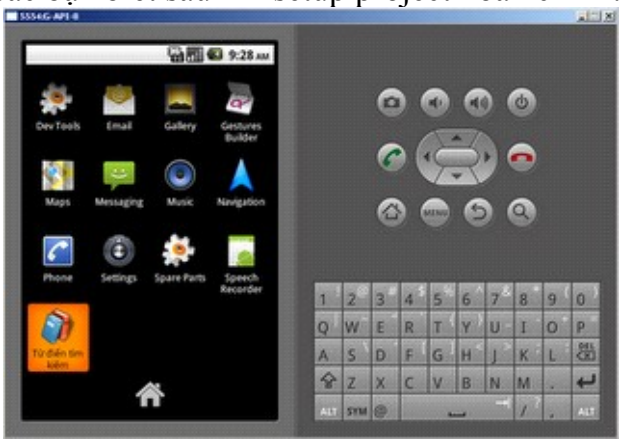
Sau khi các thiết lập và code cho activity hoàn tất, kết quả tôi có khi run demo như sau :

[Làm thế nào xây dựng từ điển tìm kiếm trong Android]

Một ứng dụng mẫu có thể hiện tìm kiếm của Android.

Ứng dụng này bao gồm một từ điển các từ. Bằng cách gọi hộp thoại tìm kiếm bên trong các ứng dụng Android (thông qua các nút tìm kiếm thiết bị hoặc Menu> Search), bạn có thể thực hiện tìm kiếm trên từ điển. Khi bạn nhập, gợi ý sẽ xuất hiện, mà bạn có thể chọn để xem định nghĩa hoàn chỉnh. Bạn cũng có thể thực hiện tìm kiếm để xem tất cả các định nghĩa từ phù hợp với văn bản nhập vào. Ứng dụng này cũng cho phép Quick Search Box (toàn hệ thống tìm kiếm của Android) để cung cấp từ điển đề nghị.

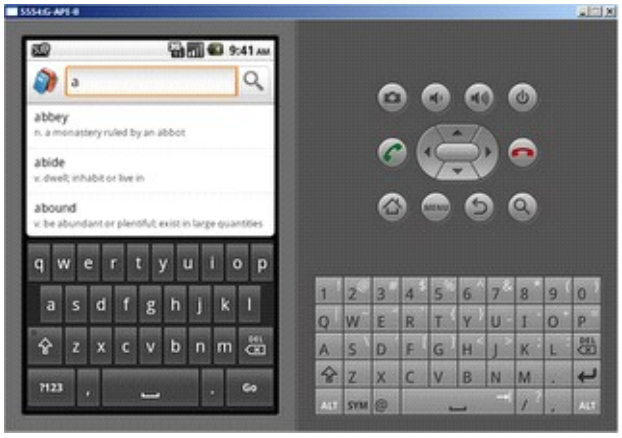
Màn hình sau cho các bạn biết sau khi setup project hoàn chỉnh.



Khi ban click vao button từ điển tìm kiếm và màn hình kế tiếp bạn sẽ thấy như sau :

Sau khi bạn lick tìm kiếm

Khi bạn gõ bất cứ một ký tự nào đó, chương trình sẽ đưa ra những gợi ý bằng những từ có ký tự đầu bằng ký tự mà bạn đã gõ vào. Ở đây tôi sẽ gõ a bạn xem thế nào?



Bạn có thể thêm từ, từ file /res/raw/definitions.txt.

Source SVN :
http://kythuatlaptrinh.googlecode.com/svn/trunk/mobile/android/SampleSearchableDictionary/

Cách sử dụng ContentProvider trong Android

ContentProvider là nền tảng Android cách chia sẻ thông tin giữa nhiều ứng dụng thông qua giao diện ContentResolver của nó. Mỗi ứng dụng có quyền truy cập cơ sở dữ liệu SQLite để duy trì thông tin của họ và điều này không thể chia sẻ với các ứng dụng khác.

Trong bản demo này, chúng tôi sử dụng thông tin cung cấp nội dung avaialable qua getContentResolver() để có được thông tin thiết bị với các trường hợp contentresolver và quierying cung cấp con trỏ. Ngoài việc truy vấn, bạn có thể dùng với các phương pháp liên quan, khả năng chèn, cập nhật, xóa và getType (để trích xuất các loại định dạng).

Kết quả cuối cùng là danh sách của hệ thống tất cả các thiết lập về người sử dụng trên thiết bị Android.

Với dụ SimpleCursorAdapter làm cho việc sử dụng của row.xml với thông tin con trỏ để định cư trong ListView trong bố trí main.xml.

Code cho **ContentUserDemo.java**

```
import android.app.Activity;
import android.content.ContentResolver;
import android.database.Cursor;
import android.os.Bundle;
import android.provider.Settings;
```

```
import android.util.Log;
import android.widget.ListView;
import android.widget.SimpleCursorAdapter;


public class ContentUserDemo extends Activity {
    private static final String TAG = "ContentUserDemo";


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


        // Get content provider and cursor
        ContentResolver cr = getContentResolver();
        Cursor cursor = cr.query(Settings.System.CONTENT_URI, null, null,
null, null);


        // Let activity manage the cursor
        startManagingCursor(cursor);
        Log.d(TAG, "cursor.getCount()=" + cursor.getCount());


        // Get the list view
        ListView listView = (ListView) findViewById(R.id.listView);
        String[] from = { Settings.System.NAME, Settings.System.VALUE };
        int[] to = { R.id.textName, R.id.textValue };
        SimpleCursorAdapter adapter = new SimpleCursorAdapter(
                this, R.layout.row, cursor, from, to);
        listView.setAdapter(adapter);
    }
}
```

Ngoài các thiết lập chính cho **AndroidManifest.xml**. Đừng quên thêm thẻ sau đây cung cấp cho người sử dụng truy cập các thông tin liên lạc.

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

Cấu hình file: **AndroidManifest.xml**

```
<?xml version="1.0" encoding="utf-8"?>
   <manifest xmlns:android="http://schemas.android.com/apk/res/android"
       package="com.android.sample" android:versionCode="1"
android:versionName="1.0.0">
       <uses-permission
android:name="android.permission.READ_CONTACTS" />
       <application android:icon="@drawable/icon"
android:label="@string/app_name">
           <activity android:name=".ContentUserDemo"
android:label="@string/app_name">
            <intent-filter>
               <action android:name="android.intent.action.MAIN" />
               <category
android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
           </activity>
       </application>
</manifest>
```

Thêm **row.xml** cho demo

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
      xmlns:android="http://schemas.android.com/apk/res/android"
      android:layout_height="wrap_content"
      android:padding="5sp"
      android:layout_width="fill_parent">
      <TextView
          android:layout_height="wrap_content"
```

```
            android:id="@+id/textName"
            android:text="Name"
            android:textSize="18sp"
            android:layout_width="fill_parent"
            android:layout_weight="1"></TextView>
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:id="@+id/textValue"
            android:text="Value"
            android:textSize="18sp"
            android:gravity="right"></TextView>
</LinearLayout>
```

Cấu hình **main.xml**

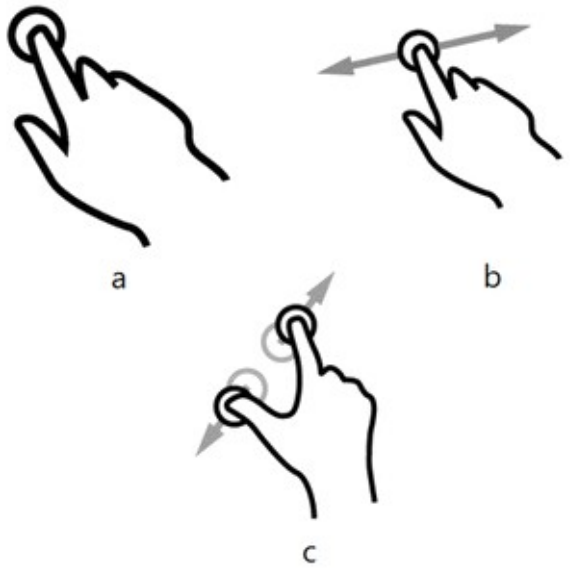
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">
    <ListView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"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

Làm thế nào để sử dụng cảm ứng đa điểm (Multi touch) ở Android

Giới thiệu cảm ứng đa điểm :

Multi-touch là chỉ đơn giản là một phần mở rộng của giao diện người dùng thông thường màn hình cảm ứng, sử dụng hai hay nhiều ngón tay thay vì một. Chúng tôi đã sử dụng một ngón tay cử chỉ trước đây, mặc dù chúng tôi đã không gọi nó là. Trong Chương 4, chúng tôi cho phép người dùng chạm vào một gạch trong các trò chơi Sudoku để thay đổi nó. Đó là gọi là "vòi" cử chỉ. Cử chỉ khác là được gọi là "kéo". Đó là nơi mà bạn giữ một ngón tay trên màn hình và di chuyển nó xung quanh, gây ra các nội dung theo ngón tay của bạn để di chuyển.

Tập, kéo, và một vài cử chỉ khác duy nhất-ngón luôn luôn có được hỗ trợ Android. Nhưng do sự phổ biến của iPhone Apple, người dùng Android đầu bị một loại cử chỉ ghen tị. IPhone hỗ trợ đa cảm ứng, trong "pinch zoom" những cử chỉ cụ thể.



Với pinch phóng to, bạn đặt hai ngón tay trên màn hình và ép chúng lại với nhau để làm cho các mục mà bạn đang xem nhỏ hơn, hoặc kéo chúng ra để làm cho nó lớn hơn. Trước khi Android 2.0 bạn phải sử dụng một clunky điều khiển zoom với các biểu tượng mà bạn ép để phóng to ra (ví dụ các setBuiltInZoomControls() trong ví dụ MyMap). Nhưng nhờ có sự hỗ trợ đa cảm ứng mới của nó, bạn có thể kẹp để phóng to trên Android quá! Khi ứng dụng hỗ trợ nó, tất nhiên.

Lưu ý: Nếu bạn cố gắng để chạy các ví dụ trong chương này trên Android 1.5 hoặc 1.6, nó sẽ sụp đổ vì những phiên bản không hỗ trợ đa cảm ứng. Chúng tôi sẽ tìm hiểu làm thế nào để làm việc xung quanh mà trong chương 13, "Viết một lần, mọi nơi thử nghiệm".

Cảnh báo: Multi-lỗi trước

Multi-touch, như thực hiện trên điện thoại Android hiện tại có lỗi. Trong thực tế nó rất nhiều lỗi mà nó không sử dụng được. API này thường xuyên báo cáo dữ liệu không hợp lệ hoặc không thể điểm, đặc biệt là trong quá trình chuyển đổi từ một ngón tay vào hai ngón tay trên màn hình và ngược lại.

Trên các diễn đàn phát triển bạn có thể tìm thấy khiếu nại của các ngón tay bị đổi chỗ, các trục x và y lật, và nhiều ngón tay đôi khi bị đối xử như là một. Với rất nhiều thử và sai, tôi đã có thể có được những ví dụ trong chương này làm việc bởi vì các cử chỉ đó thực hiện rất đơn giản. Cho đến khi Google công nhận và sửa

chữa các vấn đề, thatmay được về tất cả các bạn có thể làm. May mắn thay, nhăn nhúm zoom dường như là chỉ đa chạm cử chỉ hầu hết mọi người muốn.

Ví dụ Touch
Để chứng minh đa cảm ứng, chúng ta sẽ xây dựng một ứng dụng xem ảnh đơn giản cho phép bạn phóng to và di chuyển xung quanh một hình ảnh.

Xây dựng ví dụ Touch trong Android Phần 2

Để chứng minh đa cảm ứng, chúng ta sẽ xây dựng một ứng dụng xem ảnh đơn giản cho phép bạn phóng to và di chuyển xung quanh một hình ảnh.

Bắt đầu bằng cách tạo ra một "Hello, Android" dự án với các thông số sau trong hộp thoại Android mới dự án:

Tên dự án: Touch
Xây dựng các mục tiêu: Android 2.1
Ứng dụng hiệu: Touch
Đóng gói và tên: org.example.touch
Tạo cuối: Touch

Điều này sẽ tạo Touch.java để chứa các hoạt động chính của bạn. Hãy chỉnh sửa nó để hiển thị một hình ảnh mẫu, đặt trong một người nghe cảm ứng, và thêm một vài nhập khẩu chúng tôi sẽ cần sau:

từ Touchv1/src/org/example/touch/Touch.java:

```
package org.example.touch;

import android.app.Activity;
import android.graphics.Matrix;
import android.graphics.PointF;
import android.os.Bundle;
import android.util.FloatMath;
import android.util.Log;


import android.view.MotionEvent;
import android.view.View;
import android.view.View.OnTouchListener;
import android.widget.ImageView;

public class Touch extends Activity implements OnTouchListener {
```

```
    private static final String TAG = "Touch" ;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        ImageView view = (ImageView) findViewById(R.id.imageView);
        view.setOnTouchListener(this);
    }

    @Override
     public boolean onTouch(View v, MotionEvent event) {
        // Handle touch events here...
    }
}
```

Chúng ta sẽ làm ra rằng onTouch() trong giây lát. Trước tiên chúng ta cần định nghĩa bố trí cho hoạt động của chúng ta:

Từ **Touchv1/res/layout/main.xml:**

```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent" >
    <ImageView android:id="@+id/imageView"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:src="@drawable/butterfly"
        android:scaleType="matrix" >
     </ImageView>
</FrameLayout>
```

Giao diện toàn bộ là một ImageView kiểm soát lớn bao gồm toàn bộ màn hình. Các **android:src="@drawable/butterfly"** giá trị đề cập đến hình ảnh con bướm được sử dụng trong ví dụ. Bạn có thể sử dụng bất kỳ định dạng hình ảnh JPG hoặc PNG bạn thích, chỉ cần đặt nó trong thư mục res/drawables-nodpi. Các **android:scaleType="matrix"** cho thấy chúng ta sẽ sử dụng một ma trận để kiểm soát vị trí và quy mô của hình ảnh. Thêm về điều này sau. Các tập tin AndroidManifest.xml là bị ảnh hưởng trừ việc bổ sung các android:theme
Từ Touchv1/AndroidManifest.xml:

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="org.example.touch"
    android:versionCode="1"
    android:versionName="1.0" >
    <application android:icon="@drawable/icon"
        android:label="@string/app_name"
        android:theme="@android:style/Theme.NoTitleBar.Fullscreen" >
        <activity android:name=".Touch"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category
android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
    <uses-sdk android:minSdkVersion="3"
android:targetSdkVersion="7" />
</manifest>
```

@android:style/Theme.NoTitleBar.Fullscreen như tên cho thấy, cho Android sử dụng toàn bộ màn hình không có thanh tiêu đề hoặc thanh trạng thái ở đầu trang. Bạn có thể chạy các ứng dụng hiện nay và nó chỉ đơn giản sẽ hiển thị hình ảnh.

Tìm hiểu về các sự kiện touch trong Android

Bất cứ khi nào lần đầu tiên tôi tìm hiểu một API mới, tôi muốn đặt đầu tiên trong mã số để đổ tất cả mọi thứ vì vậy tôi có thể nhận được một cảm giác về những gì các phương pháp làm và trong những sự kiện gì để xảy ra. Vì vậy, hãy bắt đầu với điều đó. Thêm đầu một cuộc gọi đến **dumpEvent**() phương pháp bên trong **onTouch**():

Từ Touchv1/src/org/example/touch/Touch.java:

```
@Override
public boolean onTouch(View v, MotionEvent event) {
    // Dump touch event to log
    dumpEvent(event);
    return true; // indicate event was handled
}
```

Lưu ý rằng chúng ta cần phải trả về đúng với chỉ số cho Android rằng sự kiện này đã được xử lý. Tiếp theo, xác định các method **dumpEvent**(). Các tham số chỉ là sự kiện mà chúng ta muốn dump.

Từ Touchv1/src/org/example/touch/Touch.java:

```
/** Show an event in the LogCat view, for debugging */
private void dumpEvent(MotionEvent event) {
    String names[] = { "DOWN" , "UP" , "MOVE" , "CANCEL" ,
"OUTSIDE" ,
            "POINTER_DOWN" , "POINTER_UP" , "7?" , "8?" , "9?" };
    StringBuilder sb = new StringBuilder();
     int action = event.getAction();
     int actionCode = action & MotionEvent.ACTION_MASK;
     sb.append("event ACTION_" ).append(names[actionCode]);

     if (actionCode == MotionEvent.ACTION_POINTER_DOWN
          || actionCode == MotionEvent.ACTION_POINTER_UP) {
          sb.append("(pid " ).append(
          action >> MotionEvent.ACTION_POINTER_ID_SHIFT);
          sb.append(")" );
     }
     sb.append("[" );
     for (int i = 0; i < event.getPointerCount(); i++) {
          sb.append("#" ).append(i);
          sb.append("(pid " ).append(event.getPointerId(i));
          sb.append(")=" ).append((int) event.getX(i));
          sb.append("," ).append((int) event.getY(i));
          if (i + 1 < event.getPointerCount())
               sb.append(";" );
     }
     sb.append("]" );
     Log.d(TAG, sb.toString());
}
```

Kết quả sẽ đi đến các bản ghi gỡ lỗi Android, bạn có thể nhìn thấy bằng cách mở xem LogView. Cách dễ nhất để hiểu đoạn mã này là để chạy nó. Thật không may bạn không thể chạy chương trình trên Emulator (thực ra bạn có thể, nhưng các Emulator không hỗ trợ đa cảm ứng như vậy kết quả sẽ không được rất thú vị). Vì vậy, hãy làm với một điện thoại thực sự với cổng USB của bạn và chạy các mẫu có.

Làm thế nào sử dụng cảm biến trong Android

Bộ cảm biến hỗ trợ Android là khá đơn giản nhưng đa năng. Nó được xử lý qua các SensorManager được cung cấp từ ngữ cảnh ứng dụng của bạn thông qua **getSystemService()**. Một khi bạn đã quản lý bộ cảm biến, bạn phải đăng ký để nhận thông báo của mình. Để làm điều này, chúng tôi thực hiện các giao diện **SensorListener**.

Chú ý rằng chúng tôi đăng ký để nhận được cập nhật cảm biến trong **onResume**() và huỷ đăng ký trong **onPause**(). Điều này là quan trọng. Cảm biến dữ liệu đến trong khoảng thời gian thất thường và có thể tiêu tốn rất nhiều CPU và năng lượng pin. Để ứng dụng tối ưu hóa của chúng tôi để thực hiện, đó là thực hành tốt nhất để có được các thông báo chỉ khi ứng dụng của bạn là trong điều hành nhà nước.

Các thông báo đến qua **onSensorChanged**() và **onAccuracyChanged**(). Chú ý rằng mỗi một tham số cho các báo cáo cảm biến. Android API cảm biến có thể hỗ trợ một số lượng lớn các bộ cảm biến, mỗi người có riêng ID số nguyên duy nhất của nó. Những người hiện tại được thể hiện qua các hằng số của họ, chẳng hạn như **SensorManager.SENSOR_ORIENTATION.**

Cảm biến Các giá trị chúng ta có được trong **onSensorChanged()** sẽ phụ thuộc vào cảm biến cụ thể. Ví dụ, định hướng cảm biến báo cáo phương vị là giá trị ở chỉ số 0, sân ở chỉ số 1 và cuộn vào chỉ số

Code: SensorsDemo.java

```
package com.android.sample;


import android.app.Activity;
import android.hardware.SensorListener;
import android.hardware.SensorManager;
import android.os.Bundle;
import android.util.Log;
import android.widget.TextView;


public class SensorDemo extends Activity implements SensorListener {
    private static final String TAG = "SensorDemo";
    private SensorManager sensorManager;
    private TextView outView;
    private int sensor = SensorManager.SENSOR_ORIENTATION;
```

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


        // Real sensor manager
        sensorManager = (SensorManager)
getSystemService(SENSOR_SERVICE);
    }


    /** Register for the updates when Activity is in foreground */
    @Override
     protected void onResume() {
           super.onResume();
           Log.d(TAG, "onResume");
           sensorManager.registerListener(this, sensor);
     }


      /** Stop the updates when Activity is paused */
      @Override
      protected void onPause() {
            super.onPause();
            Log.d(TAG, "onPause");
            sensorManager.unregisterListener(this, sensor);
       }


       public void onAccuracyChanged(int sensor, int accuracy) {
            Log.d(TAG, String.format("onAccuracyChanged sensor: %d
accuraccy: %d",
            sensor, accuracy));
       }


       public void onSensorChanged(int sensorReporting, float[] values) {
           if (sensorReporting != sensor)
               return;
```

```
            float azimuth = Math.round(values[0]);
            float pitch = Math.round(values[1]);
            float roll = Math.round(values[2]);

            String out = String.format("Azimuth: %.2f\n\nPitch:%.2f\nRoll",
azimuth,
pitch, roll);
            Log.d(TAG, out);
            outView.setText(out);
      }
}
```

Cách bố trí cho ví dụ đơn giản chi có chứa một TextView cho đầu ra.

Code: /res/layout/main.xml

```
<?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">
       <TextView android:id="@+id/output"
android:layout_width="fill_parent"
            android:text="@string/hello" android:textSize="24sp"
            android:layout_height="fill_parent"
android:layout_margin="30dp" />
</LinearLayout>
```

Chú ý rằng cảm biến trong các giả lập không được hỗ trợ. Có một cách thông qua một công cụ OpenIntents, nhưng đòi hỏi một chút về thiết lập. Đây là những gì đầu ra trông giống như trên thiết bị thật:

Thực hiện giao diện từ xa sử dụng AIDL

AIDL (Android Interface Definition Language) là một ngôn ngữ IDL được sử dụng để tạo ra mã cho phép hai quá trình trên một thiết bị hỗ trợ Android để nói chuyện bằng cách sử dụng giao tiếp InterProcess (IPC). Nếu bạn có mã trong một quá trình (ví dụ, trong một hoạt động) mà cần phải gọi các phương thức trên một đối tượng trong quá trình khác (ví dụ, một dịch vụ), bạn sẽ sử dụng AIDL để tạo ra mã để marshall các thông số.

Các cơ chế AIDL IPC là dựa trên giao diện, tương tự như COM hay CORBA, nhưng trọng lượng nhẹ hơn. Nó sử dụng một lớp proxy để vượt qua giá trị giữa khách hàng và thực hiện.

Mỗi ứng dụng Android chạy trong quá trình riêng của mình. Một ứng dụng không thể trực tiếp truy cập không gian bộ nhớ khác của ứng dụng. Đây là ứng dụng gọi là sandboxing.

Xác định AIDL

Cú pháp AIDL rất giống với giao diện Java thường xuyên. Bạn chỉ cần xác định phương pháp chữ ký. Các kiểu dữ liệu được hỗ trợ bởi AIDL là hơi khác so với thông thường giao diện Java. Đối với một, tất cả các kiểu dữ liệu nguyên thủy được hỗ trợ Java. Vì vậy, là String, Danh sách, bản đồ, và CharSequence các lớp học. Ngoài ra, tất cả các kiểu dữ liệu AIDL khác mà bạn định nghĩa được hỗ trợ. Thêm vào đó, tất cả các lớp Parcelable được hỗ trợ, nhưng điều này sẽ không được

đề cập trong ví dụ này. Tôi đang cố gắng để giữ cho ví dụ này khá đơn giản để bắt đầu.

Code: **/src/com.android.sample/IAdditionService.aidl**

```
package com.android.sample;


// Declare the interface.
interface IAdditionService {
    // You can pass values in, out, or inout.
    // Primitive datatypes (such as int, boolean, etc.) can only be passed in.
    int add(in int value1, in int value2);
}
```

Thực hiện các dịch vụ từ xa

Một khi bạn tạo ra file AIDL của bạn và đặt nó vào đúng chỗ, công cụ Eclipse + AIDL sẽ tạo ra một file có cùng tên, nhưng phần mở rộng. java. Vì vậy, tôi bây giờ có file **/gen/com.android.sample/IAdditionService.java**. Đây là một tập tin tự động tạo ra do đó bạn không muốn chỉnh sửa nó. Điều quan trọng là nó có chứa một lớp Stub rằng chúng tôi sẽ muốn thực hiện cho các dịch vụ từ xa của chúng tôi.

Để thực hiện các dịch vụ từ xa của chúng ta, chúng ta sẽ trả lại **IBinder** từ **onBind**() trong lớp dịch vụ **AdditionService**. **IBinder** đại diện cho việc thực hiện các dịch vụ từ xa. Để thực hiện IBinder, chúng ta phân lớp **IAddtionService.Stub** lớp từ mã Java tự động tạo ra, và cung cấp thực hiện các phương pháp của chúng tôi **AIDL** xác định, trong trường hợp tiện ích này **Add()**.

Code : **/src/com.android.sample/AdditionService.java**

```
import android.app.Service;
import android.content.Intent;
import android.os.IBinder;
import android.os.RemoteException;
import android.util.Log;


/**
* This class exposes the remote service to the client
*/
public class AdditionService extends Service {
```

```
    private static final String TAG = "AdditionService";


    @Override
    public void onCreate() {
        super.onCreate();
        Log.d(TAG, "onCreate()");
    }


    @Override
    public IBinder onBind(Intent intent) {
        return new IAdditionService.Stub() {
            /**
            * Implementation of the add() method
            */
             public int add(int value1, int value2) throws RemoteException {
                 Log.d(TAG, String.format("AdditionService.add(%d,
%d)",value1, value2));
                 return value1 + value2;
             }
        };
    }


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

Phơi bày các dịch vụ cục bộ

Một khi chúng ta có các dịch vụ thực hiện các **onBind**() đúng cách, chúng ta sẵn sàng để kết nối với dịch vụ từ khách hàng của chúng ta. Trong trường hợp này, chúng ta đã **AIDL** Demo hoạt động kết nối với dịch vụ đó. Để thiết lập kết nối, chúng ta cần phải thực hiện các lớp ServiceConnection. Hoạt động trong ví dụ này cung cấp này thực hiện trong lớp **AdditionServiceConnection** bên trong bằng cách thực hiện **onServiceConnected**() và phương pháp **onServiceDiconnected**(). Những **callback** sẽ được thực hiện sơ khai của các dịch vụ từ xa khi kết nối. Chúng ta cần phải bỏ chúng từ khai để thực hiện dịch vụ AIDL của chúng tôi. Để làm được điều đó, chúng tôi sử dụng

IAdditionService.Stub.asInterface((**IBinder**) **boundService**) phương pháp giúp đỡ.

Từ thời điểm này, chúng ta có một đối tượng dịch vụ địa phương mà chúng ta có thể sử dụng để thực hiện cuộc gọi đối với các dịch vụ từ xa.

Code: /src/com.android.sample/AIDLDemo.java

```
package com.android.sample;


import android.app.Activity;
import android.content.ComponentName;
import android.content.Context;
import android.content.Intent;
import android.content.ServiceConnection;
import android.os.Bundle;
import android.os.IBinder;
import android.os.RemoteException;
import android.util.Log;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;


public class AIDLDemo extends Activity {
private static final String TAG = "AIDLDemo";
IAdditionService service;
AdditionServiceConnection connection;


/**
* This class represents the actual service connection. It casts the bound
* stub implementation of the service to the AIDL interface.
*/
class AdditionServiceConnection implements ServiceConnection {
    public void onServiceConnected(ComponentName name, IBinder
boundService) {
        service = IAdditionService.Stub.asInterface((IBinder) boundService);
        Log.d(AIDLDemo.TAG, "onServiceConnected() connected");
```

```
            Toast.makeText(AIDLDemo.this, "Service connected",
Toast.LENGTH_LONG)     .show();
        }


        public void onServiceDisconnected(ComponentName name) {
            service = null;
            Log.d(AIDLDemo.TAG, "onServiceDisconnected() disconnected");
            Toast.makeText(AIDLDemo.this, "Service connected",
Toast.LENGTH_LONG).show();
        }
}


/** Binds this activity to the service. */
private void initService() {
        connection = new AdditionServiceConnection();
        Intent i = new Intent();
        i.setClassName("com.marakana",
com.marakana.AdditionService.class.getName());
        boolean ret = bindService(i, connection,
Context.BIND_AUTO_CREATE);
        Log.d(TAG, "initService() bound with " + ret);
}


/** Unbinds this activity from the service. */
private void releaseService() {
        unbindService(connection);
        connection = null;
        Log.d(TAG, "releaseService() unbound.");
}


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

         initService();
```

```
        // Setup the UI
        Button buttonCalc = (Button) findViewById(R.id.buttonCalc);
        buttonCalc.setOnClickListener(new OnClickListener() {

        TextView result = (TextView) findViewById(R.id.result);
        EditText value1 = (EditText) findViewById(R.id.value1);
        EditText value2 = (EditText) findViewById(R.id.value2);

        public void onClick(View v) {
            int v1, v2, res = -1;
            v1 = Integer.parseInt(value1.getText().toString());
            v2 = Integer.parseInt(value2.getText().toString());
            try {
                res = service.add(v1, v2);
            } catch (RemoteException e) {
                 Log.d(AIDLDemo.TAG, "onClick failed with: " + e);
                 e.printStackTrace();
            }
            result.setText(new Integer(res).toString());
        }
});
}


/** Called when the activity is about to be destroyed. */
@Override
protected void onDestroy() {
      releaseService();
}

}
```

Các giao diện người dùng trong trường hợp này rất đơn giản. Có vài EditText TextViews và các lĩnh vực và một nút nút này xử lý các sự kiện của nó trong một bên trong lớp OnClickListener vô danh. Nút này chỉ đơn giản gọi các tiện ích () phương thức dịch vụ này như thể nó là một cuộc gọi địa phương.

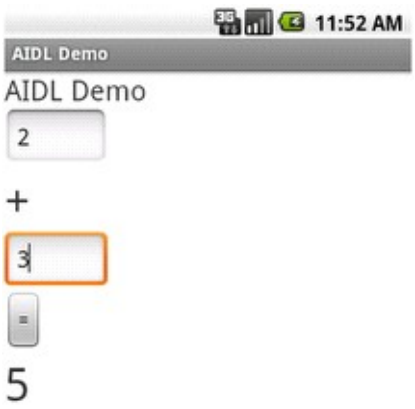
Cách bố trí cho ví dụ này không phải là quan trọng, nhưng tôi bao gồm nó ở đây cho mục đích hoàn thiện.

Code: **/res/layout/main.xml**

```
<?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">
    <TextView android:layout_width="fill_parent"
        android:layout_height="wrap_content" android:text="AIDL
Demo"
        android:textSize="22sp" />
     <EditText android:layout_width="wrap_content"
        android:layout_height="wrap_content" android:id="@+id/value1"
        android:hint="Value 1" />
    <TextView android:id="@+id/TextView01"
android:layout_width="wrap_content"
        android:layout_height="wrap_content" android:text="+"
        android:textSize="36sp" />
    <EditText android:layout_width="wrap_content"
        android:layout_height="wrap_content" android:id="@+id/value2"
        android:hint="Value 2" />
    <Button android:layout_width="wrap_content"
        android:layout_height="wrap_content"
android:id="@+id/buttonCalc"
        android:text="=" />
    <TextView android:layout_width="wrap_content"
        android:layout_height="wrap_content" android:text="result"
        android:textSize="36sp" android:id="@+id/result" />
</LinearLayout>
```

Kết quả tôi có khi Run demo ;



Kiểm tra và nhắc nhở bật sử dụng GPS

Để kiểm tra xem GPS đã được kích hoạt hay không, các mã sau đây có thể được sử dụng:

String provider = Settings.Secure.getString(getContentResolver(), Settings.Secure.LOCATION_PROVIDERS_ALLOWED);

Nếu đó là trống rỗng, có nghĩa là GPS chưa được kích hoạt. Bạn có thể bắt đầu hoạt động với **Settings.ACTION_SECURITY_SETTINGS** ý định, để chuyển sang trang thiết lập **GPS**.

Nếu tính năng GPS được thiết lập, khi ấy demo se báo cho bạn biết như sau :

Với một ít code sau sẽ giúp bạn kiểm tra tính năng này :

```
import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.provider.Settings;
import android.widget.Toast;
```

```
public class AndroidEnableGPS extends Activity {


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


    private void CheckEnableGPS(){
        String provider = Settings.Secure.getString(getContentResolver(),
        Settings.Secure.LOCATION_PROVIDERS_ALLOWED);
        if(!provider.equals("")){
            //GPS Enabled
            Toast.makeText(AndroidEnableGPS.this, "GPS Enabled: " +
provider,
            Toast.LENGTH_LONG).show();
        }else{
            Intent intent = new
Intent(Settings.ACTION_SECURITY_SETTINGS);
            startActivity(intent);
        }
    }
}
```

Làm thế nào để chơi MIDI âm thanh sử dụng MediaPlayer

Lớp MediaPlayer có thể được sử dụng để kiểm soát phát lại các tập tin audio/video từ file hay dữ liệu.

Đặt một file MIDI vào res/raw thư mục của dự án của bạn, nơi mà các Eclipse plugin (hoặc aapt) sẽ tìm thấy nó và làm cho nó thành một nguồn tài nguyên có thể được tham chiếu từ lớp R của bạn. "midi_sound.mid" trong exercise.

Sửa đổi **main.xml** có hai nút bấm để chơi và tạm dừng.

```
<?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" >
          <TextView
                android:layout_width="fill_parent"
                android:layout_height="wrap_content"
                android:text="@string/hello" />
         <Button
                android:id="@+id/play"
                android:layout_width="fill_parent"
                android:layout_height="wrap_content"
                android:text="- PLAY -" />
        <Button
                android:id="@+id/pause"
```

```
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:text="- PAUSE -" />
  </LinearLayout>
```

Sửa đổi mã nguồn, AndroidAudioPlayer.java.

```
package com.exercise.AndroidAudioPlayer;


import android.app.Activity;
import android.media.MediaPlayer;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.Toast;


public class AndroidAudioPlayer extends Activity {


    MediaPlayer mediaPlayer;


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


        mediaPlayer = MediaPlayer.create(this, R.raw.midi_sound);


        Button buttonPlay = (Button)findViewById(R.id.play);
        Button buttonPause = (Button)findViewById(R.id.pause);
        buttonPlay.setOnClickListener(buttonPlayOnClickListener);
```

```
        buttonPause.setOnClickListener(buttonPauseOnClickListener);
    }


    Button.OnClickListener buttonPlayOnClickListener = new
Button.OnClickListener(){
        @Override
        public void onClick(View v) {
            // TODO Auto-generated method stub
            if(!mediaPlayer.isPlaying()){
                mediaPlayer.start();
                Toast.makeText(AndroidAudioPlayer.this,
"mediaPlayer.start()",
                Toast.LENGTH_LONG).show();
            }
        }
    };


     Button.OnClickListener buttonPauseOnClickListener = new
Button.OnClickListener(){
        @Override
        public void onClick(View v) {
            // TODO Auto-generated method stub
            if(mediaPlayer.isPlaying()){
                mediaPlayer.pause();
                Toast.makeText(AndroidAudioPlayer.this,
"mediaPlayer.pause()",
                Toast.LENGTH_LONG).show();
            }
        }
    };
}
```

[Làm thế nào xây dựng dịch vụ báo động dùng AlarmManager]

AlarmManager class cung cấp truy cập vào các dịch vụ hệ thống báo động. Điều này cho phép bạn lên lịch các ứng dụng của bạn để chạy vào một số điểm trong tương lai. Ngay cả khi báo thức đã tắt, và đã được đăng ký với dịch vụ của hệ thống, nó sẽ tự động khởi động ứng dụng khi nó chưa được chạy.

Trong bài này, một báo động dự kiến là 10 giây sẽ bắt đầu một dịch vụ, MyAlarmService.

Sửa đổi main.xml có hai nút Start và Cancel các báo động.

```
<?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" >
      <TextView
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:text="@string/hello" />
     <Button
            android:id="@+id/startalarm"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
```

```
            android:text="Start" />
        <Button
            android:id="@+id/cancelalarm"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:text="Cancel" />
</LinearLayout>
```

AndroidAlarmService.java, và các hoạt động chính cho class.

```
package com.android.sample.AndroidAlarmService;


import java.util.Calendar;


import android.app.Activity;
import android.app.AlarmManager;
import android.app.PendingIntent;
import android.content.Intent;
import android.os.Bundle;
import android.os.SystemClock;
import android.view.View;
import android.widget.Button;
import android.widget.Toast;


public class AndroidAlarmService extends Activity {
    private PendingIntent pendingIntent;
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        Button buttonStart = (Button)findViewById(R.id.startalarm);
        Button buttonCancel = (Button)findViewById(R.id.cancelalarm);
```

```
        buttonStart.setOnClickListener(new Button.OnClickListener(){
            @Override
            public void onClick(View arg0) {
                 // TODO Auto-generated method stub
                 Intent myIntent = new Intent(AndroidAlarmService.this,
MyAlarmService.class);
                 pendingIntent =
PendingIntent.getService(AndroidAlarmService.this, 0, myIntent, 0);
                 AlarmManager alarmManager =
(AlarmManager)getSystemService(ALARM_SERVICE);
                 Calendar calendar = Calendar.getInstance();
                 calendar.setTimeInMillis(System.currentTimeMillis());
                 calendar.add(Calendar.SECOND, 10);
                 alarmManager.set(AlarmManager.RTC_WAKEUP,
calendar.getTimeInMillis(), pendingIntent);
                 Toast.makeText(AndroidAlarmService.this, "Start Alarm",
Toast.LENGTH_LONG).show();
            }});
            buttonCancel.setOnClickListener(new Button.OnClickListener(){
                @Override
                public void onClick(View arg0) {
                     // TODO Auto-generated method stub
                     AlarmManager alarmManager =
(AlarmManager)getSystemService(ALARM_SERVICE);
                     alarmManager.cancel(pendingIntent);
                     // Tell the user about what we did.
                     Toast.makeText(AndroidAlarmService.this, "Cancel!",
Toast.LENGTH_LONG).show();
                }});
        }
}
```

MyAlarmService.java, nó sẽ bắt đầu trong 10 giây được kích hoạt bởi SlarmManager

```
package com.android.sample.AndroidAlarmService;
```

```
import android.app.Service;
import android.content.Intent;
import android.os.IBinder;
import android.widget.Toast;


public class MyAlarmService extends Service {


    @Override
    public void onCreate() {
        // TODO Auto-generated method stub
        Toast.makeText(this, "MyAlarmService.onCreate()",
Toast.LENGTH_LONG).show();
    }


    @Override
    public IBinder onBind(Intent intent) {
        // TODO Auto-generated method stub
        Toast.makeText(this, "MyAlarmService.onBind()",
Toast.LENGTH_LONG).show();
        return null;
    }


    @Override
    public void onDestroy() {
       // TODO Auto-generated method stub
       super.onDestroy();
       Toast.makeText(this, "MyAlarmService.onDestroy()",
Toast.LENGTH_LONG).show();
    }
```

```
    @Override
    public void onStart(Intent intent, int startId) {
        // TODO Auto-generated method stub
        super.onStart(intent, startId);
        Toast.makeText(this, "MyAlarmService.onStart()",
Toast.LENGTH_LONG).show();
    }


    @Override
    public boolean onUnbind(Intent intent) {
        // TODO Auto-generated method stub
        Toast.makeText(this, "MyAlarmService.onUnbind()",
Toast.LENGTH_LONG).show();
         return super.onUnbind(intent);
    }
}
```

Cuối cùng, sửa đổi AndroidManifest.xml liệt kê như là dịch vụ.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.android.sample.AndroidAlarmService"
    android:versionCode="1"
    android:versionName="1.0">
    <application android:icon="@drawable/icon"
android:label="@string/app_name">
        <activity android:name=".AndroidAlarmService"
              android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category
android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <service android:name=".MyAlarmService" />
    </application>
```

```
    <uses-sdk android:minSdkVersion="4" />
</manifest>
```

Xây dựng ứng dụng đơn giản xem video (.3gp) trong Android

Demo đơn giản xem video định dạng file (.3gp) trong Android

Code cho VideoActivity.java :

```
package com.android.sample.video;


import android.app.Activity;
import android.net.Uri;
import android.os.Bundle;
import android.util.Log;
import android.widget.MediaController;
import android.widget.VideoView;


public class VideoActivity extends Activity {
    /** Called when the activity is first created. */
    @Override
```

```
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        VideoView myVideoView = (VideoView)
findViewById(R.id.videoview);
        String viewSource = "http://daily3gp.com/vids/747.3gp";
        try {
            myVideoView.setVideoURI(Uri.parse(viewSource));
            myVideoView.setMediaController(new MediaController(this));
            myVideoView.requestFocus();
            myVideoView.start();
        } catch (Exception e) {
            Log.e("SampleVideo", "error: " + e.getMessage(), e);
            if (myVideoView != null) {
                myVideoView.stopPlayback();
            }
        }
    }
}
```

Code cho AndroidManifest.xml :

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.android.sample.video"
    android:versionCode="1"
    android:versionName="1.0">
    <application android:icon="@drawable/icon"
android:label="@string/app_name">
        <activity android:name=".VideoActivity"
android:label="Media/Video View">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category
android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
```

```
    </application>
    <uses-permission android:name="android.permission.INTERNET" />
</manifest>
```

Code cho main.xml :

```
<?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">
    <TextView android:layout_width="fill_parent"
        android:layout_height="wrap_content"
android:text="@string/hello" />
    <VideoView android:id="@+id/videoview"
android:layout_width="fill_parent"
        android:layout_height="fill_parent" />
</LinearLayout>
```

Một số ví dụ về inputType trên EditText trong Android


```
<?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" >

    <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="Ordinary EditText" />

   <EditText
        android:layout_width="fill_parent"
        android:layout_height="wrap_content" />

  <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Password" />
   <EditText
        android:password="true"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content" />

   <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Multi-ine" />

   <EditText
        android:inputType="text|textMultiLine"
        android:minLines="3"
        android:gravity="top"
```

```
        android:layout_width="fill_parent"
        android:layout_height="wrap_content" />
    <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Date" />

     <EditText
        android:inputType="date"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content" />

    <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Signed Decimal Number" />

    <EditText
        android:inputType="number|numberSigned|numberDecimal"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content" />

</LinearLayout>
```

Sử dụng Gallery, ImageView trong Android

Trong Android, chúng tôi có thể hiển thị nhiều hình ảnh trong chế độ xem ảnh. Dưới đây là một ví dụ thư viện Android sẽ giải thích làm thế nào để hiển thị các hình ảnh trong thư viện xem.

Bây giờ chúng ta sẽ thấy một ví dụ thư viện đơn giản về cách sử dụng bộ sưu tập như một album ảnh như trong hình của demo sau. Đó là, khi chúng ta click vào mục trong thư viện, các hình ảnh tương ứng sẽ hiển thị bên dưới trong kích thước đầy đủ bằng cách sử dụng imageview.

Tạo một file attrs.xml ở res/values thư mục. Tập tin này được sử dụng để khai báo kiểu.

Main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout android:id="@+id/LinearLayout01"
    android:layout_width="fill_parent"
android:layout_height="fill_parent"
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical">
    <Gallery xmlns:android="http://schemas.android.com/apk/res/android"
        android:id="@+id/examplegallery"
android:layout_width="fill_parent"
        android:layout_height="wrap_content" />
        <ImageView android:id="@+id/ImageView01"
            android:layout_width="wrap_content"
android:layout_height="wrap_content"/>
</LinearLayout>
```

Code cho GalleryExample.java :

```
public class GalleryExample extends Activity {
```

```
    private Gallery gallery;
    private ImageView imgView;


    private Integer[] Imgid = { R.drawable.a_1, R.drawable.a_2,
R.drawable.a_3,
            R.drawable.a_4, R.drawable.a_5, R.drawable.a_6, R.drawable.a_7 };


    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        imgView = (ImageView) findViewById(R.id.ImageView01);
        imgView.setImageResource(Imgid[0]);
        gallery = (Gallery) findViewById(R.id.examplegallery);
        gallery.setAdapter(new AddImgAdp(this));
        gallery.setOnItemClickListener(new OnItemClickListener() {
            public void onItemClick(AdapterView parent, View v, int
position,  long id) {
                imgView.setImageResource(Imgid[position]);
            }
        });
     }


      public class AddImgAdp extends BaseAdapter {
          int GalItemBg;
          private Context cont;


          public AddImgAdp(Context c) {
              cont = c;
              TypedArray typArray =
obtainStyledAttributes(R.styleable.GalleryTheme);
              GalItemBg = typArray.getResourceId(
                 R.styleable.GalleryTheme_android_galleryItemBackground,
0);
              typArray.recycle();
          }


          public int getCount() {
```

```
            return Imgid.length;
        }


        public Object getItem(int position) {
            return position;
        }


        public long getItemId(int position) {
            return position;
        }


        public View getView(int position, View convertView, ViewGroup
parent) {
            ImageView imgView = new ImageView(cont);
            imgView.setImageResource(Imgid[position]);
            imgView.setLayoutParams(new Gallery.LayoutParams(80, 70));
            imgView.setScaleType(ImageView.ScaleType.FIT_XY);
            imgView.setBackgroundResource(GalItemBg);


            return imgView;
        }
    }
}
```

SVN : https://kythuatlaptrinh.googlecode.com/svn/trunk/mobile/android/GalleryImageview

Các kiểu Dialog trong Android

Hộp thoại Android với chọn Option

Trong Android, bằng cách sử dụng hộp thoại chúng ta có thể chọn một tùy chọn từ nhiều tùy chọn bằng cách sử dụng nút radio.

Code ví dụ cho dialog này như sau :

```
public class ExampleApp extends Activity
{
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        final String items[] = {"item1","item2","item3"};


        AlertDialog.Builder ab=new AlertDialog.Builder(ExampleApp.this);
        ab.setTitle("Title");
        ab.setSingleChoiceItems(items, 0,new
DialogInterface.OnClickListener() {
```

```
                public void onClick(DialogInterface dialog, int whichButton) {
                    // onClick Action
                }
            })
            .setPositiveButton("Ok", new DialogInterface.OnClickListener() {
                public void onClick(DialogInterface dialog, int whichButton) {
                    // on Ok button action
                }
            })
            .setNegativeButton("Cancel", new DialogInterface.OnClickListener()
{
                public void onClick(DialogInterface dialog, int whichButton) {
                    // on cancel button action
                }
            });
            ab.show();
        }
}
```

Hộp thoại Android với HTML

Trong Android, chúng ta có thể sử dụng một số thẻ HTML cho văn bản. Bằng cách sử dụng HTML tag chúng tôi có thể làm cho một văn bản như in đậm và chúng ta có thể thay đổi màu sắc, vv.

Ở đây tôi sử dụng '<b>' tag cho từ khóa '
' Bold & cho dòng mới trong một hộp thoại cảnh báo.

chúng ta có thể sử dụng mã cho văn bản textview, văn bản nút, vv ...

```
public class ExampleApp extends Activity
{
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        AlertDialog.Builder ab=new AlertDialog.Builder(ExampleApp.this);
        ab.setMessage(Html.fromHtml(" Html View " +"
Androidpeople.com"));
        ab.setPositiveButton("ok", null);
        ab.show();
```

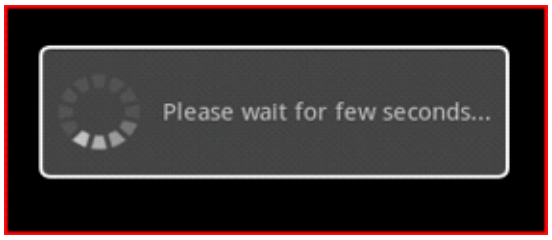
```
    }
}
```

Android Progress Dialog

Trong Android, chúng tôi có thể hiển thị thanh tiến trình thông qua hộp thoại. Đối với điều này chúng ta cần phải sử dụng ProgressDialog.

Dưới đây là một ví dụ về cách sử dụng ProgressDialog: -

```
public class ExampleApp extends Activity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        ProgressDialog dialog = ProgressDialog.show(ExampleApp.this, "",
                "Please wait for few seconds...", true);
    }
}
```



Android Dialog với Radio Buttons

Bây giờ chúng ta có thể thấy một ví dụ đơn giản về cách sử dụng các radio groups, actions, and radio trong điện thoại di động Android.

Ví dụ cho Radio Buttons Android Dialog: -

```
public class ExampleApp extends Activity {
```

```
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        final CharSequence[] PhoneModels = {"iPhone", "Nokia",
"Android"};
        AlertDialog.Builder alt_bld = new AlertDialog.Builder(this);
        alt_bld.setIcon(R.drawable.icon);
        alt_bld.setTitle("Select a Phone Model");
        alt_bld.setSingleChoiceItems(PhoneModels, -1, new
DialogInterface.OnClickListener() {
            public void onClick(DialogInterface dialog, int item) {
                Toast.makeText(getApplicationContext(), "Phone Model =
"+PhoneModels[item],   Toast.LENGTH_SHORT).show();
            }
        });
        AlertDialog alert = alt_bld.create();
        alert.show();
    }
}
```

Cho phép truy cập các tính năng trong Android

Như chúng ta đã biết bất cứ khi nào chúng ta sử dụng một tính năng đặc biệt hoặc API chúng ta cần phải yêu cầu sự cho phép trong file AndroidManifest.xml để sử dụng. Nếu chúng ta không chỉ định bất kỳ truy cập nào trong hệ thống, khi đó ứng

dụng sẽ không sử dụng các tính năng mà cần sử dụng nằn trong danh mục được cấp phép mới có quyền.

Quyền được cấp cho ứng dụng bằng cách cài đặt gói cần cài đặt. Nhưng không phải tất cả các điều khoản sẽ được cấp cho hệ thống. Có một số sự cho phép hệ thống mà sẽ không được cấp cho các ứng dụng người dùng, nhưng chỉ đến các ứng dụng hệ thống. Sau đây là một số trong những điều khoản đó không thể được cấp cho các ứng dụng người dùng.

- **android.permission.ACCESS_CHECKIN_PROPERTIES**
- **android.permission.ACCESS_SURFACE_FLINGER**
- **android.permission.ACCOUNT_MANAGER**
- **android.permission.BIND_APPWIDGET**
- **android.permission.BIND_DEVICE_ADMIN**
- **android.permission.BIND_INPUT_METHOD**
- **android.permission.BIND_WALLPAPER**
- **android.permission.BRICK**
- **android.permission.BROADCAST_PACKAGE_REMOVED**
- **android.permission.BROADCAST_SMS**
- **android.permission.BROADCAST_WAP_PUSH**
- **android.permission.CALL_PRIVILEGED**
- **android.permission.CHANGE_COMPONENT_ENABLED_STATE**

- **android.permission.CLEAR_APP_USER_DATA**
- **android.permission.CONTROL_LOCATION_UPDATES**
- **android.permission.DELETE_CACHE_FILES**
- **android.permission.DELETE_PACKAGES**
- **android.permission.DEVICE_POWER**
- **android.permission.DIAGNOSTIC**
- **android.permission.FACTORY_TEST**
- **android.permission.FORCE_BACK**
- **android.permission.GLOBAL_SEARCH**
- **android.permission.HARDWARE_TEST**
- **android.permission.INJECT_EVENTS**
- **android.permission.INSTALL_LOCATION_PROVIDER**
- **android.permission.INSTALL_PACKAGES**
- **android.permission.INTERNAL_SYSTEM_WINDOW**
- **android.permission.MANAGE_APP_TOKENS**
- **android.permission.MASTER_CLEAR**
- **android.permission.READ_FRAME_BUFFER**
- **android.permission.READ_INPUT_STATE**
- **android.permission.REBOOT**
- **android.permission.SET_ACTIVITY_WATCHER**
- **android.permission.SET_ORIENTATION**
- **android.permission.SET_PREFERRED_APPLICATIONS**
- **android.permission.SET_TIME**
- **android.permission.STATUS_BAR**
- **android.permission.UPDATE_DEVICE_STATS**
- **android.permission.WRITE_GSERVICES**
- **android.permission.WRITE_SECURE_SETTINGS**

Để có được các điều khoản, ứng dụng phải được ký kết với phím đó sử dụng để đăng các nền tảng. Điều này có thể khác nhau cho các nhà sản xuất. Vì vậy, nó thực tế không thể có được những quyền được cấp cho một ứng dụng người dùng.

Danh sách các ứng dụng Permissions thường gặp được sử dụng bởi nhà phát triển ứng dụng Android

Google Location-based Services

- **android.permission.ACCESS_COARSE_LOCATION**
- **android.permission.ACCESS_FINE_LOCATION**

Accessing the Device's Contact Database

- **android.permission.READ_CONTACTS**
- **android.permission.WRITE_CONTACTS**

Accessing User's Calendars

- **android.permission.READ_CALENDAR**
- **android.permission.WRITE_CALENDAR**

Changing Phone Settings

- **android.permission.SET_ORIENTATION**
- **android.permission.SET_TIME_ZONE**
- **android.permission.SET_WALLPAPER**

Making, Monitoring Phone Calls

- **android.permission.CALL_PHONE**
- **android.permission.CALL_PRIVILEGED**

Sending and Receiving SMS/MSS messages

- **android.permission.READ_SMS**
- **android.permission.RECEIVE_MMS**
- **android.permission.RECEIVE_SMS**
- **android.permission.RECEIVE_WAP_PUSH**
- **android.permission.SEND_SMS**
- **android.permission.WRITE_SMS**

Edit and Utilize Audio Settings

- **android.permission.RECORD_AUDIO**
- **android.permission.MODIFY_AUDIO_SETTINGS**

Access to Network Settings

- **android.permission.ACCESS_NETWORK_STATE**
- **android.permission.CHANGE_NETWORK_STATE**

Access to Wi-Fi Settings

- **android.permission.ACCESS_WIFI_STATE**
- **android.permission.CHANGE_WIFI_STATE**

Using Internet Network Socket

- **android.permission.INTERNET**

Access to Phones Hardware

- **android.permission.BLUETOOTH**
- **android.permission.CAMERA**
- **android.permission.FLASHLIGHT**
- **android.permission.VIBRATE**
- **android.permission.BATTERY_STATS**

Manage Google Accounts and Services

- **android.permission.GET_ACCOUNTS**
- **android.permission.MANAGE_ACCOUNTS**

Sử dụng MySpace SDK trong Android

Các Myspace Android SDK cung cấp các chức năng cốt lõi để tích hợp tài nguyên Myspace vào Android ứng dụng của bạn

Những hành động sau đây được tiếp xúc trực tiếp thông qua các SDK :

- Integrated Login and Logout experience
- Get and Update Status Mood

- Get Friends
- Upload Photo
- Upload Video
- newGet Friends Status Mood
- newCreate and Get Activities.

SDK này cũng đi kèm với một mẫu Myspace Android ứng dụng chứng minh sử dụng các hành động trên.

Trong vài tuần tới chúng tôi sẽ bổ sung thêm các hành động nhiều hơn trong SDK, hoặc bạn có thể thực hiện bất kỳ API đó là tiếp xúc thông qua MySpaceID trực tiếp.

Dưới đây là một vài hình ảnh mà SDK hỗ trợ cho bạn, bạn cũng có thể run ví dụ có sẵn đính kèm trong SDK.

Integrated Login

Get and Set Status Mood

Upload Photo

Get Friends

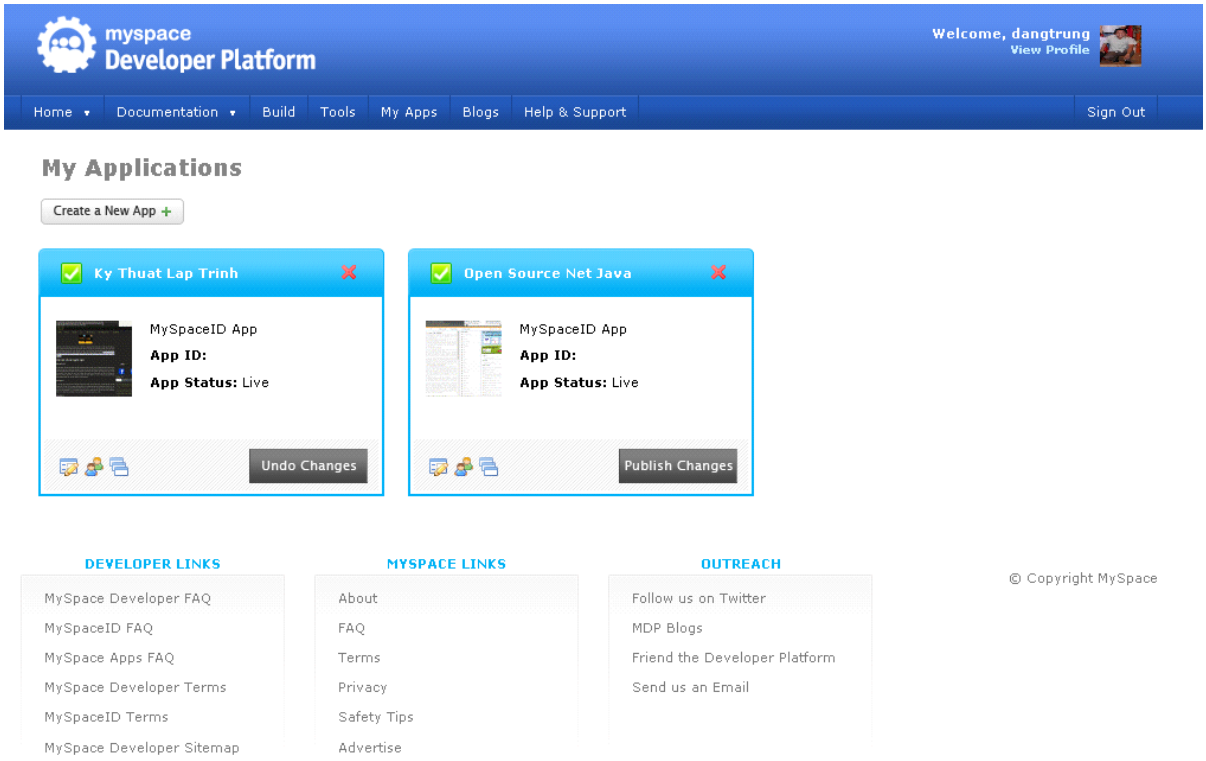
SVN : http://myspace-android-sdk.googlecode.com/svn/trunk/

Download : http://code.google.com/p/myspace-android-sdk/downloads/list

Đôi nét phát triển ứng dụng Android với mạng xã hội

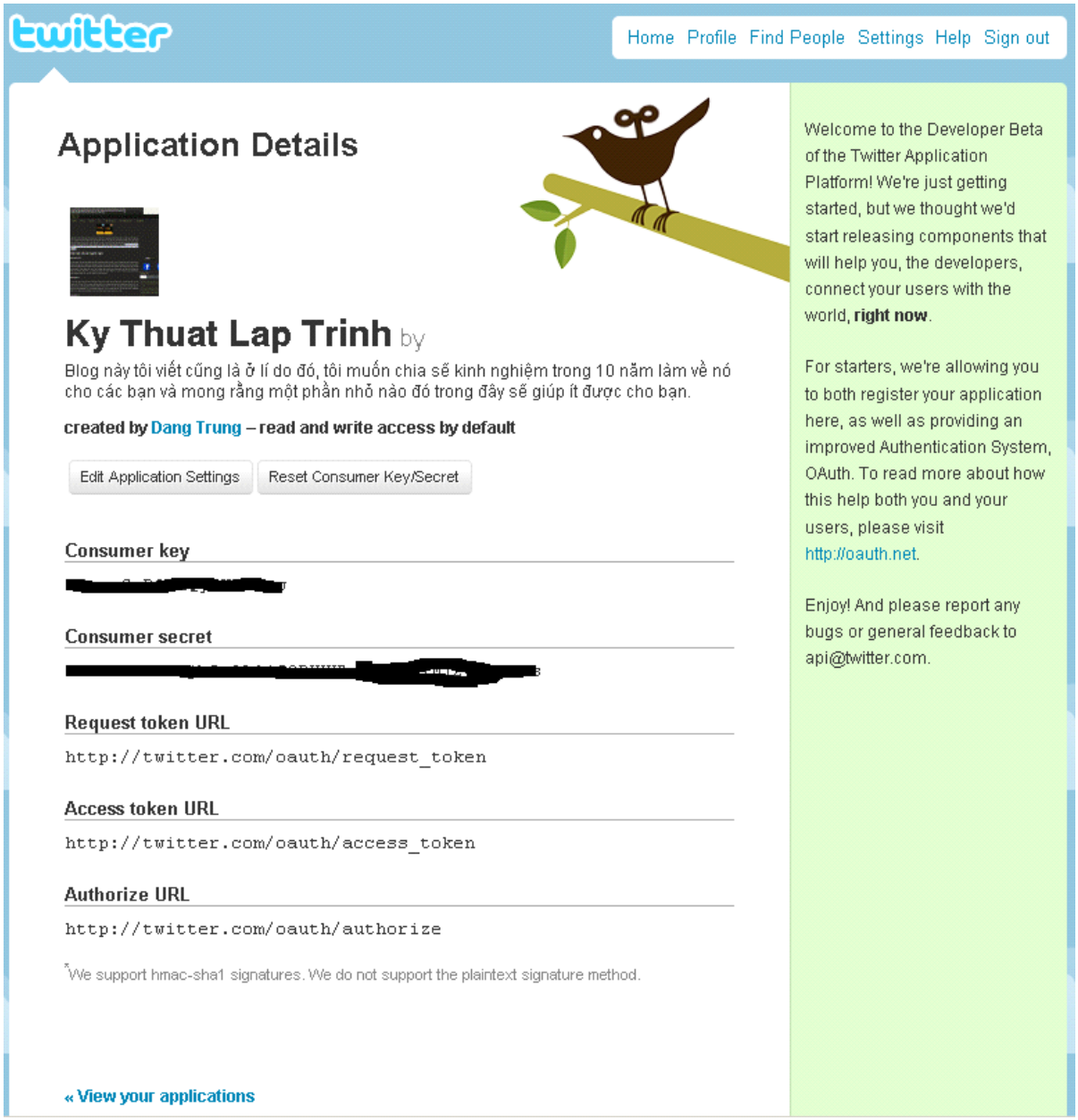
Trong phần đầu tôi đã giới thiệu sơ lược về SDK của web mạng xã hội như MySpace ngoài ra còn nhiều web site khác về mạng xã hội. Trong source mà tôi đã giới thiệu và tôi chắc chắn rằng bạn đã download về và bung nó ra, import vào eclipse.

Trong các mạng xã hội hiện nay, họ đều dùng chung phương thức chừng thực tài khoản, thông qua một nhà cung cấp thứ 3.



Nếu bạn muốn phát triển ứng dụng làm việc với các mạng xã hội, trước tiên bạn nên nghỉ sẽ phải tạo một app trên mang xã hội đó. Khi đó nó sẽ cấp (Consumer key, Consumer secret) và những thông số này là chìa khóa mở cửa trước khi bạo vào phòng.

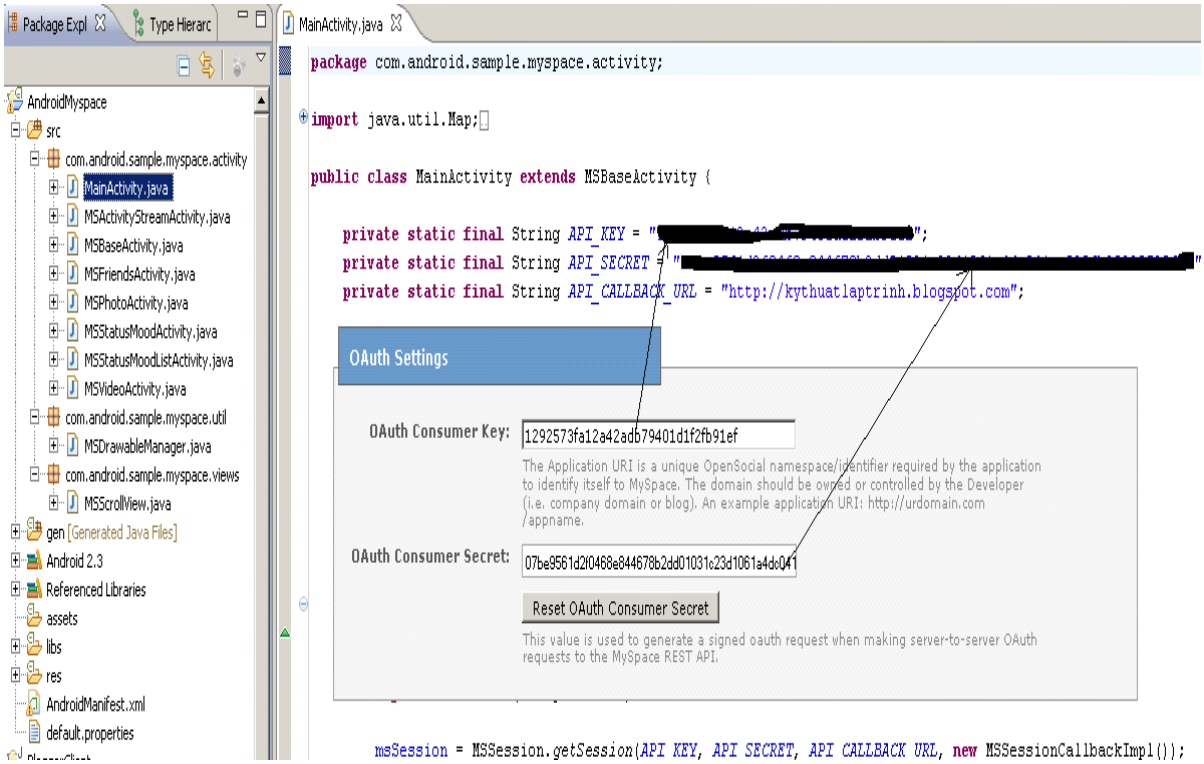
Tùy theo web xã hội dễ hay khó, tức là các thức web cung cấp đăng ký, đối với Twitter tương đối đơn giản hơn Facebook một chút.



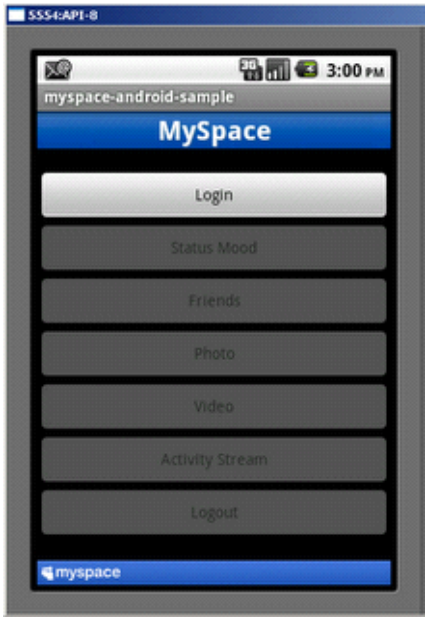
Trong các hình trên tôi đã cho các bạn thấy các đăng ký Application cho cac web xã hội như thế nào? và họ cung cấp cho mình những gì? **Consumer Key**, **Consumer Secret** là những chỉ số được sử dụng cho OAuth

Làm thế nào login MySpace trong Adnroid?

Get source từ SVN : **http://myspace-android-sdk.googlecode.com/svn/trunk/myspace-android-sample/**, Sau khi hoàn tất và import vào Eclipse phải đảm bảo rằng tất cả đầy đủ như hình sau,



Nhập key đăng ký mà MySpace đã cung cấp cho bạn (đăng ký ở đây) vào **MainActivity** theo mũi tên như hình phía bên trên. Nếu bạn làm đúng theo trình tự kết quả sẽ có sau khi run với Android.



Click login button, khi ấy bạn sẽ thấy màn hình sau :

Sau khi đăng nhập thành công, bạn sẽ thấy như sau :

Bây giờ bạn có thể làm việc trên MySpace trên Android.

Tìm hiểu về Intents và Intent Filters trong Android

Ba trong số các thành phần cốt lõi của một ứng dụng - activity, service và receivers broadcast - được kích hoạt thông qua tin nhắn, intents call. Intent message là một cơ sở cho run-time ràng buộc giữa các thành phần trong các ứng dụng giống và khác nhau. Mục đích chính một đối tượng Intent, là một cấu trúc dữ

liệu thụ động tổ chức mô tả trừu tượng của hoạt động được thực hiện - hoặc, thường trong trường hợp các **broadcasts**, mô tả về một điều gì đó đã xảy ra và đang được công bố. Có những cơ chế riêng cho việc cung cấp intents cho từng loại component:

- Một đối tượng Intent được thông qua đến **Context.startActivity()** hoặc **Activity.startActivityForResult()** để khởi động một activity hoặc có một activity hiện tại để thực hiện điều gì đó mới. (Nó cũng có thể được truyền cho **Activity.setResult()** để trả về thông tin cho hoạt động này đó được gọi là **startActivityForResult()**.)
- Một đối tượng Intent được thông qua với **Context.startService()** để bắt đầu một service hoặc cung cấp chỉ dẫn mới với một dịch vụ đang diễn ra. Tương tự như vậy, intent có thể được thông qua cho **Context.bindService()** cho thiết lập kết nối giữa các component gọi điện thoại và service một mục tiêu. Nó tùy chọn có thể bắt đầu các service nếu nó chưa được chạy.
- Các đối tượng Intent truyền cho bất kỳ method broadcast (như **Context.sendBroadcast(), Context.sendOrderedBroadcast()**, hoặc **Context.sendStickyBroadcast()**) đều gửi đến tất cả các receivers broadcast quan tâm đến. Rất nhiều loại broadcast trong mã nguồn hệ thống.

Trong mỗi trường hợp, hệ thống Android tìm thấy dịch vụ activity, thích hợp, hoặc thiết lập các receivers broadcast cho đáp ứng mục đích, instantiating chúng nếu cần thiết. Không có sự chồng chéo trong các hệ thống nhắn tin: intents Broadcast được phân phối duy nhất cho broadcast receivers , không bao giờ cho activity, service. Một intent thông qua với **startActivity()** được phân phối duy nhất cho activity có, không bao giờ cho một service hoặc receiver broadcast, vv.

Đối tượng Intent

Một đối tượng Intent là một gói thông tin. Nó chứa thông tin có ích cho các component tiếp nhận intent (chẳng hạn như các hành động được thực hiện và các dữ liệu với hoạt động trên) cộng với các thông tin quan tâm đến hệ thống Android (như các chủng loại thành phần nên xử lý các intent và hướng dẫn làm thế nào với khởi động một activity mục tiêu). Về cơ bản, nó thể chứa những điều sau đây:

Component name
Tên của các component quản lý các intent. Trường này là một đối tượng ComponentName

Tên component là tùy chọn. Nếu nó được thiết lập, các đối tượng Intent là cung cấp cho một instance của lớp được chỉ định. Nếu nó không được thiết lập, Android sử dụng các thông tin khác trong đối tượng Intent với xác định một mục tiêu phù hợp.

Tên component là thiết lập bởi **setComponent(), setClass(),** hoặc **setClassName()** và đọc bởi **getComponent().**

Action
Chuỗi đặt tên cho các action được thực thi - hoặc, trong trường hợp intents broadcast, các action đó đã diễn ra và được thông báo. Lớp Intent xác định một số constants action, bao gồm cả các:

- ***ACTION_CALL*** ---- **activity** ---- khởi đầu một cuộc gọi điện thoại.
- ***ACTION_EDIT*** ---- **activity** ---- Hiển thị dữ liệu cho người sử dụng để chỉnh sửa.
- ***ACTION_MAIN*** ---- **activity** ---- Khởi động activity ban đầu của công việc, không có dữ liệu đầu vào và đầu ra.
- ***ACTION_SYNC*** ---- **activity** ---- Đồng bộ dữ liệu trên một máy chủ với dữ liệu trên thiết bị di động.
- ***ACTION_BATTERY_LOW*** ---- **receiver broadcast** ---- Một cảnh báo đó pin yếu.
- ***ACTION_HEADSET_PLUG*** **----** **broadcast receiver** ---- Một tai nghe đã được cắm vào thiết bị, hoặc Unplugged từ nó.
- ***ACTION_SCREEN_ON*** ---- **receiver broadcast** ---- màn hình đã được bật.
- ***ACTION_TIMEZONE_CHANGED*** ---- **receiver broadcast** **----** Các thiết lập cho múi thời gian đã thay đổi.