### **Activity trong android là gì?**

**Activity** là một thành phần của ứng dụng Android nó sẽ biểu diễn giao diện của một màn hình nào đó trong ứng dụng của bạn.Một ứng dụng có thể sẽ có nhiều màn hình và mỗi màn hình có thể là một Activity (chưa nói đến Fragment).

Mỗi Activity sẽ hoạt động độc lập với nhau nhưng có thể tương tác và truyển dữ liệu qua nhau.Vì hoạt động độc lập nên có sẽ có vòng đời riêng từ lúc được khởi tạo cho đến lúc được huỷ đi.

Bây giờ bạn thử tạo new một dự án mà bài trước mình đã có hướng dẫn để xem coi Activity nó được thiết lập và làm gì nhé, nhìn xuống ảnh phía dứới đây ha.

Project đầu tiên lập trình Android

Ở trên bạn sẽ thấy một class tên là **MainActivity** đúng không nào? Đây chính là một Activity đó, không phải là nó có chữ Activity đâu nhé cái tên do mình tự đặt thôi.

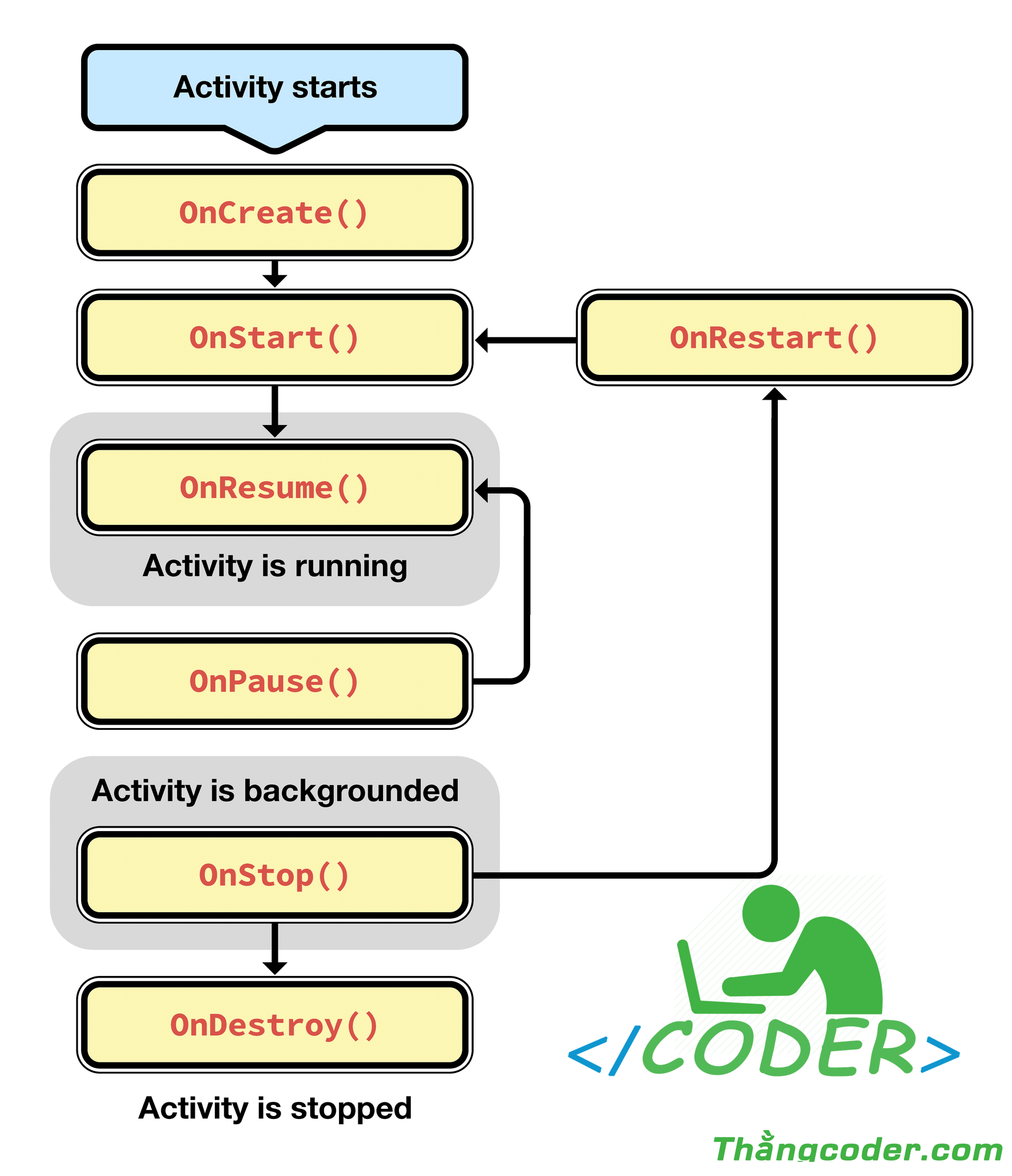
#### Thế nào là một Activity?

Một class được gọi là Activity khi nó **extend** (kế thừa) từ những class cha như : AppCompatActivity,  Activity, FragmentActivity… và nhìn trên ảnh bạn sẽ thấy một hàm tên là **onCreate** đúng không nào?

Đây là hàm bắt buộc phải có trong Activity bởi vì nó có nhiệm vụ là khởi tạo Activity đó lên và nó chỉ được khởi tạo một lần duy nhất trong 1 vòng đời của Activity, lưu ý là không có cũng chẳng bị lỗi gì nhé.Vì là chỉ khởi tạo một lần duy nhất nên nó thường là nơi vẽ ra giao diện của Activity.

Và rõ hơn hơn các bạn xem qua vòng đời của Activity nhé.

#### Vòng đời của một Activity  (Life cycle)



Vòng đời của một Activity Android

Chi tiết nó sẽ như thế này, mình viết như dạng kể chuyển cho dễ hiểu nhé:

Khi một [Activity](http://thangcoder.com/lap-trinh-android/chuyen-doi-giua-cac-activity-voi-nhau-trong-android) được khởi tạo nó sẽ gọi hàm **OnCreate()** đầu tiền, ở đây nó sẽ vẽ lên giao diện của màn hình đó lúc này giao diện vẫn chưa hiển thị ra đâu.Và tiếp sau đó nó sẽ gọi đến hàm OnStart(), lúc này giao diện đã được vẽ ra rồi và bạn có thể nhìn thấy được giao diện đó tuy nhiên vẫn chưa tương tác vào được, ví dụ như chưa click vào nút button ” ***Đăng ký*** ” chẳng hạn.

Tiếp đến nó sẽ gọi đến hàm **OnResume()** , ở đây thì giao diện đã được khởi tạo xong và người dùng đã có thể tương tác trên giao diện đó và lúc này nếu bạn thao tác gì làm ẩn Activity đi thì nó sẽ dừng ở đây, activity đang hoạt động.Lúc này sẽ có 2 trường hợp xảy ra tiếp theo:

##### Trường hợp 1 : Activity bị che mất 1 phần

Bây giờ, nếu như có một Activity khác đè lên trên Activity này và chỉ che khuất 1 phần nào đó thôi chứ không che khuất hoàn toàn, nghĩa là Activity đầu tiên bây giờ chúng ta không click hay làm gì được những vẫn thấy 1 phần có nó trên màn hình.Lúc này đây hàm **onPause()** sẽ được gọi .Và khi Activity vừa đè lên bị hủy bỏ hay xóa đi thì hàm**OnResume()** sẽ được gọi và bây giờ chúng ta có thể thao tác, click… được trên Activity này.

##### Trường hợp 2 : Activity bị che mất toàn phần

Nếu như có một cuộc gọi đến chẳng hạn thì giao diện cuộc gọi đến sẽ che mất màn hình của Activity này thì lúc này hàm OnPause() và gọi luôn hàm OnStop() ngay sa đó, nghĩa là Activity hiện tại bị che khuất hoàn toàn rồi nhé.Sa khi cuộc gọi kết thúc thì hàm **OnRestart()** sẽ được gọi, tiếp đến hàm **OnStart()** và **OnResume()** cũng được gọi ngay sau đó luôn và bây giờ Activity lai quay về trạng thái hoạt động bình thường.

Một hàm cuối cùng đó là hàm **OnDestroy()**, hàm này sẽ được gọi khi bạn bấm nut back trên điện thoại để thoát ứng dụng hoặc tắt đi một màn hình activity nào đó.Sau khi hàm nay được gọi thì Activity đó sẽ bị hủy luôn nhé, không start lại được đâu.

Cơ bản vòng đời của một Activity chỉ có vậy thôi, mục đích của vòng đời này là giúp chúng ta có thể xử lí các sự kiện khi activity ở các trạng thái khác nhau.Ví dụ như ở màn hình A bạn có start một Broadcast Receiver (sẽ học ở phần nâng cao) lắng nghe trạng thái mạng của điện thoại ở hàm **OnStart()** thì nếu muốn tắt nó đi thì mình sẽ hủy nó ở hàm **OnDestroy()**, vì lúc này màn hình Activity này đã mất nên mình hủy nó luôn.

Cơ bản về [vòng đời Activity](http://thangcoder.com/lap-trinh-android/hoc-lap-trinh-android-can-ban/activity-va-vong-doi-cua-no-trong-lap-trinh-android)nó chỉ như thế thôi, và gần cuối serie này bạn sẽ học về thằng Fragment nó cũng tương tự như Activity nhưng rất là hay luôn.

Và dưới đây là video và code vi dụ trong bài viết này nếu như bạn chưa hiểu có thể xem và tải về thực hành nhé.

# **Kiến trúc của hệ điều hành Android**

Hệ điều hành Android là 1 ngăn xếp các thành phần phần mềm, được chia thành 5 phần và 4 lớp chính như trong hình bên dưới.



**Linux Kernel**

Dưới cùng là lớp Linux - Linux 3.6 cùng với khoảng 115 bản vá. Lớp này cung cấp 1 cấp độ trừu tượng giữa phần cứng của thiết bị và các thành trình điều khiển phần cứng thiết yếu như máy ảnh, bàn phím, màn hình hiển thị... Đồng thời, hạt nhân (kernel) còn xử lý tất cả các thứ mà Linux có thể làm tốt như mạng kết nối và 1 chuỗi các trình điều khiển thiết bị, giúp cho giao tiếp với các thiết bị ngoại vi dễ dàng hơn.

**Các thư viện**

Ở trên lớp nhân Linux là tập các thư viện bao gồm WebKit - trình duyệt Web mã nguồn mở, được biết đến như thư viện libc, cơ sở dữ liệu SQLite - hữu dụng cho việc lưu trữ và chia sẻ dữ liệu ứng dụng, các thư viênj chơi và ghi âm audio, video, hay các thư viện SSL chiụ trách nhiệm bảo mật Internet...

**Các thư viện Android**

Đây là các thư viện dựa trên Java phục vụ cho việc phát triển Android. Ví dụ của các thư viện này bao gồm các thư viện ứng dụng dùng để xây dựng giao diện người dùng, vẽ đồ họa hay truy cập cơ sở dữ liệu. 1 số thư viện chính của Android:

* android.app - Cung cấp quyền truy cập vào ứng dụng và là nền tảng của tất cả ứng dụng Android.
* android.content - Cung cấp quyền truy cập nội dung (content), truyền tải thông điệp giữa các ứng dụng hay các thành phần của ứng dụng.
* android.database - Đựoc sử dụng để truy cập dữ liệu của content provider và cơ sở dữ liệu SQLite
* android.opengl - giao diện các phuơng thức Java để sử dụng OpenGL ES
* android.os - Cung cấp các ứng dụng với quyền truy cập vào các dịch vụ của hệ điều hành bao gồm thông điệp, các dịch vụ hệ thống và các giao tiếp nội tại
* android.text - Đựoc sử dụng để hiển thị và điều chỉnh chữ trên màn hình thiết bị
* android.view - Các thành phần cơ bản trong việc xây dựng giao diện người dùng của ứng dụng.
* android.widget - Tập các thành phần giao diện người dùng đã được xây dựng sẵn như các nút, các nhãn (label), list view,....
* android.webkit - Tập các lớp cho phép xây dựng khả năng duyệt web.

**Android Runtime**

Đây là phần thứ 3 của kiến trúc và nằm ở lớp thứ 2 từ dưới lên. Phần này cung cấp 1 bộ phận quan trọng là Dalvik Vỉtual Machine - là 1 loại Java Virtual Machine được thiết kế đặc biệt để tối ưu cho Android.

Dalvik VM sử dụng các đặc trưng của nhân Linux như quản lý bộ nhớ và đa luồng, những thứ mà đã có sẵn trong Java. Dalvik VM giúp mọ ứng dụng Android chạy trong tiến trình riêng của nó, với các thể hiện (instance) riêng của Dalvik virtual Machine.

Android Runtime cũng cung cấp 1 tập các thư viện chính giúp các nhà phát triển ứng dụng Android có thể viết ứng dụng Android bằng Java

**Application Framework**

Lớp Android Framework cung cấp các dịch vụ cấp độ cao hơn cho các ứng dụng dưới dạng các lớp Java. Các nhà phát triển ứng dụng được phép sử dụng các dịch vụ này trong ứng dụng của họ.

Android Framework bao gồm các dịch vụ chính sau:

* Activitty Manager - Kiểm soát tất cả khía cạnh của vòng đời ứng dụng và ngăn xếp các Activity.
* Content Providers - Cho phép các ứng dụng chia sẽ dữ liệu với các ứng dụng khác.
* Resource Manager - Cung cấp quyền truy cập vào các tài nguyên như các chuỗi, màu sắc, các layout giao diện người dùng...
* Notifications Manager - Cho phép các ứng dụng hiển thị cảnh báo và các thông báo cho người dùng.
* View System - Tập các thành phần giao diện (view) được sử dụng để tạo giao diện người dùng.

**Application**

Lớp trên cùng của kiến trúc là Application. Các ứng dụng bạn tạo ra sẽ được cài đặt trên lớp này. Ví dụ như: Danh bạ, nhắn tin, trò chơi...

**Threads and Networking**

* **Threads, AsyncTasks & Handlers - Part 1**

Các hệ thống cầm tay, giống như tất cả các thiết bị điện toán ngày nay, ngày càng chứa nhiều lõi (core) máy tính. Có nghĩa là nhiều chương trình hoặc luồng thực thi có thể chạy trên thiết bị của bạn cùng một lúc. Nó cho phép bạn làm được nhiều việc hơn trong một khoảng thời gian ngắn hơn.

Nhưng nó cũng có thể làm cho chương trình của bạn phức tạp hơn nhiều, dẫn đến lỗi và các vấn đề về hiệu suất, nếu bạn không cẩn thận. Vì vậy, trong bài học này, chúng ta sẽ nói về việc viết các chương trình đa luồng (**multi-threaded**) cho Android. Và chúng ta cũng sẽ thảo luận về một số lớp mà Android cung cấp để hỗ trợ điều này.

Cụ thể, trong bài học này tôi sẽ bắt đầu với một cuộc thảo luận ngắn gọn về việc xâu chuỗi chính nó. Tiếp theo, tôi sẽ nói về luồng giao diện người dùng của Android, luồng chính trong đó các ứng dụng Android thực hiện hầu hết công việc của họ. Và tôi cũng sẽ thảo luận về cách điều này ảnh hưởng đến việc thiết kế phần mềm ứng dụng của bạn.

Sau đó, tôi sẽ nói về lớp AsyncTask, giúp đơn giản hóa việc xâu chuỗi trong Android.

Và cuối cùng, tôi sẽ kết thúc với một cuộc thảo luận về lớp Handler, một cơ chế phân luồng Android khác.

**Thread là gì?**

* Về mặt khái niệm, một luồng (a thread) là một trong nhiều tính toán (computation) có thể chạy cùng một lúc, trong một quy trình hệ điều hành.
* Về mặt triển khai, mỗi luồng có bộ đếm chương trình và thời gian chạy chương trình riêng, nhưng chia sẻ vùng heap và vùng nhớ tĩnh với các luồng khác đang chạy trong một quy trình của hệ điều hành.

***Mở rộng: Process và Thread***

*Process là quá trình hoạt động của một ứng dụng. Điều đó nghĩa là gì? Có thể lấy một ví dụ như sau, khi bạn click đúp chuột vào biểu tượng MS Word, một process chạy ứng dụng Word được khởi tạo.* ***Thread là một bước điều hành bên trong một process, Thread được định nghĩa là một luồng dùng để thực thi một chương trình****. Một process dĩ nhiên có thể chứa nhiều thread bên trong nó. Khi chúng ta chạy ứng dụng Word, hệ điều hành tạo ra một process và bắt đầu chạy các thread chính của process đó.*

*Điểm quan trọng nhất cần chú ý là một thread có thể làm bất cứ nhiệm vụ gì một process có thể làm. Tuy nhiên, vì một process có thể chứa nhiều thread, mỗi thread có thể coi như là một process nhỏ. Vậy, điểm khác biệt mấu chốt giữa thread và process là công việc mỗi cái thường phải làm.*

*Một điểm khác biệt nữa đó là nhiều thread nằm trong cùng một process dùng một không gian bộ nhớ giống nhau, trong khi process thì không. Điều này cho phép các thread đọc và viết cùng một kiểu cấu trúc và dữ liệu, giao tiếp dễ dàng giữa các thread với nhau. Giao thức giữa các process, hay còn gọi là IPC (inter-process communication) thì tương đối phức tạp bởi các dữ liệu có tính tập trung sâu hơn.*

*Vậy Thread là gì? Thread được định nghĩa là một luồng dùng để thực thi một chương trình. Java Virtual Machine cho phép một chương trình có thể có nhiều Thread thực thi đồng thời. Mỗi Thread đều có độ ưu tiên của nó. Rồi mỗi Thread có thể được đánh dấu là daemon . Daemon Thread là một loại Thread có độ ưu tiên thấp, cung cấp dịch vụ cho người dùng, là Thread duy trì hoạt động cho đến khi tất cả các Threads khác hoàn thành công việc hay chết đi thì nó cũng mới chết theo. Ví dụ cụ thể là trình dọn rác trong Java là một Daemon Thread. Để tạo mới Thread ta có hai cách. Cách thứ nhất là kế thừa (extends) từ class Thread:*

*private class MyThread extends Thread{*

*@Override*

*public void run() {*

*//TODO*

*}*

*}*

*...*

*new MyThread().start();*

*Cách thứ 2 là thực thi (implements) interface Runnable:*

*private class MyRunnable implements Runnable{*

*@Override*

*public void run() {*

*//TODO*

*}*

*}*

*...*

*new Thread(new MyRunnable()).start();*

***Sau đây là bảng tổng kết sự khác nhau giữa thread và process:***

* *Để tạo nhiều thread thì dễ dàng hơn so với process vì chúng không cần các địa chỉ nhớ riêng rẽ.*
* *Việc chạy đa luồng cần được lập trình một cách chi tiết vì các thread chia sẻ các cấu trúc chung mà chỉ sử dụng được bởi từng thread vào mỗi thời điểm. Khác với thread, các process không dùng chung địa chỉ nhớ.*
* *Thread được xếp hạng “nhẹ cân” bởi vì chúng sử dụng ít tài nguyên hơn so với các process.*
* *Các process chạy độc lập với nhau. Các thread thì sử dụng chung các địa chỉ nhớ liên kết với nhau, vì thế cần thận trọng tránh việc thread này nhảy sang thread khác. (Điều đã được nhắc đến trong ý thứ 2 vừa trên)*
* *Một process có thể chứa nhiều thread****.***

*Trong Android, khi chương trình được khởi chạy, hệ thống sẽ start một Thread ban đầu cùng với một Process. Thì Thread đó chính là* ***Main Thread****. Vậy vì sao Main Thread lại thường được gọi là UI Thread thì có 2 lý do chính đáng sau đây.*

* *Thread này có nhiệm vụ gửi các sự kiện đến widget, tức là đến các view ở giao diện điện thoại, thậm chí cả các sự kiện vẽ.*
* *Ngoài ra Thread này cũng phải tương tác với bộ công cụ Android UI (Android UI toolkit) gồm hai gói thư viện là android.widget và android.view.*

***??Có khi nào Main Thread lại không được gọi là UI Thread không?***

***=>*** *Đó là khi một chương trình có nhiều hơn một Thread phụ trách việc xử lý giao diện.*

*=> Còn một trường hợp nữa là Worker Thread, chính là Thread mà bạn tạo thêm cho chương trình để nó thực thi một công việc nào đó không liên quan đến giao diện, Thread này cũng được gọi là* ***Background Thread****.*

Ví dụ: Ở đây, tôi đang hiển thị một thiết bị máy tính giả định. Bây giờ thiết bị này có hai CPU, CPU 1 và CPU 2. Mỗi CPU này có thể thực hiện các hướng dẫn tạo nên các ứng dụng đang chạy trên thiết bị của bạn. Bây giờ trên CPU 2, tôi đang hiển thị hai quá trình đang chạy, p3 và p4. Bây giờ, một cách để suy nghĩ về các quy trình là chúng là các môi trường thực thi khép kín. Họ có các tài nguyên như bộ nhớ, tệp mở, kết nối mạng và những thứ khác mà họ quản lý và tách biệt với các quy trình khác trên thiết bị của bạn. Và trong một trong các quy trình này, p4, tôi đang hiển thị hai luồng đang chạy, T7 và T8. Bây giờ, mỗi luồng này là một luồng hướng dẫn thực hiện tuần tự, với ngăn xếp cuộc gọi riêng của nó. Nhưng vì chúng nằm trong cùng một quy trình, mỗi người có thể truy cập các tài nguyên quy trình được chia sẻ, bao gồm bộ nhớ nhiệt và các biến tĩnh.

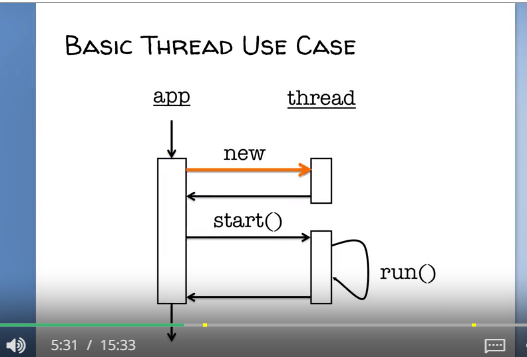
**Trong Java, các luồng (threads) được đại diện bởi một đối tượng của luồng loại trong gói java.lang. Các luồng Java thực hiện giao diện runnable. Điều đó có nghĩa là họ phải có một phương thức công khai gọi là run, không có đối số và không có giá trị trả về.**

Bây giờ một số phương thức luồng mà chúng ta sẽ thấy trong bài học này bao gồm phương thức bắt đầu (start method) để bắt đầu 1 luồng.

Và phương pháp ngủ (the sleep method) để tạm thời đình chỉ một luồng. Một số phương thức đối tượng mà bạn có thể cần khi bạn đang sử dụng, khi bạn đang sử dụng các luồng bao gồm phương thức chờ (wait method), cho phép luồng hiện tại từ bỏ khóa mà nó giữ và đợi cho đến khi một luồng khác gọi một phương thức tương ứng, chẳng hạn như thông báo, hoặc thông báo tất cả. Và khi điều này xảy ra, chuỗi chờ có thể yêu cầu khóa mà nó đã từ bỏ khi nó được gọi là chờ và có thể tiếp tục thực thi. Phương thức thông báo (notify method) đánh thức một luồng đang chờ trên đối tượng này. Bây giờ để sử dụng một luồng bạn thường làm những điều sau đây:

* Đầu tiên, bạn tạo luồng. Ví dụ bằng cách sử dụng lệnh luồng mới. Bây giờ các chủ đề không tự động bắt đầu khi bạn tạo chúng. Để bắt đầu chuỗi, bạn cần gọi phương thức bắt đầu (start method) của luồng.

Làm điều này cuối cùng dẫn đến phương thức chạy của luồng được gọi và luồng tiếp tục thực thi cho đến khi phương thức chạy (run method) đó kết thúc.



Đồ họa này giúp thể hiện hành vi này. Đầu tiên, một ứng dụng đang chạy sẽ tạo một lệnh mới để tạo một đối tượng luồng mới. Khi cuộc gọi này kết thúc, ứng dụng sẽ tiếp tục và đôi khi sau đó gọi phương thức bắt đầu (start method) của luồng. Và cuộc gọi này quay trở lại ứng dụng, nhưng cũng khiến mã trong phương thức chạy (run method) của luồng cũng chạy. Và khi chương trình tiếp tục, bây giờ có hai luồng thực thi. Và tất nhiên, bạn có thể làm điều này nhiều lần, tạo và thực hiện nhiều luồng như bạn muốn.

Vì vậy, chúng ta hãy xem xét một ứng dụng trong đó phân luồng sẽ hữu ích. Ứng dụng đầu tiên mà chúng ta sẽ thảo luận trong bài học này được gọi là **ThreadingNoThreading**. Và như bạn sẽ thấy trong một giây, ứng dụng sẽ hiển thị giao diện người dùng đơn giản với hai nút. Nút đầu tiên được gắn nhãn LoadIcon. Khi người dùng nhấp vào nút này, ứng dụng sẽ mở và đọc một tệp, chứa bitmap. Và khi đã xong, nó hiển thị bitmap vừa được tải trên màn hình. Và ý tưởng ở đây là thao tác này cần một lượng thời gian đáng chú ý.

Nút thứ hai được dán nhãn Nút khác. Khi người dùng nhấp vào nút này, một thông báo bánh mì (toast message) sẽ bật lên hiển thị một số văn bản. Và ý tưởng ở đây là nếu bạn nhìn thấy văn bản, thì bạn biết rằng nút đang hoạt động. Bây giờ nếu bạn không thể nhấp vào nút hoặc bạn không nhìn thấy văn bản, thì có gì đó không đúng. Cụ thể, người dùng sẽ có thể nhấp vào một trong hai nút bất kỳ lúc nào và hệ thống sẽ hoạt động. Vì vậy, hãy chạy một phiên bản của ứng dụng này không sử dụng bất kỳ luồng nào. Bây giờ bạn nghĩ điều gì sẽ xảy ra?

Tôi có thể nhấn cả hai nút bất cứ khi nào tôi muốn không? Hãy xem nào. Ở đây, tôi sẽ khởi động ứng dụng ThreadingNoThreading. Như bạn có thể thấy, có hai nút mà chúng ta đã nói đến.Trước tiên tôi sẽ nhấn nút Khác.Và như bạn thấy, tôi có thể nhấp vào nó và thông báo đã hứa xuất hiện trên màn hình. Bây giờ tôi sẽ làm hai việc. Trước tiên tôi sẽ nhấn nút Load Icon, nút này sẽ bắt đầu hoạt động tốn thời gian đọc trong bitmap từ một tệp, sau đó hiển thị nó. Và ngay sau khi tôi nhấn nút Tải biểu tượng.Tôi sẽ nhấn nút khác một lần nữa. Bây giờ, tôi sẽ nhấn nút Tải biểu tượng và bây giờ tôi sẽ nhấn Nút khác, được rồi. Vì vậy, những gì đang xảy ra ở đây? Nút khác dường như bị kẹt. Tại sao vậy?

Chà, câu trả lời là khi tôi cố nhấn nút Other, Android vẫn tải biểu tượng từ phía sau khi tôi nhấn nút Tải biểu tượng. Và hoạt động đầu tiên đó đã ngăn cản hoạt động thứ hai diễn ra.

Được rồi, vì vậy một giải pháp có vẻ rõ ràng nhưng cuối cùng không chính xác cho vấn đề này, sẽ là đi đến người nghe được gắn vào nút Biểu tượng tải. Và chỉ cần tạo một luồng mới tải bitmap, và sau đó hiển thị nó.

Vì vậy, tôi đã thực hiện phương pháp đó trong một ứng dụng có tên **Threadingsimple**. Chúng ta hãy xem ứng dụng đó và nói về lý do tại sao nó không thực sự hoạt động.

Vì vậy, đây là mã cho ứng dụng Threadingsimple. Đây là trình nghe nút, cho nút Tải biểu tượng. Nó gọi phương thức loadIcon, được liệt kê ngay bên dưới. Mã này tạo ra một luồng mới, mất một lúc để tải bitmap. Và sau đó cố gắng thiết lập bitmap trên chế độ xem hình ảnh là một phần của bố cục. Vì vậy, hãy chạy mã này. Bây giờ, tôi sẽ khởi động ứng dụng. Và bây giờ tôi sẽ nhấn nút Biểu tượng tải, và bây giờ tôi sẽ nhấn Nút khác. Trước hết, tôi nhấn Nút khác và thấy nó phản hồi. Vì vậy, tải biểu tượng không xuất hiện để chặn nhấn Nút khác. Vì vậy, đó là tốt, chúng tôi đã đạt được một số tiến bộ. Tuy nhiên, bạn có thể thấy rằng chúng ta có một vấn đề lớn hơn bây giờ. Chúng tôi đã bị lỗi ứng dụng. Nếu chúng tôi điều tra đầu ra logcat, chúng tôi thấy rằng có một thông báo cho chúng tôi biết rằng chỉ có chủ đề ban đầu tạo ra hệ thống phân cấp chế độ xem mới có thể chạm vào chế độ xem của nó. Vì vậy, Android đơn giản sẽ không cho phép các luồng bắt đầu lộn xộn với các chế độ xem được tạo bởi các luồng khác.

Vì vậy, điều đó có nghĩa là trong khi luồng mới mà chúng ta tạo để tải bitmap có thể thực hiện công việc đó, thì thực tế nó không thể thực hiện bước cuối cùng và thêm bitmap kết quả vào màn hình. Vì vậy, chủ đề nào thực sự tạo ra hệ thống phân cấp xem của ứng dụng này? Vâng, tất cả các ứng dụng Android đều có một luồng chính, còn được gọi là **luồng UI**.

Các thành phần ứng dụng chạy trong cùng một quy trình, tất cả chúng đều làm theo mặc định, sử dụng cùng một luồng UI. Trong tất cả các phương thức vòng đời mà chúng ta đã nói về OnCreate, OnStart, v.v., tất cả chúng đều được xử lý trong luồng UI. Và ngoài ra, bộ công cụ UI không phải là chủ đề an toàn.

Bây giờ tất cả những gì có nghĩa là nếu bạn chặn luồng UI bằng một số thao tác chạy dài, thì bạn sẽ ngăn ứng dụng của bạn phản ứng với những thứ khác mà người dùng đang làm. Trong thực tế, chúng tôi thấy rằng trong luồng không có ứng dụng luồng. Vì vậy, các hoạt động chạy dài cần phải được đặt trong luồng nền (background threads).

Tuy nhiên, trong cùng 1 lúc, chúng tôi không thể truy cập bộ công cụ UI từ một luồng không phải UI. Và đó là điều khiến chúng tôi gặp rắc rối với các ứng dụng đơn giản

Vì vậy, chúng ta cần thực hiện công việc trong một luồng nền, nhưng khi công việc đó được thực hiện, chúng ta cần thực hiện các cập nhật giao diện người dùng trong luồng UI.

Và Android trong thực tế cho chúng ta rất nhiều cách để làm điều đó. Cụ thể, Android cung cấp một số phương thức được đảm bảo để chạy trên luồng UI.

Hai trong số các phương thức đó là post class the activity classes chạy chạy trên luồng UI. Cả hai phương thức này đều có tham số runnable. Ví dụ, runnable này có thể chứa mã cập nhật màn hình trong các ví dụ gần đây của chúng tôi. Vì vậy, nếu chúng ta đang sử dụng các phương thức này, chúng ta sẽ tải bitmap trong một luồng nền, và khi hoạt động đó hoàn thành, chúng ta sẽ sử dụng một trong các phương thức này để thực thi một chạy được, sau đó thiết lập bitmap trên màn hình. Chúng ta hãy xem điều đó trong hành động.

Và tôi sẽ bắt đầu ứng dụng xem bài viết luồng. Và có hai nút. Tôi sẽ nhấn nút Tải biểu tượng. Và ngay sau đó, tôi sẽ nhấn nút Khác. Và tôi hy vọng sẽ thấy rằng hoạt động Nút khác không bị chặn bởi hoạt động Biểu tượng tải. Và, tôi hy vọng sẽ thấy rằng biểu tượng thực sự tải mà không làm hỏng ứng dụng. Vì vậy, ở đây đi. Bây giờ, tôi sẽ nhấn nút Tải biểu tượng và bây giờ tôi sẽ nhấn Nút khác. Có văn bản từ Nút khác và cuối cùng, có bitmap. Chúng ta hãy xem mã nguồn là tốt. Vì vậy, đây là ứng dụng của tôi mở trong IDE. Tôi sẽ mở hoạt động chính cho ứng dụng này và tôi sẽ chuyển thẳng sang phương thức loadIcon, được gọi khi người dùng nhấn nút Tải biểu tượng. Như trước đây, mã này tạo ra một luồng mới, sau đó tải bitmap. Nhưng sau khi tải bitmap, bạn thấy rằng bây giờ chúng ta có lệnh gọi view.post truyền vào một runnable có mã thực sự gọi phương thức bitmap hình ảnh được đặt để đặt bitmap vừa được tải trên chế độ xem hình ảnh đó.

* **Threads, AsyncTasks & Handlers - Part 2**

Lớp hỗ trợ luồng (threading support class) tiếp theo mà chúng ta sẽ thảo luận là lớp **Asynctask**. Lớp này cung cấp một khung chung để quản lý các tác vụ, như trong các ví dụ trước của chúng tôi, liên quan đến công việc phải được chia sẻ giữa luồng nền (background thread) và luồng UI. Luồng công việc chung mà bạn theo dõi khi bạn đang sử dụng AsyncTask là công việc đó được phân chia giữa một luồng nền và luồng U-I. The **background thread** thực hiện các hoạt động chạy dài và có thể tùy ý báo cáo tiến độ của nó. Mặt khác, **luồng UI**, chịu trách nhiệm thiết lập ban đầu cho hoạt động chạy dài. Nó chịu trách nhiệm xuất bản thông tin tiến trình trung gian. Và nó chịu trách nhiệm hoàn thành thao tác sau khi luồng nền đã hoàn thành công việc.

**ASYNCTAS**K là một lớp chung. Phải mất ba tham số loại, thông số, tiến độ và kết quả (It takes three types parameters, params, progress, and result). Params là loại tham số được nhập vào A, với AsyncTask. Tiến trình là loại báo cáo tiến trình gián đoạn và kết quả là loại kết quả được AsyncTask tính toán.

**Quy trình làm việc của AsyncTask** diễn ra như sau. Đầu tiên, phương thức onPreExecute được chạy trong UI Thread trước khi phương thức doInBackground bắt đầu. onPreExecute thường sẽ thiết lập hoạt động chạy dài. Sau đó, phương thức doInBackground thực hiện phần lớn thao tác trong luồng nền. Và phương thức này lấy một danh sách các tham số đầu vào khác nhau và nó trả về kết quả của loại Kết quả. Bây giờ, trong khi doInBackground đang chạy, nó có thể tùy chọn gọi phương thức publishProgress  đi ​​qua một danh sách các giá trị thay đổi, có lẽ cung cấp một số dấu hiệu cho thấy tiến trình hoạt động lâu dài. Nếu luồng nền thực hiện gọi để xuất bản tiến trình, thì một số cuộc gọi đến onProgressUpdate thường sẽ được thực hiện trong luồng UI miễn là luồng nền vẫn đang chạy. Và cuối cùng onPostExecute sẽ được gọi trong luồng UI với kết quả re, với kết quả được trả về bởi luồng nền làm tham số của nó.

Hãy xem xét một phiên bản của ứng dụng tải biểu tượng của chúng tôi được triển khai với tác vụ Async. Ở đây tôi sẽ khởi động ứng dụng Thread AsyncTask. Nó trông giống như các ví dụ trước, nhưng tôi đã thêm một yếu tố UI mới, một thanh tiến trình, thể hiện số lượng tải bitmap đã được thực hiện. Vì vậy, ở đây tôi sẽ nhấn nút biểu tượng tải. Sau đó, bạn có thể thấy rằng một thanh tiến trình nhỏ đã xuất hiện và đang dần được điền vào. Bây giờ tôi sẽ nhấn nút khác và chúng tôi thấy văn bản quen thuộc bật lên. Và cuối cùng, có bitmap. Hãy xem mã nguồn của ứng dụng này. >> Ở đây, tôi sẽ khởi động ứng dụng tác vụ Async. >> Bây giờ tôi sẽ mở tệp hoạt động chính.

Và ở đây, tôi đang hiển thị trình nghe nút cho nút biểu tượng tải. Mã này tạo một phiên bản mới của LoadIconTask. Và sau đó gọi thực thi trên nó, chuyển vào id tài nguyên biểu tượng như một tham số. Vì vậy, hãy xem xét lớp LoadIconTask chi tiết hơn.

LoadIconTask là một AsyncTask và chu vi loại của nó là số nguyên, cho tham số, số nguyên cho tiến trình và bitmap cho kết quả. Phương pháp đầu tiên chúng ta sẽ xem xét là onPreExecute. Phương pháp này được thực thi trong luồng UI và mục đích của nó là làm cho thanh tiến trình hiển thị trên màn hình. Phương thức tiếp theo là doInBackground. Phương thức này nhận một số nguyên dưới dạng tham số. Và số nguyên này là tài nguyên cho bitmap được truyền vào phương thức thực thi LoadiconTasks. doInBackground thực hiện công việc tải bitmap. Khi thực hiện điều này, nó định kỳ gọi tiến trình xuất bản chuyển qua một số nguyên biểu thị phần trăm tải được thực hiện cho đến nay. Và một lần nữa, ví dụ này là một chút giả định, với hy vọng giữ cho mọi thứ đơn giản. Ví dụ có thể thực tế hơn một chút, nếu chúng ta đang tải xuống một hình ảnh từ Internet hoặc nếu chúng ta đang chờ kết quả từ một truy vấn cơ sở dữ liệu. Nhưng hy vọng điều này vẫn mang đến cho bạn một hương vị tốt về cách thức hoạt động của Asynct task.

Phương pháp tiếp theo là **onProgressUpdate**. Phương thức này chạy trong luồng (thread) UI, nhận số nguyên được chuyển vào PublishProgress và sau đó đặt thanh tiến trình để phản ánh tỷ lệ phần trăm công việc được thực hiện. Và cuối cùng, phương thức cuối cùng là onPostExecute. Phương thức này, một lần nữa, chạy trong luồng UI và nó nhận được justLoadedBitmap làm tham số. Đầu tiên, nó làm cho thanh tiến trình trở nên vô hình vì điều đó không còn cần thiết nữa, và sau đó nó sẽ thiết lập bitmap được tải trên chế độ xem hình ảnh.

**Mở rộng: *AsyncTask*** *là một đối tượng thích hợp dùng để giao tiếp giữa một Thread với UI Thread, nó rất dễ dùng. Nó cho phép bạn thực hiện công việc ở background và sau đó cập nhật giao diện ở UI Thread. AsyncTask chính là sự tổ hợp của một Thread và một Handler.*

*AsyncTask là một đối tượng có 3 tham số Generic Type truyền vào là Params, Progress, Result và 4 phương thức onPreExecute(), doInBackground(), onProgressUpdate() và onPostExecute(). 3 tham số đều phải là kiểu Object, nếu không có thì để là Void. Trên hình vẽ bạn có thể thấy chỉ có doInBackground() chạy ở background, còn 3 phương thức còn lại chạy trên UI Thread. Khi bạn cho một lớp kế thừa từ AsyncTask bắt buộc bản phải override phương thức doInBackground().*

*Params là tham số có thể được nhận từ execute(), có thể là một mảng các tham số con. và Params sẽ là input của doInBackground(), Process là input của onProgressUpdate(), đầu ra này nhận từ doInBackground() thông qua phương thức publishProgress(). onProgressUpdate() có thể cập nhật giao diện lúc runtime. Result là đầu ra của doInBackground() và chính là kết quả trả về ở onPostExecute(). AsyncTask chạy trên Worker Thread còn Handler chạy trên Main Thread (hay Thread sinh ra nó).*

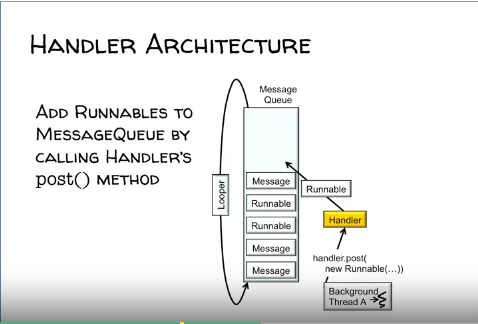
Điều cuối cùng tôi muốn nói trong bài học này là lớp xử lý (**HANDLER CLASS**).

Giống như AsyncTask, lớp trình xử lý (handler class) được thiết kế để hỗ trợ xử lý công việc giữa hai luồng. Lớp xử lý linh hoạt hơn, tuy nhiên, ở chỗ nó sẽ hoạt động cho bất kỳ hai luồng nào, không chỉ cho một luồng nền và luồng UI. Một xử lý được liên kết với một luồng cụ thể.

Một luồng có thể chuyển công việc sang luồng khác bằng cách gửi tin nhắn hoặc bằng cách đăng runnables lên một trình xử lý có liên quan đến luồng khác. Vì vậy, trước tiên, hãy thảo luận về tin nhắn runnables (**messages runnables**) và sau đó chúng ta sẽ đi vào kiến ​​trúc của lớp xử lý (class handler).

Bây giờ, bạn đã biết về runnables. Bạn sử dụng chúng khi người gửi biết chính xác các bước công việc mà nó muốn thực hiện, nhưng nó muốn công việc đó được thực hiện trên luồng của trình xử lý. Mặt khác, một tin nhắn (A message) là một lớp có thể chứa dữ liệu như mã tin nhắn, một đối tượng dữ liệu tùy ý và một số giá trị nguyên. Và bạn sử dụng tin nhắn khi luồng người gửi muốn chỉ ra một thao tác nên được thực hiện trong luồng khác. Nhưng nó để lại việc thực hiện thao tác đó cho chính người xử lý. Vì vậy, bây giờ hãy nói về cách xử lý sử dụng các tin nhắn và runnables này. Mỗi luồng Android được liên kết với một messageQueue và một looper. MessageQueue là một cấu trúc dữ liệu chứa các thông điệp và runnables.

Looper lấy những tin nhắn này và chạy ra khỏi messageQueue và gửi chúng đi khi thích hợp.



Bây giờ, đồ họa này mô tả một chủ đề A đã tạo ra một runnable và đã sử dụng một đối tượng xử lý để đăng có thể chạy đến chủ đề của trình xử lý. Khi luồng A thực hiện điều này, runnable được đặt trên messageQueue của luồng được liên kết với trình xử lý. Bây giờ, một cái gì đó khá tương tự xảy ra với các tin nhắn. Và đồ họa này mô tả một chủ đề B đã tạo ra một thông điệp và đã sử dụng phương thức của trình xử lý, phương thức sendMessage của trình xử lý để gửi thông điệp đó đến chủ đề của trình xử lý. Khi Thread B thực hiện điều này, thông báo sẽ được đặt trên MessageQueue được liên kết với trình xử lý đó. Bây giờ, trong khi tất cả những điều này đang diễn ra, đối tượng looper đang ngồi đó chỉ chờ công việc xuất hiện trên messageQueue.

Và khi công việc đó xuất hiện, looper phản ứng theo một trong hai cách tùy thuộc vào loại công việc vừa đến. Bây giờ, nếu công việc đó là một tin nhắn, looper sẽ xử lý tin nhắn bằng cách gọi phương thức handMessage của trình xử lý và tự gửi tin nhắn. Nếu thay vào đó, công việc đó là một runnable, thì looper sẽ xử lý nó bằng cách gọi phương thức chạy của runnable đó.

Bây giờ, đây là một số phương pháp mà bạn sử dụng khi đăng runnables lên một trình xử lý.

Chúng ta đã thấy phương pháp post. Có một số phương pháp khác cho phép bạn lên lịch làm việc để thực hiện vào các thời điểm khác nhau. Chẳng hạn, bạn có thể sử dụng phương thức postAtTime để thêm một runnable vào messageQueue. Nhưng để chạy nó vào một thời điểm cụ thể, đó cũng là một phương thức postDelayed. Và điều đó cho phép bạn thêm một runnable vào messageQueue, nhưng để chạy nó sau một độ trễ được chỉ định. Nếu bạn muốn gửi tin nhắn, trước tiên bạn cần tạo tin nhắn. Một cách để làm điều đó là sử dụng phương thức getMessage của trình xử lý, cung cấp cho bạn một thông báo với trình xử lý đã được đặt. Bạn cũng có thể sử dụng phương thức thu được các lớp thông báo và khi bạn có tin nhắn, bạn sẽ muốn đặt dữ liệu hoặc tin nhắn. Và có một số biến thể để làm điều này vì vậy xin vui lòng kiểm tra tài liệu. Như với runnables, có một số phương thức mà bạn có thể sử dụng để gửi tin nhắn. Có phương thức sendMessage mà chúng ta vừa nói đến. Ngoài ra còn có một phiên bản cho phép bạn đặt tin nhắn ở phía trước của messageQueue để nó thực thi càng sớm càng tốt.

Có một phương thức sendMessageAtTime để xếp hàng tin nhắn theo thời gian đã chỉ định.

Ngoài ra còn có một phương thức sendMessageDelayed xếp hàng tin nhắn tại thời điểm hiện tại cộng với độ trễ được chỉ định. Hãy xem mã nguồn cho các phiên bản của ví dụ đang chạy của chúng tôi đã được triển khai bằng trình xử lý.

Ví dụ: Đây là ứng dụng xử lý luồng. Bây giờ tôi sẽ mở ra hoạt động chính của ứng dụng này. Và trước tiên, bạn thấy rằng mã này đang tạo một trình xử lý mới. Trình xử lý này sẽ được tạo bởi luồng UI chính. Vì vậy, các runnables mà trình xử lý này nhận được sẽ được thực thi trong luồng UI. Bây giờ, ở đây tôi đang hiển thị cho bạn trình nghe nút cho nút loadIcon. Khi người dùng nhấn nút loadIcon, mã này sẽ tạo và bắt đầu một luồng mới có phương thức chạy được xác định bởi tác vụ loadIcon có thể chạy được. Hãy nhìn vào lớp học đó. Bây giờ phương thức chạy này bắt đầu bằng cách đăng một runnable mới mà khi được thực thi, sẽ làm cho thanh tiến trình hiển thị. Nó tiếp tục bằng cách tải bitmap. Và trong khi nó đang làm điều đó, nó định kỳ xuất bản tiến trình của mình bằng cách đăng một chương trình có thể chạy khác gọi setProgressOnTheProgressBar. Sau đó, nó đăng một runnable đặt bitmap mới được tải lên màn hình và nó kết thúc bằng cách đăng một runnable cuối cùng làm cho thanh tiến trình vô hình. Chúng ta cũng hãy xem phiên bản thứ hai của ứng dụng này gửi tin nhắn thay vì đăng các bản chạy.

Đây là ứng dụng ThreadingHandlerMessages. Bây giờ tôi sẽ mở ra hoạt động chính của ứng dụng này. Và trước tiên, bạn thấy rằng mã này đang tạo một trình xử lý mới. Và một lần nữa, trình xử lý này sẽ được tạo bởi luồng UI chính. Công việc mà trình xử lý này thực hiện sẽ được thực hiện trong luồng UI. Như bạn có thể thấy, trình xử lý này có một handMessageMethod trong đó nó thực hiện các loại công việc khác nhau. Phương pháp này bắt đầu bằng cách kiểm tra mã thông báo trong tin nhắn và sau đó thực hiện hành động thích hợp cho mã tin nhắn đó. Chẳng hạn, nếu mã được đặt\_Progress\_bar\_visibility, thì mã này sẽ đặt trạng thái hiển thị của thanh tiến trình. Nếu mã thay vì Progress\_update, thì mã này sẽ đặt trạng thái tiến trình trên thanh tiến trình. Nếu mã là set\_bitmap, thì mã sẽ đặt bitmap trên màn hình. Bây giờ, hãy đi đến trình nghe nút cho nút loadIcon. Và giống như trước đây, khi người dùng nhấn nút loadIcon, mã này sẽ tạo và bắt đầu một luồng mới có phương thức chạy được xác định bởi loadIconTask có thể chạy được. Và phương thức chạy này bắt đầu bằng cách lấy một thông báo với mã được đặt thành set\_ Progress\_bar\_visibility. Và với một đối số chỉ ra rằng thanh tiến trình sẽ được hiển thị. Sau đó nó sẽ gửi tin nhắn đó đến bộ xử lý, nó sẽ xử lý nó và hiển thị thanh tiến trình. Sau đó nó tiếp tục bằng cách tải bitmap. Và trong khi thực hiện điều đó, nó định kỳ xuất bản tiến trình bằng cách nhận và gửi tin nhắn với mã process\_update và với một đối số chỉ ra tỷ lệ phần trăm, tỷ lệ phần trăm công việc được thực hiện. Điều này sẽ dẫn đến trình xử lý gọi setProgressOnTheProgressBar. Sau đó, nó nhận được và gửi một thông báo để thiết lập bitmap mới được tải trên màn hình. Và cuối cùng, nó sẽ gửi một thông điệp cuối cùng để làm cho thanh tiến trình trở nên vô hình.

**Mở rộng: HANDLER là gì?**

*Là một đối Android cung cấp dùng để liên kết, trao đổi giữa các Thread với nhau, là trao đổi giữa Thread sinh ra Handler và các Thread khác. Thường là Main Thread (UI Thread) với các Worker Thread (Background Thread). Handler có nhiệm vụ gửi và thực thi các Message hoặc Runnable tới Message Queue của Thread sinh ra nó (Handler). Handler luôn được gắn kết với một Thread (Thread sinh ra nó) cũng với Message Queue (của Thread đó). Các Message và Runnable sẽ được thực thi khi đi ra khỏi Message Queue. Có 2 nhiệm vụ mà Handler thường làm đó là:*

* *Lên lịch thực thi các Message và Runnable ở các thời điểm trong tương lai.*
* *Sắp xếp một hành động được thực hiện trong một Thread khác.*

*Mesage Queue có thể xem là một hàng đợi, nó năm giữ một list các Message được gửi tới bởi đối tương Looper. Các Message không được add ngay lập tức vào Message Queue này mà phải thông qua Handler kết hợp với Looper. Looper là một đối tượng dùng để chạy vòng lặp trong Message trong Thread. Các bạn có thể hiểu Message Queue là một đường hầm, còn tập hợp các Task (Message or Runnable) là một con tàu, mỗi Task là một toa tàu. Còn Looper là người lái tàu, và người lái tàu này lại lái tàu qua khỏi hầm rồi lại vòng lại đi vào hầm tiếp, nó duy trùy công việc liên tục trong Main Thread. Nếu Handler sinh ra ở Main Thread thì đã có mặc định một Looper sinh ra, nhưng nếu nó sinh ra một Thread không phải là Main Thread thì phải tạo Handler trong cặp lệnh Looper.prepare() và Looper.loop().*

* **Networking - Part 1**

Một trong những đặc điểm xác định của các hệ thống cầm tay hiện đại, là chúng có thể giữ cho chúng tôi kết nối và kết nối mạng mà không cần buộc chúng ta vào một vị trí. Trong bài học này, chúng ta sẽ khám phá phần mềm và thực hành lập trình mà chúng ta sẽ cần để kết nối các ứng dụng của bạn với mạng. Tôi sẽ bắt đầu bài học này bằng cách thảo luận về mạng (networking) nói chung. Cuộc thảo luận đó sẽ tập trung vào việc kết nối các ứng dụng của bạn với internet bằng giao thức truyền siêu văn bản hoặc HTTP, cụ thể bằng cách sử dụng các yêu cầu HTTP.

Sau đó, tôi sẽ trình bày một số lớp mà Android cung cấp để hỗ trợ loại mạng này, và cuối cùng tôi sẽ thảo luận về cách các ứng dụng của bạn có thể xử lý dữ liệu họ nhận được để đáp ứng yêu cầu nhận HTTP này. Cụ thể tôi sẽ nói về hai ngôn ngữ định dạng dữ liệu phổ biến. Một, ngôn ngữ ký hiệu đối tượng JavaScript hoặc JSON, và hai, ngôn ngữ đánh dấu mở rộng hoặc XML. Và tôi sẽ nói về cách bạn phân tích hoặc hiểu ý nghĩa của các phản hồi HTTP này khi chúng được hình thành bằng một trong những ngôn ngữ này. Vì vậy, các thiết bị cầm tay sớm, đã cho chúng tôi di động. Bạn có thể di chuyển từ nơi này đến nơi khác và vẫn thực hiện tính toán hữu ích. Tuy nhiên, khả năng kết nối mạng của họ còn sơ khai theo tiêu chuẩn ngày nay. Hiện đang chuyển tiếp các thiết bị ngày nay kết hợp bộ xử lý điện với kết nối mạng nhanh qua WiFi và mạng di động. Do đó, các ứng dụng cầm tay thường muốn sử dụng, là những khả năng kết nối mạng này để truy cập và cung cấp dữ liệu và dịch vụ.

Bây giờ để giúp bạn làm điều này, Android bao gồm nhiều lớp hỗ trợ mạng bao gồm các lớp socket và URL trong các gói Java.net. Các lớp HttpRequest và HttpResponse trong các gói org.appache và các lớp URI AndroidHttpClient và AudioStream trong các gói android.net.

Để làm cho ứng dụng này hoạt động, đoạn mã cần tạo một yêu cầu http, gửi nó đến máy tính của máy chủ, truy xuất kết quả và sau đó hiển thị các kết quả đó. Android cung cấp một số lớp để trợ giúp việc này. Ba chúng ta sẽ nói về bây giờ là lớp socket, lớp HTTPURLConnection và AndroidHTTPClient.

Ví dụ:Tôi sẽ khởi chạy ứng dụng ổ cắm mạng. Như bạn có thể thấy, ứng dụng này ban đầu hiển thị một nút có nhãn Tải dữ liệu. Khi tôi nhấn nút đó, ứng dụng sẽ đưa ra yêu cầu HTTP GET cho máy chủ bên ngoài và máy chủ đó sẽ phản hồi với một số văn bản phức tạp, chứa dữ liệu động đất được yêu cầu. Được rồi, vậy bây giờ tôi sẽ nhấn nút Tải dữ liệu, và ở đó bạn có thể xem dữ liệu được yêu cầu. Hãy xem mã nguồn để xem những gì nó cần để có được dữ liệu đó. Bây giờ, ở đây tôi đã mở ứng dụng trong IDE. Bây giờ, tôi sẽ mở hoạt động chính cho ứng dụng này và ở đây tôi đang hiển thị trình nghe cho nút tải dữ liệu. Khi nhấn nút này, ứng dụng sẽ tạo và sau đó thực thi một tác vụ không đồng bộ có tên là HTTP get task. Hãy nhìn vào class đó.

Lớp nhiệm vụ HTTP get trước tiên khai báo một số biến được sử dụng trong việc tạo yêu cầu nhận HTTP. Khi phương thức thực thi được gọi trên tác vụ nhận HTTP. Việc làm trong phương thức nền được gọi. Phương pháp đó bắt đầu bằng cách tạo một ổ cắm mới sẽ được kết nối với máy tính chủ API.geoname.org trên cổng http tiêu chuẩn, cổng 80. Tiếp theo, mã nhận được luồng đầu ra của ổ cắm, sau đó viết lệnh http get, và điều này chuỗi sẽ được gửi đến máy tính chủ. Điều này diễn giải nó như một yêu cầu HTTP GET, và sau đó trả lời bằng cách gửi lại dữ liệu phản hồi, thích hợp và sau đó mã này tiếp tục bằng cách lấy luồng đầu vào của socket và bằng cách chuyển nó đến một phương thức gọi là luồng đọc. Phương thức luồng đọc cuối cùng đọc dữ liệu phản hồi từ luồng đầu vào của socket và sau đó trả về phản hồi dưới dạng một chuỗi. Và chuỗi này sau đó được chuyển đến phương thức thực thi bài viết thực thi trên luồng chính và hiển thị phản hồi trong chế độ xem văn bản.

Chúng tôi quay lại ứng dụng. Bạn sẽ nhận thấy rằng văn bản phản hồi không chỉ bao gồm dữ liệu động đất mà còn cả các tiêu đề phản hồi HTTP. Bây giờ thông thường tôi sẽ không muốn hiển thị văn bản đó ở đây. Tôi thực sự chỉ muốn cho bạn thấy dữ liệu động đất. Vì vậy, trong trường hợp này tôi nên phân tích cú pháp phản hồi và rút ra dữ liệu mà tôi muốn. Ngoài ra, bạn có thể nhận thấy rằng tôi đã không viết bất kỳ mã xử lý lỗi nào mà bạn thực sự cần để làm cho ứng dụng này mạnh mẽ. Và những điểm này nắm bắt khá tốt sự đánh đổi của việc sử dụng ổ cắm. Họ ở cấp độ rất thấp. Bạn có thể viết bất cứ điều gì bạn muốn trên ổ cắm, nhưng bù lại, bạn phải xử lý tất cả nhiều chi tiết thực hiện các yêu cầu http, tất cả xử lý lỗi và xử lý các phản hồi http.

Việc triển khai tiếp theo chúng ta sẽ xem xét sử dụng lớp kết nối URL http (**http URL connection class**). Lớp này cung cấp giao diện cấp cao hơn xử lý nhiều chi tiết mạng hơn lớp ổ cắm, nhưng, như chúng ta sẽ thấy trong giây lát, nó cũng có API kém linh hoạt hơn so với tùy chọn cuối cùng của chúng tôi, ứng dụng khách HTTP, Android lớp học.

Bây giờ đã nói rằng, tôi cũng chỉ ra rằng nhóm Android sẽ không tích cực làm việc trên Máy khách HTTP Android nữa và nó sẽ nỗ lực cải thiện lớp này trong tương lai. Vì vậy, hãy xem ứng dụng ví dụ được triển khai lần này với lớp kết nối URL HTTP.

Bây giờ tôi sẽ khởi chạy ứng dụng URL mạng. Như trước đây, ứng dụng này ban đầu hiển thị một nút có nhãn dữ liệu tải và như trước đây, khi tôi nhấn vào nút đó, ứng dụng sẽ đưa ra yêu cầu nhận HTTP. Đến một máy chủ bên ngoài. Và máy chủ đó sẽ phản hồi với một số văn bản phức tạp chứa dữ liệu động đất được yêu cầu. Đuợc. Vì vậy, bây giờ tôi sẽ nhấn nút Dữ liệu thấp.

Ở đó bạn có thể thấy dữ liệu được yêu cầu xuất hiện trong chế độ xem văn bản. Tuy nhiên, lưu ý rằng lần này các tiêu đề phản hồi HTTP đã bị loại bỏ. Hãy xem mã nguồn và xem nó hoạt động như thế nào. Bây giờ ở đây tôi đã mở ứng dụng trong IDE. Bây giờ tôi sẽ mở hoạt động chính cho ứng dụng này và ở đây tôi đang hiển thị trình nghe cho nút tải dữ liệu. Như trước đây, khi nhấn nút này, ứng dụng sẽ tạo và sau đó thực thi một tác vụ không chính thức có tên là httpGetTask. Hãy nhìn vào lớp học đó.

Khi phương thức thực thi được gọi trên HttpGetTask, phương thức doInBackground được gọi. Phương pháp đó bắt đầu bằng cách tạo một đối tượng URL mới và truyền chuỗi URL cho dịch vụ mong muốn làm tham số. Đoạn mã sau đó gọi phương thức kết nối mở trên đối tượng url. Trả về kết nối url http. Đối tượng này sau đó được lưu trữ trong một biến gọi là kết nối url http. Mã tiếp tục bằng cách lấy luồng đầu vào của kết nối url http và bỏ qua nó đến phương thức luồng đọc. Và như trước đây, phương thức luồng đọc đọc dữ liệu phản hồi từ luồng đầu vào của ổ cắm và sau đó trả về phản hồi dưới dạng một chuỗi. Tuy nhiên, lần này, kết nối URL http sẽ loại bỏ các tiêu đề phản hồi http và xử lý việc kiểm tra lỗi cho bạn. Bây giờ chuỗi này sau đó được chuyển đến phương thức thực thi onPost. Hiển thị phản hồi trong chế độ xem văn bản.

Lớp thứ ba là **AndroidHTTPClient**. Lớp này là một triển khai của DefaultHttpClient của Dự án Apache. Và nó cho phép rất nhiều tùy biến. Cụ thể, lớp phá vỡ một giao dịch HTTP thành một đối tượng yêu cầu và thành một đối tượng phản hồi. Vì vậy, bạn có thể tạo các lớp con tùy chỉnh việc xử lý các yêu cầu và phản hồi của chúng. Bây giờ đến lúc này bạn đã biết ứng dụng trông như thế nào rồi, hãy nhảy thẳng vào mã và xem xét việc thực hiện.

Bây giờ ở đây tôi đã có ứng dụng NetworkAndroidHttpClient. Mở trong IDE. Bây giờ, tôi sẽ mở hoạt động chính cho ứng dụng này và hãy đi thẳng đến lớp nhiệm vụ http, lớp đó bắt đầu bằng cách tạo một đối tượng máy khách http http mới, bằng cách gọi các phương thức cá thể mới của lớp. Bây giờ, khi phương thức doInBackground được gọi, mã sẽ tạo một đối tượng HttpGet, chuyển vào chuỗi URL cho yêu cầu đó. Tiếp theo, nó tạo ra một đối tượng FeedbackHandler. Đối tượng này có trách nhiệm xử lý các phản ứng. Đối với yêu cầu http, trong trường hợp này, trình xử lý phản hồi thuộc loại, Trình xử lý phản hồi cơ bản, sẽ trả về phần thân phản hồi. Bây giờ chúng ta sẽ thấy một trình xử lý phản hồi phức tạp hơn sau này trong bài học này. Và cuối cùng, yêu cầu và trình xử lý phản hồi được chuyển vào phương thức thực thi sẽ gửi yêu cầu, nhận phản hồi, chuyển nó qua trình xử lý phản hồi và kết quả của tất cả điều này sau đó được chuyển sang thực thi bài. Hiển thị phản hồi trong chế độ xem văn bản.

**1. SOCKET là gì?** *Socket là một phương pháp để thiết lập kết nối truyền thông giữa một chương trình yêu cầu dịch vụ ( client) và một chương trình cung cấp dịch vụ (server) trên mạng LAN, WAN hay Internet. Một socket trên máy yêu cầu dịch vụ có địa chỉ mạng được cấp sẵn để gọi một socket trên máy cung cấp dịch vụ. Một khi socket đó được thiết lập phù hợp, hai bên có thể trao đổi dịch vụ và dữ liệu.*

* **Networking - Part 2**

Cho đến nay, ứng dụng ví dụ của chúng tôi đã yêu cầu dữ liệu, và sau đó chỉ hiển thị dữ liệu đó trong chế độ xem văn bản. Nhưng như bạn đã thấy, dữ liệu đó có định dạng phức tạp. Đó thực sự là dành cho xử lý máy. Và không phải để hiển thị cho con người.

Trên thực tế, đây là một cách ngày càng phổ biến để vận chuyển dữ liệu. Trên Internet, và nhiều dịch vụ web hiện cung cấp dữ liệu ở các định dạng như vậy. Cụ thể, hai định dạng mà chúng ta sẽ nói đến bây giờ là Ký hiệu đối tượng JavaScript, JSON và Ngôn ngữ đánh dấu mở rộng, XML. Chúng ta hãy nói về từng thứ một, từng cái một.

Định dạng đầu tiên chúng ta sẽ nói đến là JSON ký hiệu đối tượng JavaScript. Định dạng này được dự định có trọng lượng nhẹ và giống với các cấu trúc dữ liệu được tìm thấy trong các ngôn ngữ lập trình truyền thống. Dữ liệu JSON được đóng gói theo hai loại cấu trúc dữ liệu. Một: Bản đồ, về cơ bản là tập hợp các cặp khóa và giá trị, Và hai: Danh sách được sắp xếp. Nếu bạn muốn biết thêm chi tiết về JSON, vui lòng xem trang web này.

Bây giờ, hãy quay trở lại ứng dụng ví dụ của chúng tôi. Như bạn nhớ rằng ứng dụng đã yêu cầu một dịch vụ web cho một số dữ liệu về động đất. Chà, phản hồi đã trở lại thực sự được định dạng trong JSON. Vì vậy, đây là dữ liệu. Chúng ta hãy phá vỡ nó. Đầu tiên, dữ liệu bao gồm một đối tượng JSON và đối tượng đó là bản đồ và bản đồ đó có một cặp giá trị chính. Khóa được gọi là động đất và giá trị là một danh sách theo thứ tự. Bây giờ danh sách đó. Có một vài đối tượng bên trong nó, và mỗi đối tượng đó là một bản đồ. Và mỗi bản đồ đó lại chứa các cặp giá trị chính.

Ví dụ. Có một khóa gọi là eqid và giá trị của nó là earthquake id. Ngoài ra còn có một chìa khóa gọi là lng. Và giá trị của nó là kinh độ mà trận động đất là trung tâm. Và còn có một loạt các khóa khác, và tất cả các giá trị này cung cấp dữ liệu cho một trận động đất. Chúng ta hãy xem một ứng dụng ví dụ lấy dữ liệu này từ internet và sau đó xử lý nó để tạo ra một màn hình dễ đọc hơn cho con người. Và bây giờ tôi sẽ khởi chạy the networking Android http client JSON application. Như trước đây, ứng dụng này ban đầu hiển thị một nút có nhãn dữ liệu tải và như trước đây, khi tôi nhấn nút đó, ứng dụng sẽ đưa ra yêu cầu nhận HTTP đến máy chủ bên ngoài và máy chủ đó sẽ phản hồi với một số văn bản phức tạp chứa dữ liệu động đất được yêu cầu.

Lần này, dữ liệu sẽ được tóm tắt và trình bày trong chế độ xem danh sách.Được rồi, vì vậy bây giờ tôi sẽ nhấn nút Tải dữ liệu và ở đó bạn có thể thấy dữ liệu được yêu cầu được tóm tắt và trình bày trong chế độ xem danh sách. Hãy xem mã nguồn để xem cách thức hoạt động của nó.

Ở đây tôi đã mở ứng dụng trong IDE và bây giờ tôi sẽ mở tệp tải xuống và hiển thị. Và tôi sẽ bỏ qua ngay đến lớp HttpGetTask. Bây giờ ở đây phương thức do trong nền tương tự như những gì chúng ta đã thấy trước đây nhưng lần này nó sử dụng lớp Trình xử lý phản hồi JSON để xử lý phản hồi. Hãy cuộn xuống và xem lớp đó.

Phương thức chính trong lớp này là phương thức trả lời xử lý (**handle response**). Phương thức này bắt đầu bằng cách chuyển phản hồi thô thông qua trình xử lý phản hồi cơ bản, nó chỉ trả về phần thân phản hồi mà không cần các tiêu đề phản hồi HTTP. Tiếp theo mã sử dụng mã thông báo JSON để phân tích phản hồi JSON thành đối tượng Java, và sau đó để trả về đối tượng cấp cao nhất, trong trường hợp này là bản đồ. Tiếp theo, mã trích xuất giá trị liên quan đến khóa của trận động đất. Và trong trường hợp này, đó là một danh sách theo thứ tự.

Tiếp theo, mã lặp lại trong danh sách của trận động đất. Và với mỗi yếu tố cho danh sách đó, nó nhận được dữ liệu liên quan đến một trận động đất duy nhất và dữ liệu này được lưu trữ trong bản đồ.

Tiếp theo, mã tóm tắt các phần khác nhau của dữ liệu động đất chuyển đổi chúng thành một chuỗi và thêm chuỗi đó vào danh sách được gọi là kết quả.

Và cuối cùng kết quả được đưa trở lại phương thức gọi (calling).

Bây giờ sau khi phương thức làm và nền kết thúc. Phương thức thực thi bài được gọi, và điều này cung cấp kết quả là tham số của nó và bạn có thể thấy. Phương pháp này tạo và thiết lập một bộ điều hợp danh sách. Đối với chế độ xem danh sách đi qua trong danh sách kết quả đã được tính toán lại trong phản hồi xử lý. Định dạng dữ liệu thứ hai mà chúng ta sẽ nói đến là Ngôn ngữ đánh dấu mở rộng, XML. XML là một ngôn ngữ đánh dấu để tạo tài liệu XML. Tài liệu XML chứa đánh dấu và nội dung. Đánh dấu mã hóa một mô tả về bố cục lưu trữ và cấu trúc logic của tài liệu. Nội dung là mọi thứ khác. Và đặc biệt nội dung đó bao gồm dữ liệu phản hồi. Khi XML được sử dụng để mã hóa phản hồi HTTP. Bây giờ, nếu bạn muốn biết thêm chi tiết về XML, vui lòng xem trang web này. Bây giờ, hãy quay trở lại ứng dụng ví dụ của chúng tôi. Nếu chúng tôi cung cấp một URL hơi khác nhau.

Sau đó, dịch vụ web đó sẽ trả về dữ liệu động đất ở định dạng XML chứ không phải ở định dạng JSON. Vì vậy, đây là dữ liệu. Chúng ta hãy phá vỡ nó. Đầu tiên có một yếu tố gọi là geoname. Được lồng bên trong phần tử đó là một loạt các phần tử động đất và mỗi phần tử động đất chứa các phần tử khác cung cấp dữ liệu cho một trận động đất. Vì vậy, có thể so sánh với những gì chúng ta đã thấy với định dạng JSON có một yếu tố gọi là eqid và giá trị của nó là một id động đất. Ngoài ra còn có phần tử LNG và giá trị của nó là kinh độ mà trận động đất được tập trung và giống như trong ví dụ JSON cũng có một loạt các yếu tố khác. Vì vậy, nếu ứng dụng của chúng tôi nhận được dữ liệu XML từ internet, nó sẽ cần phân tích tài liệu XML để nó có thể tạo danh sách bạn hiển thị mà chúng ta đã thấy trước đó. Android cung cấp một số loại trình phân tích cú pháp XML khác nhau. Bao gồm các trình phân tích cú pháp DOM. DOM là viết tắt của Mô hình Đối tượng Tài liệu. Và các trình phân tích cú pháp DOM đọc toàn bộ tài liệu XML và chuyển đổi nó thành cấu trúc Mô hình Đối tượng Tài liệu. Một cây, và sau đó ứng dụng xử lý nó trên cấu trúc cây này. Bây giờ loại trình phân tích cú pháp này đòi hỏi nhiều bộ nhớ hơn, nhưng không cho phép ứng dụng thực hiện những việc như xử lý nhiều tài liệu.

Trình phân tích cú pháp SAX. Các trình phân tích cú pháp này đọc tài liệu XML dưới dạng luồng. Và khi họ gặp các thực thể tài liệu khác nhau, họ gọi lại vào ứng dụng. Mà sau đó có thể xử lý thông tin của tài liệu. Bây giờ, các trình phân tích cú pháp này sử dụng ít bộ nhớ hơn trình phân tích cú pháp DOM. Nhưng họ bị giới hạn trong việc xử lý trong một lần chuyển tài liệu.

Kéo trình phân tích cú pháp, như trình phân tích cú pháp sax, đọc tài liệu dưới dạng luồng nhưng trình phân tích cú pháp kéo sử dụng cách tiếp cận dựa trên trình vòng lặp trong đó ứng dụng thay vì trình phân tích cú pháp quyết định khi nào sẽ di chuyển quá trình phân tích cú pháp. Và giống như trình phân tích cú pháp sax, trình phân tích cú pháp kéo cũng sử dụng ít bộ nhớ hơn trình phân tích cú pháp dom, nhưng trình phân tích cú pháp kéo cũng cung cấp cho ứng dụng quyền kiểm soát lớn hơn quá trình phân tích cú pháp so với trình phân tích cú pháp sax.

Bây giờ ứng dụng ví dụ trông giống hệt như ứng dụng mà chúng tôi đã hiển thị để phân tích các phản hồi json. Vì vậy, tôi muốn hiển thị ứng dụng này cho bạn bây giờ. Thay vào đó, hãy xem mã nguồn của ứng dụng này. [SOUND] Bây giờ, ở đây tôi đã mở ứng dụng trong IDE và bây giờ tôi sẽ mở tệp tải xuống và hiển thị. Và, một lần nữa, tôi sẽ chuyển ngay sang lớp HTTPGetTask. Phương thức doInBackground tương tự như những gì chúng ta đã thấy trước đây. Nhưng bây giờ, nó sử dụng lớp xử lý phản hồi XML để xử lý phản hồi. Vì vậy, hãy mở lớp đó và xem nó hoạt động như thế nào.

Bây giờ cũng như trước đây, phương thức chính trong lớp này là phương thức trả lời xử lý. Và phương thức này bắt đầu bằng cách tạo đối tượng PullParser.

Tiếp theo, mã đặt đầu vào của trình phân tích cú pháp thành tài liệu XML được trả về phần thân của phản hồi HTTP.

Và sau đó, mã nhận được sự kiện trình phân tích cú pháp đầu tiên và sau đó bắt đầu lặp lại trên tài liệu XML.

Bây giờ bên trong vòng lặp while có 3 sự kiện mà mã này kiểm tra.

Xem phần bắt đầu của thẻ XML, xem phần cuối của thẻ XML và xem nội dung phần tử và khi sự kiện là sự kiện bắt đầu. Phương thức startTag được gọi, truyền vào phần tử đang được bắt đầu.

Phương pháp này xác định nếu phần tử dữ liệu này là một phần tử cần được lưu và nếu có, nó ghi lại bằng cách đặt các biến nhất định.

Khi sự kiện là sự kiện kết thúc, phương thức endTag được gọi. Đi qua trong các yếu tố đang được kết thúc. Và một lần nữa, phương pháp này xác định xem phần tử dữ liệu này có phải là phần tử đang được lưu hay không và nếu có thì nó ghi lại rằng nếu đây là phần cuối của thẻ động đất, thì chuỗi kết quả cho phần dữ liệu động đất này được thêm vào danh sách kết quả .

Khi sự kiện là một sự kiện văn bản, phương thức văn bản được gọi. Truyền nội dung của phần tử.

Phương thức này xác định thẻ nào hiện đang được phân tích cú pháp và sau đó lưu nội dung để sử dụng sau này và như trước đây, sau khi phương thức nền trong kết thúc. Phương thức thực hiện trên bài được gọi. Với kết quả được thông qua như là một tham số.

Và phương thức đó tạo và thiết lập một bộ điều hợp danh sách cho chế độ xem danh sách, chuyển vào danh sách kết quả, được tính lại trong phản hồi xử lý.

## User Notifications, BroadcastReceivers, and Alarms

* [**User Notifications - Part 1**](https://www.coursera.org/lecture/android-programming-2/user-notifications-part-1-YLjc2)

Trong bài học này, chúng ta sẽ nói về thông báo (**notifications**). Thông báo về cơ bản là tin nhắn, đôi khi khá phong phú. Các ứng dụng đó hiển thị cho người dùng bên ngoài giao diện người dùng thông thường của ứng dụng.

Ví dụ: giả sử bạn có một ứng dụng có thể tải xuống sách điện tử từ internet. Trong trường hợp đó, có lẽ bạn muốn cho phép người dùng chọn một cuốn sách để tải xuống. Nhưng sau đó tiếp tục sử dụng ứng dụng hoặc thậm chí thoát tất cả cùng nhau, trong khi cuốn sách đang tải xuống. Và, nếu vậy, có lẽ bạn cũng muốn cho người dùng biết khi quá trình tải xuống kết thúc. Và nếu nó đã thành công. Để làm điều đó, bạn cần làm một vài điều, bạn cần tìm ra khi quá trình tải xuống kết thúc. Và sau đó bạn cần hiển thị một số loại tin nhắn để cung cấp cho người dùng thông tin đó.

Trong bài học này, tôi sẽ nói về hai loại thông báo người dùng khác nhau mà Android hỗ trợ. Loại đầu tiên là tin nhắn toast (toast messages). Loại thứ hai là khu vực thông báo hoặc thông báo trên thanh trạng thái (notification area or status bar notifications).

Các thông báo này có thể nhìn thấy và có thể được truy cập trong khu vực thông báo do hệ thống kiểm soát ở đầu thiết bị. Và một lần nữa, bạn đã thấy các ví dụ về điều này.

Ví dụ, trong bài học cơ bản về ứng dụng, tôi đã chỉ ra cách một thiết bị nhận được tin nhắn MMS đến. Và, bạn có thể nhớ rằng, khi tin nhắn MMS thực sự đến, một biểu tượng xuất hiện ở phía trên bên trái của điện thoại. Và người dùng sau đó có thể kéo xuống khu vực thông báo để hiển thị thêm thông tin về tin nhắn MMS đến đó.

Nhưng một tin nhắn bật lên vài giây sau khi bạn nhấn nút gửi, ví dụ, nó cho bạn biết, tin nhắn đã được gửi thành công. Hoặc một biểu tượng xuất hiện ở đầu thiết bị của bạn, cho biết tin nhắn MMS đã đến. Mặc dù bạn hiện đang đọc một trang web với ứng dụng trình duyệt. Những thứ đó là thông báo của người dùng và đó là những thứ chúng ta sẽ nói về ngày hôm nay.

Bây giờ, một số thông báo người dùng Android tồn tại để cung cấp phản hồi cho người dùng về những điều họ đang làm. Và một trong những cơ chế như vậy là các tin nhắn chúc mừng mà chúng ta đã nói đến.

Một cái khác là đối thoại (dialogues). Bây giờ chúng tôi đã xem qua các hộp thoại khá chi tiết trong bài học về các lớp giao diện người dùng, vì vậy tôi sẽ không nói về các hộp thoại trong bài học này.

Một loại thông báo người dùng khác là thông báo khu vực thông báo (the notification area notifications). Và chúng thường được sử dụng để thông báo các sự kiện không thể đoán trước cho người dùng và thực hiện theo cách không làm gián đoạn. Bất cứ điều gì khác họ sẽ làm tại thời điểm đó. Và một lần nữa, hãy nghĩ về tình huống tin nhắn SMS đến. Đặt một biểu tượng trong vùng thông báo cho bạn biết rằng một tin nhắn đã đến. Nhưng nó không làm điều đó theo một cách khó hiểu hoặc khinh miệt.

Thông điệp là có, và bạn có thể tự do đối phó với nó. Bất cứ khi nào nó thuận tiện cho bạn.

Tin nhắn toast (**Toast messages**), như bạn đã thấy, là những tin nhắn tạm thời bật lên trên màn hình. Chẳng hạn, để cho người dùng biết rằng một thao tác đã hoàn thành thành công. Toasts tự động mờ dần trong và mờ dần ra khỏi tầm nhìn. Và công việc của họ là cung cấp thông tin cho người dùng, nhưng họ không có intent thu thập thông tin sẽ được gửi lại cho ứng dụng lưu trữ.

Bạn có thể tạo tin nhắn toast bằng cách sử dụng các lớp toast làm phương thức văn bản. Phương thức đó có một vài tham số, bao gồm cả văn bản mà bạn muốn hiển thị. Và lượng thời gian mà bạn muốn văn bản đó hiển thị. Và sau khi bạn đã tạo toast, bạn có thể hiển thị nó bằng cách gọi phương thức hiển thị trên toast bạn vừa tạo. Vì vậy, hãy xem một ứng dụng ví dụ sử dụng tin nhắn toast.

Ví dụ: Ở đây tôi đang bắt đầu ứng dụng Toast thông báo. Ứng dụng này hiển thị một nút duy nhất có nhãn Show Toast, đó chính xác là những gì nó sẽ làm khi tôi nhấn nó. Vì vậy, tôi sẽ nhấn nút đó ngay bây giờ. Và ở gần phía dưới màn hình, bạn thấy một cửa sổ bật lên nhỏ nói rằng, bạn đang toast. Bây giờ nếu chúng ta mở mã nguồn ứng dụng trong IDE, chúng ta sẽ thấy cách thức này được thực hiện. Ở đây tôi nhấn mạnh trình nghe nút cho nút toast hiển thị. Bên trong, bạn thấy một cuộc gọi đến phương thức makeText, truyền vào văn bản. Và chuyển qua Toast không đổi.LENGTH\_LONG, trong đó kết thúc làm cho văn bản hiển thị trong khoảng ba giây rưỡi.

Và ở cuối dòng đó. Có một cuộc gọi đến phương thức hiển thị thực sự hiển thị toast. Bây giờ nếu bạn muốn tin nhắn toast fancier, bạn cũng có thể tạo chế độ xem tùy chỉnh cho toast của bạn. Ví dụ: bạn có thể tạo bố cục tùy chỉnh trong XML, thổi phồng nó. Và đính kèm khung nhìn bị thổi phồng vào tin nhắn toast với cuộc gọi đến phương thức setView. Hãy xem một ví dụ.

Và bây giờ tôi sẽ bắt đầu một toast thông báo với ứng dụng Custom View.

Giống như ví dụ toast đơn giản. Ứng dụng này hiển thị một nút có nhãn hiển thị Toast. Và tôi có thể nhấn nút đó để hiển thị thông báo Toast. Tuy nhiên, trong trường hợp này, khi tôi nhấn nút, bạn sẽ thấy chế độ xem Toast tùy chỉnh thay vì cửa sổ bật lên màu xám đơn giản mà chúng ta đã thấy trước đây. Vì vậy, ở đây chúng tôi đi. Và có tin nhắn toast tùy chỉnh của bạn với một văn bản, toast của bạn và một nhãn cầu thêm để giúp lái xe về nhà tin nhắn.

Vì vậy, hãy mở ra hoạt động chính cho ứng dụng này và xem cách chúng tôi tạo ra thông điệp toast.Bây giờ bạn có thể thấy khi nhấn nút hiển thị toast, mã đầu tiên tạo ra một đối tượng toast mới. Hai dòng tiếp theo đặt vị trí của toast trên màn hình. Và chỉ định khoảng thời gian mà nó sẽ hiển thị. Sau đó, có một lệnh gọi để đặt chế độ xem, trong đó tham số đầu tiên là kết quả của việc tăng bố cục XML trong tệp XML dấu gạch dưới tùy chỉnh gạch dưới. Hãy mở tập tin đó.

Bây giờ ở đây chúng ta có thể thấy rằng chế độ xem tùy chỉnh là tương đốiLayout chứa hai con. Đứa trẻ đầu tiên là chế độ xem hình ảnh giữ nhãn cầu. Đứa con thứ hai là một chế độ xem văn bản hiển thị văn bản, Bạn đang Toast! Và sau đó, trở lại hoạt động chính, có một dòng cuối cùng gọi phương thức hiển thị để hiển thị toast.

**Mở rộng:**  *Notification trong Android là một user interface element, nó được hiện thị ở bên ngoài user interface của ứng dụng để thông báo cho người dùng những thông tin cần thiết. Người dùng có thể xem được những thông báo này ở màn hình khóa ( đối với Android Lollipop ) hoặc từ Notification Drawer.*

* **User Notifications - Part 2**

Một loại thông báo người dùng khác mà chúng ta sẽ nói đến trong bài học này, là các thông báo xuất hiện trong khu vực thông báo (notification area). Và như tôi đã nói trước đó, loại thông báo người dùng này xuất hiện trong khu vực kiểm soát hệ thống ở đầu thiết bị, được gọi là khu vực thông báo hoặc thanh trạng thái. Các ứng dụng và chính hệ thống Android, có thể sử dụng khu vực này, để thông báo cho người dùng về sự xuất hiện của các sự kiện khác nhau. Vùng thông báo cũng cung cấp một thành phần giao diện người dùng, được gọi là ngăn kéo, và người dùng có thể kéo xuống khu vực thông báo, để mở ngăn kéo. Và một khi nó mở, bạn có thể thấy thông tin bổ sung về các thông báo khác nhau, đã được đặt trong khu vực thông báo.

Hãy xem ví dụ về cách các thông báo này được sử dụng trong Android. Và tôi sẽ mở ứng dụng điện thoại, và bây giờ tôi sẽ bắt đầu quay số điện thoại. Và bây giờ tôi sẽ nhấn nút quay số và điện thoại sẽ bắt đầu quay số và kết nối cuộc gọi điện thoại. Bây giờ hãy nói rằng, ở giữa cuộc gọi này, tôi cần lấy một số thông tin từ internet. Vì vậy, tôi nhấn nút Home để quay lại màn hình chính. Và từ đó, tôi sẽ mở ứng dụng trình duyệt. Và bây giờ hãy chú ý rằng ở phía trên bên trái thiết bị của tôi, một biểu tượng mới đã xuất hiện, đó là một thông báo. Khi tôi rút lui khỏi ứng dụng điện thoại, Android đã tạo đối tượng thông báo này và đặt nó vào vùng thông báo. Và thông báo này đóng vai trò như một lời nhắc nhở với tôi rằng cuộc gọi điện thoại vẫn được kết nối và nó cũng là cách để tôi nhanh chóng quay lại cuộc gọi đó. Bây giờ quay lại trình duyệt, tôi sẽ truy cập www.google.com và tôi sẽ thực hiện tìm kiếm. Giả sử tại thời điểm này tôi được trang bị thông tin mà tôi cần và tôi muốn quay lại cuộc gọi điện thoại của mình.

Vì vậy, bây giờ, tôi sẽ kéo xuống khu vực thông báo để mở ngăn thông báo. Và một khi nó mở, tôi có thể thấy một chế độ xem, hiển thị cho tôi một số thông tin về cuộc gọi. Và nó cho phép tôi kết nối lại với cuộc gọi hoặc gác máy. Bây giờ trong trường hợp này, tôi muốn tiếp tục cuộc gọi. Vì vậy, tôi sẽ nhấp vào khu vực thông báo. Và điều đó mang đến ứng dụng điện thoại, đưa nó trở lại nền trước và cho phép tôi tiếp tục nói chuyện.

Khi bạn muốn gửi thông báo, có một số điều mà bạn cần xem xét. Đầu tiên, có thông báo cơ bản, phải có ít nhất văn bản cho tiêu đề và nội dung của nó, và cũng là một biểu tượng nhỏ. Khi thông báo được gửi, cuối cùng nó sẽ đến khu vực thông báo, nơi biểu tượng nhỏ sẽ được hiển thị.

Ngoài ra, bạn có thể đặt văn bản đánh dấu của thông báo, trong trường hợp đó văn bản đó cũng sẽ được hiển thị khi thông báo xuất hiện lần đầu tiên trong khu vực thông báo.Cuối cùng, nếu người dùng mở ngăn thông báo, sẽ có chế độ xem mà họ sẽ thấy. Bây giờ, theo mặc định, điều này bao gồm tiêu đề, chi tiết hoặc văn bản nội dung, biểu tượng nhỏ và dấu thời gian.

Bạn cũng cần xác định bất kỳ hành động nào sẽ xảy ra nếu người dùng nhấp vào hình vẽ thông báo, chế độ xem ngăn kéo. Bây giờ khi bạn đã tạo thông báo, bạn có thể muốn gửi nó. Cập nhật nó, hủy bỏ nó hoặc những thứ như thế. Các hoạt động này được quản lý bởi một dịch vụ hệ thống Android có tên là Trình quản lý thông báo.

Vì vậy, hãy xem hai ứng dụng gửi thông báo và sau đó chúng tôi sẽ xem xét một số mã nguồn để xem tất cả những điều này được thực hiện như thế nào.

Vì vậy, ở đây tôi sẽ mở ứng dụng thanh trạng thái thông báo.

Và giao diện người dùng này hiển thị một nút duy nhất có nhãn Thông báo.

Khi tôi nhấp vào nút này, một thông báo sẽ được tạo và gửi và cuối cùng nó sẽ xuất hiện trong thông báo ở trên cùng của điện thoại. Vì vậy, hãy để tôi nhấp vào nút bây giờ. Và thông báo hiện đang đến, và con gà trống gáy bạn nghe thấy là âm thanh mà tôi đính kèm với thông báo đó. Bạn cũng có thể thấy văn bản đánh dấu cuộn lên trong khu vực thông báo. Bây giờ sau khi văn bản đánh dấu kết thúc cuộn, nó sẽ biến mất và biểu tượng thông báo sẽ vẫn hiển thị. Vì vậy, bây giờ hãy mở ngăn thông báo. Bạn có thể thấy chế độ xem ngăn kéo hiển thị biểu tượng, văn bản tiêu đề thông báo, văn bản chi tiết thông báo, hiển thị số một trong ngoặc đơn, cho biết rằng nút thông báo đã được nhấn một lần. Và cuối cùng, cũng có một dấu ấn thời gian. Bây giờ, tại thời điểm này, tôi sẽ chỉ đóng, ngăn thông báo và quay lại và nhấn nút Thông báo thêm một lần nữa. Bây giờ, tôi sẽ mở ngăn Thông báo một lần nữa và bạn có thể thấy rằng văn bản chi tiết đã được cập nhật để cho thấy rằng đây là lần thứ hai nút Thông báo được nhấn. Tại thời điểm này, tôi sẽ nhấp vào thông báo và bạn có thể thấy rằng một hoạt động mới đã bắt đầu, in ra dòng chữ: Có intent. Và vấn đề ở đây tất nhiên là bạn có thể đính kèm một intent, vào chế độ xem ngăn kéo thông báo. Để đưa người dùng đến ứng dụng nên xử lý bất kỳ hành động tiếp theo nào mà thông báo được dự định kích động. Chúng ta hãy nhìn vào một thanh trạng thái thông báo ứng dụng thứ hai, với chế độ xem tùy chỉnh. Ứng dụng này làm điều tương tự như ví dụ cuối cùng. Tuy nhiên, nó hiển thị chế độ xem tùy chỉnh khi ngăn thông báo được mở. Bây giờ tôi sẽ khởi động ứng dụng và nhấn nút thông báo như trước đây. Thông báo được tạo và cuối cùng xuất hiện trong vùng thông báo. Tuy nhiên, khi tôi mở ngăn thông báo, tôi sẽ không thấy chế độ xem tiêu chuẩn, tôi sẽ thấy chế độ xem tùy chỉnh của riêng mình.

Hãy để tôi mở ngăn thông báo bây giờ. Và, có nhãn cầu của chúng tôi và các từ, bạn đã được thông báo, với số trong ngoặc đơn. Bây giờ, tôi sẽ đóng ngăn kéo và tôi sẽ nhấn nút Thông báo lại. Tôi sẽ mở lại ngăn kéo và bạn có thể thấy mọi thứ đều giống nhau ngoại trừ số một đã trở thành số hai, để cho thấy rằng đây là thông báo thứ hai, cập nhật thông báo thứ nhất. Và cuối cùng, tôi sẽ nhấp vào giao diện ngăn kéo thông báo, khởi động một hoạt động mới hiển thị các từ Got The Intent.

Được rồi, vì vậy hãy xem mã cho ứng dụng thứ hai đó, thanh trạng thái thông báo với chế độ xem tùy chỉnh. Và ở đây tôi quay lại IDE và tôi sẽ mở hoạt động chính của ứng dụng. Bắt đầu từ đầu, mã này tạo ID cho thông báo mà nó sẽ gửi. Và điều này cho phép trình quản lý thông báo cập nhật thông báo này, sau khi nó được gửi lần đầu tiên. Và tiếp theo, có một số biến chứa các thành phần văn bản của thông báo, bao gồm văn bản đánh dấu, tiêu đề và nội dung của nó. Sau đó, mã sẽ thiết lập một số thông tin được sử dụng để phát âm thanh và làm rung thiết bị khi có thông báo này. Tiếp theo, mã tạo chế độ xem tùy chỉnh sẽ được hiển thị trong ngăn thông báo. Bố cục cho chế độ xem đó nằm trong tệp custom\_notification.xml. Hãy xem xét điều đó.

Như bạn có thể thấy, khung nhìn này là một bố cục tuyến tính, với hai khung nhìn con.

Một cái là hình ảnh, cái hiển thị nhãn cầu. Và cái còn lại là chế độ xem văn bản và nó sẽ hiển thị văn bản, bạn đã được thông báo. Quay lại hoạt động chính, chúng ta hãy xem phương thức tạo. Và ở đây mã tạo ra một intent, được gọi là intent thông báo M. Mục đích này sẽ kích hoạt rõ ràng hoạt động phụ thông báo. Dòng mã tiếp theo, là thứ mà chúng ta chưa từng nói đến trước đây.

Dòng này tạo ra một cái gì đó gọi là intent chờ xử lý. Dựa trên mục đích thông báo M, đã được tạo trên dòng trước đó. Một mục đích đang chờ xử lý về cơ bản là một giấy phép cho phép một đoạn mã đứng trong một đoạn mã khác. Và điều tôi muốn nói là, giấy phép này cho phép đoạn mã thứ hai, kích hoạt mục đích cơ bản như thể nó là đoạn mã đầu tiên. Và đó là, nó làm điều đó với nó, các quyền và danh tính của đoạn mã đầu tiên đó.

Vì vậy, tiếp tục, người nghe của nút thông báo trước tiên cập nhật văn bản nội dung. Chúng cho biết số lần nhấn các nút. Sau đó, nó xây dựng thông báo thực tế, sử dụng lớp notify.builder. Một mã tạo ra đối tượng thông báo mới. Đối tượng, sau đó nó đặt văn bản đánh dấu,

đặt biểu tượng nhỏ và sau đó đặt tự động hủy thành đúng. Bây giờ, điều này nói với Android để hủy thông báo, là người dùng nhấp vào giao diện ngăn kéo.

Tiếp theo, nó đặt mục đích nội dung và đây là mục đích đang chờ xác định. hành động cần thực hiện, khi người dùng nhấp vào giao diện ngăn kéo. Tiếp theo, nó đặt âm thanh và kiểu rung sẽ phát khi có thông báo đến.

Và cuối cùng, nó đặt chế độ xem tùy chỉnh sẽ được hiển thị khi người dùng kéo ngăn kéo thông báo xuống. Bây giờ thông báo được thiết lập, mã sau đó nhận được tham chiếu đến trình quản lý thông báo bằng cách gọi dịch vụ hệ thống get, nhập ID của dịch vụ thông báo. Cuối cùng, các cuộc gọi mã thông báo trên trình quản lý thông báo, chuyển qua ID của thông báo.

Điều này cho phép các thông báo hiện có được cập nhật và nó cũng vượt qua kết quả của việc gọi xây dựng trên đối tượng thông báo .builder. Và phương thức xây dựng đó là những gì thực sự tạo ra, đối tượng thông báo thực tế.

* **BroadcastReceiver - Part 1**

Các lớp thu phát sóng (BroadcastReceiver) và luồng công việc cơ bản mà bạn theo dõi khi bạn sử dụng nó. Tiếp theo tôi sẽ thảo luận về cách các máy thu quảng bá (broadcast receivers) được đăng ký với hệ thống Android.

**BroadcastReceiver** là một lớp cơ sở cho các thành phần, với mục đích là chờ đợi một số sự kiện nhất định xảy ra, để nhận các sự kiện đó và sau đó phản ứng với chúng. Và cách thức hoạt động của tất cả, là các máy thu phát sóng riêng lẻ, đăng ký để nhận các sự kiện cụ thể mà chúng quan tâm.

Ví dụ: có một máy thu phát sóng trong Android có nhiệm vụ nghe các tin nhắn MMS gửi đi. Bây giờ, ở một nơi khác sau đó, một thành phần thực hiện điều gì đó mà nó muốn thông báo cho người nhận phát sóng. Vì vậy, nó tạo ra một intent đại diện cho sự kiện đó và sau đó nó phát sóng intent đó. Vì vậy, có các thành phần khác trong Android chuẩn bị tin nhắn MMS, để sau đó chúng có thể được gửi qua mạng. Khi tin nhắn MMS đã sẵn sàng thì sẽ có một thành phần tạo ra intent action\_send\_message và sau đó phát đi intent đó. Android sau đó thực hiện intent vừa phát sóng và đưa nó đến thành phần mà chúng ta đã nói trước đó, trước đó đã đăng ký để nhận các intent action\_send\_message. Và thành phần đó cuối cùng cũng nhận được intent bằng cách nhận một cuộc gọi đến phương thức onReceive của nó với intent đó là một trong các tham số cho phương thức. Bây giờ, đối với ví dụ MMS của chúng tôi, thành phần nhận được intent action\_send\_message.Và cuối cùng đặt ra một dịch vụ để gửi tin nhắn MMS.

2:48

Vì vậy, để tóm tắt lại, **quy trình làm việc cơ bản** là một, BroadcastReceivers đã đăng ký để nhận các intent cụ thể. Hai, một số thành phần tạo ra một intent, và sau đó phát nó đến hệ thống. Ba, Android cung cấp intent đó cho các máy thu quảng bá đã đăng ký để nhận nó. Và bốn người nhận phát sóng sau đó nhận được một cuộc gọi đến phương thức onReceive của họ. Trong đó họ xử lý các sự kiện đến. Vì vậy, hãy nói về từng bước một.

Để đăng ký một máy thu phát sóng, các nhà phát triển có hai tùy chọn. Một, họ có thể đăng ký tĩnh máy thu quảng bá (broadcast receiver statically) bằng cách đưa một số thông tin vào tệp AndroidManifest.XML của ứng dụng mà máy thu quảng bá thuộc về. Hai, họ có thể đăng ký máy thu phát sóng một cách linh hoạt bằng cách gọi một số phương thức trong thời gian chạy.

3:55

Vì vậy, để đăng ký một máy thu phát sóng tĩnh (broadcast receiver statically). Bạn thêm một thẻ người nhận trong ứng dụng của mình tệp AndroidManifest.XML. Và sau đó trong thẻ người nhận đó, bạn đặt ít nhất một thẻ lọc intent. Và thông tin ở đó cho Android biết rằng khi một intent phù hợp với bộ lọc intent này được phát đi. Trình thu phát này muốn biết về nó và định dạng cho thẻ người nhận trông giống như thế này.

Bạn bắt đầu với từ khóa người nhận và sau đó bạn thêm một số thuộc tính sau. Và một số thuộc tính đó bao gồm android: enable, cho phép bạn bật hoặc tắt một bộ thu cụ thể. Một cái khác là android: xuất khẩu. Nếu được đặt thành true, thì máy thu này có thể nhận các chương trình phát sóng từ bên ngoài ứng dụng của nó, trong khi nếu nó được đặt thành false và máy thu chỉ có thể nhận các intent được phát bởi các thành phần khác trong ứng dụng này. Một thuộc tính khác là android: name cung cấp tên của lớp thực hiện bộ thu này. Ngoài ra còn có android: allow xác định chuỗi quyền. Người gửi intent phải có để người nhận này nhận được intent từ họ.

Bây giờ khi bạn đã tạo thẻ người nhận. Bạn chèn và thẻ lọc intent là một trong những đứa trẻ của nó. Và cũng giống như với các bộ lọc intent được sử dụng để bắt đầu các hoạt động, các thẻ bộ lọc intent này có thể chỉ định những thứ như một hành động, dữ liệu và danh mục. Bây giờ, nếu bạn đăng ký một người nhận tĩnh. Thông tin đó sẽ được đọc và xử lý khi hệ thống khởi động.

Hoặc khi gói ứng dụng được thêm vào, nếu điều đó xảy ra trong khi hệ thống đang chạy. Vì vậy, chúng ta hãy xem một ứng dụng đăng ký tĩnh một máy thu phát sóng duy nhất, để nhận được một mục đích tùy chỉnh mà tôi sẽ gọi intent toast chương trình. Và tôi sẽ khởi động ứng dụng đăng ký tĩnh phát sóng duy nhất. Ứng dụng này hiển thị một nút duy nhất có nhãn Broadcast Intent. Nhấn nút này sẽ khiến intent được phát, sau đó được chuyển đến và nhận bởi một chương trình phát nhận được, sau đó hiển thị thông báo toast.

Bây giờ, tôi sẽ nhấn nút. Và ở đó bạn thấy tin nhắn toast, chỉ ra rằng người nhận đã có intent.

Bây giờ, ở đây tôi đã mở ứng dụng trong IDE. Bây giờ, hãy mở hoạt động chính và mã này trước tiên xác định chuỗi hành động có intent sẽ được sử dụng để xác định intent này. Cuộn xuống, có một nút nghe, gọi phương thức context.send, Broadcast, phương thức. Vượt qua trong một intent, và một chuỗi quyền. Mục đích sẽ được kết hợp với các bộ lọc intent đã đăng ký.

Mặc dù chuỗi quyền chỉ ra rằng intent này, chỉ có thể được gửi đến các máy thu quảng bá có quyền này. Bạn cũng có thể đăng ký máy thu phát sóng theo chương trình vào thời gian chạy, ví dụ nếu bạn muốn máy thu phát sóng. Để chỉ trả lời các lần thử trong khi hoạt động của bạn ở phía trước. Vậy thì bạn có thể tự động đăng ký và hủy đăng ký chúng trong các hoạt động của mình, trên sơ yếu lý lịch và các phương thức tạm dừng chẳng hạn. Và để làm điều này, trước tiên bạn sẽ tạo một đối tượng bộ lọc intent, chỉ định intent mà bạn muốn đăng ký.

Tiếp theo bạn tạo bộ thu phát sóng. Và sau đó bạn đăng ký nó bằng cách gọi một phương thức nhận đăng ký. Và có những triển khai khác nhau của phương pháp này. Một là trong lớp quản lý đài truyền hình địa phương. Và cái này là dành cho phát sóng, chỉ dành cho ứng dụng này.

Và do đó, chúng không cần phải được phát sóng trên toàn hệ thống. Việc thực hiện khác là trong lớp bối cảnh. Và điều này một intent phát sóng toàn hệ thống. Vì vậy, những intent này có khả năng có thể được nhận bởi bất kỳ ứng dụng nào trên thiết bị của bạn. Cuối cùng, khi cần thiết, bạn có thể gọi một phương thức thu chưa đăng ký để hủy đăng ký các máy thu quảng bá này. Hãy xem mã nguồn cho ứng dụng đăng ký động phát sóng đơn.

Ứng dụng này trông giống hệt như ứng dụng trước.

Có một nút mục đích phát sóng và khi bạn nhấn nó, một intent được phát ra. Một máy thu quảng bá nhận được intent, và sau đó hiển thị một thông báo toast, tuy nhiên, bên trong, việc thực hiện nó có một chút khác biệt. Vì vậy, mã này bắt đầu bằng cách tạo một chuỗi hành động có intent và tiếp theo nó tạo ra một bộ lọc intent, đối tượng với chuỗi hành động vừa tạo. Sau đó, nó tạo ra một thể hiện máy thu quảng bá được gọi là máy thu. Và sau đó trong phương thức tạo, mã sẽ lấy một thể hiện của trình quản lý quảng bá cục bộ. Như tôi đã nói, được sử dụng để phát và nhận intent, chỉ trong ứng dụng này. Tiếp theo mã sử dụng trình quản lý phát sóng cục bộ, để đăng ký đối tượng người nhận cho mục đích hiển thị toast. Và sau đó nếu nhấn nút mục đích phát sóng, người nghe gọi trình quản lý phát sóng cục bộ. Phát sóng truyền phát trong intent. Khi nhấn nút, trình quản lý quảng bá cục bộ sẽ gửi intent đến đối tượng người nhận, hoạt động giống như trong ứng dụng cuối cùng tạo và hiển thị thông báo toast. Bây giờ ví dụ của tôi cho đến nay, chỉ đơn giản là phát một intent duy nhất đến một người nhận duy nhất. Nhưng Android hỗ trợ nhiều cách khác nhau để có thể phát sóng intent. Ví dụ, intent có thể được phát sóng bình thường hoặc theo thứ tự. Phát sóng bình thường được gửi đến các máy thu phát đã đăng ký theo thứ tự không xác định. Vì vậy, nếu bạn có hai máy thu phát sóng sẽ nhận được một mục đích duy nhất, có thể cả hai máy thu phát sóng đều có thể xử lý intent đó cùng một lúc.

Mặt khác, phát sóng theo thứ tự, cung cấp intent cho nhiều máy thu phát cùng một lúc theo thứ tự ưu tiên. Và chương trình phát sóng cũng có thể dính hoặc không dính. Một intent dính dính xung quanh sau khi phát sóng ban đầu của nó. Vì vậy, một máy thu phát sóng được đăng ký sau khi phát sóng intent ban đầu.

Có thể vẫn có thể nhận được nó, mặc dù nó không xuất hiện khi intent thực sự được phát sóng. Và tính năng này rất hữu ích, ví dụ, để ghi lại các thay đổi trạng thái hệ thống, chẳng hạn như thay đổi mức pin hoặc trạng thái sạc. Trong những trường hợp này, nó không quan trọng đối với người nhận khi trạng thái thay đổi.

Phải, họ chỉ muốn biết tình trạng hiện tại là gì. Phát sóng không dính ngược lại được loại bỏ sau khi phát sóng ban đầu của họ. Và những chương trình phát sóng này phù hợp hơn do đó, để báo cáo rằng một sự kiện nào đó đã xảy ra.

Trong những trường hợp này, nếu người nhận không đăng ký nhận phát sóng khi nó xảy ra, thì ở đó, họ sẽ không nhận được. Và cuối cùng, như chúng ta đã thấy trong các ví dụ, một số phương thức phát sóng cũng có một chuỗi quyền. Giới hạn phát sóng đối với những máy thu quảng bá có sự cho phép được chỉ định.

Bây giờ, nếu bạn thấy mình gặp rắc rối, hãy phát sóng đến đúng máy thu phát sóng. Có một vài điều mà bạn có thể làm là yêu cầu Android đăng nhập thêm thông tin về quy trình giải quyết intent của nó. Và bạn có thể làm điều này bằng cách đặt cờ FLAG\_DEBUG\_LOG\_RESOLNING theo intent mà bạn sẽ phát.

Một cách khác là sử dụng công cụ ADB để tìm hiểu những máy thu quảng bá nào đã được đăng ký. Và bạn có thể sử dụng lệnh dumpsys như thế này, để hiển thị các máy thu quảng bá được đăng ký động. Và bạn có thể sử dụng nó như thế này, để hiển thị những cái đã được đăng ký tĩnh.

**Mở rộng:**

*Android* ***BROADCAST RECEIVER*** *là một thành phần nơi bạn có thể đăng ký sự kiện của hệ thống hay ứng dụng. Bạn sẽ nhận được thông báo về các sự kiện đã đăng ký trước đó. Việc phát tin broadcast có thể bắt nguồn từ hệ thống hoặc từ các ứng dụng.*

* *Broadcast Receiver là một trong 4 component lớn trong Android, với mục đích là lắng nghe các sự kiện, trạng thái của hệ thống phát ra thông qua Intent nhờ đó mà các lập trình viên có thể xử lí được các sự kiện hệ thống ở bên trong ứng dụng của mình.*
* *Broadcast Receiver có thể hoạt động được cả khi ứng dụng bị tắt đi, nghĩa là ở background chính vì vậy nó thường được sử dụng với service*

*Ví dụ: một số broadcast từ hệ thống như thông báo pin yếu, bật tắt màn hình, kết nối hay ngắt kết thiết bị ngoại vi…*

*Với cấp độ ứng dụng có thể kể ra như khi bạn download một file nhạc. Ứng dụng nhạc này sẽ nhận được thông báo về việc download, sau khi kết thúc sẽ đưa bài hát này vào danh sách để bật. Để thực hiện được việc này, ứng dụng nhạc cần thiết phải đăng ký cho sự kiện. Không giống như Activity, Broadcast receiver trong android không hề có giao diện người dùng. Mặc dù nó có thể tạo thông báo trên thanh status bar. Để truyền thông tin, Broadcast receiver sử dụng Intent để đóng gói dữ liệu. Nếu bạn đã đọc bài viết của mình về Intent trong Android thì sẽ dễ dàng hiểu về Broadcast receiver trong Android.*

***Cách tạo Broadcast Receiver:***

*class MyReceiver:BroadcastReceiver() {*

*fun onReceive(context:Context, intent:Intent) {*

*// This method is called when this BroadcastReceiver receives an Intent broadcast.*

*Toast.makeText(context, "Action: " + intent.getAction(), Toast.LENGTH\_SHORT).show()*

*}*

*}*

*Chúng ta sẽ tạo một class là MyReciever và được kế thừa từ BroadcastReceiver. Do BroadcastReceiver là một abstract class nên chúng ta bắt buộc phải override hàm onReceiver(). Bất cứ khi nào có một sự kiện xảy ra, Android sẽ gọi hàm onReceiver(). Bạn để ý rằng, hàm onReceiver() có 2 tham số được truyền vào là: Intent và Content. Với Intent, bạn sẽ nhận được những thông tin cần thiết từ hệ thống. Còn Context sẽ giúp bạn có thể làm được một việc như start một Activity hay Service kiểu như context.startService(new Intent(this, TestService.class));*

***Hướng dẫn đăng kí để nhận thông báo từ Broadcast Receiver:***

*Chúng ta đã hoàn thành việc tạo một Broadcast Receiver trong Android, nhưng để nó có thể nhận được thông báo từ hệ thống hay ứng dụng khác thì nó cần phải đăng kí. Có 2 cách đăng kí:*

* *Đăng kí bằng cách khai báo trong Manifest file*

*<receiver*

*android:name="com.pycitup.pyc.MyReceiver"*

*android:enabled="true"*

*android:exported="true" >*

*<intent-filter>*

*<action android:name="com.pycitup.BroadcastReceiver" />*

*</intent-filter>*

*</receiver>*

*Chúng ta sử dụng thẻ <receiver>để đăng kí với một intent filter.*

*Về cơ bản có thể hiểu nôm na: Ứng dụng sử dụng intent filter để thông báo với hệ thống là ứng dụng của tôi đăng kí nhận thông báo. Nhưng không phải nhận tất cả mà chỉ những thông báo nào phù hợp thì mới nhận. Ở đây thì chỉ những thông báo nào có action là: com.pycitup.BroadcastReceiverthì mới nhận được*

* *Sử dụng code để đăng kí Broadcast Receiver*

*Chúng ta có thể thực hiện việc đăng kí bằng code Kotlin như sau:*

*var filter = IntentFilter("com.pycitup.BroadcastReceiver")*

*var myReceiver = MyReceiver()*

*fun registerReceiver(myReceiver, filter)*

*Tương tự như cách đăng kí bằng cách khai báo trong Manifest. Chúng ta cũng chỉ đăng kí nhận những thông báo có action là: com.pycitup.BroadcastReceiver. Cơ bản action này cũng chỉ là một String mà bạn muốn đặt là gì cũng được. Thông thường thì người ta sẽ đặt tên action giống với tên package ứng dụng. Có một sự khác biết lớn giữa 2 cách đăng kí là:Nếu đăng kí bằng cách sử dụng code thì khi activity bị stop thì việc lắng nghe broadcast cũng bị tạm dừng theo. Trong khi đăng kí qua manifest thì sẽ lắng nghe vĩnh viễn. Do vậy, nếu bạn đăng kí receiver thông qua code thì nên hủy đăng kí khi activity bị stop. Kiểu như sau*

*protected fun onPause() {*

*unregisterReceiver(mReceiver)*

*super.onPause()*

*}*

***Cách gửi Broadcast Event/Intent:***

*val intent = Intent()*

*intent.addFlags(Intent.FLAG\_INCLUDE\_STOPPED\_PACKAGES)*

*intent.setAction("com.pycitup.BroadcastReceiver")*

*intent.putExtra("Foo", "Bar")*

*sendBroadcast(intent)*

*Chúng ta tạo một intent, sau đó thiết lập action cho intent này. Ở đây là com.pycitup.BroadcastReceiver. Chính là action mà chúng ta đã sử dụng để đăng kí ở phần phía trên bài viết. Dữ liệu sẽ được đưa vào intent thông qua hàm putExtra()*

***Trong đó: FLAG\_INCLUDE\_STOPPED\_PACKAGES.*** *Cơ bản thì flag này cho phép ứng dụng vẫn nhận thông báo trong trường hợp ứng dụng được cài đặt nhưng chưa chạy lần nào hoặc bị tắt bởi công cụ quản lý hệ thống như task manager.*

***Cuối cùng gửi Broadcast bằng hàm sendBroadcast():***

*Nhìn chung, có 2 hình thức để gửi broadcasts:*

* *Gửi Broadcasts theo cách thông thường: Tức là chúng ta sử dụng hàm Context.sendBradcast() như bạn vừa thấy ở trên. Đây là cách gửi broadcast không đồng bộ.Và phía receiver cũng vậy. Các receiver được chạy theo một thứ tự không xác định, thường là cùng một lúc. Nhược điểm của cách gửi này là các reciever không thể sử dụng kết quả của nhau*
* *Gửi Broadcasts theo thứ tự: Để làm được điều này, chúng ta sử dụng hàm Context.sendOrderedBroadcast() và nó chỉ định rõ receiver được nhận tại một thời điểm. Thứ tự Receiver nhận kiểm soát bởi thuộc tính android:priority. Reciever có cùng mức độ ưu tiên sẽ được thực hiện theo thứ tự ngẫu nhiên. Khi mỗi reciever thực hiện, nó có thể chuyển kết quả đến receiver kế tiếp hoặc hủy bỏ toàn bộ chuỗi broadcasts để không cho reciever khác nhận được*

***Vấn đề bảo mật khi sử dụng Broadcast Receiver:***

*Nếu bạn không muốn sendBroadcast cho các đối tượng ở bên ngoài ứng dụng của bạn thì bạn có thể sử dụng LocalBroadcast vì LocalBroadcastManager hiệu quả hơn nhiều (không cần giao tiếp liên tục) và cho phép bạn tránh suy nghĩ về bất kỳ vấn đề bảo mật nào liên quan đến các ứng dụng khác có thể nhận hoặc gửi các broadcast của bạn.*

*Không phát thông tin nhạy cảm bằng cách sử dụng intent không tường minh. Thông tin có thể được đọc bởi bất kỳ ứng dụng nào đăng ký để nhận broadcast. Có ba cách để kiểm soát ai có thể nhận broadcast của bạn:*

*Bạn có thể set các quyền cho Broadcast*

*Set package mà bạn muốn gửi Broadcast đến quá setPackage(String)*

*Sử dụng Local Broadcast*

*Khi bạn đăng ký nhận Broadcast thì bất cứ ứng dụng nào mà bạn đăng ký đều có thể gửi những thông tin độc hại đến ứng dụng của bạn. Có các cách phòng tránh sau đây:*

*Set các quyền bạn cho là "Nguy hiểm"*

*Bạn có thể set android:exported=false*

*Local Broadcast*

*Chú ý cách đặt tên hành động lắng nghe sao cho tránh trùng lặp tên*

*Không nên chạy task nặng trong hàm onReceive() nếu bạn muốn chạy thì chúng ta nên sử dụng một trong hai cách sau:*

*Sử dụng goAsync() trong onReceive()*

*Sử dụng JobScheduler*

*Không nên start Activity từ Broadcast vì lúc này trải nghiệm người dùng đang chập chờn, nhất là khi có nhiều hơn một người nhận. Thay vào đó hãy xem xét hiện thị các thông báo.*

* **BroadcastReceiver - Part 2**

Bây giờ, cuối cùng nó sẽ được chuyển đến phù hợp máy thu phát sóng, thông qua một cuộc gọi đến phương thức onReceive của máy thu phát sóng. Và phương pháp này có hai tham số. Một, bối cảnh trong đó máy thu đang chạy và hai, ý định đã được phát sóng.

Vì vậy, có một số điều cần xem xét khi bạn viết mã xử lý các chương trình phát sóng đến. Đầu tiên, để truyền phát, Android có thể phải khởi động ứng dụng của người nhận quá trình, bởi vì nó có thể không thực sự chạy, khi ý định được phát sóng.

Và trong khi onreceive đang chạy, quá trình đó có mức độ ưu tiên cao. Vì vậy, phát sóng và ý định, đặc biệt là nếu nó đi đến nhiều ứng dụng, có thể là một hoạt động tương đối tốn kém.

Thứ hai, phương pháp onreceive chạy trong luồng chính của quá trình nướng (toasting process). Và giống như bất cứ điều gì bạn làm trên luồng chính, nó sẽ khá ngắn, và nó không nên chặn chủ đề chính. Đặc biệt, nếu bạn muốn phát sóng là tốn thời gian, sau đó bạn nên xem xét bắt đầu thay vì thực hiện công việc trong onReceive. Trên thực tế, đó chính xác là những gì ứng dụng MMS làm khi có tin nhắn MMS đến. Một máy thu phát sóng nhận được một phát sóng, thông báo rằng tin nhắn MMS đã đến.

Và trong phương thức onReceive của nó, khi được gọi, nó ngay lập tức chuyển thông điệp, đến một dịch vụ khác xử lý tất cả công việc tải xuống tin nhắn, lưu trữ nó trong cơ sở dữ liệu, v.v.

Một vấn đề cuối cùng cần xem xét, đó là một máy thu quảng bá chỉ được coi là hợp lệ miễn là OnReceive đang chạy. Trên thực tế, một khi OnReceive Returns android sẽ đôi khi chấm dứt máy thu phát sóng cơ bản. Đặc biệt, điều này có nghĩa là máy thu phát sóng không thể bắt đầu hoạt động cần gọi lại cho người nhận một cách không đồng bộ ngay sau đó. Và tất nhiên điều đó có ý nghĩa, bởi vì không có gì đảm bảo rằng máy thu quảng bá thậm chí sẽ tồn tại khi cuộc gọi lại đó xảy ra.

Bây giờ ví dụ về các cuộc gọi lại không đồng bộ này bắt đầu một cuộc đối thoại. Chúng tôi đang bắt đầu một hoạt động thông qua hoạt động bắt đầu hoặc phương pháp kết quả.

Ví dụ về các ứng dụng sử dụng máy thu quảng bá. Ở đây, tôi sẽ bắt đầu ứng dụng phát sóng

Nó có một nút có nhãn mục đích phát sóng. Mà khi ấn, sẽ được sử dụng để phát sóng để hiển thị. Lần này, tuy nhiên, Có ba máy thu phát sóng sẽ nhận được ý định này. Vì vậy, bây giờ tôi sẽ nhấn nút. Và bên dưới bạn thấy một toast message từ người nhận 1. Cái thứ hai từ người nhận 2, và một cái thứ ba từ người nhận 3. Bởi vì các ứng dụng phát sóng ghép được sử dụng gửi phát sóng. Các chương trình phát sóng được gửi đi bình thường. Và điều đó có nghĩa là lệnh đến và xử lý, không được xác định. Trong trường hợp này, việc phát sóng đã được xử lý theo một thứ tự cụ thể. Trong các trường hợp khác, hoặc trên một thiết bị khác, vàng nếu Android thay đổi trong tương lai, thứ tự đó có thể khác. Vì vậy, nếu bạn thực sự muốn các máy thu phát sóng nhận được phát sóng một đơn đặt hàng cụ thể hoặc nếu bạn muốn mỗi máy thu quảng bá có độc quyền truy cập vào ý định trong khi nó đang tồn tại, sau đó bạn sẽ muốn mở rộng bạn sẽ muốn chương trình phát sóng được đặt hàng. Và, bạn sẽ làm điều đó bằng cách sử dụng phương thức sendOrderedBroadcast. Và bạn có thể làm điều đó bằng cách sử dụng một số phương pháp, một số phương thức sau từ lớp ngữ cảnh. Đầu tiên, gửi một vấn đề đến các máy thu phát sóng có đã cho phép theo thứ tự ưu tiên.

Thứ hai làm điều tương tự, biện pháp bổ sung để kiểm soát tốt hơn. Hãy xem xét một số ứng dụng sử dụng các phương thức này để sendOrderedBroadcasts. Tại đây, tôi sẽ khởi động ứng dụng Phát sóng theo thứ tự hợp chất.

Ứng dụng này hiển thị một nút có nhãn Broadcast Intent và Khi bạn nhấn nút, ý định Toast được phát. Đối với ứng dụng này, có ba máy thu phát sóng đã đăng ký, một loại người nhận một, người nhận hai và người nhận ba. Và mỗi trong số đó, có một ưu tiên khác nhau để nhận được phát sóng. Người nhận hai có mức ưu tiên cao nhất, Người nhận một, mức cao nhất tiếp theo và Người nhận ba có mức ưu tiên thấp nhất. Vì vậy, chúng tôi hy vọng người nhận sẽ nhận được phát sóng theo thứ tự đó, người nhận hai, người nhận một, người nhận ba.

Tuy nhiên, tôi đã nhập một số mã. Tôi nhận được một số mã trong máy thu một, hủy bỏ việc phát sóng. Vì vậy, trong trường hợp này, chỉ có hai người nhận và sau đó một người nhận, sẽ nhận được phát sóng. Vì vậy, hãy xem điều đó trong hành động.

Bây giờ tôi sẽ nhấn phát sóng bạn có thể thấy một Toast Message nói rằng nó đã được nhận bởi người nhận hai. Và sau đó một người khác nói rằng nó đã được nhận bởi một người nhận. Và rõ ràng người nhận kém đang thoát khỏi cái lạnh. Hãy xem mã nguồn của ứng dụng này.

Bây giờ ở đây, tôi đã mở ứng dụng trong IDE. Và tôi sẽ bắt đầu bằng cách mở tệp tệp kê khai Android. Và như bạn có thể thấy, tôi đã được đăng ký tĩnh nhận2 với mức độ ưu tiên trong số 10 và Người nhận3 tôi đã đăng ký với mức độ ưu tiên là 1. Bây giờ trở lại trong hoạt động chính, mã tạo một thể hiện của Receiver1 ý định bánh mì nướng hiển thị, và sau đó đặt ưu tiên thành ba. Vì vậy, bất kỳ trường hợp nhận2 có ưu tiên 10. Ví dụ nhận1 này là ưu tiên 3.

Và sau đó người nhận của bạn ba trường hợp là ưu tiên 1. Bây giờ khi tin nhắn được nhấn, một người nghe gọi sendOrderedBroadcast, chuyển qua một chương trình bánh mì nướng mới có ý định. Thủ tục tố tụng này là một ví dụ đầu tiên sau đó bởi một ví dụ người nhận 1 đã được tạo trong tệp này. Hãy mở mã nhận 1. Bây giờ ở đây trong nhận, bạn thấy rằng mã kiểm tra xem liệu đây có phải là một chương trình phát sóng được đặt hàng hay không. Và nếu vậy, nó kêu gọi phát sóng, tiêu thụ phát sóng, và trong trường hợp này, ngăn không cho nó được gửi đến người nhận 3. Bây giờ chúng ta cũng xem xét việc sử dụng phát sóng hủy bỏ phức tạp hơn một chút. Trong ứng dụng này, chúng ta sẽ thấy rằng một toast message được hiển thị, cho thấy tất cả các máy thu phát sóng và điều đó chỉ ra thứ tự mà họ nhận được nó. Bây giờ ở đây, tôi sẽ bắt đầu ứng dụng. Bây giờ tôi sẽ nhấn nút Broadcast Intent và có toast message. Nó cho thấy rằng người nhận 2 đã nhận được ý định, sau đó nhận 1, sau đó nhận 3.

Hãy xem mã nguồn. Đây là ứng dụng, mở trong IDE. Bây giờ tôi sẽ mở hoạt động chính. Và đi đến người nghe cho nút mục đích phát sóng. Như bạn có thể thấy, mã này sử dụng hình thức thứ hai của phương thức phát sóng. Một trong những thông số thú vị của phương pháp này, là một máy thu quảng bá. Tôi sẽ gọi nó là người nhận kết quả. Điều đó sẽ được nhận sau khi các máy thu phát khác, đã có cơ hội nhận được nó. Và điều này rất hữu ích nếu các máy thu phát ban đầu tính toán một số giải quyết. Bởi vì kết quả đó sẽ có sẵn cho người nhận cuối cùng này. Và nó có thể nhận được kết quả đó bằng cách gọi getResultData. Như chúng ta thấy ở đây. Bây giờ chúng ta hãy xem một trong các lớp học phát sóng để xem họ như thế nào tính kết quả Người nhận này cuối cùng sẽ hiển thị trong Toast Message. Bây giờ đây là lớp Receiver1. Và như bạn có thể thấy, phương thức onReceive gọi getResultData. Để có được dữ liệu kết quả hiện tại. Và sau đó, nó xử lý chuỗi của chính nó vào cuối. Và sau đó lưu chuỗi mới bằng cách gọi setResultData.

Bây giờ, kết quả cuối cùng dữ liệu là một chuỗi duy nhất nhận được người nhận. Và thứ tự mà họ nhận được nó. Loại phát sóng cuối cùng mà tôi đang nói đến hôm nay, là các chương trình phát thanh dính. Như tôi đã nói trước đó, các chương trình phát sóng không dính chỉ ra rằng một cái gì đó có đã xảy ra tại một thời điểm cụ thể. Họ thông báo các sự kiện. Vì vậy, sự kiện đã kết thúc, nó đã kết thúc. Android xử lý sự kiện và di chuyển trên. Nếu một máy thu phát sóng không được đăng ký nói ngày hôm qua, sau đó không nên nói về nó bây giờ.

Bởi vì có thể sự kiện vừa xảy ra, và điều đó có thể dẫn đến tất cả các loại vấn đề thời gian và những khó khăn khác. Tuy nhiên, các sự kiện khác chỉ ra rằng những thay đổi có thể tồn tại theo thời gian.

Máy thu phát sóng muốn biết về trạng thái thiết bị hiện tại ví dụ, chúng tôi vẫn muốn thông tin đó ngay cả khi họ đã không được đăng ký khi nhà nước cụ thể thay đổi. Ví dụ: nếu mức pin xuống rất thấp, Android sẽ phát sóng thực tế này trong hệ thống.

Ứng dụng tốt sẽ muốn trở thành công dân tốt. Và không ở trạng thái thấp này. Vì vậy, trong trường hợp này, nếu một ứng dụng mới khởi động, và đăng ký để nghe về trạng thái pin, ứng dụng đó cần biết, ngay bây giờ, nếu thiết bị ở trạng thái pin yếu.

Và vì điều này, Android phát sóng pin Vì vậy, các ý định dính được lưu trữ bởi Android và các chương trình phát sóng dính của một mục đích nhất định ghi đè lên bất kỳ bộ nhớ cache nào từ các chương trình phát thanh dính trước đó về ý định phù hợp.

Khi một máy thu quảng bá được đăng ký động, bất kỳ ý định lưu trữ sẽ được phát sóng tới nó. Ngoài ra, một ý định phù hợp với bộ nhớ cache, sẽ được trả lại cho người gọi của người nhận đăng ký. Bây giờ, tương tự như những gì chúng ta đã thấy với các chương trình phát sóng không dính, phát sóng dính có thể được gửi bình thường, đó là, không có thứ tự xác định, hoặc tuần tự, theo thứ tự ưu tiên. Phát sóng dính bình thường có thể sử dụng phương pháp này, và đặt hàng phát sóng dính có thể sử dụng phương pháp này. Và cuối cùng, các ứng dụng muốn phát sóng Dính ý định phải có sự cho phép của BROADCAST\_STICKY. Hãy xem xét một ứng dụng ví dụ sử dụng Phát sóng dính. Bây giờ ở đây tôi sẽ chạy ứng dụng ý định dính trong trình giả lập. Tôi cũng đã mở một cửa sổ đầu cuối, và bắt đầu với phiên telnet với trình giả lập đó. Bây giờ, tôi sẽ khởi động ứng dụng, hiển thị mức pin hiện tại. Và hiển thị chuỗi, Đọc có thể cũ, và chuỗi cuối cùng này chỉ nói rằng mức đọc pin, Chú ý, không phải từ một chương trình phát sóng mới. Nếu bây giờ tôi đi đến cửa sổ thiết bị đầu cuối và thay đổi mức pin, một ý định mới sẽ được phát sóng, phản ánh mức pin được cập nhật. Vì vậy, bây giờ tôi sẽ nhập công suất 95 và nhấn Return. Như bạn có thể thấy, màn hình đã được cập nhật. Và văn bản hiển thị đó đã thay đổi để chỉ ra rằng người nhận có thể phân biệt giữa các chương trình phát sóng mới và một bộ nhớ cache.

Và bây giờ chúng ta sẽ mở hoạt động chính cho ứng dụng này. Nó gọi máy thu thanh ghi và truyền vào máy thu quảng bá như được định nghĩa trong dòng. Chúng ta hãy xem phương pháp onReceive của máy thu phát sóng đó. Đầu tiên, nó kiểm tra xem nó nhận được pin\_ đã thay đổi ý định hay chưa. Nếu vậy nó sẽ xác định xem chương trình phát sóng là mới hay bắt đầu phát sóng dính. Phương thức này trả về true, nếu đây là giá trị được lưu trữ. Và sau đó mã gửi văn bản để hiển thị.

**Mở rộng:**

* **Alarms**

Các ứng dụng ví dụ chúng tôi đã nghiên cứu, cho đến nay, đã đưa ra quyết định và sau đó hành động.

00:25

Nhưng nếu ứng dụng của bạn đưa ra quyết định nhưng muốn thực hiện một hành động một giờ kể từ bây giờ, ngày mai lúc nửa đêm, hoặc bao giờ 15 phút kể từ bây giờ?

00:35

Trong trường hợp này, ứng dụng của bạn sẽ cần tạo và đặt báo thức.

00:43

Trong bài học này, tôi sẽ bắt đầu bằng cách thảo luận về báo động là gì và cách chúng được sử dụng.

00:48

Tiếp theo, tôi sẽ thảo luận về Trình báo động Android và các API mà nó cung cấp để cài đặt và hủy báo động.

00:58

Sau đó, tôi sẽ thảo luận về các loại báo thức khác nhau mà Android hỗ trợ. Và cuối cùng, tôi sẽ trình bày và thảo luận về một ứng dụng sử dụng báo động.

01:10

Tóm lại, báo động là một cơ chế cho gửi ý định tại một số điểm hoặc điểm trong tương lai. Và điều này rất hữu ích, vì nó cho phép bạn áp dụng một số mã khác để thực thi, ngay cả khi ứng dụng đó không còn chạy.

01:32

Khi báo thức đã được tạo và đăng ký xem ngay cả khi thiết bị đi ngủ. Và nếu chúng ta sẽ nói về sau, thiết bị tắt trong khi thiết bị đang ngủ, thiết bị có thể được đánh thức để xử lý báo thức hoặc báo thức có thể được giữ cho đến lần tiếp theo thiết bị thức dậy.

01:56

Một báo động sẽ tiếp tục hoạt động cho đến khi thiết bị tắt. Khi tắt máy, tất cả các báo động đã đăng ký sẽ bị hủy. Để cung cấp cho bạn một số ví dụ về báo động, một ngày khác, tôi đã đào qua một số nguồn mã Android và cam trên một số ứng dụng sử dụng báo động. Chẳng hạn, ứng dụng nhắn tin MMS sử dụng báo động để bắt đầu dịch vụ có thể tìm thấy tin nhắn MMS chưa được gửi có thể thử, một lần nữa, để cung cấp cho họ. Ứng dụng cài đặt có thể giúp thiết bị có thể khám phá qua bluetooth. Và khi có, ứng dụng đó sẽ đặt và báo động khi báo thức kêu, ứng dụng sẽ khiến thiết bị không thể phát hiện được nữa.

02:45

Và ứng dụng điện thoại giữ một bộ đệm thông tin người dùng.

02:50

Ứng dụng này sử dụng báo động để định kỳ cập nhật bộ đệm đó.

02:56

Nếu bạn muốn sử dụng báo thức trong các ứng dụng của riêng bạn, bạn làm như vậy bằng cách tương tác với dịch vụ AlertManager.

03:04

Để có được một tài liệu tham khảo cho dịch vụ này getSystemService chuyển qua tên của dịch vụ, trong trường hợp này, ALARM\_SERVICE làm tham số. Khi bạn có một tham chiếu đến Trình quản lý báo thức, bạn có thể sử dụng một số phương pháp của nó để tạo và đặt báo thức. Chẳng hạn, bạn có thể sử dụng phương thức set để đặt một báo thức, và phương pháp này có ba tham số. Một loại, mà chúng ta sẽ thảo luận trong thời gian ngắn. Dài, đại diện cho thời gian mà báo thức sẽ tắt.

03:43

Và tùy thuộc vào việc đóng gói hoạt động xảy ra khi báo thức cuối cùng không hoạt động.

04:05

Bạn có thể sử dụng phương thức này, setRepeat, để đặt báo thức liên tục tắt trong các khoảng thời gian cụ thể. Phương pháp này có bốn tham số. Ba cái mà chúng ta đã thấy trong phương thức set và thêm một phần dài lượng thời gian giữa mỗi lần liên tiếp báo thức sẽ tắt.

04:30

Một phương thức AlertManager khác là setInexactRepeat. Phương pháp này tương tự như setRepeat, trong đó báo động nên đi định kỳ, nhưng phương pháp này cho Android linh hoạt hơn trong việc quyết định chính xác thời điểm bắn báo động. Chẳng hạn, Android có thể xử lý nhiều báo thức kiểu này bắn chúng cùng một lúc, rất nhiều như vậy.

05:00

Nếu bạn muốn có loại hành vi này thì thời gian của bạn phải là một các hằng số sau, chỉ định khoảng thời gian 15 phút, 30 phút, 1 giờ, 12 giờ và 24 giờ.

05:17

Và nếu bạn không sử dụng một trong các hằng số này, thì hành vi của các báo động là điều tương tự mà bạn đã nhận được đã sử dụng setRepeat thay thế.

05:28

Bây giờ, tôi chỉ cho bạn một tham số gọi là loại. Bây giờ hãy nói về các loại báo động.

05:37

Android cung cấp hai mức cấu hình, liên quan đến báo động.

05:43

Một, phải làm với thông tin như thế nào? Và cái khác, chẳng hạn như Android cách phản hồi nếu thiết bị đang ngủ khi báo thức kêu. Chúng ta hãy nhìn vào từng cái một.

05:58

Đầu tiên, bạn nhớ rằng mỗi phương thức cài đặt báo thức mất nhiều thời gian một tham số, và tôi đã nói rằng từ lâu đã đại diện cho một thời gian.

06:09

Báo động Android có thể diễn giải giá trị dài hạn này theo hai cách khác nhau. Một, nó có thể được coi là một thời gian đồng hồ treo tường thực sự. Và trong trường hợp này, tên dài biểu thị số mili giây kể từ nửa đêm ngày 1 tháng 1 năm 1970. Và hai, chúng ta có thể diễn giải thời gian hoạt động của hệ thống. Và đó là khoảng thời gian mà lần cuối cùng khởi động. Vấn đề thứ hai là Android nên làm gì

06:44

nếu khi báo thức kêu, thiết bị đang ngủ.

06:49

Một khả năng là đánh thức thiết bị và đưa ra ý định.

06:54

Một lựa chọn khác là để thiết bị tiếp tục ngủ để cung cấp lần sau khi thiết bị thức dậy.

07:03

Vì vậy, đặt tất cả những thứ đó lại với nhau, có bốn khả năng, được định nghĩa như sau. RTC\_WAKEUP, bật báo thức vào thời gian đồng hồ treo tường được chỉ định.

07:15

Nếu thiết bị đang ngủ, hãy đánh thức nó ngay và đưa ra ý định. RTC, kích hoạt báo động theo quy định nhưng nếu thiết bị đang ngủ, đừng đánh thức nó ngay bây giờ. Thay vào đó, cung cấp ý định khi thiết bị tiếp theo thức dậy.

07:36

Và có ELAPSED\_REALTIME và ELAPSED\_REALTIME\_WAKEUP. Đối với hai loại báo thức này, Android kích hoạt báo thức khi thiết bị đã hoạt động nếu thiết bị đang ngủ khi báo thức kêu nó sẽ không được đánh thức với ELAPSED\_REALTIME, nó sẽ được đánh thức với ELAPSED\_REALTIME\_WAKEUP. Phần cuối cùng của API trình quản lý báo thức mà chúng ta sẽ thảo luận là PendingIntent.

08:14

Như chúng ta đã thảo luận lại trong thông báo người dùng, PendingIntent giữ thường xuyên nó được phép để lại một giây thành phần để sử dụng mục đích của công ty con như thể nó là thành phần đầu tiên.

08:32

Ba phương thức có thể được sử dụng để tạo Pendingintent là, getActivity, trong đó trả về một mục đích đang chờ xử lý có thể được sử dụng để bắt đầu một hoạt động, getBroadcast, mà trả về PendingIntent có thể được sử dụng để phát sóng và ý định. Và nhận dịch vụ, trả về PendingIntent có thể được sử dụng để bắt đầu dịch vụ.

08:57

Vì vậy, bây giờ chúng ta đã học về báo động, chúng ta hãy xem một ứng dụng ví dụ có tên là AlertCreate. Ứng dụng này sử dụng báo động để nhẹ nhàng khuyến khích học sinh dừng chơi trò chơi điện tử và quay trở lại việc học của mình. Hãy xem nào. Bây giờ tôi sẽ khởi động ứng dụng AlertCreate.

09:21

Ứng dụng hiển thị giao diện người dùng đơn giản với bốn nút.

09:26

Một nút, có nhãn Đặt báo thức đơn,

09:30

sẽ đặt một báo thức duy nhất để tắt trong hai phút.

09:34

Một nút có nhãn Lặp lại Bộ báo thức, đặt báo thức lặp lại sẽ tắt trong 2 phút và sau đó cứ sau 15 phút.

09:46

Một nút có nhãn Đặt báo thức lặp lại không chính xác, đặt báo thức lặp lại. Nó sẽ tắt sau mỗi 15 phút, bắt đầu trong khoảng 2 phút.

09:58

Bây giờ bởi vì đây là một báo động không chính xác, Android sẽ cố gắng báo thức cứ sau 15 phút, mục tiêu sẽ thực hiện rất nhiều tính linh hoạt khi chính xác những báo thức đó tắt.

10:12

Và cuối cùng, một nút bấm, cả hai đều hủy báo thức lặp lại. Và điều đó sẽ hủy bỏ bất kỳ báo động lặp lại hiện đang hoạt động, hoặc báo động lặp lại không chính xác.

10:22

Bây giờ, tôi sẽ nhấp vào nút có nhãn, Đặt báo thức đơn.

10:28

Và như bạn thấy, toast message gợi ý rằng ứng dụng hiện đã đặt một báo thức.

10:34

Và trong trường hợp này, báo thức sẽ tắt trong hai phút. Và khi nó xảy ra, mã sẽ đặt một thông báo người dùng trong khu vực thông báo.

10:46

Chúng ta hãy quay lại điểm này và xem điều gì sẽ xảy ra.

10:50

Vì vậy, bây giờ chúng tôi trở lại xem thiết bị.

10:54

Và có thông báo hỏi liệu tôi có chơi lại Angry Birds không.

11:00

Hãy để tôi kéo xuống khu vực thông báo.

11:03

Chế độ xem thông báo cho tôi biết rằng đây là một lời nhắc nhở tử tế để trở lại học tập

11:10

Nếu tôi nhấp vào xem thông báo này ứng dụng Tạo báo thức đang được sao lưu.

11:18

Bây giờ, trước khi chúng ta tiến lên phía trước, hãy xem mã nguồn cho ứng dụng này.

11:26

Bây giờ ở đây tôi đã mở ứng dụng trong IDE và bây giờ tôi sẽ mở hoạt động chính.

11:34

Chúng ta hãy xem phương pháp onCreate.

11:37

Đầu tiên, mã được tham chiếu đến dịch vụ quản lý báo động.

11:42

Tiếp theo, mã tạo ra một ý định, có mục tiêu là lớp nhận thông báo.

11:50

Ý định này sau đó được gói gọn trong nguyên nhân chờ xử lý sẽ gây ra ý định này sẽ được phát sóng.

11:57

Và sau đó, mã tạo ra một giây được gọi là AlertLoggerReceiver.

00:07

Và như trước đây, bức thư này được gói trong PendingIntent cuối cùng sẽ được gửi đến người nhận.

00:16

Bây giờ, cuộn xuống, mã thiết lập nút mà chúng ta đã thấy trước đó. Nút đầu tiên, khi được nhấn, sẽ thiết lập hai báo động một lần bắn.

00:28

Báo thức đầu tiên dự kiến ​​sẽ tắt sau khi nhấn nút.

00:34

Thứ hai của cặp này, sẽ tắt sau năm giây sau. Nút thứ hai khi nhấn, sẽ là một cặp báo động lặp lại. Báo thức đầu tiên dự kiến ​​sẽ tắt trong hai phút

00:50

sau khi nhấn nút, và sau đó cứ sau 15 phút lại tắt. Báo động thứ hai của đồng nghiệp

01:03

Nút thứ ba, khi được nhấn, sẽ thiết lập hai báo thức lặp lại không chính xác. Báo thức đầu tiên dự kiến ​​sẽ tắt, khoảng 15 phút một lần, bắt đầu 2 phút sau khi nhấn nút. Và một lần nữa, báo động thứ hai

01:19

đã sẵn sàng để đi sau năm giây sau lần đầu tiên. Bây giờ hãy nhớ, bởi vì đây là một báo động lặp lại không chính xác và vàng nó đã xem xét tính linh hoạt trong thời gian chính xác của cả hai báo động này. Đặc biệt, Android sẽ cố gắng giảm thiểu nó đang ngủ Ví dụ, bằng cách nhóm các báo động riêng biệt tắt cùng một lúc.

01:49

Cuối cùng, nút thứ tư, khi được nhấn, sẽ hủy các báo động hiện có.

01:54

Đặc biệt, điều này rất quan trọng để lặp lại các báo động, sẽ tiếp tục tắt cho đến khi chúng bị hủy hoặc cho đến khi thiết bị tắt.

02:07

Quay trở lại ứng dụng đang chạy, Bây giờ tôi sẽ nhấn nút Lặp lại.

02:14

Và điều này đặt ra một số báo thức lặp lại mới sẽ tắt sau 2 phút, và sau đó cứ sau 15 phút.

02:25

Hãy quay lại khi chúng ta sẵn sàng.

02:27

Được rồi, vì vậy chúng tôi trở lại bây giờ, và đã được khoảng hai phút kể từ khi báo thức được đặt.

02:33

Có thông báo cho thấy báo thức đã tắt.

02:38

Báo thức tiếp theo sẽ đến trong khoảng 15 phút. Hãy nghỉ ngơi ngay bây giờ, và khi chúng tôi quay lại, chúng tôi sẽ kiểm tra đầu ra logcat cho ứng dụng này.

02:50

Vì vậy, hơn 15 phút đã trôi qua, và đây là ứng dụng mở trong IDE.

02:57

Bây giờ tôi sẽ mở rộng chế độ xem logcat và tôi sẽ lọc đầu ra logcat để chỉ hiển thị thông điệp tường trình từ ứng dụng này. Tôi sẽ gõ, tag: alarm, và đó chỉ là thông điệp mà chúng ta quan tâm bây giờ.

03:19

Như bạn có thể thấy, có bốn tin nhắn. 2 từ lần đầu tiên báo động nổ, và Thêm 2 từ khi báo thức đã tắt sau 15 phút.

03:31

Và lưu ý rằng hai báo động giữa họ, bạn nghĩ gì?

03:41

Vì vậy, những người đã lặp lại báo động. Chúng ta hãy quay lại ứng dụng và xem điều gì sẽ xảy ra khi chúng ta sử dụng phương thức setInexactRepeat để đặt các báo thức tương tự.

03:54

Vì vậy, đây là thiết bị của tôi một lần nữa và bây giờ tôi sẽ kéo xuống khu vực thông báo, và sử dụng nó để quay lại ứng dụng.

04:03

Đầu tiên, tôi sẽ hủy các báo động hiện có, và tiếp theo tôi sẽ đặt báo thức lặp lại không chính xác.

04:12

Vì vậy, hãy để ứng dụng chạy một lúc và sau đó chúng ta sẽ có cái nhìn khác về đầu ra logcat cho ứng dụng này. Vì vậy, ở đây chúng tôi trở lại trong IDE, và khoảng 20 phút đã trôi qua kể từ lần cuối chúng tôi nói chuyện.

04:26

Chúng ta hãy quay lại chế độ xem LogCat. Và ở đây bạn có thể thấy bài viết mới. Hai từ lần đầu tiên báo động nổ, và thêm hai từ lần thứ hai những báo động đã tắt.

04:41

Và một số điều cần chú ý lúc 11:19 chúng tôi

04:45

đặt báo thức để tắt sau hai phút,

04:50

bộ báo động đầu tiên thực sự đã tắt sau bốn hoặc năm phút. Điều thứ hai cần chú ý, là mặc dù cặp báo thức đã được lên kế hoạch đi trong một giây trì hoãn giữa họ, Android thực sự đã sa thải họ cùng một lúc. Và một lần nữa, vì đây là những báo động lặp lại không chính xác, Android đã miễn phí sửa đổi thời gian báo thức chính xác.

## Graphics and Animation - Part 1

## Các thiết bị cầm tay ngày nay đi kèm với CPU mạnh mẽ và màn hình mật độ cao sáng. Và các ứng dụng có thể sử dụng các khả năng này để trình bày các yếu tố đồ họa phong phú cho người dùng. Và để làm động các yếu tố đó để cung cấp cho người dùng trải nghiệm hình ảnh mượt mà và năng động.

## Trong bài học này, chúng ta sẽ nói về cách các ứng dụng thực hiện điều này thông qua việc sử dụng cẩn thận đồ họa và hoạt hình hai chiều ( two dimensional graphics and animation).

## Tôi sẽ bắt đầu bài học này bằng cách thảo luận về sự hỗ trợ của Android cho đồ họa hai chiều hoặc 2D. Tôi sẽ nói về cách các ứng dụng có thể vẽ các yếu tố thay đổi tĩnh và động vào màn hình của chúng. Sử dụng lớp ImageView. Và sử dụng lớp Canvas. Tiếp theo, tôi sẽ nói về những cách khác nhau mà bạn có thể dễ dàng tạo hiệu ứng cho các chế độ xem để cung cấp các hiệu ứng đơn giản như thay đổi kích thước và vị trí của chế độ xem và làm mờ dần chế độ xem trong và ngoài. Và cuối cùng, tôi sẽ kết thúc với một cuộc thảo luận tổng quát hơn về hoạt hình thuộc tính, cung cấp cho các ứng dụng một khung chung để hoạt hình không chỉ các thuộc tính xem đơn giản, mà về cơ bản là bất kỳ thuộc tính nào khác.

## Khi ứng dụng của bạn muốn đưa đồ họa 2D lên màn hình. Nó có thể làm điều đó theo những cách khác nhau. Đặc biệt, nó có thể vẽ đồ họa để xem.

## Hoặc nó có thể vẽ lên một bức tranh. Vẽ để xem đơn giản hơn nhưng kém linh hoạt. Bạn sẽ sử dụng tùy chọn này khi đồ họa bạn muốn vẽ đơn giản. Và khi bạn không có kế hoạch cập nhật chúng quá thường xuyên, nếu có.

## Vẽ vào một bức tranh phức tạp hơn nhưng cũng mạnh mẽ hơn và linh hoạt hơn. Và bạn sẽ đi theo con đường này khi đồ họa bạn muốn vẽ phức tạp hơn và khi bạn muốn cập nhật những đồ họa đó khá thường xuyên.

## Có nhiều cách để vẽ với quan điểm. Nhưng trong bài học này, tôi sẽ tập trung vào việc vẽ bằng cách sử dụng lớp có thể vẽ được.

## Một drawable đại diện cho một cái gì đó có thể được rút ra. Những thứ như bitmap, màu sắc, hình dạng, và nhiều hơn nữa. Một số drawable đơn giản bao gồm một lớp ShapeDrawable, đại diện cho một hình dạng như hình chữ nhật hoặc hình bầu dục. Lớp BitmapDrawable, đại diện cho một ma trận pixel. Và lớp ColorDrawable, đại diện cho một màu đơn sắc.

## Trong các ứng dụng ví dụ của chúng tôi cho bài học này. Chúng ta thường tạo một đối tượng có thể vẽ và gắn nó vào chế độ xem hình ảnh. Và sau đó chúng tôi sẽ để chế độ xem hình ảnh xử lý tất cả các bản vẽ thực tế cho chúng tôi.

## Cũng như các tính năng giao diện người dùng Android mà chúng ta đã thấy, bạn có thể thực hiện việc này thông qua các tệp XML hoặc bạn có thể thực hiện thông qua các hướng dẫn chương trình rõ ràng.

## Các ứng dụng ví dụ đầu tiên của chúng tôi được gọi là GraphicsBubble XML và Chương trình GraphicsBubble. Các ứng dụng đơn giản này đều hiển thị một ImageView duy nhất và ImageView giữ hình ảnh bitmap của bong bóng xà phòng. (soap bubble.)

## Hãy xem nào.

## Bây giờ, tôi sẽ bắt đầu một trong những ứng dụng, GraphicsBubble XML.Và ở đó bạn có thể thấy hình ảnh bong bóng (bubble image) đơn giản. Vì vậy, hãy xem mã nguồn cho cả hai ứng dụng này bắt đầu bằng mã cho GraphicsBubble XML. Vì vậy, đây là ứng dụng mở trong IDE. Bây giờ tôi sẽ mở hoạt động chính cho ứng dụng này.

## Như bạn có thể thấy nó rất đơn giản. Tất cả những gì nó làm là gọi setContentView bằng cách sử dụng tệp bố cục main.xml. Hãy mở tập tin đó. Đây là tệp XML và nó chỉ định rằng toàn bộ bố cục là một bố cục tương đối. Và lồng bên trong bố cục tương đối là một ImageView. ImageView này có chiều rộng bố cục và chiều cao bố cục, 250 pixel độc lập mật độ hoặc dp. ImageView cũng được tập trung vào bên trong cha mẹ của nó. Bố cục tương đối.

## Và cuối cùng, bitmap thực sự cho bong bóng là một trong những thư mục có thể vẽ được và nó được gọi là B128. Chúng ta cũng hãy xem một ứng dụng làm điều tương tự. Nhưng điều đó xây dựng giao diện người dùng của nó theo chương trình.

## Vì vậy, đây là ứng dụng GraphicsBubbleProgram mở trong IDE. Bây giờ tôi sẽ mở hoạt động chính cho ứng dụng này và ứng dụng này cũng gọi setContentView bằng cách sử dụng tệp bố cục main.xml.

## Nhưng trong trường hợp này, bố cục đó chỉ bao gồm bố cục tương đối bên ngoài nhất mà không có gì bên trong nó.

## Hãy mở tập tin đó. Vì vậy, đây là tệp XML và như tôi đã nói, nó chỉ xác định rằng toàn bộ bố cục là một bố cục tương đối, nhưng nó không có bất kỳ khung nhìn con nào bên trong nó. Vì vậy, quay trở lại hoạt động chính, mã này tiếp tục bằng cách tạo ImageView. Tiếp theo, nó đặt bitmap b128 làm hình ảnh có thể vẽ được cho chế độ xem hình ảnh.

## Sau đó, mã tiếp tục bằng cách đặt tất cả các thuộc tính bố cục mà chúng ta đã thấy trước đó trong phiên bản XML.

## Đầu tiên, nó đặt chiều cao và chiều rộng của ImageView. Các giá trị này được lưu trữ trong một tệp khác gọi là dimens.xml. Điều đó được lưu trữ trong thư mục res / value.

## Tiếp theo, mã tạo một đối tượng RelativeLayout.LayoutParams với chiều cao và chiều rộng chính xác.

## Sau đó, mã thêm một quy tắc cho đối tượng LayoutParams. Điều này cho Android biết trung tâm chế độ xem hình ảnh này bên trong bố cục tương đối. Sau đó, mã đặt các tham số bố cục hoặc thuộc tính bố cục này trên ImageView.

## Và cuối cùng, nó bổ sung ImageView như một đứa con của relativelayout.

## Vì vậy, hãy nói về một số loại khác của drawables.

## Một loại có thể rút ra là ShapeDrawable.

## ShapeDrawables được sử dụng để vẽ các hình đơn giản.Các hình dạng khác nhau được đại diện bởi các lớp con khác nhau của Các lớp hình dạng, bao gồm. PathShape cho các đoạn đường và đường cong. RectShape cho hình chữ nhật. Và OvalShape cho hình bầu dục & nhẫn.

## Các ứng dụng ví dụ tiếp theo của chúng tôi là XML GraphicsShapeDraw và Chương trình GraphicsShapeDraw. Các ứng dụng này hiển thị hai hình bầu dục trong RelativeLayout. Hai hình dạng có màu sắc khác nhau. và là bán trong suốt. Hãy chạy các ứng dụng đó.

## Và bây giờ tôi sẽ bắt đầu một trong những ứng dụng. Hình dạng đồ họa vẽ XML. Và ở đó bạn có thể thấy hai hình bầu dục. Cái bên trái có màu lục lam. Và cái bên phải là màu đỏ tươi. Như bạn có thể thấy, các hình bầu dục chồng lên nhau. Và nơi trùng lặp, màu sắc của chúng đã trộn lẫn để tạo thành một loại màu tím.

## Hãy xem mã nguồn của các ứng dụng này. Vì vậy, đây là ứng dụng vẽ đồ họa XML mở trong IDE. Tôi sẽ mở hoạt động chính cho ứng dụng này.

## Một lần nữa, ứng dụng chỉ gọi setContentView sử dụng tệp bố cục main.xml. Hãy mở tập tin đó.

## Đây là tệp XML.

## Và nó xác định rằng toàn bộ bố cục là một bố cục tương đối. Và lồng bên trong bố cục tương đối này là hai chế độ xem hình ảnh. Cả hai chế độ xem hình ảnh đều có chiều rộng bố cục và chiều cao bố cục 250 dp.

## Cả hai thêm một số không gian hoặc đệm xung quanh nội dung của họ.

## Và cả hai đều được căn giữa theo chiều dọc bên trong bố trí tương đối.

## Tuy nhiên, chế độ xem hình ảnh đầu tiên được căn chỉnh về phía bên trái của cha mẹ,

## trong khi hình ảnh thứ hai được căn chỉnh sang phải.

## Và cuối cùng, hình ảnh thực tế được định nghĩa bằng thuộc tính android: src. Cyan\_shade.

## Hãy mở tập tin đó. Nó nằm trong thư mục res / drawable.

## Tập tin này là cụ thể cho hình dạng của nó hình dạng là một hình bầu dục, và màu này được cho bởi giá trị thập lục phân này.

## Tất nhiên, có một tệp tương tự cho hình dạng màu đỏ tươi.

## Và như trước đây, chúng ta có thể làm chính xác những điều tương tự theo lập trình.

## Chúng ta hãy xem ứng dụng chương trình đồ họa shape.draw. Mà tôi cũng đã mở trong IDE.

## Tôi sẽ mở hoạt động chính cho ứng dụng này. Một lần nữa, ứng dụng chỉ gọi setContentView bằng cách sử dụng tệp bố cục main.xml. Tập tin đó chỉ xác định rằng toàn bộ bố cục là một bố cục tương đối.

## Bây giờ, mã tìm thấy chiều rộng bố trí, chiều cao bố trí và phần đệm. Tiếp theo, mã được tham chiếu đến bố cục tương đối.

## Và sau đó, nó tạo ra một ShapeDrawable mới có OvalShape.

## Nó tiếp tục bằng cách thiết lập màu sắc của hình dạng, chiều cao và chiều rộng của nó

## Tiếp theo mã tạo ImageView và đặt hình dạng mới vào đó. Nó cũng đặt phần đệm trên ImageView.

## Và tiếp tục, mã đặt một số tham số bố cục cho ImageView.

## Cụ thể, nó tập trung ImageView theo chiều dọc trong bố cục tương đối. Và nó căn chỉnh hình ảnh này ở bên trái của cha mẹ. Đoạn mã sau đó kết thúc bằng cách thực hiện những điều tương tự cho MagentaView.

## Bây giờ, nếu bạn muốn vẽ nhiều hơn, bạn cũng có thể vẽ bằng Canvas. Và để làm điều này bạn cần bốn điều. Một bitmap, về cơ bản là ma trận các pixel mà bạn muốn vẽ.

## Canvas, nơi lưu trữ các cuộc gọi vẽ sẽ cập nhật bitmap bên dưới.

## Một bản vẽ nguyên thủy, đại diện cho hoạt động vẽ cụ thể mà bạn muốn phát hành. Và một đối tượng sơn, cho phép bạn thiết lập nhiều màu sắc khác nhau phong cách cho bản vẽ bạn muốn làm.

## Chúng ta sẽ đi vào chi tiết hơn về lớp vải chỉ trong một chút. Aim, canvasses cung cấp một loạt các phương pháp vẽ. Ví dụ: bạn có thể vẽ văn bản, dấu chấm, màu sắc, hình bầu dục và bitmap bằng các phương pháp này.

## Khi bạn vẽ, bạn có thể sử dụng lớp Paint để đặt tham số kiểu. Chẳng hạn, bạn có thể chỉ định những thứ như độ dày của đường, kích thước của văn bản, màu sắc bạn đang vẽ và có hoặc không không áp dụng các tối ưu hóa khác nhau, chẳng hạn như khử răng cưa. Được sử dụng để làm mịn các cạnh lởm chởm của hình ảnh.

## Hãy xem xét một ứng dụng đơn giản rút ra một số hộp, mỗi trong số đó giữ một số văn bản. Nhưng nó làm như vậy bằng cách sử dụng sơn khác nhau cho mỗi hộp.

## Bây giờ tôi sẽ bắt đầu ứng dụng vẽ đồ họa.

## Ứng dụng khởi động và lò nướng thẳng đứng hình chữ nhật ra cái tiếp theo.

## Mỗi hình chữ nhật này có một số văn bản, mỗi trong số đó là một kích thước và phong cách khác nhau.Mỗi hình chữ nhật có chiều rộng đường viền khác nhau. Và phong cách viền, và có một màu nền khác nhau.

## Hãy xem mã nguồn của các ứng dụng này. Chúng tôi sẽ chọn ra một vài trong số các tham số kiểu này và xem cách chúng được chỉ định.

## Vì vậy, đây là ứng dụng GraphicsPaint mở trong IDE.

## Giống như một số trong những người chúng ta đã thấy trước đây, phương pháp onCreate của ứng dụng này chỉ gọi setContentView, chuyển tham chiếu đến tệp bố cục main.xml. Hãy mở tập tin đó ra.

## Đây là tệp XML.

## Và nó xác định rằng toàn bộ bố cục là một bố cục tuyến tính và bố cục tuyến tính đó có bốn con, mỗi con là một TextView. Nếu chúng ta nhìn vào phần đầu của các TextView này, chúng ta có thể thấy rằng nó đặt một số thuộc tính kiểu văn bản. Chẳng hạn. Cái này đặt màu văn bản của nó thành giá trị thập lục phân này.

## Kích thước văn bản 32 pixel hoặc sp độc lập.

## Đó là phong cách để in đậm và in nghiêng và đó là kiểu chữ bình thường.

## Nếu bạn nhìn vào các TextView khác, bạn sẽ thấy rằng họ thực hiện các lựa chọn phong cách khác nhau.

## ThisTextView cũng chỉ định một nền trong tệp có tên là SQ1.xml, trong thư mục res / drawable. Hãy mở tập tin đó.

## Vì vậy, đây là tệp SQ1.xml. Và như bạn có thể thấy, tập tin này xác định một hình dạng.

## Hình dạng nào là hình chữ nhật và nó có màu đặc.

## Trong trường hợp này, một màu trắng, được xác định bởi Android.

## Và cuối cùng hình dạng có đường viền, với chiều rộng ba pixel Nó có màu nền trong trường hợp này là màu đen mờ hoàn toàn.

## Graphics and Animation - Part 2

## Nếu bạn muốn vẽ đồ họa phức tạp hơn và bạn muốn cập nhật những đồ họa này thường xuyên, sau đó bạn có thể thực hiện bản vẽ của mình bằng một khung vẽ. Và tôi đã nói trước đó, một bức tranh là một loại vàng bối cảnh cơ chế để vẽ một bitmap cơ bản. Và bạn có thể truy cập vào một khung vẽ thông qua một lớp con xem đặc biệt gọi là SurfaceView.

## Bạn thường sẽ có thể kiểm tra khi bạn không thường xuyên. Và nếu bạn thực hiện phương pháp này, ứng dụng của bạn sẽ tạo một lớp con dạng xem tùy chỉnh và sau đó là hệ thống sẽ cung cấp khung vẽ cho khung nhìn của bạn thông qua một cuộc gọi đến phương thức onDraw của nó.

## Nếu thay vào đó, ứng dụng của bạn cần cập nhật đồ họa, sau đó bạn có thể xem xét vẽ qua SurfaceView. Với cách tiếp cận này, ứng dụng tạo ra một lớp con SurfaceView tùy chỉnh và nó cũng sẽ tạo ra một chủ đề thứ cấp SurfaceView sẽ được thực hiện.

## Trong thời gian chạy, ứng dụng sau đó có thể có được khung vẽ riêng của mình và do đó kiểm soát nhiều hơn về cách thức và thời điểm vẽ.

## Ứng dụng ví dụ tiếp theo này được gọi là GraphicsCanvasBubble. Và ý tưởng đằng sau ứng dụng này là nó sẽ vẽ một khung nhìn và sau đó cập nhật khung nhìn đó, nhưng các bản cập nhật có phần không thường xuyên, khoảng mỗi giây hoặc lâu hơn. Vì vậy, ứng dụng này có một luồng nội bộ thức dậy mỗi giây một lần hoặc lâu hơn và di chuyển khung nhìn, sau đó sử dụng một khung vẽ khi nó vẽ lại khung nhìn ở vị trí mới. Chúng ta hãy xem điều đó trong hành động.

## Vì vậy, đây là thiết bị của tôi. Và bây giờ, tôi sẽ khởi động ứng dụng GraphicsCanvasBubble.

## Ứng dụng bắt đầu với bong bóng được vẽ tại một vị trí được chọn ngẫu nhiên.

## Và cứ sau một giây, bạn có thể thấy bong bóng bị xóa, di chuyển và sau đó vẽ lại ở vị trí mới.

## Hãy xem mã nguồn của ứng dụng này.

## Vì vậy, đây là ứng dụng GraphicsCanvasBubble mở trong IDE. Bây giờ tôi sẽ mở hoạt động chính. Ở đây bạn có thể thấy rằng ứng dụng tải một bitmap từ thư mục tài nguyên. Và sau đó, nó sử dụng bitmap đó để tạo một thể hiện của lớp xem tùy chỉnh được gọi là bubbleView. Tiếp theo, mã thêm đối tượng bubbleView vào bố cục. Và sau đó nó tạo và bắt đầu một chủ đề mới.Và chủ đề đó đi vào một vòng lặp. Và trên mỗi lần lặp, nó gọi phương thức bubbleViews.move.

## Bây giờ, như chúng ta sẽ thấy trong một giây, phương thức này thay đổi vị trí của bubbleView và sau đó trả về giá trị đúng hay sai tùy thuộc vào vị trí mới của bubbleView có hay không vẫn hiển thị trên màn hình.

## Tiếp theo mã gọi phương thức postInvalidate của lớp xem. Phương pháp này bảo Android vẽ lại khung nhìn này. Và sau đó, sợi chỉ đi ngủ một giây trước khi thức dậy và bắt đầu quá trình một lần nữa. Bây giờ, tôi sẽ bỏ qua hầu hết các chi tiết về cách bubbleView tự di chuyển và thay vào đó tôi sẽ tập trung vào cách nó được vẽ lại.

## Vì vậy, khi luồng vẽ gọi phương thức postInvalidate, điều này cho Android biết rằng bubbleView nghĩ rằng nó cần phải được vẽ lại.

## Nếu vậy, Android cuối cùng sẽ gọi phương thức onDraw của bubbleView, chuyển qua khung vẽ mà bubbleView được vẽ. Hãy xem phương pháp đó. Và như bạn có thể thấy, phương thức này đưa khung vẽ được truyền vào nó và gọi phương thức drawBitmap của nó, truyền vào bitmap để vẽ, chuyển qua tọa độ trên cùng và bên trái của vị trí để vẽ bitmap và cuối cùng, đi qua một đối tượng sơn xác định tham số phong cách cho hoạt động vẽ này.

## Bây giờ, nếu chúng ta thực sự muốn tăng tần suất mà chúng ta vẽ lại bong bóng để làm cho nó lướt nhẹ hơn trên màn hình, thì chúng ta có thể muốn sử dụng một khung vẽ và SurfaceView để làm điều đó. Và như tôi đã đề cập trước đó, SurfaceViews cần một luồng không phải UI riêng để thực hiện công việc của chúng để các hoạt động của SurfaceView không can thiệp vào các hoạt động của luồng UI. Vì vậy, hãy nói nhiều hơn về lớp SurfaceView. SurfaceView quản lý một khu vực vẽ mức thấp gọi là Surface. Và Surface này là một khu vực vẽ chuyên dụng được định vị trong hệ thống phân cấp chế độ xem của ứng dụng.

## Để xác định SurfaceView, bạn cần tạo một lớp tùy chỉnh là lớp con của SurfaceView. Và lớp tùy chỉnh này cũng phải thực hiện giao diện SurfaceHolder.Callback.

## Để sử dụng SurfaceView mới này, bạn cần thực hiện hai điều. Một, bạn cần thiết lập SurfaceView. Và hai, bạn cần vẽ lên SurfaceView này mà bạn vừa thiết lập. Chúng ta hãy nói về từng bước một. Để thiết lập SurfaceView, trước tiên bạn sử dụng phương thức getHolder của SurfaceView để có được SurfaceHolder của SurfaceView.

## Tiếp theo, bạn đăng ký các cuộc gọi lại của SurfaceView hoặc SurfaceHolder bằng cách gọi phương thức addCallback của SurfaceHolder. Bây giờ, các phương thức này là một, SurfaceCreated, được gọi khi SurfaceView được tạo.

## Cho đến khi phương thức này được gọi, bạn không thể vẽ trên bề mặt. Nó chưa sẵn sàng.

## Hai, bề mặt Thay đổi. Phương pháp này được gọi bất cứ khi nào thay đổi cấu trúc, chẳng hạn như thay đổi kích thước bề mặt, xảy ra. Và ba, bề mặt. Và phương pháp này được gọi ngay trước khi một bề mặt bị phá hủy. Khi phương thức này trở lại, bạn không còn có thể gọi các thao tác sẽ vẽ trên Surface.

## Và bước thiết lập cuối cùng là tạo luồng sẽ được sử dụng để thực hiện các thao tác vẽ trên SurfaceView này. Và hãy nhớ rằng các phương thức gọi lại của SurfaceHolder thường sẽ được gọi từ luồng chính, không phải từ luồng của SurfaceView. Vì vậy, bạn sẽ phải đảm bảo rằng bạn đồng bộ hóa quyền truy cập vào bất kỳ dữ liệu nào cần thiết cho cả hai luồng.

## Vì vậy, khi bạn đã thiết lập SurfaceView, bạn có thể bắt đầu vẽ lên nó.

## Để vẽ, trước tiên bạn sẽ có được một khóa trên khung vẽ bằng cách gọi phương thức lockCanvas.

## Tiếp theo, bạn thực hiện bất kỳ thao tác vẽ nào bạn muốn làm trên khung vẽ. Ví dụ: bằng cách gọi một phương thức canvas như drawBitmap.

## Và cuối cùng, bạn mở khóa khung vẽ, cho phép Android cập nhật màn hình. Và bạn sẽ làm điều này bằng cách gọi phương thức UnlockCanvasAndPost của SurfaceHolder.

## Vì vậy, hãy xem xét một triển khai khác của ứng dụng ví dụ cuối cùng của chúng tôi được gọi là GraphicsBubbleCanvas SurfaceView.

## Tôi sẽ khởi động ứng dụng GraphicsBubbleCanvasSurfaceView. Và khi tôi làm, ứng dụng sẽ khởi động và vẽ bong bóng tại một vị trí được chọn ngẫu nhiên trên màn hình. Nhưng lần này, thay vì cập nhật mỗi giây hoặc lâu hơn, ứng dụng này sẽ cố gắng thực hiện càng nhiều cập nhật càng tốt.

## Ngoài ra, ứng dụng cũng sẽ xoay BubbleView để tạo ra sự xuất hiện của bong bóng đang xoay tròn trong không gian. Vì vậy, ở đây đi.

## Và như bạn có thể thấy, bong bóng đang hoạt hình trơn tru, cả di chuyển và xoay khi nó trôi theo. Hãy xem mã nguồn của ứng dụng này.

## Vì vậy, đây là ứng dụng GraphicsBubbleCanvasSurfaceView mở trong IDE. Bây giờ, tôi sẽ mở hoạt động chính. Và trong onCreate, mã này lại tải một bitmap từ thư mục tài nguyên và sau đó sử dụng bitmap đó để tạo một thể hiện của lớp xem tùy chỉnh được gọi là bubbleView.

## Chúng ta hãy nhìn vào lớp BubbleView.

## Lớp BubbleView mở rộng SurfaceView và thực hiện giao diện SurfaceHolder.Callback. Các nhà xây dựng cho lớp này làm rất nhiều vệ sinh. Và sau đó, ở cuối phương thức, có một lệnh gọi phương thức getHolder, trả về SurfaceHolder. Mã lấy SurfaceHolder đó và sau đó đăng ký BubbleView này cho callbacks.

## Hãy xem điều gì xảy ra khi những cuộc gọi lại cuối cùng cũng đến.

## Đầu tiên, chúng ta thấy rằng khi Surface cho SurfaceView được tạo, mã này sẽ tạo ra một luồng mới và sau đó khởi động nó.

## Và phương thức chạy của luồng đó sau đó đi vào một vòng lặp. Và trên mỗi lần lặp của vòng lặp, nó sẽ kiểm tra xem luồng có bị gián đoạn hay không. Và nếu không, nó sẽ gọi phương thức di chuyển, giống như trước đó, di chuyển BubbleView và trả về Boolean cho biết liệu BubbleView có rời khỏi màn hình hay không.

## Nếu các kiểm tra này đánh giá là đúng, thì mã sẽ cố gắng khóa canvas của SurfaceHolder và nếu thành công, mã sẽ gọi phương thức drawBubble, chuyển vào canvas bị khóa.

## Và cuối cùng, ứng dụng mở khóa khung vẽ và cho phép Android cập nhật màn hình. Chúng ta hãy quay lại một giây và xem phương thức drawBubble. Bây giờ, phương pháp này trước tiên vẽ lại nền của khung vẽ, sau đó nó xoay khung vẽ và sau đó vẽ lại bong bóng trên khung vẽ.

## Graphics and Animation - Part 3

## Trong các ví dụ về chế độ xem bong bóng (bubble View) mà chúng ta vừa thấy, tôi đã trình diễn một loại hoạt hình đơn giản. Tôi đã có một cái nhìn duy nhất và tôi đã thay đổi một số thuộc tính của nó. Cụ thể, vị trí và định hướng của nó và tôi đã làm điều này trong một khoảng thời gian.

## Hoạt hình như thế này là khá phổ biến. Các ứng dụng thường làm động các thay đổi đối với sự vật, đến các thuộc tính của chế độ xem như kích thước, vị trí, độ trong suốt, hướng của nó và hơn thế nữa.

## Để làm cho hoạt hình dễ dàng hơn, Android cung cấp một số lớp hỗ trợ xem hoạt hình khác nhau. Và ba cái mà chúng ta sẽ nói đến bây giờ là lớp TransitionDrawable để tạo hiệu ứng chuyển tiếp giữa hai chế độ xem.

## AnimationDrawable để tạo khung theo hình động. Và lớp Animation để tạo tween hoạt hình. Nơi bạn chỉ định một số khung hoặc khoảnh khắc nhất định trong hoạt hình và Android sẽ nội suy hoặc điền vào, các điểm ở giữa. Hãy xem xét từng lớp một. Lớp hoạt hình đầu tiên chúng ta sẽ thảo luận là TransitionDrawable. Lớp này chỉ định hai lớp có thể vẽ được. Và khi nó được hiển thị, người dùng sẽ thấy lớp đầu tiên và sau đó một lát sẽ thấy lớp thứ hai.

## Ứng dụng ví dụ của chúng tôi được gọi là GraphicsTransitionDrawable. Và ứng dụng này đưa chúng ta trở lại các ứng dụng GraphicsShapeDraw mà chúng ta đã thấy trước đó. Nhưng lần này, thay vì hiển thị cả hai hình dạng cùng một lúc. Ứng dụng này đầu tiên hiển thị hình trái màu lục lam nhất.

## Sau đó mờ dần thành một màn hình hiển thị màu đỏ tươi nhất bên phải.

## Hãy xem cách nó hoạt động.

## Và khi tôi khởi động ứng dụng GraphicsTransitionDrawable, bạn sẽ thấy hình dạng đầu tiên mờ dần thành hình thứ hai.

## Vì vậy, ở đây chúng tôi đi. Đây là hình dạng đầu tiên.Và đây là cái thứ hai. Vì vậy, hãy xem mã nguồn.Vì vậy, đây là ứng dụng GraphicsTransitionDrawable mở trong IDE.

## Bây giờ, tôi sẽ mở hoạt động chính.

## Trong OnCreate, ứng dụng này tải TransitionDrawable từ một tệp có tên là shape\_transition.xml. Hãy mở tập tin đó. Tập tin này xác định một tài nguyên chuyển tiếp.

## Tài nguyên thuộc loại này có thể có tối đa hai thẻ mục. Và mỗi thẻ mục mô tả một drawable.

## Trong trường hợp này, các drawables là màu lục lam và đỏ tươi quen thuộc.

## Bây giờ trở lại trong hoạt động chính, mã gọi setCorssFadeEnables với tham số true.

## Và điều này làm cho drawable đầu tiên mờ dần thành drawable thứ hai.

## Tiếp theo, mã đặt chuyển đổi là có thể vẽ được cho chế độ xem hình ảnh bằng cách gọi setImageDrawable.

## Và cuối cùng, nó gọi bắt đầu khi quá trình chuyển đổi có giá trị 5000 cho thời lượng hoạt ảnh. Loại hoạt hình tiếp theo mà chúng ta sẽ nói đến là AnimationDrawable.

## Và drawable này hoạt hình một loạt các drawable khác, hiển thị mỗi cái trong một khoảng thời gian. Ứng dụng ví dụ tiếp theo của chúng tôi được gọi là GraphicsFrameAnimation.

## Ứng dụng này sử dụng và hoạt hình có thể vẽ được để hiển thị một loại màn hình giật gân. Mà chính nó trình bày một loạt các hình ảnh đếm ngược đến khi bắt đầu ứng dụng chính.

## Chúng ta hãy xem những gì nó làm.

## Khi tôi khởi động ứng dụng GraphicsFrameAnimation. Bạn sẽ thấy một loạt các hình ảnh đếm ngược đến một hình ảnh cuối cùng.

## Được rồi, hãy bắt đầu. Chín, tám, bảy, sáu, năm, bốn, ba, hai, một, và cuối cùng là hình ảnh mà tất cả chúng ta đang chờ đợi. Vì vậy, hãy đi đến mã nguồn cho ứng dụng này.

## Vì vậy, đây là ứng dụng GraphicsFrameAnimation mở trong IDE.

## Bây giờ tôi sẽ mở hoạt động chính. Và trong OnCreate, ứng dụng này tải AnimationDrawable từ một tệp có tên là view\_animation.xml. Và sau đó, nó đặt AnimationDrawable làm nền trên chế độ xem hình ảnh.

## Mã này sau đó lưu trữ có thể rút ra trong một biến gọi là mAnim. Bây giờ chúng ta trước khi chúng ta đi tiếp, hãy xem tệp viewx animation .xml.

## Vì vậy, đây là tệp view\_animation.xml và tệp này xác định tài nguyên danh sách hoạt hình.

## Tài nguyên này chứa danh sách các thẻ mục trong đó mỗi thẻ đại diện cho một drawable. Và trong trường hợp này, cái đầu tiên có thể rút ra chỉ là một màn hình đen và hiển thị trong một giây.

## Và điều này được theo sau bởi chín hình ảnh khác, mỗi hình ảnh cũng được hiển thị trong một giây.

## Bây giờ trở lại trong hoạt động chính. Mã chờ cho đến khi phương thức onWindowF FocusChanged được gọi.

## Và phương pháp này được gọi khi cửa sổ ứng dụng tăng hoặc giảm trọng tâm. Trong phương thức này, trước tiên mã kiểm tra xem cửa sổ hiện có đang lấy nét hay không và nếu có thì nó sẽ bắt đầu hoạt ảnh.

## Lớp tiếp theo là lớp Animation. Và lớp này được sử dụng để tạo hình ảnh động trong đó các phép biến đổi được áp dụng cho nội dung của một khung nhìn.

## Các ứng dụng có thể chơi với thời gian của các biến đổi này. Để trình tự và kết hợp các biến đổi khác nhau để làm cho hình ảnh động phức tạp hơn.

## Ứng dụng GraphicsTweenAnimation cho thấy việc sử dụng lớp hoạt hình. Khi ứng dụng này chạy, nó sẽ hiển thị chế độ xem bong bóng. Và sau đó tiến hành làm động một số thay đổi cho bong bóng đó. Chúng ta hãy xem nó hoạt động.

## Khi tôi khởi động ứng dụng GraphicsTweenAnimation, bạn sẽ thấy bong bóng. Và sau đó bạn sẽ thấy một loạt các biến đổi được áp dụng cho bong bóng đó. Bạn sẽ thấy nó mờ dần, xoay, di chuyển, thay đổi kích thước và cuối cùng, mờ dần.

## Và bạn cũng sẽ thấy rằng thời gian của các biến đổi này không phải lúc nào cũng tuyến tính. Đó là, một số biến đổi sẽ xảy ra ở một tốc độ thống nhất ,. Một số sẽ bắt đầu chậm và sau đó tăng tốc độ. Một số sẽ bắt đầu nhanh và sau đó giảm tốc độ của họ. Một số với cả tốc độ tăng và giảm tốc độ tại các điểm khác nhau trong hình ảnh động.

## Tôi sẽ làm chậm video ở đây một chút để bạn có thể thấy những hiệu ứng này rõ ràng hơn. Vì vậy, ở đây chúng tôi đi.

## Đầu tiên, bong bóng mờ dần. Và tiếp theo bong bóng quay hai lần, nhanh hơn khi nó đi.

## Bây giờ bong bóng sẽ di chuyển, vượt qua vị trí cuối cùng của nó và sau đó kéo lại một chút.

## Sau đó, bong bóng sẽ co lại một chút trước khi nhanh chóng nhân đôi kích thước của nó.

## Và cuối cùng, bong bóng mờ dần khỏi tầm nhìn. Nhanh chóng lúc đầu và sau đó chậm hơn gần cuối.

## Chúng ta hãy xem mã nguồn cho ứng dụng này.

## Vì vậy, đây là ứng dụng GraphicsTweenAnimation mở trong IDE.

## Bây giờ tôi sẽ mở hoạt động chính. Và ứng dụng này đặt, gọi setContentView, truyền vào tệp .xml chính.

## Và sau đó nó tìm thấy một ImageView trong bố cục đó.

## Và sau đó, nó đọc một hình ảnh động từ tệp view\_animation.xml được lưu trữ trong thư mục res / nm.

## Đây là tài nguyên và được lồng trong thẻ set. Có một số thẻ khác. Mỗi trong số đó chỉ ra một chuyển đổi cụ thể được áp dụng cho chế độ xem. Đầu tiên là một chuyển đổi alpha.

## Thời lượng của hình ảnh động là ba giây. Nó đi từ alpha bằng 0 hoặc hoàn toàn trong suốt đến alpha của một, hoàn toàn mờ đục.

## Việc chuyển đổi cũng sử dụng bộ nội suy tuyến tính. Vì vậy, hiệu ứng được áp dụng thống nhất trong suốt hình ảnh động. Thẻ tiếp theo là một chuyển đổi xoay. Thời lượng được đặt thành bốn giây. Nhưng nó cũng được thiết lập để bắt đầu chỉ sau ba giây trôi qua. Vì vậy, sự chuyển đổi đầu tiên sẽ xảy ra và kết thúc và sau đó lần chuyển đổi này sẽ bắt đầu.

## Hoạt hình này đi từ 0 đến 720 độ, hoặc hai lượt đầy đủ. Và bộ nội suy của nó tăng tốc hoặc tăng tốc khi hoạt ảnh tiếp tục. Và bạn có thể tự mình xem tất cả các thẻ khác khi chúng tôi nghỉ ngơi.

## Quay trở lại hoạt động chính, chúng ta hãy đi đến phương thức onWindowF FocusChanged.

## Một lần nữa, mã này kiểm tra xem cửa sổ hiện đang tập trung. Và nếu vậy, nó bắt đầu Animation.

## Khái niệm cuối cùng tôi sẽ nói đến là Hoạt hình tài sản.

## Các ví dụ trước cho thấy các lớp được thiết kế để tạo hiệu ứng cho một tập hợp các thay đổi thuộc tính đơn giản trên các khung nhìn. Nhưng đôi khi bạn muốn làm động nhiều hơn chỉ những thứ đó. Và để hỗ trợ, Android đã phát triển một hệ thống để thay đổi các thuộc tính chung của các đối tượng chung trong một khoảng thời gian nhất định.

## Và hệ thống hoạt hình tài sản này có một số thành phần. Đầu tiên có ValueAnimator và đây là lớp chính điều khiển hoạt hình. ValueAnimator chứa một bộ nội suy thời gian. Mà xác định cách các giá trị thay đổi như là một hàm của thời gian. Ví dụ, theo thời gian, các thay đổi xảy ra thống nhất. Hoặc là họ tăng tốc, làm chậm hoặc sử dụng kết hợp cả hai.

## ValueAnimator cũng định nghĩa một giao diện người nghe gọi là AnimatorUpdateListener. Và giao diện này xác định phương thức onAnimationUpdate. Mà được gọi mỗi khi một khung hình động mới được tạo.

## Và cuối cùng, trong khi Android biết cách làm động các giá trị của các loại phổ biến như số nguyên và số float. Nó không biết làm thế nào để làm động các loại tùy chỉnh mà bạn tạo.

## Vì vậy, nếu bạn cần làm động các kiểu của riêng bạn hoặc làm động các kiểu hiện có theo những cách mới, bạn có thể thực hiện giao diện đánh giá kiểu. Giao diện này xác định một phương thức đánh giá được gọi để đặt các giá trị hoạt hình tại một điểm cụ thể trong hoạt hình.

## Phần cuối cùng của Thuộc tính hoạt hình, là lớp Animatorset. Lớp này cho phép bạn kết hợp các đối tượng Animator như giá trị Animator mà chúng ta vừa thảo luận thành các Animations phức tạp hơn. Vì vậy, chúng ta hãy xem xét một số ứng dụng ví dụ sử dụng Thuộc tính Hoạt hình.

## Ứng dụng đầu tiên được gọi là GraphicsValueAnimator. Và ứng dụng này sử dụng một trình hoạt hình giá trị để tạo hiệu ứng thay đổi màu nền của ImageView. Hãy xem ứng dụng này. Bây giờ tôi sẽ khởi động ứng dụng GraphicsValueAnimator. Và như bạn có thể thấy nó trình bày một nút duy nhất có nhãn Animate.

## Khi tôi nhấn nút đó, một hình chữ nhật màu đỏ sẽ xuất hiện ở giữa màn hình.

## Và trong khoảng thời gian khoảng mười giây, hình chữ nhật sẽ đổi màu cho đến khi cuối cùng nó có màu xanh.

## Tôi sẽ nhấn nút bây giờ.

## Có hình chữ nhật màu đỏ. Và bây giờ hãy xem khi nó từ từ chuyển sang màu xanh.

## Hãy mở mã nguồn và xem chúng tôi đã làm điều đó như thế nào.

## Vì vậy, đây là ứng dụng GraphicsValueAnimator mở trong IDE.

## Bây giờ, tôi sẽ mở hoạt động chính. Ứng dụng này bắt đầu bằng cách đặt hai giá trị nguyên,

## đại diện cho các điểm nêu và kết thúc cho hình ảnh động. Giá trị đầu tiên là biểu diễn số nguyên của màu đỏ. Giá trị thứ hai là biểu diễn số nguyên của màu xanh lam.

## Trong onCreate mã tạo một nút sẽ bắt đầu hoạt hình. Và như bạn thấy ở đây khi nhấp vào nút, phương thức startAnimation được chạy. Hãy nhìn vào phương pháp đó. Ở đây phương thức startAnimation tạo ra một đối tượng ValueAnimator được gọi là Anim.

## Và nó đã tạo ra đối tượng đó bằng cách gọi phương thức ValueAnimators.ofObject.

## Tham số đầu tiên cho phương thức này là một bộ đánh giá kiểu. Trong trường hợp này, bộ đánh giá kiểu thực sự là một đối tượng đánh giá ARGB. Và lớp này biết làm thế nào để nội suy các biểu diễn số nguyên của màu sắc. Các tham số thứ hai và thứ ba cho phương thức ofObject là điểm bắt đầu và điểm kết thúc cho hình ảnh động. Các màu đỏ và xanh.

## Tiếp theo mã thêm một trình nghe cập nhật hoạt hình và điều đó sẽ được gọi lại. Mỗi lần tạo khung hình động mới. Và mã đó gọi phương thức getAnimatedValue để lấy giá trị màu hiện tại.

## Và sau đó, nó làm cho màu đó thành màu nền của ImageView được hiển thị trong bố cục.

## Và cuối cùng, mã này đặt thời lượng mười giây cho hình động và sau đó bắt đầu hoạt hình chạy.

## Cũng xem xét một ứng dụng hoạt hình thuộc tính thứ hai được gọi là GraphicsViewPropertyAnimator.

## Ứng dụng này sẽ triển khai cùng một ứng dụng mà chúng tôi đã tạo cho ứng dụng GraphicsTweenAnimation.

## Bạn sẽ thấy một bong bóng mờ dần theo vòng quay, di chuyển thay đổi kích thước và sau đó mờ dần.

## Và phiên bản này của ứng dụng tuy nhiên. Sẽ sử dụng lớp ViewPropertyAnimator, đây là một loại hoạt họa được đơn giản hóa cho chỉ các thuộc tính xem. Hãy xem ứng dụng đó hoạt động.

## Và tôi sẽ khởi động ứng dụng GraphicsViewPropertyAnimator và chúng tôi sẽ xem bong bóng trải qua một loạt các biến đổi. Chúng ta đi đây.

## Hãy mở mã nguồn cho ứng dụng này.

## Vì vậy, đây là ứng dụng GraphicsViewPropertyAnimator. Mở trong IDE.

## Bây giờ tôi sẽ mở hoạt động chính và hãy cuộn qua phương thức onWindowF FocusChanged.

## Khi phương thức này được gọi là ứng dụng đảm bảo rằng cửa sổ hoạt động đang tập trung và nếu vậy, nó gọi phương thức chạy của đối tượng fadeIn. Và đối tượng fadeIn đó là một Runnable.

## Bây giờ bên trong phương thức chạy của nó, mã của nó gọi hoạt hình trên ImageView, trả về một đối tượng ViewPropertyAnimator.

## Đối tượng này, hoặc lớp này, sử dụng giao diện trôi chảy. Giống như những gì chúng ta đã thấy với các thông báo khu vực thông báo. Và do đó, bạn có thể xây dựng một hình ảnh động bằng cách thực hiện các cuộc gọi phương thức khác nhau. Trong trường hợp này, cuộc gọi tiếp theo được đặtDuration thành ba giây. Sau đó setInterpolator đến bộ nội suy tuyến tính. Sau đó, một phương thức được gọi là phương pháp alpha để thay đổi độ trong suốt thành mờ hoàn toàn.

## Và sau đó là một cuộc gọi cuối cùng, thông báo cho ViewPropertyAnimator rằng khi hoạt ảnh này kết thúc, nó sẽ gọi phương thức chạy của một runnable khác. Gọi là xoay.

## Và như bạn có thể đoán quay có thể chạy này tạo ra bước xoay của hình ảnh động. Và sau đó kết thúc bằng cách gọi bước chuyển động của hình ảnh động và điều này tiếp tục cho đến khi tất cả các bước đã được hoàn thành.

## Touch and Gestures - Part 1

## Nếu bạn sử dụng các ứng dụng phổ biến hiển thị bản đồ hoặc trang web thì có lẽ bạn đã sử dụng các cử chỉ như vuốt, để cuộn chế độ xem. Hoặc véo và tháo ngón tay cái và ngón trỏ của bạn để phóng to hoặc thu nhỏ.

## Trong bài học này, tôi sẽ bắt đầu bằng cách thảo luận về MotionEvents. Android sử dụng lớp này để thể hiện sự chuyển động trong các thiết bị đầu vào khác nhau. Những thứ như chuột, bi xoay và phổ biến nhất là ngón tay của bạn.

## Tiếp theo, tôi sẽ thảo luận về cách Android thực hiện các sự kiện chuyển động này và đưa chúng đến chế độ xem và các đối tượng khác để ứng dụng của bạn có thể phản hồi với chúng và cuối cùng tôi sẽ kết thúc cuộc thảo luận về cách Android nhận ra các kiểu chuyển động hoặc cử chỉ phức tạp. Những thứ như pinch để phóng to mà tôi đã đề cập trước đó.

## Android sử dụng lớp MotionEvent để thể hiện các chuyển động trong một thiết bị đầu vào như bút, bóng theo dõi, chuột hoặc ngón tay của bạn.

## Một sự kiện phong trào cá nhân có chứa một số thông tin. Nó có một mã hành động, cho biết loại chuyển động đã xảy ra. Nó cũng chứa nhiều loại dữ liệu liên quan đến chuyển động đó, ví dụ, nó có thông tin về thời gian xảy ra sự kiện, sự kiện đến từ thiết bị nào, địa điểm của sự kiện. Và nếu thích hợp, thiết bị đã được ép cứng như thế nào và hơn thế nữa. Và như tôi vừa nói, thông tin này sẽ thay đổi tùy thuộc vào loại thiết bị đầu vào có liên quan. Trong phần còn lại của bài học, tôi sẽ tập trung đặc biệt vào các sự kiện chạm ngón tay được đọc bằng cách nhấn vào màn hình cảm ứng.

## Nhiều thiết bị màn hình cảm ứng ngày nay là thiết bị MultiTouch. Điều đó có nghĩa là họ có thể đăng ký và theo dõi nhiều lần chạm cùng một lúc. Trong Android, các thiết bị MultiTouch phát ra một dấu vết chuyển động trên mỗi nguồn cảm ứng. Và mỗi nguồn cảm ứng này được gọi là một con trỏ.

## Khi Android gặp một con trỏ mới, nó sẽ tạo một ID duy nhất không đổi cho đến khi con trỏ đó hoạt động. Trong một số trường hợp, Android sẽ nhóm nhiều con trỏ trong một sự kiện chuyển động. Và trong trường hợp đó, mỗi con trỏ trong sự kiện chuyển động có thể được truy cập bởi chỉ mục của nó, nhưng lưu ý rằng chỉ mục đó không giống với ID con trỏ. ID con trỏ không đổi miễn là con trỏ hoạt động. Tuy nhiên, chỉ mục trong đó dữ liệu con trỏ được lưu trữ. Có thể không.

## Vì vậy, hãy nói về các sự kiện chuyển động chi tiết hơn. Bây giờ, như tôi đã nói các sự kiện chuyển động trước đó có mã hành động. Khi một cử chỉ bắt đầu, các sự kiện chuyển động sẽ được tạo và chúng sẽ chứa một số mã hành động sau. Hành động xuống, chỉ ra rằng một ngón tay đầu tiên đã, đã bắt đầu chạm vào màn hình.

## Con trỏ hành động xuống, điều đó có nghĩa là chúng ta đã có một hành động xuống và bây giờ chúng ta có một ngón tay khác đã bắt đầu chạm vào màn hình.

## Con trỏ hành động lên, chúng ta đã có một hành động\_pulum và một hành động xuống. Và bây giờ một trong những ngón tay đã ngừng chạm vào màn hình. Action\_move, một số ngón tay chạm vào màn hình, đã thay đổi vị trí của chúng. Action\_up, ngón tay cuối cùng chạm vào màn hình giờ đã ngừng chạm vào nó.

## Và hành động, một cái gì đó đã sớm hủy bỏ cử chỉ hiện tại.

## Trong khi một cử chỉ đang diễn ra, Android sẽ cố gắng đảm bảo rằng các sự kiện chuyển động của nó tuân theo các quy tắc sau. Chạm sẽ đi xuống một lần.

## Họ sẽ di chuyển như một nhóm, vì vậy một sự kiện chuyển động duy nhất có thể đề cập đến nhiều con trỏ. Và họ sẽ đưa ra từng cái một hoặc bị hủy bỏ.

## Khi bạn cần xử lý các sự kiện chuyển động, bạn có thể sử dụng một số phương pháp sau: getActionMasked. Trả về mã hành động liên quan đến sự kiện chuyển động.

## GetAction Index, trả về chỉ mục của con trỏ được liên kết với mã hành động này. Ví dụ: nếu mã hành động là con trỏ hành động xuống, thì bạn có thể sử dụng phương thức này để tìm chỉ mục của con trỏ cụ thể vừa chạm xuống.

## Nhận ID con trỏ. Đưa ra một chỉ mục, phương thức này trả về ID ổn định của con trỏ được liên kết với chỉ mục đó.

## Lấy con trỏ đếm. Trả về số lượng con trỏ liên quan đến sự kiện chuyển động. GetX và getY, trả về tọa độ x và y của con trỏ được lưu trữ tại chỉ mục hiện tại.

## Và findPulum Index, trả về chỉ mục được liên kết với một ID con trỏ đã cho.

## Khi một lần chạm xảy ra trên một chế độ xem, Android sẽ tạo ra một sự kiện chuyển động và sau đó cố gắng phân phối sự kiện đó đến nhiều đối tượng khác nhau, một trong số đó là chính chế độ xem đó.

## Android cung cấp sự kiện chuyển động, thông qua phương thức sự kiện chạm.

## Phương pháp này có thể xử lý sự kiện chuyển động và nên kết thúc bằng cách trả về true, nếu sự kiện chuyển động đã được tiêu thụ và sai nếu không.

## Các đối tượng quan tâm đến việc nhận các sự kiện chuyển động xảy ra trên một chế độ xem nhất định có thể đăng ký để nhận các sự kiện đó. Bằng cách thực hiện các giao diện nghe trên giao diện người nghe.

## Và bằng cách đăng ký đối tượng với phương thức view.setOnTouchListener. Phương thức onTouch của người nghe sau đó sẽ được gọi khi xảy ra một sự kiện như nhấn, thả hoặc kéo. Phương pháp này sẽ được gọi trước khi chế độ xem được chạm có cơ hội xử lý cảm ứng đó. Và một lần nữa, onTouch sẽ trả về true nếu nó tiêu thụ sự kiện chuyển động hoặc sai nếu không.

## Trong trường hợp đơn giản nhất, bạn có thể xử lý từng sự kiện chạm một cách độc lập. Nhưng các ứng dụng thường cần hoặc muốn xử lý nhiều lần chạm là một phần của cử chỉ phức tạp hơn. Để làm điều này, mã của bạn sẽ cần xác định và xử lý các kết hợp chạm cụ thể. Ví dụ, một chạm hai lần sẽ liên quan đến một hành động xuống.

## Và sau đó một hành động lên. Và sau đó một hành động khác xuống. Và cuối cùng, một hành động lên. Tất cả liên tiếp nhanh chóng. Để đưa ra một số ví dụ, giả sử bạn bắt đầu một cử chỉ bằng cách đặt một ngón tay xuống màn hình.

## Điều đó sẽ tạo ra một sự kiện xuống hành động. Và có thể gán id con trỏ bằng 0 cho con trỏ đó. Nếu bạn giữ ngón tay đó xuống và di chuyển nó trên màn hình, bạn có thể nhận được một số sự kiện di chuyển hành động liên quan đến id con trỏ.

## Giả sử bây giờ bạn đặt một ngón tay thứ hai xuống.

## Trong trường hợp đó, bạn sẽ nhận được một sự kiện action\_pulum\_down. Và con trỏ mới này có thể nhận được một ID nói về một. Nếu bạn giữ những ngón tay đó xuống và bạn di chuyển chúng, thì bạn có thể nhận được một số sự kiện action\_move liên quan đến số 0 và một của con trỏ ID.

## Và nếu bây giờ bạn nhấc ngón tay đầu tiên thì bạn sẽ nhận được một sự kiện hành động con trỏ lên liên kết với con trỏ số không. Và sau đó nếu cuối cùng bạn nhấc ngón tay cuối cùng, bạn sẽ nhận được một sự kiện hành động liên quan đến một con trỏ id.

## Trong ví dụ tiếp theo, chúng tôi sẽ bắt đầu như trước khi đặt ngón tay đầu tiên xuống. Di chuyển nó, đặt ngón tay thứ hai xuống và sau đó di chuyển những ngón tay đó một lần nữa, tuy nhiên lần này, chúng ta sẽ nhấc ngón tay thứ hai trước. Trong trường hợp này, chúng ta nhận được một hành động con trỏ lên hành động được liên kết với con trỏ ID một, và cuối cùng khi chúng ta nhấc ngón tay cuối cùng, chúng ta sẽ thực hiện hành động đó. Hành động liên quan đến con trỏ ID 0. Và trong ví dụ cuối cùng của chúng tôi, chúng tôi sẽ sử dụng ba ngón tay. Chúng tôi sẽ đặt ngón tay thứ nhất xuống, rồi ngón thứ hai và thứ ba. Và sau đó chúng tôi sẽ di chuyển các ngón tay và sau đó chúng tôi sẽ nâng chúng lên. Đầu tiên nâng ngón tay thứ hai. Sau đó, ngón tay đầu tiên và cuối cùng nâng ngón tay thứ ba.

## Ứng dụng ví dụ đầu tiên của chúng tôi trong bài học này được gọi là TouchIndicateTouchLocation.

## Ứng dụng này vẽ một vòng tròn bất cứ nơi nào người dùng chạm vào màn hình. Màu vòng tròn được chọn ngẫu nhiên. Và ứng dụng cũng sau đó vẽ lại vòng tròn, theo ngón tay của người dùng nếu nó di chuyển trên màn hình. Và cuối cùng, khi người dùng chạm vào màn hình ở nhiều vị trí. Kích thước của các vòng tròn được vẽ sẽ thay đổi để phản ánh số lần chạm hiện đang hoạt động. Chúng ta hãy xem ứng dụng này hoạt động. Vì vậy, đây là thiết bị của tôi.

## Và bây giờ tôi sẽ khởi động ứng dụng Touch Indicate Touch Location.

## Khi nó khởi động, màn hình trống vì tôi không chạm vào màn hình ngay bây giờ.

## Bây giờ, tôi sẽ đặt ngón tay lên màn hình. Và điều đó khiến một vòng tròn duy nhất được vẽ tại nơi tôi chạm vào màn hình. Khi tôi trượt ngón tay dọc theo màn hình, bạn có thể thấy rằng vòng tròn được vẽ lại để theo dõi chuyển động ngón tay của tôi. Bây giờ, tôi sẽ đặt một ngón tay thứ hai lên màn hình và điều đó khiến một vòng tròn thứ hai được vẽ dưới ngón tay đó. Và như bạn có thể thấy kích thước của hai vòng tròn hiện chỉ bằng một nửa so với những gì bạn đã thấy khi chỉ có một vòng tròn duy nhất. Bây giờ, ở đây tôi sẽ lấy đi ngón tay thứ hai và vòng tròn thứ nhất trở về kích thước ban đầu. Bây giờ tôi sẽ đặt ngón tay thứ hai trở lại. Và một lần nữa hai vòng tròn xuất hiện ở một nửa kích thước.Và tôi có thể kéo hai ngón tay này quanh màn hình, và các vòng tròn sẽ theo chuyển động của tôi.

## Cuối cùng, ở đây tôi sẽ đặt thêm ngón tay, bốn, sáu, tám, mười. Bây giờ tôi đã ra khỏi ngón tay.

## Vì vậy, bây giờ tôi sẽ di chuyển chúng xung quanh. Và bây giờ tôi sẽ bắt đầu lấy đi một số ngón tay. Tám, sáu, bốn, hai, một. Bây giờ hãy xem mã nguồn cho ứng dụng này.

## Đây là ứng dụng mở trong IDE.

## Bây giờ tôi sẽ mở hoạt động chính. Và cos này, mã này trước tiên tạo ra một nhóm các khung nhìn tùy chỉnh được gọi là các khung nhìn đánh dấu.

## Chế độ xem đánh dấu sẽ được sử dụng để đánh dấu vị trí của một lần chạm.

## Tiếp theo, mã xác định một bộ giữ các chế độ xem đánh dấu hiện đang hiển thị trên màn hình.

## Và bây giờ trong onCreate, mã có bố cục khung đại diện cho khung nhìn chính của hoạt động này.

## Và sau đó, nó tạo ra một trình nghe cảm ứng và đặt nó là người nhận của người nghe đó khi gọi lại liên lạc.

## Hãy xem phương pháp đó.

## Khi người dùng chạm vào màn hình, phương thức cảm ứng của người nghe này được gọi.

## Phương pháp đó bắt đầu bằng cách kiểm tra mã hành động cho sự kiện chuyển động mới.

## Nếu mã hành động là hành động xuống hoặc con trỏ hành động xuống, thì đã có một liên lạc mới. Vì vậy, mã tạo và hiển thị một giao diện đánh dấu mới.

## Và mã thực hiện điều này bằng cách ghi lại ID con trỏ và chỉ mục con trỏ cho sự kiện này.

## Sau đó, nó lấy chế độ xem đánh dấu từ danh sách không hoạt động và sau đó thêm chế độ xem đánh dấu đó vào bộ hoạt động, sử dụng ID con trỏ của nó làm khóa cho chế độ xem này.

## Tiếp theo, nó đặt vị trí cho chế độ xem đánh dấu này và sau đó nó cập nhật tổng số lần chạm cho mỗi chế độ xem đánh dấu hiện có thể nhìn thấy.

## Và sau đó, nó thêm chế độ xem đánh dấu vào chế độ xem chính của hoạt động. Bây giờ, nếu thay vào đó, mã hành động đã hành động. Hoặc hành động con trỏ lên, điều đó có nghĩa là một ngón tay đã được nhấc ra khỏi màn hình.

## Vì vậy, mã về cơ bản hoàn tác những gì chúng ta vừa nói xong.

## Bây giờ như trước khi mã bắt đầu bằng cách ghi ID con trỏ và chỉ mục con trỏ cho sự kiện này.

## Sau đó, nó sẽ xóa chế độ xem đánh dấu được liên kết với ngón tay vừa được nhấc khỏi bộ hoạt động.

## Sau đó, nó thêm chế độ xem đánh dấu trở lại danh sách không hoạt động và tiếp theo nó cập nhật tổng số lần chạm cho mỗi chế độ xem đánh dấu hiện có thể nhìn thấy.

## Và sau đó nó xóa chế độ xem đánh dấu khỏi chế độ xem chính của hoạt động.

## Và cuối cùng, nếu mã hành động được di chuyển, thì mã sẽ điều chỉnh vị trí của các khung nhìn đánh dấu bị ảnh hưởng và bắt đầu vẽ lại. Mã này thực hiện điều này bằng cách lặp qua các con trỏ trong sự kiện chuyển động.

## Đối với mỗi cái, nó nhận được khung nhìn đánh dấu cho con trỏ đó, và sau đó nó kiểm tra xem con trỏ có đi được một khoảng cách tối thiểu không. Nếu vậy, nó đặt một vị trí mới cho chế độ xem đánh dấu đó. Và sau đó các cuộc gọi không hợp lệ trên chế độ xem đánh dấu, cho biết chế độ xem đánh dấu muốn được vẽ lại.

## Touch and Gestures - Part 2

## Android cung cấp một lớp có tên GestureDetector mà các ứng dụng có thể sử dụng để nhận ra các cử chỉ chạm thông thường.

## Lớp này có thể nhận ra các cử chỉ như một lần chạm được xác nhận, một lần chạm hai lần, về cơ bản là hai lần chạm liên tiếp và một cú ném, đó là một lần nhấn theo sau là một động tác kéo và nhả có tốc độ khá cao.

## Để sử dụng GestureDetector, hoạt động của bạn sẽ phải tạo một thể hiện của lớp GestureDetector và sẽ phải cung cấp cho nó một đối tượng thực hiện giao diện GestureDetector.OnGestureListener.

## Và sau đó, hoạt động sẽ cần ghi đè phương thức onTouchEvent của nó,

## đó là phương thức được gọi khi hoạt động nhận được một sự kiện chạm. Và phương thức này sau đó sẽ ủy thác sự kiện chuyển động cho phương thức onTouchEvent của trình phát hiện cử chỉ.

## Hãy xem xét một ứng dụng ví dụ sử dụng GestureDetector để nhận ra cử chỉ fling.

## Ứng dụng này được gọi là TouchGestureViewFlipper và khi khởi động, nó hiển thị một TextView hiển thị một số. Nếu người dùng thực hiện cử chỉ lật từ phải sang trái, thì TextView sẽ cuộn khỏi phía bên trái màn hình.

## Và trong khi thực hiện, một TextView mới hiển thị một số mới sẽ cuộn phía sau nó từ bên phải. Hãy xem ứng dụng đó hoạt động.

## Bây giờ tôi sẽ khởi động ứng dụng TouchGestureViewFlipper. Khi khởi động, màn hình hiển thị TextView hiển thị số 0. Bây giờ, nếu tôi thực hiện cử chỉ vẩy, nghĩa là, nếu tôi nhấn và giữ chế độ xem, sau đó nhanh chóng vuốt về phía bên trái màn hình và cuối cùng, nhấc ngón tay ra khỏi màn hình, sau đó chúng ta sẽ thấy hình ảnh động mà tôi đã đề cập trước đó. Hãy để tôi làm điều đó bây giờ. Và như bạn có thể thấy, TextView với số 0 trượt khỏi màn hình, đi về phía bên trái và TextView mới hiển thị số 1 trượt trong màn hình từ bên phải.

## Hãy để tôi làm điều đó một vài lần nữa. Và lưu ý rằng cử chỉ này chỉ hoạt động nếu tôi vuốt từ phải sang trái. Nếu tôi thử nó theo hướng khác, sẽ không có gì xảy ra. Chúng ta hãy xem mã nguồn cho ứng dụng này. Vì vậy, đây là ứng dụng mở trong IDE.

## Bây giờ tôi sẽ mở hoạt động chính.

## Trước hết, ứng dụng này sử dụng lớp ViewFlipper để xử lý các hình ảnh động. Bây giờ, tôi sẽ không đi sâu vào vấn đề đó ở đây, nhưng hãy thoải mái nghiên cứu mã sau khi chúng tôi hoàn thành phân đoạn này.

## Bây giờ, hãy tập trung vào cách ứng dụng này phát hiện cử chỉ fling.

## Vì vậy, trong onCreate, bạn có thể thấy rằng mã tạo ra một GestureDetector mới.

## Và trong hàm tạo cho đối tượng này, mã truyền vào một SimpleOnGestureListener mới.

## Và đối tượng này định nghĩa một phương thức onFling.

## Khi GestureDetector phát hiện cử chỉ fling, phương thức này sẽ được gọi. Bây giờ, chúng ta sẽ quay lại với phương pháp này trong vài giây. Ngay bây giờ, hãy xem phương thức onTouchEvent cho hoạt động này. Phương thức này được gọi khi xảy ra sự kiện chạm và không có chế độ xem nào trong hoạt động xử lý nó.

## Khi phương thức này được gọi, nó sẽ chỉ ủy thác cuộc gọi đến GestureDetector.

## Nếu GestureDetector cuối cùng quyết định rằng nó đã thấy một cử chỉ fling hoàn chỉnh, phương thức onFling mà tôi vừa chỉ ra cho bạn sẽ được gọi. Và phương thức onFling này nhận được một tham số. Trong trường hợp này, nó được gọi là velocityX, cho biết tốc độ vuốt được thực hiện nhanh và theo hướng nào.

## Trong ví dụ này, nếu thao tác vuốt chuyển từ phải sang trái với tốc độ hơn mười pixel mỗi giây, thì mã sẽ gọi một phương thức gọi là switchLayoutStateTo, khiến hoạt hình của TextViews bắt đầu. Ví dụ, nếu vận tốc không đáp ứng tiêu chí đó, nếu đó là lực kéo chậm thay vì bay, hoặc nếu nó đi sai hướng, từ trái sang phải thay vì phải sang trái, thì cử chỉ bay sẽ bị bỏ qua.

## Để nhận ra các cử chỉ phức tạp hơn, bạn có thể sử dụng ứng dụng GestureBuilder của Android để tạo và sau đó lưu các cử chỉ tùy chỉnh. Ứng dụng này đi kèm với SDK. Khi chạy, bạn có thể sử dụng lớp GestureL Library để tải các cử chỉ tùy chỉnh của mình và để nhận ra khi nào người dùng thực hiện một trong những cử chỉ đó.

## Để thực hiện công việc này, bạn bao gồm một GestureOverlayView trong ứng dụng của bạn và chế độ xem này về cơ bản chặn các cử chỉ của người dùng. Và sau đó, nó gọi mã ứng dụng của bạn để xử lý các cử chỉ đó.

## Đây là một ảnh chụp màn hình của ứng dụng GestureBuilder.

## Như bạn có thể thấy, tôi đã tạo ra bốn cử chỉ tùy chỉnh. Tiếp theo, đó là một cú vuốt ngang từ trái sang phải. Không, trông giống như một, một x mà bạn thực hiện bằng cách sử dụng một nét. Trước đó, hoặc trước đó, là một cú vuốt ngang từ phải sang trái. Và vâng, trông giống như một dấu kiểm.

## Trên trình giả lập, GestureBuilder lưu các cử chỉ tùy chỉnh của bạn vào một tệp có tên / mnt / sdcard / cử chỉ. Để sử dụng những cử chỉ này, bạn sẽ cần sao chép tệp này vào thư mục res / raw của ứng dụng.

## Hãy xem ứng dụng TouchGestures. Ứng dụng này hiển thị một chế độ xem nhỏ với màu ứng cử viên cho toàn bộ nền ứng dụng.

## Màu nền cho toàn bộ ứng dụng ban đầu được đặt thành màu xám và người dùng có thể sử dụng bốn cử chỉ tùy chỉnh mà tôi đã trình bày trước đó để tương tác với ứng dụng này. Ví dụ: nếu người dùng thực hiện cử chỉ Tiếp theo, màu nền sẽ chuyển tiếp. Nếu người dùng thực hiện cử chỉ Trước đó, màu nền sẽ quay trở lại. Nếu người dùng thực hiện cử chỉ Có, ứng dụng sẽ đặt toàn bộ nền của ứng dụng thành màu hiện tại. Và nếu người dùng thực hiện Không có cử chỉ, màu nền của ứng dụng sẽ được đặt lại thành màu xám.

## Hãy xem ứng dụng đang chạy.

## Vì vậy, đây là thiết bị của tôi. Bây giờ tôi sẽ khởi động ứng dụng TouchGestures. Và khi khởi động, nền của ứng dụng thường có màu xám, nhưng có một hình vuông màu ở giữa màn hình.

## Nếu tôi vuốt màn hình từ trái sang phải, màu của hình vuông ở giữa sẽ thay đổi. Và nếu tôi làm lại, màu sắc sẽ thay đổi một lần nữa.

## Và tôi có thể quay lại màu trước bằng cách vuốt, lần này từ phải sang trái, thay vì trái sang phải.

## Nếu tôi quyết định rằng tôi thích màu hiện tại, tôi có thể thực hiện cử chỉ Có, như vậy.

## Và như bạn thấy, toàn bộ ứng dụng hiện có nền của màu đó.

## Nhưng nếu tôi đổi ý, tôi có thể thực hiện Không cử chỉ, như vậy.

## Và như bạn có thể thấy, nền của ứng dụng trở lại màu xám ban đầu. Hình vuông màu xuất hiện lại ở giữa bố cục.

## Và tôi có thể giữ nó, đưa ra những cử chỉ để nhìn vào màu sắc ứng cử viên mới.

## Chúng ta hãy xem mã nguồn cho ứng dụng này.

## Đây là ứng dụng mở trong IDE. Và bây giờ, tôi sẽ mở hoạt động chính. Và bạn nhận thấy rằng hoạt động này thực hiện giao diện OnGesturePerformedListener, có nghĩa là nó cung cấp một phương thức onGesturePerformed. Trong onCreate, mã được tham chiếu đến bố cục khung, nó lưu trữ trong một biến gọi là mFrame và đây là nơi xuất hiện màu nền ứng cử viên. (where the candidate background colors appear).

## Mã này cũng nhận được một tham chiếu đến một bố cục tương đối, mà nó lưu trữ trong một biến gọi là mLayout và đây là cách bố trí cho toàn bộ ứng dụng.

## Bây giờ, đây là mã đọc tệp cử chỉ từ thư mục res / raw, sử dụng phương thức fromRawResource của GestureL Library. Phương thức này trả về một đối tượng GestureL Library và sau đó mã tiếp tục gọi phương thức tải cho GestureL Library. Sau đó, mã tìm thấy GestureOverlayView, nằm trong bố cục và thêm hoạt động hiện tại dưới dạng trình nghe cho các cử chỉ bị chặn bởi GestureOverlayView. Khi GestureOverlayView phát hiện một cử chỉ, nó gọi phương thức onGesturePerformed được hiển thị ở đây.

## Và phương pháp này trước tiên gọi phương thức nhận dạng, phân tích cử chỉ được phát hiện và sau đó chấm điểm từng cử chỉ tùy chỉnh về mức độ cử chỉ được phát hiện giống với cử chỉ tùy chỉnh được ghi trong tệp cử chỉ. Và tiếp theo, mã được dự đoán xếp hạng cao nhất, và sau đó nếu dự đoán đó có điểm đủ cao, mã sẽ thực hiện hành động được liên kết với cử chỉ đó. Ví dụ: nếu cử chỉ là cử chỉ CÓ, thì mã sẽ đặt màu nền của bố cục thành màu ứng cử viên hiện tại.

## Multimedia - Part 1

## Thiết bị cầm tay cho phép người dùng tạo và tiêu thụ một lượng lớn nội dung đa phương tiện phong phú. Và nội dung này bao gồm nội dung âm thanh, như khi bạn nghe nhạc hoặc ghi lại giọng nói. Nội dung hình ảnh, như khi bạn chụp và xem ảnh. Và nội dung video, như khi bạn lấy và xem phim.

## Trong bài học này, chúng ta sẽ nói về các lớp đa phương tiện mà Android cung cấp. Và chúng ta sẽ tìm hiểu các API và các ứng dụng ví dụ phát âm thanh, cho phép bạn xem âm thanh, cho phép bạn ghi lại âm thanh và cuối cùng, cho phép bạn sử dụng máy ảnh để chụp ảnh. Vậy hãy bắt đầu.

## Android cung cấp một số lớp và khả năng để hỗ trợ mã hóa và giải mã các định dạng phương tiện phổ biến.

## Ứng dụng của bạn có thể sử dụng những thứ này để phát và ghi lại âm thanh, hình ảnh tĩnh và video.

## Một số lớp chúng ta sẽ nói đến hôm nay sẽ là lớp AudioManager và SoundPool, cho phép các ứng dụng phát hiệu ứng âm thanh và tệp âm thanh và kiểm soát phần cứng liên quan đến âm thanh của thiết bị, như loa và tai nghe không dây.

## Chúng ta sẽ nói về Nhạc chuông và nhạc chuông, và đây là những âm thanh bạn thường nghe khi có cuộc gọi điện thoại, khi nhận được thông báo và khi báo thức kêu.

## Và MediaPlayer, cho phép các ứng dụng phát các tệp âm thanh và video.

## MediaRecorder, cho phép các ứng dụng ghi lại âm thanh và video. Và cuối cùng, chúng ta sẽ kết thúc bằng cách nhìn vào lớp Camera, cho phép các ứng dụng điều khiển cứng, các camera phần cứng trên thiết bị.

## Lớp đầu tiên tôi sẽ nói đến là AudioManager. Lớp này quản lý các khả năng âm thanh cơ bản như thao tác âm lượng thiết bị, phát hiệu ứng âm thanh của hệ thống và thay đổi chế độ chuông của thiết bị.

## Các ứng dụng có được một tham chiếu đến AudioManager bằng cách gọi Context.getSystemsService, truyền vào giá trị, Context.AUDIO\_SERVICE.

## Và một khi nó có tham chiếu đến AudioManager, một ứng dụng có thể tải và sau đó phát hiệu ứng âm thanh, điều chỉnh âm lượng của thiết bị và điều khiển phần cứng thiết bị. Chẳng hạn, bằng cách tắt tiếng micrô hoặc bật tai nghe Bluetooth.

## Một lớp liên quan khác có thể được sử dụng để phát âm thanh phức tạp là lớp SoundPool. SoundPool đại diện cho một tập hợp các mẫu âm thanh hoặc luồng và SoundPool cho phép bạn trộn lẫn với nhau và phát nhiều mẫu cùng một lúc.

## Chúng ta hãy xem ứng dụng ví dụ đầu tiên của chúng tôi cho bài học này.

## Ứng dụng đó được gọi là AudioVideoAudioManager. Và khi chạy, ứng dụng này trình bày hai nút, được gắn nhãn Lên và Xuống, được sử dụng để tăng và giảm âm lượng của thiết bị.

## Ứng dụng cũng hiển thị một nút có nhãn Play, khi nhấn, sẽ phát một nút bật âm thanh ở mức âm lượng hiện tại.

## Hãy chạy ứng dụng ngay bây giờ. Vì vậy, đây là thiết bị của tôi. Và bây giờ, tôi sẽ khởi động ứng dụng AudioVideoAudioManager. Và như đã hứa, hiển thị ba nút, Lên, Xuống và Chơi. Ứng dụng này cũng hiển thị mức âm lượng hiện tại theo thang điểm từ 0 đến 10.

## Ngay bây giờ, mức âm lượng được đặt thành sáu.

## Hãy để tôi nhấn nút Play để bạn có thể nghe thấy âm thanh bật ra bong bóng.

## Ở đây đi. [SOUND] Và bây giờ, tôi sẽ nhấn nút Lên một vài lần để đạt âm lượng tối đa. Và bây giờ tôi sẽ nhấn nút Play một lần nữa. [SOUND] Bây giờ, tôi sẽ nhấn nút Xuống và nút Phát một vài lần và bạn sẽ nghe thấy tiếng bong bóng bật ở mức âm lượng ngày càng thấp. Ở đây đi. [SOUND] Hãy xem mã nguồn cho ứng dụng này.

## Đây là ứng dụng AudioVideoAudioManager mở trong IDE.

## Bây giờ tôi sẽ mở hoạt động chính. Ở đây, phương thức onCreate lấy tham chiếu đến AudioManager. Tiếp theo, nó bắt đầu thiết lập giao diện người dùng của nó.

## Đầu tiên, có TextView để hiển thị mức âm lượng hiện tại. Tiếp theo, có upButton và khi được nhấp, mã này sẽ phát âm thanh nhấp phím và sau đó cố gắng tăng mức âm lượng và cập nhật hiển thị mức. Tiếp theo, chúng ta thấy nút xuống. Và điều này gần giống như mã cho upButton, nhưng nó giảm hơn là tăng mức âm lượng. Tiếp theo, mã được tham chiếu đến playButton và sau đó vô hiệu hóa nút này. Sau đó, mã tạo ra một đối tượng SoundPool, nó sẽ sử dụng để phát âm thanh popping bong bóng. Các tham số cho đối tượng SoundPool này cho biết rằng nó sẽ chỉ có một luồng âm thanh và luồng đó được phát trên luồng nhạc âm thanh. Tiếp theo, mã tải âm thanh pop bong bóng. Tải âm thanh là một hoạt động không đồng bộ. Vì vậy, mã đặt một OnLoadCompleteListener, mã này sẽ được gọi khi âm thanh cuối cùng được tải.

## Và khi phương thức onLoadComplete được gọi là mã sẽ kiểm tra xem thao tác có thành công hay không. Và nếu vậy, nó cho phép playButton.

## Trước đó, mã đã vô hiệu hóa playButton. Và nó đã làm điều đó để đảm bảo rằng nó đã không thử phát âm thanh cho đến khi âm thanh được tải đúng cách.

## Tiếp theo, mã đặt một trình nghe trên playButton. Khi nhấn, mã này phát âm thanh bật bong bóng ở mức âm lượng hiện tại.

## Và sau đó, mã yêu cầu tập trung âm thanh, có nghĩa là về cơ bản nó muốn chịu trách nhiệm về âm thanh được phát bởi thiết bị.

## Ngoài ra còn có một số mã trong các phương thức onResume và onPause.

## Phương thức onResume bật loa ngoài của thiết bị và sau đó tải các hiệu ứng âm thanh của hệ thống, như hiệu ứng âm thanh nhấn phím mà chúng tôi đã sử dụng ở trên.

## Phương thức onPause bắt đầu bằng cách tải SoundPool và giải phóng tài nguyên của nó.

## Và sau đó nó tắt loa ngoài của thiết bị và giải phóng hiệu ứng âm thanh của hệ thống.

## Các lớp tiếp theo chúng ta sẽ nói đến là Nhạc chuông và Nhạc chuông. Nhạc chuông Trình quản lý cung cấp quyền truy cập vào các clip âm thanh mà bạn nghe thấy, ví dụ như khi có cuộc gọi điện thoại, khi bạn nhận được một tin nhắn email mới hoặc khi báo thức kêu. Và thông qua Trình tạo nhạc chuông, các ứng dụng có thể nhận và đặt nhiều nhạc chuông khác nhau và chúng có thể phát và dừng phát chúng. Chúng ta hãy xem một ứng dụng ví dụ sử dụng Trình điều khiển nhạc chuông.

## Bây giờ, ứng dụng này được gọi là AudioVideoRing toneManager. Và ứng dụng này trình bày ba nút, tương ứng được gắn nhãn Nhạc chuông, Thông báo và Báo thức. Nhấn một trong các nút này sẽ khiến nhạc chuông mặc định được liên kết phát.

## Thử một lần đi.

## Bây giờ, tôi sẽ khởi động ứng dụng AudioVideoRing toneManager. Như bạn có thể thấy, nó hiển thị ba nút, Nhạc chuông, Thông báo và Báo thức. Hãy để tôi nhấn một số các nút này.

## Trước tiên, hãy nghe nhạc chuông mặc định khi tôi nhấn nút Nhạc chuông. [SOUND] Bây giờ, hãy thử nhạc chuông thông báo mặc định. [SOUND] Và cuối cùng, hãy thử nhạc chuông báo thức mặc định. [SOUND] Sau khi phân đoạn kết thúc, hãy thử tải xuống, cài đặt và chạy ứng dụng này trên thiết bị của bạn. Và sau đó, đi đến ứng dụng Cài đặt của bạn và thay đổi nhạc chuông mặc định của bạn.

## Khi bạn chạy lại ứng dụng này, bạn sẽ thấy rằng nó phát nhạc chuông mới của bạn chứ không phải nhạc chuông cũ.

## Vì vậy, hãy nhìn vào mã nguồn cho ứng dụng này. Đây là ứng dụng AudioVideoRing toneManager mở trong IDE. Bây giờ, tôi sẽ mở hoạt động chính. Trong phương thức onCreate, mã tạo ra ba nút. Hãy xem nhạc chuôngButton làm ví dụ.

## Bây giờ, ở đây, chúng ta thấy rằng khi nhấp vào nút này, mã sử dụng Trình điều khiển nhạc chuông để lấy URI cho nhạc chuông chuông điện thoại mặc định.

## Tiếp theo, mã nhận được nhạc chuông được liên kết với URI đó bằng cách gọi phương thức RingstyleManager.getRingTone, truyền vào URI. Kết quả của tất cả điều này sau đó được chuyển vào một phương thức gọi là playRing tone.

## Hãy cuộn xuống phương pháp đó. Bây giờ, phương pháp này kiểm tra xem nhạc chuông hiện đang phát. Và nếu có, thì nhạc chuông đó sẽ bị dừng bằng cách gọi phương thức dừng của lớp Nhạc chuông. Nhạc chuông hiện tại sau đó được đặt trong biến mCienRing tone. Và nếu nhạc chuông hiện tại không có giá trị thì mã sẽ bắt đầu phát.

## Multimedia - Part 2

## Class MediaPlayer

## MediaPlayer kiểm soát phát lại các luồng và tệp âm thanh và video. Và điều này cho phép bạn có được âm thanh và video trong các ứng dụng của mình và để cho các ứng dụng và người dùng kiểm soát phát lại đó.

## Lớp này hoạt động theo một máy trạng thái phức tạp. Mà tôi sẽ không đi qua đây trong bài học này.

## Vì vậy, hãy xem trang web sau để biết thêm thông tin.

## Một số phương thức mà bạn có thể sử dụng khi bạn sử dụng MediaPlayer bao gồm :

## setDataSource(): Nó cho người chơi phương tiện truyền thông nào phát.

## Prepare(): khởi tạo trình phát phương tiện và tải các luồng cần thiết. prepare method là đồng bộ và thông thường bạn sẽ sử dụng nó khi nội dung phương tiện được lưu trữ trong tệp của bạn trên thiết bị. Bây giờ cũng có một phiên bản không đồng bộ của phương pháp này. Mà có thể được sử dụng, ví dụ, khi phương tiện truyền thông được truyền phát từ internet.

## Start(): Một phương pháp tạm dừng để dừng chơi. Phương thức tìm kiếm để di chuyển đến một vị trí cụ thể trong luồng.

## Stop(): Một phương pháp dừng để dừng chơi phương tiện truyền thông.

## Release(): giải phóng các tài nguyên được sử dụng bởi trình phát phương tiện hiện tại.

## Class VideoView

## Một lớp khác có thể được sử dụng để xem video là lớp VideoView. Lớp này là một lớp con của SurfaceView. Và trong nội bộ sử dụng trình phát phương tiện mà chúng ta vừa nói đến.

## Lớp này có thể tải video từ các nguồn khác nhau và nó bao gồm một số phương thức và điều khiển để giúp xem nội dung video dễ dàng hơn.

## Ứng dụng ví dụ tiếp theo của chúng tôi là AudioVideoVideoPlay. Và ứng dụng này hiển thị chế độ xem đơn giản cho phép người dùng phát tệp video. Trong trường hợp này, bộ phim là một đoạn phim từ bộ phim năm 1902, A Trip to The Moon của George Melies. Hãy xem nào.

## Và bây giờ, tôi sẽ khởi động ứng dụng AudioVideoVideoPlay.

## Và bây giờ chạm vào màn hình. Bạn có thể thấy một bộ điều khiển phát lại xuất hiện. Và bây giờ, tôi sẽ nhấn vào một hình tam giác duy nhất và sau đó video sẽ bắt đầu phát. Chúng ta đi đây.

## Chúng ta hãy xem mã nguồn cho ứng dụng này.

## Đây là ứng dụng phát video âm thanh mở trong IDE.

## Và bây giờ, tôi sẽ mở hoạt động chính.

## Trong onCreate, mã đầu tiên nhận được tham chiếu đến VideoView trong bố cục của hoạt động này.

## Tiếp theo, nó tạo một MediaContoder, đây là một khung nhìn chứa các điều khiển để điều khiển trình phát phương tiện. Đoạn mã tiếp tục bằng cách vô hiệu hóa phương tiện sau đó bằng cách đính kèm MediaContoder này vào VideoView. Với một cuộc gọi đến phương thức setMediaControll của VideoView.

## Tiếp theo mã xác định tệp phương tiện để phát, chuyển qua URI trỏ đến tệp được lưu trữ trong thư mục res / raw.

## Sau khi đặt mã trên VideoView. Bây giờ mã này sẽ được gọi khi phương tiện được tải và sẵn sàng để chơi. Và khi điều đó xảy ra, mã sẽ kích hoạt MediaContoder mà người dùng có thể bắt đầu phát phim.

## Và cuối cùng, trong phương thức onPause, mã sẽ tắt VideoView.

## Lớp tiếp theo sẽ thảo luận, là MediaRecorder. Bây giờ lớp này có thể được sử dụng để ghi cả âm thanh và video. Các lớp hoạt động theo các máy trạng thái. Mà bạn có thể đọc thêm về URL này.

## Bây giờ, một số phương thức MediaRecorder mà bạn có thể sẽ sử dụng bao gồm:

## setAudioSource(): đặt nguồn đầu vào cho micrô cho âm thanh hoặc camera cho video.

## setVideoSource(): đặt nguồn đầu vào cho micrô cho âm thanh hoặc camera cho video.

## setOutputFormat(): đặt định dạng đầu ra cho bản ghi. Ví dụ, mp4.

## Prepare(): trong đó sẵn sàng ghi âm để được chụp và mã hóa dữ liệu.

## Start(): bắt đầu quá trình ghi thực tế.

## Stop(): dừng quá trình ghi âm.

## Release(): trong đó phát hành các tài nguyên được tổ chức bởi máy ghi âm phương tiện này.

## AudioVideoAudioRecply

## Ứng dụng ví dụ tiếp theo của chúng tôi là AudioVideoAudioRecply. Bây giờ ứng dụng này ghi lại âm thanh từ người dùng và nó có thể phát lại âm thanh cho người dùng. Hãy sử dụng ứng dụng này để ghi lại giọng nói của tôi.

## Và bây giờ, tôi sẽ khởi động ứng dụng AudioVideoAudioRecext.

## Ứng dụng này hiển thị hai nút chuyển đổi. Một nhãn bắt đầu ghi và một nhãn bắt đầu phát lại.

## Khi tôi nhấn nút bắt đầu ghi, ứng dụng sẽ bắt đầu ghi. Các nút sẽ thay đổi để dừng ghi âm.

## Và nút phát lại sẽ bị vô hiệu hóa.

## Khi tôi nhấn nút bắt đầu ghi lại, quá trình ghi sẽ dừng lại. Các nút sẽ thay đổi trở lại và nút sẽ được kích hoạt lại.

## Hãy thử xem.

## Tôi sẽ nhấn nút bắt đầu ghi âm.

## Kiểm tra. Kiểm tra. Một hai ba. Kiểm tra.

## Và bây giờ tôi đã nhấn nút lưu lại và nút bắt đầu phát lại hiện được bật. Hãy để tôi nhấn cái đó ngay bây giờ.

## Kiểm tra, thử nghiệm, một, hai, ba, thử nghiệm.

## Và bây giờ nhấn nút đó một lần nữa, và chúng tôi trở lại nơi chúng tôi bắt đầu.

## Hãy xem mã nguồn của ứng dụng này.

## Đây là ứng dụng AudioVideoAudioRecext mở trong IDE.

## Bây giờ tôi sẽ mở hoạt động chính. Trong onCreate, mã đầu tiên nhận được tham chiếu đến hai nút chuyển đổi.

## Tiếp theo, nó được thiết lập trênCheckedChangeListener trên mỗi nút chuyển đổi.

## Mã này được gọi khi trạng thái kiểm tra của nút chuyển đổi thay đổi.

## Hãy nhìn vào nút chuyển đổi đầu tiên, đó là nút ghi âm.

## Khi trạng thái kiểm tra của nút này thay đổi, hãy nói từ tắt sang bật ,.

## Mã này trước tiên sẽ vô hiệu hóa nút phát và sau đó sẽ gọi phương thức onRecordPression

## Đầu tiên, nó thay đổi trạng thái kích hoạt của nút ghi, vô hiệu hóa nếu người dùng muốn bắt đầu phát lại. Hoặc kích hoạt nó nếu người dùng muốn dừng phát lại.

## Sau đó, nó gọi phương thức onPlayPression. Trước tiên hãy xem phương thức onRecordPression. Như bạn có thể thấy, Phương thức này lấy Boolean làm tham số, được gọi sẽ bắt đầu ghi.

## Nếu nên bắt đầu ghi là đúng ,. Sau đó, mã gọi phương thức startRecext. Nếu không, nó gọi phương thức stopRecext.

## Phương thức startRecext trước tiên tạo MediaRecorder mới và sau đó đặt nguồn của nó là micrô.

## Sau đó, nó đặt định dạng đầu ra.

## Và sau đó tập tin đầu ra nơi ghi âm sẽ được lưu. Và sau đó nó đặt bộ mã hóa cho tệp âm thanh. Bây giờ tiếp tục, các cuộc gọi mã đã sẵn sàng để ghi âm sẵn sàng. Và cuối cùng, nó gọi phương thức start để bắt đầu ghi.

## Phương thức stopRecext thay vào đó dừng MediaRecorder và sau đó giải phóng tài nguyên của nó.

## Nếu người dùng thay vào đó đã nhấn nút trênPlayPression sẽ được gọi.

## Nếu nút được chọn, thì tham số, sẽ bắt đầu phát, sẽ đúng. Và nếu vậy, thì phương thức startPlay được gọi. Mặt khác, phương thức stopPlay được gọi. Phương thức startPlay bắt đầu bằng cách tạo MediaPlayer.

## Và sau đó theo dõi bởi dữ liệu nguồn của nó.

## Sau đó, bằng cách gọi chuẩn bị trên MediaPlayer. Và sau đó, bằng cách gọi phương thức bắt đầu.

## Phương thức stopPlay sẽ dừng trình phát phương tiện sau đó giải phóng tài nguyên của trình phát đa phương tiện.

## Class camera

## Lớp cuối cùng chúng ta đang nói đến trong bài học này là class camera. Lớp này cho phép các ứng dụng truy cập dịch vụ camera, mã cấp thấp trên thiết bị của bạn.

## Bây giờ, thông qua lớp này, ứng dụng của bạn có thể quản lý các cài đặt để chụp ảnh.

## Stop/start chức năng xem trước, cho phép bạn sử dụng các thiết bị như một loại công cụ tìm chế độ xem camera. Và quan trọng nhất, nó cho phép bạn chụp ảnh và quay video.

## Để sử dụng các tính năng của máy ảnh, bạn sẽ cần đặt một số quyền và tính năng. Bạn sẽ cần ít nhất là bao gồm thẻ tính năng người dùng trong tệp tệp kê khai Android của bạn. Điều đó chỉ định sự cần thiết của một máy ảnh.

## Và bạn có thể muốn chỉ định rằng các tính năng phụ ứng dụng của bạn như tự động lấy nét hoặc đèn flash.

## Mặc dù bạn có thể dễ dàng sử dụng máy ảnh để chụp ảnh. Bạn có thể muốn thêm một số tính năng cho một ứng dụng máy ảnh truyền thống. Hoặc bạn có thể muốn sử dụng máy ảnh cho các mục đích khác. Trong trường hợp này, bạn có thể làm theo các bước sau. Đầu tiên, bạn sẽ có được một ví dụ máy ảnh. Tiếp theo, bạn có thể thiết lập mọi thứ bạn cần.

## Sau đó, bạn sẽ muốn thiết lập màn hình xem trước để người dùng có thể thấy những gì camera nhìn thấy.

## Tiếp theo, bạn sẽ bắt đầu xem trước và bạn sẽ tiếp tục chạy cho đến khi người dùng chụp ảnh.

## Và người dùng chụp ảnh, ứng dụng của bạn sẽ nhận được quá trình hình ảnh. Và cuối cùng, ứng dụng của bạn sẽ phát hành camera, để các ứng dụng khác có thể truy cập vào nó. Ứng dụng ví dụ cuối cùng cho bài học này, được gọi là AudioVideoCamera. Ứng dụng này chụp ảnh tĩnh, sử dụng màn hình của thiết bị làm kính ngắm của máy ảnh. Thử một lần đi.

## Vì vậy, đây là thiết bị của tôi và bây giờ tôi sẽ khởi động ứng dụng AudioVideoCamera.

## Như bạn có thể thấy ứng dụng hiện có thể nhìn thấy qua ống kính của máy ảnh. Và nếu bạn di chuyển máy ảnh, hình ảnh sẽ thay đổi.

## Nếu người dùng hài lòng với hình ảnh thì họ có thể chỉ cần chạm vào màn hình để chụp ảnh.

## Và khi anh ấy hoặc cô ấy như vậy, máy ảnh sẽ chụp ảnh sau đó đóng băng cửa sổ trong khoảng hai giây. Để người dùng có thể nhìn thấy bức ảnh họ vừa chụp. Hãy để tôi làm điều đó. Tôi sẽ chạm vào màn hình để chụp ảnh. Và bây giờ xem trước trong hai giây. Và bây giờ máy ảnh đã sẵn sàng để chụp một bức ảnh khác.

## Hãy xem mã nguồn của ứng dụng này.

## Đây là ứng dụng AudioVideoCamera mở trong ID E.

## Bây giờ, tôi sẽ mở hoạt động chính.

## Và cho phép cuộn đến phương thức onCreate.

## Và một trong những điều chúng ta thấy ở đây là mã gọi phương thức getCamera để có được một tham chiếu đến đối tượng máy ảnh. Hãy cuộn xuống phương pháp đó.

## Phương thức này gọi các lớp camera mở phương thức. Mà quay trở lại máy ảnh mặt sau đầu tiên trên thiết bị này.

## Nếu thiết bị của bạn có nhiều camera, bạn có thể sử dụng các phiên bản khác của phương pháp mở để có được các camera cụ thể.

## Bây giờ cuộn lại lên onCreate. Mã bây giờ thiết lập một trình nghe cảm ứng trên giao diện chính. Và khi người dùng chạm vào màn hình, người nghe này sẽ được gọi phương pháp onTouch.

## Và phương pháp này sẽ đưa các máy ảnh để chụp ảnh. Bây giờ chúng ta sẽ quay lại phương pháp này sau vài giây.

## Tiếp theo, mã thiết lập SurfaceView được sử dụng để hiển thị bản xem trước,

## trong đó cho người dùng thấy những gì máy ảnh hiện đang nhìn thấy.

## Và những bước này chỉ là những gì chúng ta đã nói trong bài học trước về đồ họa. Đầu tiên, mã lấy SurfaceHolder cho SurfaceView. Và sau đó, nó thêm một đối tượng Callback vào SurfaceHolder.

## Và đối tượng Callback được định nghĩa dưới đây. Hãy cuộn xuống nó.

## Bây giờ bạn nhớ giao diện SurfaceHolder.Callback. Nó tìm thấy ba phương pháp. SurfaceCreated, SurfaceChanged, và SurfaceDestroyed. Phương thức SurfaceCreated bắt đầu với SurfaceHolder mà máy ảnh sẽ hiển thị bản xem trước của nó. Và sau đó, mã bắt đầu xem trước của máy ảnh.

## Khi bề mặt thay đổi định dạng cấu trúc vàng, phương thức SurfaceChanged được gọi. Và phương pháp này vô hiệu hóa các phím trên bố cục và sau đó dừng xem trước camera. Và tiếp theo, mã thay đổi các thông số camera. Trong trường hợp này, mã tìm thấy một kích thước phù hợp cho bản xem trước của máy ảnh.

## Và sau đó đặt kích thước xem trước.

## Và sau đó, hãy quay trở lại với máy ảnh.

## Bây giờ các tham số được đặt, mã khởi động lại phần xem trước. Bằng cách gọi phương thức startPreview. Và cuối cùng, mã kích hoạt lại các phím trên bố cục.

## Vì vậy, chúng tôi đã tiến hành thiết lập quản lý màn hình xem trước, hãy quay lại và xem việc chụp ảnh thực tế.

## Vì vậy, cuộn trở lại lên onTouchListener.

## Khi người dùng khóa, màn hình sẽ hiển thị phương thức TakePicture.

## Trong phương thức này mã ở đây

## Một là ShutterCallback và hai là CameraCallback.

## ShutterCallback được gọi vào khoảng thời gian người dùng chụp ảnh. Về cơ bản máy ảnh đang chăm sóc máy ảnh.

## CameraCallback đã được sử dụng sau ảnh khi hình ảnh nén có sẵn. Khi điều này xảy ra, phương thức CameraCallbacks onPictureTaken được gọi.

## Trong ví dụ này, mã chỉ cần ngủ trong hai giây và khởi động lại phần xem trước.

## Và bạn có thể nhận thấy rằng ứng dụng cụ thể này không thực sự lưu hình ảnh. Nhưng bạn sẽ không muốn làm điều đó nếu vậy, bạn thường làm điều đó ngay tại đây trong phương pháp này. Phương pháp cuối cùng tôi đang nói đến là onPause. Tại đây, mã vô hiệu hóa các phím trên màn hình, tắt chế độ xem trước rồi nhả camera để các ứng dụng khác có thể sử dụng nó.

## Programming Cloud Services for Android Handheld Systems

## Vậy chính xác thì Cloud Service là gì? làm thế nào để chúng ta phân biệt và hiểu làm thế nào từ các ứng dụng điển hình mà chúng ta đã sử dụng để viết.

## Vì vậy, hãy xem một ví dụ và hy vọng Hiểu rõ hơn về Dịch vụ đám mây là gì và phạm vi của cuộc đua này là gì và những thách thức mà chúng ta sẽ gặp phải khi xây dựng các dịch vụ đó. Vì vậy, hãy giả sử rằng chúng tôi đang viết cho ứng dụng Java. Và những gì ứng dụng Java cho phép chúng ta làm thì chúng ta có thể chạy một dòng nó sẽ đi và lưu video trên hệ thống tệp bên dưới. Và nó không chỉ đi và lưu video trong hệ thống tệp cơ bản mà còn đưa nó vào cơ sở dữ liệu để chúng tôi có bản ghi tất cả các video mà chúng tôi đã nhập vào hệ thống của mình. Vì vậy, đây là dịch vụ của chúng tôi trong trường hợp hoặc ứng dụng này, mà chúng tôi sẽ xây dựng. Và chúng tôi sẽ cho rằng không có gì đặc biệt về điều này. Nó chỉ là một ứng dụng Java.

## 1. Cách viết

## Chà, nếu chúng ta sẽ thực hiện điều này ở đây, một trong những điều có lẽ chúng ta sẽ muốn làm, nếu chúng ta viết điều này bằng Java, chúng ta sẽ có một Main method( phương thức chính ).

## Public static void main(String args []){

## //look for flags

## //look for video part

## .

## .

## .

## System.out.println(“…”);

## }

## Vì vậy, chúng ta Public static void main và có một số đối số và chúng ta có thể ra ngoài và đánh vần những gì chúng là args . Và sau đó, trong phương thức chính đó, chúng ta có thể đi và làm những việc như, bạn biết đấy, hãy tìm những lá cờ mà chúng ta đang vượt qua dòng lệnh. Vì vậy, chúng tôi có thể tìm kiếm các cờ, chúng tôi có thể tìm tệp mà họ muốn thêm, video. Vì vậy, hãy tìm video và chúng ta có thể có một loạt các thứ khác. Nhưng điều quan trọng là, chúng tôi đã có phương pháp này, một phần của ứng dụng này. Và mỗi lần, trong ứng dụng, họ sẽ làm việc với ứng dụng của chúng tôi. Vì vậy, những gì chúng ta có là một dòng lệnh đang được truyền vào ứng dụng của chúng ta. Vì vậy, mỗi khi người dùng đi và gọi thứ này từ dòng lệnh, họ thực sự giỏi về nó. Và thuế mà họ nhập vào phương thức Public static void main. Nơi chúng tôi có thể bắt đầu ứng dụng của mình, hãy hành động dựa trên dữ liệu mà họ đã chuyển trong cơ sở dữ liệu hoặc làm việc khác. Chúng tôi cũng có thể muốn phát hành đơn đặt hàng cho ứng dụng của mình, rằng chúng tôi sẽ truy vấn thông tin chúng tôi đã lưu trữ. Hoặc được xử lý dựa trên thông tin mà chúng tôi đã cảm nhận được ứng dụng. Vì vậy, một trong những điều chúng ta có thể phải thực hiện phản hồi cho người dùng đang gửi lệnh đến ứng dụng. Vì vậy, chúng tôi có thể có thứ gì đó trong tay đang in ra phản hồi bằng cách sử dụng dòng system.out.print cho người dùng để người dùng có thể thấy kết quả của các lệnh họ đang phát hành nhìn thấy thông tin từ cơ sở dữ liệu người dùng đang truy vấn.

## Vì vậy, chúng tôi không chỉ có một tình huống mà chúng tôi đang gửi lệnh và yêu cầu. Ứng dụng đi và lưu trữ dữ liệu, nhưng chúng tôi cũng có những tình huống cần đến và cung cấp phản hồi cho người dùng về sự thất bại vàng thành công của các lệnh đó và bất kỳ dữ liệu nào liên quan đến chúng.

## Vì vậy, ví dụ: nếu người dùng đưa ra một truy vấn để xem tất cả các video được tạo trong một khoảng thời gian nhất định, có lẽ chúng tôi sẽ muốn in lại dòng lệnh. Tương tự, nếu người dùng đưa ra một truy vấn không đúng định dạng, có lẽ chúng ta sẽ mong đợi ứng dụng được in trên thông báo lỗi dòng lệnh cho biết người dùng đã làm gì sai giúp họ tìm ra cách thành công Hoặc nếu ứng dụng đi và xử lý lệnh nó có một lỗi trong chương trình và nó bị lỗi hoặc làm hỏng nó, chúng tôi hy vọng sẽ nhận được một số phản hồi rằng ứng dụng bị sập. Vì vậy, những gì chúng ta sẽ thấy là kiến trúc cơ bản của các lệnh được cảm nhận cho một ứng dụng. Xử lý dữ liệu ở một số định dạng, lưu trữ dữ liệu hoặc thực hiện một số phân tích cung cấp khả năng thất bại vàng tốt nhất của các lệnh đó và kết quả thực hiện các lệnh đó. Đó là kiến trúc cơ bản mà chúng ta sẽ làm việc trong suốt lớp này.

## 2. khác biệt

## Sự khác biệt lớn là, khi chúng ta đi vào mô hình cloud services , chúng ta sẽ ở đâu để gửi đơn đặt hàng ứng dụng từ một thiết bị di động không ngồi trên cùng một ứng dụng, cùng một máy chủ đó là ứng dụng đó. Vì vậy, thiết bị di động này sẽ gửi các lệnh này từ xa qua internet đến ứng dụng này sau đó một số ứng dụng sẽ xuất hiện trong Cloud. Và nhận các lệnh đó, xử lý dữ liệu thông qua một số lưu trữ dữ liệu. Lưu thông tin phải gửi từ xa trở lại thiết bị di động kết quả của các lệnh đó. Vì vậy, chúng tôi có sự tương tác này dựa trên ứng dụng và kết quả quay trở lại mà chúng tôi sẽ làm việc với. Chúng tôi sẽ gửi dữ liệu từ thiết bị di động, là ứng dụng khách đến ứng dụng. Nó sẽ xử lý thông tin đó. Và trong trường hợp này, ứng dụng sẽ trở thành một máy chủ. Và nó sẽ sống trong Đám mây(Cloud) và nó sẽ gửi kết quả trở lại thiết bị di động, để nó có thể biết chuyện gì đã xảy ra. Vì vậy, nó có thể đang gửi lại kết quả mà nói, đây là kết quả của việc chạy lệnh của bạn. Có thể bạn không thể nói rằng, bạn biết đấy, đã không xây dựng lệnh của bạn chỉ nói,máy chủ đã có một vấn đề. Đây là vấn đề và nó đã sụp đổ. Và đó là kiến trúc mà chúng ta sẽ xây dựng thông qua lớp này. Đây có phải là kiến trúc phân tán từ xa nơi khách hàng di động, thiết bị di động của bạn trên Android kết nối internet với dịch vụ hoặc máy chủ sống trong Đám mây. Và họ đang được xử lý kết quả đang được gửi lại cho khách hàng di động. Và một trong những câu hỏi chính chúng ta phải trả lời địa chỉ là làm thế nào để chúng ta nhận được các lệnh này từ thiết bị di động đến máy chủ và truyền lại kết quả. Các yêu cầu và quy tắc cho giao tiếp với máy chủ từ xa là gì? Làm thế nào để chúng tôi đảm bảo nó đáng tin cậy? Làm thế nào để chúng tôi đảm bảo rằng nó có trật tự và được thực hiện theo cách mà máy chủ có thể hiểu và xử lý các lệnh cho chúng tôi.

## Introduction to Amazon Guest Reading - Analytics and A / B Testing

## Bất cứ khi nào các nhà phát triển ngồi xuống để viết các ứng dụng di động đã đoán được những gì người dùng thực sự muốn từ ứng dụng đó. Một số trong những quyết định đó sẽ đúng, và một số thì không. Một số sẽ khá tốt trong một khoảng thời gian ban đầu, nhưng sau đó mất hiệu quả theo thời gian. Vì vậy, nó có thể rất quan trọng đối với các nhà phát triển để biết quyết định của họ thực sự hiệu quả như thế nào. Bây giờ một cách để có được một số ý kiến thông tin từ người dùng. Và điều đó tốt, nhưng dữ liệu đó có thể chậm đến chỉ có thể phản ánh những người dùng có tiếng nói nhất của bạn. Một cách khác để có được thông tin đó có thể là tập hợp các nhóm tập trung để bạn có thể hỏi trực tiếp người dùng về ứng dụng của mình. Điều đó cũng tuyệt vời, nhưng cũng có xu hướng là một quá trình chậm, và cũng có thể khá tốn kém, vì vậy nó thường chỉ giới hạn ở một mẫu nhỏ người dùng. Tùy chọn thứ ba, một lựa chọn mà chúng ta sẽ nghe về ngày hôm nay, là xây dựng ứng dụng của bạn để ứng dụng có thể cho bạn biết nó được sử dụng như thế nào. Để thảo luận về vấn đề này, bạn bè của chúng tôi tại Amazon phân tích, cách xây dựng bộ sưu tập dữ liệu vào ứng dụng của bạn và nói về thử nghiệm A / B, một cách tiếp cận cho phép bạn cung cấp trải nghiệm khác nhau cho những người dùng khác nhau và sau đó ảnh hưởng trực tiếp đến ảnh hưởng của những khác biệt đó. Và khi được thực hiện đúng, thử nghiệm A / B cho phép bạn thử nghiệm và sau đó tự động tối ưu hóa ứng dụng của bạn cho bất kỳ tiêu chí nào bạn cho là quan trọng nhất. Vì vậy, vui lòng truy cập liên kết này để bạn bè của chúng tôi tại Amazon chỉ cho bạn cách hướng dẫn dữ liệu theo cách của bạn.

## Quiz:

**Sensors, Location and Maps, and Data Management**

Thiết bị cầm tay cho phép tính toán theo ngữ cảnh và điều đó có nghĩa là các ứng dụng có thể phản ứng hoặc hành xử khác nhau dựa trên các yếu tố theo ngữ cảnh mà chúng đang được sử dụng, cách chúng được người dùng giữ, mức độ ánh sáng xung quanh hoặc tốc độ của nó người dùng đang đi du lịch. Để làm điều này, các ứng dụng đọc thông tin từ một loạt các cảm biến được tích hợp với các thiết bị cầm tay ngày nay.

Trong bài học này, tôi sẽ nói về các cảm biến mà các thiết bị Android có thể hỗ trợ và tôi sẽ thảo luận về cách các ứng dụng có thể truy cập các cảm biến này.

Tiếp theo, tôi sẽ thảo luận về các sự kiện cảm biến, lớp sử dụng Android đại diện cho các lần đọc cảm biến và tôi sẽ thảo luận về các trình lắng nghe sự kiện cảm biến được sử dụng để truyền thông tin từ cảm biến đến ứng dụng của bạn.

Sau đó, tôi sẽ thảo luận về một số kỹ thuật phổ biến được sử dụng để làm mịn các giá trị cảm biến bộ lọc vàng để các ứng dụng có thể được sử dụng theo nhiều cách khác nhau. Và chúng ta đi qua bài học, tôi sẽ sử dụng nhiều ứng dụng sử dụng các cảm biến thông thường.

Sensors( cảm biến ) là thành phần phần cứng đo môi trường vật lý xung quanh thiết bị và các cảm biến này có ba loại.

Có những cảm biến đo MOTION ( chuyển động ), ví dụ, bạn di chuyển nhanh như thế nào.

Có các cảm biến đo POSITION ( vị trí ) của thiết bị. Chẳng hạn, bạn đang ở đâu trên thế giới hoặc định hướng của thiết bị là gì. Và có những cảm biến đo ENVIRONMENT ( môi trường ), chẳng hạn như lượng ánh sáng, áp suất hoặc độ ẩm xung quanh thiết bị. Ví dụ: thiết bị của tôi có gia tốc kế 3 trục để đo các lực tác dụng lên thiết bị, ví dụ như khi tôi lắc nó.

Nó cũng có từ trường 3 trục, có thể được sử dụng để đo vị trí của nó so với hướng từ trường của Trái đất. Và chúng ta sẽ thấy rằng trong hành động sau này trên một trong những ứng dụng ví dụ.

Và cuối cùng, thiết bị của tôi có một phong vũ biểu đo áp suất khí quyển.

SensorManager ( Bộ cảm biến ) là dịch vụ hệ thống bao gồm các cảm biến.

Các ứng dụng có được một tham chiếu đến Bộ cảm biến bằng cách gọi phương thức getSystemService, chuyển vào giá trị Context.SENSOR\_SERVICE. Để truy cập vào một cảm biến cụ thể, các ứng dụng sử dụng phương thức getDefaultSensor của SensorManager, truyền vào một hằng số tương ứng với cảm biến mong muốn.

getsystemService(

Context.SENSOR\_SERVICE)

SensorManager(

getDefaultSensor( int index );

)

Một số hằng số loại cảm biến bao gồm Sensor.TYPE\_ACCELEROMETER cho gia tốc kế. Sensor.TYPE\_MAGNETIC\_FIELD cho cảm biến từ trường. Hoặc Sensor.TYPE\_PRESSURE cho phong vũ biểu.

Nếu một ứng dụng muốn nhận thông tin từ một cảm biến, thì nó sẽ phải triển khai một SensorEventListener. Và giao diện này xác định các phương thức gọi lại được gọi khi độ chính xác của cảm biến thay đổi và khi cảm biến có được cách đọc mới.

Khi độ chính xác của cảm biến thay đổi, Android sẽ gọi phương thức onAccuracyChanged, chuyển qua cảm biến đã thay đổi và chuyển qua độ chính xác mới của nó.

Khi một cảm biến có cách đọc mới, phương thức onSensorChanged được gọi, chuyển qua Bộ cảm biến tương ứng sang cách đọc mới.

Tuy nhiên, trước khi ứng dụng của bạn có thể nhận được SensorEvents, nó sẽ cần phải đăng ký một SensorEventListener, và nó đã được thực hiện với cảm biến, bạn cũng muốn hủy đăng ký cảm biến, SensorListener, để tránh lãng phí pin.

Để đăng ký một SensorEventListener cho một cảm biến nhất định, bạn gọi phương thức registerListener, chuyển qua SensorEventListener sẽ được gọi trở lại cảm biến mà bạn muốn nghe và tốc độ bạn muốn cảm biến được thăm dò.

Ví dụ, để hủy đăng ký một trình lắng nghe, hãy gọi phương thức unregisterListener, truyền vào SensorEventListener và truyền vào một bitmask chỉ ra các cảm biến mà bạn không còn muốn nghe nữa. Các bài đọc cảm biến được biểu diễn dưới dạng các thể hiện của lớp SensorEvent. Tất nhiên, dữ liệu mà lớp này nắm giữ sẽ phụ thuộc vào loại cảm biến cụ thể sẽ bao gồm loại cảm biến, dấu thời gian, độ chính xác của việc đọc và dữ liệu đo liên quan đến cách đọc mới đó. Để hiểu được dữ liệu, bạn sẽ cần biết cách đo cảm biến cụ thể.

Chẳng hạn, nhiều cảm biến sử dụng hệ tọa độ ba chiều. Khi hướng mặc định là dọc và khi thiết bị nằm thẳng, ngửa mặt lên bàn, các trục của hệ tọa độ được hiển thị ở đây.

Trục x chạy từ phải sang trái. Trục y chạy xuống phía dưới. Và trục z chạy xuống.

Và đừng quên rằng hệ tọa độ được định hướng xung quanh hướng mặc định của thiết bị và điều đó không thay đổi ngay cả khi thiết bị thay đổi hướng hiện tại. Chẳng hạn, ngay cả khi thiết bị chuyển từ chế độ dọc sang chế độ nằm ngang, hệ thống tọa độ không thay đổi.

Ví dụ đầu tiên của chúng tôi cho bài học này được gọi là SensorRawAccelerometer. Ứng dụng này chỉ đơn giản hiển thị các giá trị thô mà nó nhận được từ gia tốc kế của thiết bị. Hãy xem nó hoạt động gia tốc kế của e.

Chúng ta hãy xem nó hoạt động.

Bây giờ, tôi sẽ khởi động ứng dụng SensorRawAccelerometer.

Như bạn có thể thấy, ứng dụng này có ba chế độ xem văn bản với các số trong đó.

Và những con số này tương ứng với các giá trị x, y và z được đọc từ gia tốc kế của thiết bị này. Như bạn có thể thấy, lực lớn nhất hiện đang được tác dụng lên trục Y.

Trong khi tôi đang thực hiện cuộc biểu tình này, tôi đang cố gắng, rõ ràng là không thành công, để giữ thiết bị hoàn toàn thẳng lên và xuống. Tay tôi run lên một chút, và vì thế, những con số sẽ nhảy xung quanh một chút.

Bây giờ tôi sẽ xoay thiết bị ngược chiều kim đồng hồ 90 độ quanh trục z, bạn sẽ thấy rằng lực lớn nhất đang được tác dụng lên trục x.

Chúng ta hãy xoay thiết bị thêm 90 độ nữa và bây giờ bạn sẽ thấy rằng, một lần nữa, lực lớn nhất đang được tác dụng lên trục y, nhưng lần này, lực đó là âm và đó là vì trục y hiện đang lộn ngược.

Và cuối cùng, tôi sẽ xoay thiết bị thêm 90 độ. Và một lần nữa, bạn sẽ thấy rằng lực lớn nhất được tác dụng lên trục x. Và lực lượng đó đang hoạt động theo hướng tiêu cực.

Hãy xem mã nguồn của ứng dụng này.

Đây là ứng dụng SensorRawAccelerometer mở trong IDE. Bây giờ tôi sẽ mở hoạt động chính.

Lưu ý rằng lớp này thực hiện giao diện SensorEventListener. Vì vậy, chúng tôi có thể nhận được cuộc gọi lại từ Bộ cảm biến.

Bây giờ, trong onCreate, ứng dụng nhận được một tham chiếu đến Bộ cảm biến.

Tiếp theo, nó là một tham chiếu đến gia tốc kế của thiết bị bằng cách gọi SensorManager.getDefaultSensor, truyền vào hằng số loại tương ứng với gia tốc kế. Trong phương thức onResume, ứng dụng đăng ký trình nghe lớp này cho sự kiện gia tốc bằng cách gọi phương thức registerListener. Tham số cuối cùng là SensorManager.SENSOR\_DELAY\_UI và điều này tương ứng với tỷ lệ bỏ phiếu tần số tương đối thấp.

Tiếp theo, phương thức onPause hủy đăng ký lớp này với tư cách là người nghe cho nó có thể nghe.

Cuộn xuống, bây giờ chúng ta đến với phương thức onSensorChanged. Phương pháp này trước tiên kiểm tra để đảm bảo rằng sự kiện này là đọc gia tốc.

Tiếp theo, nó kiểm tra xem một lượng thời gian nhất định đã trôi qua kể từ lần đọc cuối cùng được hiển thị. Và nếu vậy, mã này ghi lại các giá trị X, Y và Z của máy gia tốc.

Và sau đó nó sẽ hiển thị những giá trị đó trên màn hình.

**Sensors, Location and Maps, and Data Management**

**Sensors - Part 1**

**1.Giới thiệu**

Các thiết bị cầm tay có thể tính toán để nhận biết được bối cảnh. Điều đó có nghĩa là ứng dụng có thể phản hồi khác nhau dựa trên các bối cảnh khác nhau, như vị trí nơi chúng được sử dụng, cách chúng được giữ, ánh sáng môi trường xung quanh, hay tốc độ mà người dùng đang di chuyển. Để làm được điều đó, các ứng dụng đọc thông tin từ nhiều cảm biến được lắp trong các thiết bị cầm tay ngày nay.

Trong khóa học này, tôi sẽ nói về các sensor mà thiết bị Android có thể hổ trợ, và sẽ trình bày làm thế nào mà ứng dụng có thể truy cập những sensor đó.

Tiếp theo, tôi sẽ trình bày sensor event, lớp đó dùng để đọc cảm biến và tôi sẽ trình bày về sensor event listener được đùng để chuyển đổi thông tin từ sensor đến ứng dụng.

Sau đó, tôi sẽ trình một vài công nghệ thông dụng được dùng để làm mượt hoặc lọc giá trị sensor, để ứng dụng có thể dùng giá trị của nhiều cách khác nhau. Và khi chúng ta hoàn thành bài học, tối sẽ chứng minh bằng một vài ứng dụng, ví dụ tận dụng các sensor thông dụng.

**2.Senor là gì?**

**Sensor là thành phần phần cứng để đo môi trường vật lý xung quanh thiết bị**, những sensor này có 3 loại.

*a.Motion*

Có các sensor để đo di chuyển, ví dụ như đo tốc độ di chuyển.

b.position

Có các senor để đo vị trí của của thiết bị. Ví dụ, vị trí của bạn trên thế giới, hoặc sự định hướng của thiết bị như thế nào (đứng hay nằm).

c.Emviroment

Và có các sensor để đo môi trương, như là độ sáng, sức ép hoặc độ ẩm xung quanh thiết bị.

Ví dụ,

Thiết bị của tôi có **Motion-3-axis Accelerometer** (gia tốc kế 3 trục), nó đo các các lực tác dụng lên thiết bị, ví dụ khi tôi lắc nó.

Nó cũng có **Position-3-axis Magnetic field** (trường từ tính 3 trục), thứ có thể được dùng để đo vị trí của nó hoặc định hướng liên quan đến vùng từ trường của trái đất. Và chúng ta sẽ xem điều đó sau trong một ứng dụng mẫu.

Và sau cùng, thiết vị của tôi có một **Eviroment-pressure** (phong vũ biểu (barometer)) để đo sức ép khí quyển.

**3. SensorManager là gì?**

Để cho một ứng dụng dùng sensor, điều đầu tiên là để nó tham chiếu đến SensorManager. SensorManager là hệ thống phục vụ để quản lý sensor.

Ứng dụng sẽ tham chiếu đến SensorManager bằng cách gọi phương thức getSystemService, truyền vào giá trị Context.SENSOR\_SERVICE.

**getSystemService(Context.SENSOR\_SERVICE);**

Để mà truy cập một sensor chỉ định. Ứng dụng dùng phương thức getDefaultSensor của SensorManager, truyền vào một hàng số tương ứng với sensor mong muốn.

**SensorManager. getDefaultSensor(int type);**

Một vài hằng số của kiểu sensor bao gồm Sensor.

Sensor.TYPE\_ACCELEROMETER cho gia tốc.

Sensor.TYPE\_MAGNETIC\_FIELD cho cảm biến vừng từ trường .

Sensor.TYPE\_PRESSURE cho phong vũ biểu.

**4. SensorEventListener là gì?**

Nếu một ứng dụng muốn nhận thông tin từ sensor, thì nó sẽ phải thực hiện SensorEventListener. Và interface này định nghĩa một phương thức gọi lại, thứ này được gọi khi độ chính xác của cảm biến thay đổi và khi sensor nhận được reading mới.

Khi một độ chính xác của sensor thay đổi. Android gọi phương thức onAccuracyChanged, truyền vào sensor được thay đổi đó cùng với độ chính xác mới của nó.

**void onAccuracyChanged(Sensor sensor,int accuracy)**

Khi một sensor có một reading mới, phương thức onSensorChange được gọi, truyền vào SensorEvent tương ứng với reading mới.

**void onSensorChange(SensorEvent event)**

Tuy nhiên, trước lúc ứng dụng của bạn có thể nhận được SensorEvent, nó sẽ cần đăng ký một SensorEventListener, một khi nó hoàn thành việc với sensor, bạn cũng sẽ muốn hủy đăng ký, SensorListener, để tránh lảng phí pin.

Để đăng ký một SensorEventListener cho một sensor nhất định, bạn gọi phương thức registerListener, truyền vào SensorEventListener, nó sẽ được gọi về sensor bạn muốn nghe cùng tỷ lệ, thứ mà bạn muốn sensor thăm dò.

Public boolean registerListener(SensorEventListener listener, Sensor sensor, int rate)

Để hủy đăng ký một listener cho toàn bộ sensor đã được đăng ký, là có thể, ví dụ, gọi phương thức unregisterListener, truyền vào SenserListener và bitmask chỉ ra các sensor bạn không còn muốn dùng.

**publuc void unregisterListener(SensorListener, Sensor sensor)**

**5.SenserEvent**

Senser reading được đại diện như các thể hiện của SensorEvent class.

Dử liệu mà lớp này nắm giữ sẽ phụ thuộc vào **loại cụ thể của sensor** để tạo ra reading, nhưng sẽ bao gồm **kiểu sensor**, **timestamp**, **độ chính xác** của reading. Và **việc đo lường dữ liệu** liên kết với reading mới. Để mà tạo nên sự rỏ ràng của dử liệu, bạn sẽ cần phải giải thích việc đo lường như thế nào đối với các sensor cụ thể.

Ví dụ, nhiều sensor sử dụng hệ thống tọa độ ba chiều. Khi định hướng mặc định là chân dung và khi thiết bị nằm phẳng, mặt hướng lên trên bàn, các trục của hệ thống tọa độ được hiển thị ở đây.

Trục X chạy từ phải sang trái. Trục Y chạy từ trên xuống dưới. Trục Z chạy từ sau đến trước.

Và đừng quên ràng hệ thống tọa độ được định hướng dựa trên định hướng mặc định của thiết bị và không thay đổi thậm chí nếu thiết bị thay đổi định hướng hiện tại. ví dụ, nếu thiết bị chuyển từ chế độ chân dung sang phong cảnh, hệ thống tọa độ không thay đổi.

**6.Các ví dụ**

Ví dụ đầu tiên của chúng ta với bài học này được gọi là SensorRawAccelerometer. ứng dụng này trình bày đơn giản dử liệu thô, thứ nó nhận được từ gia tốc kế của thiết bị. để xem nó hoạt động.

Now, tôi sẽ khởi động ứng dụng SensorRawAccelerometer.

Như bạn có thể thấy, ứng dụng này trình bày 3 text view với các con số trong đó.

Và các con số đó tương ứng các giá trị x, y, z được đọc từ gia tốc kế của thiết bị. như bạn thấy, lực lượng lớn nhất đang được áp dụng lên trục Y.

Trong khi tôi trình bày, tôi đã cố gắng, rõ ràng là không thành công, để giữ thiết bị thẳng lên và xuống một cách hoàn hảo. Nhưng tất nhiên, tay tôi run lên một chút, và vì vậy, những con số sẽ nhảy xung quanh một chút.

Bây giờ tôi sẽ xoay thiết bị ngược chiều kim đồng hồ 90 độ quanh trục z và bạn sẽ thấy rằng bây giờ lực lớn nhất đang được tác dụng lên trục x.

Chúng ta hãy xoay thiết bị thêm 90 độ nữa và bây giờ bạn sẽ thấy rằng, một lần nữa, lực lớn nhất đang được tác dụng lên trục y, nhưng lần này, lực đó là âm và đó là vì trục y hiện đang lộn ngược.

Và cuối cùng, tôi sẽ xoay thiết bị thêm 90 độ. Và một lần nữa, bạn sẽ thấy rằng lực lớn nhất được tác dụng lên trục x. Và lực lượng đó đang hoạt động theo hướng đối lập

Hãy xem mã nguồn của ứng dụng này.

Đây là ứng dụng SensorRawAccelerometer mở trong IDE. Bây giờ tôi sẽ mở hoạt động chính.

Lưu ý rằng lớp này thực hiện SensorEventListener interface. Vì vậy, chúng tôi có thể nhận được lời gọi lại từ SensorManager.

Tiếp theo, nó nhận được một tham chiếu đến gia tốc kế của thiết bị bằng cách gọi SensorManager.getDefaultSensor, truyền vào hằng số loại tương ứng với gia tốc kế. Trong phương thức onResume, ứng dụng đăng ký lớp này như một trình lắng nghe cho sự kiện gia tốc bằng cách gọi phương thức registerListener. Tham số cuối cùng là SensorManager.SENSOR\_DELAY\_UI và điều này tương ứng với tỷ lệ bỏ phiếu tần số tương đối thấp.

Tiếp theo, phương thức onPause hủy đăng ký lớp này với tư cách là người nghe cho bất kỳ và tất cả các cảm biến mà nó có thể đang nghe.

Cuộn xuống, bây giờ chúng ta đến với phương thức onSensorChanged. Phương pháp này trước tiên kiểm tra để đảm bảo rằng sự kiện này là đọc gia tốc.

Tiếp theo, nó kiểm tra xem một lượng thời gian nhất định đã trôi qua kể từ lần đọc cuối cùng được hiển thị. Và nếu vậy, mã này ghi lại các giá trị X, Y và Z của máy gia tốc.

Và sau đó nó sẽ hiển thị những giá trị đó trên màn hình.

Sensors - Part 2

Trong ứng dụng ví dụ mà chúng ta vừa xem, tôi đã cố gắng giữ thiết bị hoàn toàn thẳng lên. Và nếu tôi có thể làm điều đó, gia tốc kế lý tưởng sẽ báo cáo các giá trị xung quanh x bằng 0 mét trên giây bình phương, y bằng 9,81 mét mỗi giây bình phương, hoặc z bằng 0 mét trên giây bình phương.

Nhưng như bạn đã thấy trong ứng dụng ví dụ, các giá trị của máy gia tốc dao động. Tất cả các ứng dụng sẽ trải nghiệm loại điều này, do chuyển động của người dùng tự nhiên, bề mặt không phẳng, tiếng ồn điện, v.v.

Khi tạo các ứng dụng kích hoạt cảm biến, các nhà phát triển sẽ thường áp dụng các biến đổi cho dữ liệu thô để làm mượt nó. Hai loại biến đổi phổ biến được gọi là bộ lọc thông thấp( low pass filters) và bộ lọc thông cao(high pass filters). Chúng ta hãy nói về từng thứ một. Bộ lọc thông thấp được sử dụng để nhấn mạnh những thay đổi lực nhỏ, thoáng qua trong khi nhấn mạnh các lực không đổi dài hạn.

Ví dụ, bạn có thể sử dụng bộ lọc thông thấp khi ứng dụng của bạn cần chú ý đến lực hấp dẫn không đổi, chẳng hạn, và bạn không muốn bị ảnh hưởng chỉ vì tay bạn run lên một chút.

Một ví dụ thực tế về điều này sẽ giống như trình độ của thợ mộc. Bong bóng cần phải di chuyển dựa trên trọng lực, không dựa trên co giật tay nhỏ.

Ngược lại, bạn sử dụng bộ lọc thông cao khi bạn muốn nhấn mạnh những thay đổi lực thoáng qua. Và bạn muốn khử các thành phần lực không đổi.

Bạn có thể sử dụng bộ lọc thông cao khi ứng dụng của bạn nên bỏ qua lực hấp dẫn không đổi. Nhưng nên đáp ứng với các động thái cụ thể mà người dùng thực hiện. Một ví dụ thực tế về điều này có thể là một nhạc cụ gõ giống như một bộ marica. Bạn không thực sự quan tâm đến trọng lực ở đây. Bạn quan tâm đến cách người dùng lắc nhạc cụ.

Ứng dụng tiếp theo được gọi là SensorFilteredAccelerometer. Ứng dụng này áp dụng cả bộ lọc thông thấp và bộ lọc thông cao cho các giá trị gia tốc thô. Và sau đó nó sẽ hiển thị các giá trị được lọc. Hãy chạy nó.

Bây giờ tôi sẽ khởi động ứng dụng cảm biến gia tốc được lọc.

Như bạn có thể thấy, ứng dụng này hiển thị 9 tex view với các số trong đó.

Những con số này tương ứng với các giá trị x, y và z được đọc từ gia tốc kế của thiết bị. Các giá trị thô sau khi áp dụng bộ lọc thông thấp và các giá trị thô đó sau khi áp dụng bộ lọc thông cao.

Nếu chúng tôi để ứng dụng chạy một lúc, bạn sẽ thấy rằng các giá trị thông thấp bắt đầu gần đúng với số đọc gia tốc lý tưởng của chúng tôi. Khoảng không cho các trục x và z. Và khoảng 9,81 cho trục y. Đồng thời bạn có thể thấy rằng tất cả các giá trị vượt qua đều có xu hướng về không. Nếu tôi xoay thiết bị ngược chiều kim đồng hồ. Bạn thấy giá trị High Pass x đi theo chiều dương. Và nếu tôi xoay thiết bị theo chiều kim đồng hồ, bạn sẽ thấy giá trị High Pass X bị âm.

Hãy xem mã nguồn của ứng dụng này.

Đây là ứng dụng gia tốc được lọc cảm biến được mở trong IDE.

Bây giờ, tôi sẽ mở hoạt động chính. Và lưu ý một lần nữa rằng lớp này thực hiện giao diện trình nghe sự kiện cảm biến. Vì vậy, nó có thể nhận được gọi lại từ sensor manager.

Bây giờ trong OnCreate, ứng dụng sẽ tham chiếu đến sensor manager.

Tiếp theo, nó nhận được một tham chiếu đến gia tốc kế của thiết bị bằng cách gọi SensorManager.getDefaultSensor, truyền vào hằng số loại tương ứng với gia tốc kế.

Trong phương thức onResume, ứng dụng đăng ký lớp này như một listener. Đối với các sự kiện gia tốc bằng cách gọi phương thức đăng ký listener.

Và tiếp theo phương onPause hủy đăng ký lớp này như một listener cho bất kỳ cảm biến nào.

Mà nó có thể đang lắng nghe.

Di chuyển xuống. Bây giờ chúng ta đến với phương pháp thay đổi cảm biến.

Và như trước đây, phương pháp này trước tiên kiểm tra để đảm bảo rằng sự kiện này là đọc gia tốc. Và sau đó, nó kiểm tra xem một khoảng thời gian nhất định đã trôi qua kể từ lần đọc cuối cùng được hiển thị. Nếu có, mã sẽ ghi lại các giá trị gia tốc x, y và z. Và sau đó áp dụng bộ lọc thông thấp cho từng giá trị thô, sau đó mã áp dụng bộ lọc thông cao cho từng giá trị thô.

Hãy xem mã cho hai bộ lọc. Đây là phương pháp low-pass, tính toán bộ lọc low-pass. Phương pháp này có hai tham số, một giá trị hiện tại và trung bình dài hạn.

Sau đó, nó tính giá trị được lọc là a, như một loại trung bình có trọng số. Trong trường hợp này, giá trị được lọc bằng 80% giá trị trung bình dài hạn cộng với 20% giá trị đọc hiện tại. Theo thời gian, tính toán này chuyển sang các giá trị lý tưởng mà chúng ta đã nói trước đó.

Cuộn xuống, đây là phương thức highPass để tính toán các giá trị được lọc highPass. Và phương pháp này cũng có hai tham số, đọc hiện tại và trung bình dài hạn, thực sự được tính bằng phương pháp thông thấp mà chúng ta vừa nói đến.

Mã này sau đó trừ đi trung bình dài hạn khỏi số đọc hiện tại và do đó đại diện cho phần đọc không phải do trọng lực.

Ứng dụng ví dụ này được gọi là SensorCompass. Ứng dụng này sử dụng gia tốc kế của thiết bị và từ kế của thiết bị để định hướng mũi tên la bàn về phía bắc từ tính.

Bây giờ, tôi sẽ khởi động ứng dụng SensorCompass. Như bạn có thể thấy, ứng dụng này hiển thị một vòng tròn màu xanh lá cây với một mũi tên màu trắng.

Ngay bây giờ, mũi tên này hướng về phía bắc từ tính.

Tuy nhiên, nếu tôi bắt đầu xoay thiết bị, bạn sẽ thấy rằng mũi tên tiếp tục hướng về phía bắc. Điều đó, tất nhiên, chính xác là những gì một la bàn phải làm.

Hãy xem mã nguồn của ứng dụng này.

Đây là ứng dụng la bàn cảm biến mở trong IDE.

Bây giờ, tôi sẽ mở hoạt động chính.

Hãy cuộn xuống phương thức onCreate.

Cũng như các ứng dụng khác, ứng dụng này bắt đầu bằng cách thiết lập giao diện người dùng.

Và đặc biệt, nó tạo ra một chế độ xem tùy chỉnh chứa mũi tên la bàn. Và sau đó nó thêm quan điểm đó vào chế độ xem chính của hoạt động.

Sau đó, nó được tham chiếu đến trình quản lý cảm biến.

Sau đó, nó được tham chiếu đến gia tốc kế của thiết bị. Và nó được tham chiếu đến từ kế của thiết bị. Bằng cách gọi Cảm biến anager.get cảm biến mặc định .Và bằng cách chuyển qua các hằng số loại thích hợp.

Trong phương thức resume, mã đăng ký lớp này như một listener cho các sự kiện gia tốc kế và cho các sự kiện từ kế bằng cách gọi phương thức đăng ký listener.

Phương thức pause hủy đăng ký lớp của nó như một listener cho tất cả các cảm biến.

Phương pháp thay đổi cảm biến xử lý các sự kiện cảm biến đến. Phương pháp này trước tiên kiểm tra xem sự kiện là gia tốc kế hay sự kiện từ kế. Và sau đó sao chép dữ liệu sự kiện thích hợp.

Tiếp theo, nếu có các số đọc từ mỗi trong hai cảm biến, mã sẽ gọi phương thức SensorManager.getRotationMatrix, chuyển qua các số đọc cảm biến và một mảng để lưu trữ ma trận xoay.

Nếu phương thức đó thành công, thì mã được gọi là phương thức get SensManager.getOrientation. Truyền vào ma trận xoay mà chúng ta vừa nhận được từ lệnh gọi để có rotationMatrix. Nó cũng vượt qua trong một mảng khác gọi là directionMatrix.

Khi phương thức này trả về, ma trận định hướng sẽ lưu giữ thông tin mà ứng dụng cần để xác định cách thiết bị được định hướng đối với hướng Bắc của Trái đất. Đoạn mã sau đó lấy giá trị này từ ma trận định hướng.

Và vì giá trị này được đo bằng radian, mã sau đó chuyển đổi giá trị radian thành độ.

Sau đó, mã làm mất hiệu lực chế độ xem mũi tên la bàn và sau đó xóa các mảng giữ các số đọc cảm biến.

Chúng ta hãy nhìn vào khung nhìn mũi tên la bàn để xem cách nó sử dụng thông tin định hướng mới.

Cuộn xuống phương thức onDraw, trước tiên mã sẽ lưu khung vẽ hiện tại và sau đó nó xoay chế độ xem này trên khung vẽ với số tiền bằng một lần trừ m một lần xoay theo độ. Vì vậy, về cơ bản, ý tưởng ở đây là nếu thiết bị đang chỉ cách 90 độ so với hướng bắc, thì mũi tên la bàn phải quay ngược 90 độ để mũi tên la bàn tiếp tục hướng về phía bắc.

# **Location and Maps - Part 1**

**1. Giới thiệu**

Vài năm trước, gia đình tôi đã dành một thời gian ở Montreal, Canada. Và ngay khi chúng tôi đến, trời bắt đầu có tuyết, tuyết và tuyết. Và tất nhiên, chúng tôi nhanh chóng phát hiện ra rằng hai cô con gái của tôi đã quên gói những giày tuyết của chúng. Vì vậy, chúng tôi cần mua ủng và chúng tôi cần mua chúng từ một cửa hàng gần chúng tôi. Vì vậy, tôi đã rút điện thoại di động của mình, và tôi đã tìm kiếm một giày ở khu vực địa phương. Ứng dụng đã trả về một danh sách các cửa hàng giày được vẽ trên bản đồ. Ứng dụng này cũng cho phép tôi nhận chỉ đường từ vị trí hiện tại của mình đến các cửa hàng. Chúng tôi theo những chỉ dẫn đó, mua một vài đôi giày và lưu lại kỳ nghỉ của chúng tôi. Bây giờ tôi chắc chắn nhiều bạn có thể kể lại những trải nghiệm tương tự. Đôi khi thông tin chúng ta cần phụ thuộc rất nhiều vào thông tin vị trí hiện tại của chúng ta. Để giúp đỡ trong những tình huống này, Android đã hỗ trợ cho vị trí và bản đồ.

Trong bài học hôm nay, tôi sẽ nói về một số hỗ trợ. Tôi sẽ nói về thông tin vị trí là gì. Và tôi sẽ đi qua các lớp mà ứng dụng của bạn sẽ sử dụng để có được thông tin đó.

Tôi sẽ nói về bản đồ, cho phép bạn lấy thông tin vị trí. Và hiển thị nó trực quan cho người dùng của bạn. Và tôi sẽ kết thúc bằng cách đi qua các lớp mà Android cung cấp để cho phép bạn hiển thị và tùy chỉnh Bản đồ.

Vì vậy, nói chung, các ứng dụng di động có thể được hưởng lợi từ việc nhận biết vị trí. Đó là biết họ đang ở đâu và những thứ khác tại một thời điểm cụ thể.

Và do đó, Android cho phép các ứng dụng xác định và thao tác thông tin vị trí.

Bây giờ trước đó, tôi đã đưa ra một ví dụ về việc sử dụng khả năng vị trí để tìm các cửa hàng gần vị trí hiện tại của tôi.

Và sau đó để được chỉ đường từ vị trí hiện tại của tôi đến một trong những cửa hàng đó. Ngoài ra, các ứng dụng cũng có thể sử dụng các khả năng này để thực hiện những việc như xác định khu vực địa lý hoặc địa chất. Và sau đó để bắt đầu các hành động khi người dùng đi vào hoặc thoát khỏi khu vực địa chất.

**2. Location**

Android cung cấp một số lớp hỗ trợ để thực hiện tất cả những điều này có thể. Một trong những lớp đó là lớp Location.

Một Location đại diện cho một vị trí trên trái đất.

Một Loacation chứa các thông tin như là vĩ độ. Kinh độ, dấu thời gian và tùy chỉnh, độ chính xác ước tính, độ cao, tốc độ và khả năng chịu lực. Thông tin vị trí đến từ LocationProviders và thiết bị có thể có quyền truy cập vào nhiều LocationProviders.

**3. LocationProvider**

Bây giờ dữ liệu thực tế có thể đến từ các nguồn như vệ tinh GPS, tháp điện thoại di động và các điểm truy cập WiFi.

Cụ thể, các ứng dụng có thể yêu cầu thông tin từ nhà cung cấp mạng (network provider), nhà cung cấp GPS (GPS provider) và nhà cung cấp thụ động (passive provider).

Chúng ta hãy tìm hiểu vào từng cái một.

a. Network provider

Nhà cung cấp mạng xác định vị trí dựa trên các điểm truy cập WiFi và tháp di động. Và nếu bạn muốn sử dụng nhà cung cấp này thì bạn phải khai báo hoặc là quyền access\_coarse\_location hoặc là quyền access\_fine\_location.

b. GPS provider

Nhà cung cấp GPS lấy thông tin vị trí của nó từ các vệ tinh GPS. Để sử dụng nhà cung cấp này, bạn phải khai báo quyền ACCESS\_FINE\_LOCATION.

C. pasive provider

Nhà cung cấp thụ động không thực sự bật bất kỳ thiết bị nào. Nó chỉ trả về các vị trí đã được tính toán thông qua các yêu cầu của ứng dụng khác. Vì vậy, bằng cách sử dụng nhà cung cấp này, yêu cầu bạn khai báo quyền ACCESS\_FINE\_LOCATION.

Bây giờ, tại sao bạn có thể nhận thông tin vị trí từ mỗi nguồn khác nhau này, mỗi nguồn cung cấp một bộ đánh đổi khác nhau liên quan đến chi phí. Độ chính xác, sẵn có và kịp thời của dữ liệu mà nó cung cấp. Hãy xem xét một số nhà cung cấp có sẵn trong Android.

Nhà cung cấp GPS dựa vào việc liên lạc với một vệ tinh. Và, do đó, nhà cung cấp này thường đắt nhất, nhưng cho kết quả chính xác nhất. Nó cũng mất nhiều thời gian nhất để cung cấp việc đọc rất chính xác. Và người dùng cần có một cái nhìn rõ ràng về bầu trời khi họ đang liên lạc với một vệ tinh GPS.

Nhà cung cấp mạng rẻ hơn nhà cung cấp GPS. Nhưng nó có thể cho thông tin ít chính xác hơn. Sẽ mất ít thời gian hơn để trả về thông tin vị trí, nhưng nó chỉ hữu ích khi bạn ở trong phạm vi của tháp di động hoặc điểm truy cập Wi-Fi. Các nhà cung cấp thụ động là rẻ nhất để sử dụng. Về cơ bản, bạn chỉ đang sử dụng lại các phép đo đã được thực hiện. Vì vậy, nó nhanh, nhưng nó có thể chỉ ra rằng không có thông tin gần đây khi ứng dụng của bạn yêu cầu thông tin đó.

**4. Location Manager**

Một cách để truy cập thông tin vị trí là sử dụng lớp location manager. Location manager là một hệ thống dịch vụ để truy cập dữ liệu vị trí. Bạn có được một tham chiếu đến location manager bằng cách gọi các lớp context lấy phương thức dịch vụ hệ thống, truyền vào ID cho service, context.location\_service.

getSystemService(Context.LOCATION\_SERVICE)

Một khi bạn có một tham chiếu đến Location manager.

Bạn có thể sử dụng nó để lấy và sử dụng thông tin vị trí. Chẳng hạn, bạn có thể xác định lần đọc cuối cùng được thực hiện bởi một nhà cung cấp cụ thể.

Bạn có thể đăng ký cập nhật vị trí để tìm hiểu khi có thông tin vị trí mới. Và bạn cũng có thể đăng ký để nhận gợi ý. Khi một thiết bị gần hoặc di chuyển ra khỏi một khu vực địa lý nhất định.

Nếu bạn muốn được thông báo khi xác định vị trí mới, bạn có thể triển khai và sử dụng location listener.  Location listener interface xác định các phương thức gọi lại, chúng được gọi khi vị trí thay đổi. Hoặc khi trạng thái của một nhà cung cấp vị trí thay đổi.

Một location listener interface bao gồm các phương pháp sau.

onLocationChanged, được gọi khi xác định vị trí mới.

**void onLocationChanged(Location location)**

onProviderDisabled và onProviderEnabled, được gọi khi người dùng vô hiệu hóa sự cho phép của một nhà cung cấp cụ thể.

**Void onProviderDisabled(String provider)**

**Void onproviderEnabled(String provider)**

onStatusChanged được gọi khi trạng thái của nhà cung cấp thay đổi.

**void onStatusChanged(String provider, int status, Bundle extras)**

**5. obtaining location**

Nếu ứng dụng của bạn không thể nhận được một lần đọc gần đây từ hệ thống thì nó sẽ cần phải có có cách đọc của chính nó. Và để làm điều này, ứng dụng của bạn thường sẽ thực hiện các bước sau.

Trước tiên, chúng tôi sẽ bắt đầu lắng nghe cập nhật từ các nhà cung cấp vị trí bằng cách đăng ký location listener.

Tiếp theo, nó sẽ duy trì một cập nhật, một ước tính tốt nhất hiện tại khi nó bắt đầu nhận được cập nhật vị trí.

Và khi quá trình này diễn ra, ứng dụng sẽ xác định khi nào ước tính tốt nhất hiện tại là đủ tốt. Và tại thời điểm đó, nó nên ngừng lắng nghe cập nhật vị trí bằng cách hủy đăng ký location listener.

Và cuối cùng nó có thể sử dụng ước tính tốt nhất đó là vị trí hiện tại.

**6. Determining best location**

Bây giờ khi bạn xác định xem vị trí của bạn có đủ tốt hay không, có một số yếu tố bạn có thể muốn xem xét.

Ví dụ, trong bao lâu bạn nên tiếp tục đo.

Ví dụ, một hệ thống định vị có thể cần đo liên tục. Trong khi một ứng dụng tìm nhà hàng có thể chỉ cần một phép đo duy nhất.

Một câu hỏi khác là bạn thực sự cần một phép đo như thế nào? Và một lần nữa, một hệ thống điều hướng cần biết vị trí của bạn để nói trong vòng mười mét hoặc lâu hơn.

Một ứng dụng nhà hàng có thể chỉ cần biết bạn đang ở thành phố nào và trong trường hợp đó, bạn chỉ cần biết vị trí của bạn để nói trong một km hoặc thậm chí có thể ít hơn.

Và, tất nhiên, các lựa chọn bạn thực hiện ở đây ảnh hưởng rõ ràng đến việc sử dụng pin.

**7. Ứng dụng ví dụ**

Ứng dụng ví dụ được gọi là LocationGetLocation. Ứng dụng này trước tiên mua lại và hiển thị một vị trí được biết đến cuối cùng từ tất cả các nhà cung cấp trên thiết bị. Nếu những bài đọc này quá cũ, hoặc có độ chính xác quá thấp. Sau đó, ứng dụng mua lại và hiển thị bài đọc mới từ tất cả các nhà cung cấp trên thiết bị. Hãy chạy ứng dụng.

Bây giờ tôi sẽ bắt đầu ứng dụng Địa điểm, Nhận vị trí.

Khi ứng dụng khởi động, nó sẽ hiển thị ước tính vị trí tốt nhất trước đó từ thiết bị. Thông tin này sau đó được hiển thị bằng văn bản màu đỏ.

Vì cách đọc này không đủ bực bội hoặc không đủ độ chính xác, ứng dụng tiếp tục thu được ước tính vị trí mới. Và, bạn có thể thấy rằng các reading mới này được hiển thị bằng văn bản màu xám.

Hãy xem mã nguồn của ứng dụng này.

Vì vậy, đây là ứng dụng mở trong IDE.

Bây giờ, tôi sẽ mở ra hoạt động chính.

Và hãy cuộn xuống phương thức tạo.

Và ở đây, chúng ta thấy rằng mã có được một tham chiếu đến trình quản lý vị trí.

Tiếp theo nó gọi một phương thức gọi là tốt nhất, vị trí được biết đến cuối cùng.

Phương pháp này sẽ tìm vị trí được biết đến cuối cùng từ mọi nhà cung cấp vị trí và sau đó chúng tôi sẽ trả lại chính xác nhất các phép đo này nhưng cũng đáp ứng các tiêu chí nhất định.

Nếu không đọc đáp ứng các tiêu chí đó, thì phương thức trả về null.

Tiếp theo, mã hiển thị thông tin về lần đọc cuối cùng.

Sau này, mã tiếp tục bằng cách xác định một trình nghe vị trí. Một lần nữa, phương pháp thay đổi vị trí của người nghe, mã xác định xem vị trí mới có tốt hơn ước tính tốt nhất hiện tại hay không. Nếu vậy, mã sau đó cập nhật ước tính tốt nhất và sau đó cập nhật màn hình. Bây giờ nếu độ chính xác nhỏ hơn min\_accuracy, thì vị trí hiện tại được coi là đủ tốt. Và do đó, mã hủy đăng ký người nghe vị trí.

Bây giờ cuộn xuống, chúng ta thấy phương thức onResume.

Phương pháp này kiểm tra xem liệu ước tính tốt nhất hiện tại có độ chính xác thấp hay đã được thực hiện hơn hai phút trước.

Nếu vậy, mã đăng ký người nghe cho cả nhà cung cấp mạng và cho nhà cung cấp GPS.

Sau đó, lịch trình mã có thể chạy được sẽ hủy đăng ký người nghe sau một khoảng thời gian cố định. Mã cũng sẽ hủy đăng ký người nghe vị trí nếu phương thức tạm dừng của hoạt động bình tĩnh.

Dưới đây là một số mẹo bạn có thể sử dụng để tiết kiệm pin khi tạo ứng dụng nhận biết vị trí.

Đầu tiên, luôn luôn kiểm tra các phép đo được biết đến cuối cùng. Nếu đủ tốt thì không cần phải thực hiện các phép đo mới.

Trả về các cập nhật không thường xuyên nhất có thể và giới hạn tổng thời gian đo.

Một số ứng dụng, chẳng hạn như một ứng dụng theo dõi người chạy bộ cần cập nhật thường xuyên hơn. Và cần tiếp tục đo trong khi ứng dụng đang chạy, vì vị trí của người dùng đang thay đổi.

Tuy nhiên, các ứng dụng như ứng dụng chúng ta vừa thấy chỉ cần một phép đo tốt duy nhất. Vì vậy, họ có thể đo lường không thường xuyên và ít thời gian hơn.

Sử dụng phép đo ít chính xác nhất cần thiết. Và chỉ sử dụng GPS nếu bạn thực sự cần.

Tắt các cập nhật trong onpause.

# Location and Maps - Part 2

**1. Giới thiệu**

Vị trí chỉ ra những nơi thực sự xung quanh chúng ta. Để nó có ý nghĩa khi trực quan hơn, các vị trí sẽ được thể hiện trên bản đồ. Android cung cấp hỗ trợ lập bản đồ thông qua API Android V2 của Google Maps. API này cung cấp một số loại bản đồ khác nhau:

a. Normal maps

cụ thể là bản đồ thông thường, trông giống như bản đồ đường truyền thống,

b. Satellite map

bản đồ vệ tinh, hiển thị ảnh chụp từ trên không của một khu vực,

c. Hybrid – Satellite+ road map

bản đồ lai, kết hợp ảnh vệ tinh và bản đồ đường

d. terrain maps

bản đồ địa hình, hiển thị chi tiết địa hình chẳng hạn như độ cao.

Android cho phép ứng dụng của bạn tùy chỉnh bản đồ theo nhiều cách. Chẳng hạn, bạn có thể thay đổi khu vực bản đồ hiển thị cho người dùng. Bạn có thể thêm các biểu tượng được gọi là điểm đánh dấu tại các địa điểm cụ thể trên bản đồ. Và bạn có thể thêm hình ảnh lớp phủ trên đầu bản đồ.

Bạn có thể làm cho bản đồ phản ứng với các cử chỉ, chẳng hạn như duỗi hai ngón tay và chụm lại để phóng to và xoay hai ngón tay để xoay bản đồ.

Và bạn cũng có thể có bản đồ chỉ ra vị trí hiện tại của người dùng bằng cách đặt một điểm đánh dấu đặc biệt trên bản đồ.

**2. một số lớp bản đồ**

Để hiển thị bản đồ, ứng dụng của bạn có thể sẽ sử dụng một số lớp sau:

a. GoogleMap

GoogleMap, đại diện và quản lý bản đồ,

b. MapFragment

MapFragment, hiển thị bản đồ Google trong một đoạn,

c. Camera

Camera, xác định phần bản đồ có thể nhìn thấy trên màn hình và xác định quan điểm mà người dùng đang nhìn thấy bản đồ

d. Marker

Marker, đại diện cho các biểu tượng, đôi khi có cửa sổ bật lên cho biết các vị trí trên bản đồ và cho phép ứng dụng của bạn hiển thị thông tin được liên kết với các vị trí đó.

**3. Setting up a máp application**

Để thiết lập và chạy ứng dụng Bản đồ, bạn cần thực hiện thêm một số bước. Chẳng hạn, bạn sẽ cần định cấu hình SDK dịch vụ Google Play.

Bạn sẽ cần lấy khóa API xác định ứng dụng của bạn.

Bạn sẽ cần chỉ định quyền, cài đặt và khóa API trong tệp .XML của tệp Manifesh Android.

Và sau đó, bạn sẽ cần thêm bản đồ vào ứng dụng của mình.

Để sử dụng bản đồ, bạn cần bao gồm một số quyền. Bạn sẽ cần sự cho phép của internet để có thể tải xuống hình ảnh bản đồ từ máy chủ Google maps.

Bạn cũng sẽ cần quyền ACCESS\_NETWORK\_STATE mà API Maps sử dụng để xác định xem nó có thể tải xuống dữ liệu hay không.

**4. Permission**

Để sử dụng bản đồ, bạn cần bao gồm một số quyền. Bạn sẽ cần sự cho phép của internet để có thể tải xuống hình ảnh bản đồ từ máy chủ Google maps. Bạn cũng sẽ cần quyền ACCESS\_NETWORK\_STATE mà API Maps sử dụng để xác định xem nó có thể tải xuống dữ liệu hay không.

Bạn sẽ cần quyền WRITE\_EXTERNAL\_STORAGE, vì dữ liệu bản đồ cần được ghi vào vùng lưu trữ ngoài của thiết bị và bạn sẽ cần quyền của google.android.providers.gsf.R- EAD\_GSERVICE. Và đó là để API của bản đồ có thể truy cập các dịch vụ Web của Google.

Và nếu bản đồ của bạn cần lấy thông tin vị trí, ví dụ, để hiển thị vị trí hiện tại của người dùng, thì bạn cũng cần chỉ định một hoặc nhiều quyền ACCESS\_COARSE\_LOCATION và ACCESS\_FINE\_LOCATION. Ứng dụng ví dụ tiếp theo của chúng tôi được gọi là MapEarthQuakeMap.

**5. MapEarthQuakeMap**

Và ứng dụng này thu thập dữ liệu động đất từ máy chủ và sau đó hiển thị dữ liệu đó trên bản đồ, sử dụng các điểm đánh dấu có thể nhấp. Hãy xem nào.

Trước khi tôi chỉ cho bạn ứng dụng MapEarthQuakeMap, hãy để tôi đưa bạn trở lại bài học về kết nối mạng. Và nếu bạn còn nhớ, trong bài học đó, tôi đã cho bạn thấy ứng dụng máy khách Android HTTP, đã đưa ra yêu cầu mạng về dữ liệu động đất từ dịch vụ web api.geonames.org và yêu cầu đó đã trả về một danh sách các trận động đất đã xảy ra trong khu vực địa lý cụ thể và sau đó ứng dụng hiển thị dữ liệu đó trong chế độ xem danh sách đơn giản.

Bây giờ tôi sẽ bắt đầu ứng dụng đó, vì vậy bạn có thể nhớ giao diện người dùng của nó trông như thế nào.

**6. Ứng dụng mẫu**

Như bạn có thể thấy, ứng dụng lấy dữ liệu động đất và sau đó trình bày nó trong chế độ xem danh sách.

Bây giờ, điều này chắc chắn là tốt. Dữ liệu tôi muốn có trên màn hình.

Nhưng giao diện người dùng không hữu ích cho tôi.

Ví dụ: tôi không thể thực sự hình dung được vị trí trên trái đất thực sự ở đâu trên trái đất. Và tôi không thể dễ dàng phân biệt các trận động đất lớn với các trận động đất có cường độ thấp hơn.

Vì vậy, bây giờ hãy xem ứng dụng MapEarthQuakeMap. Và chúng ta sẽ thấy cùng một dữ liệu nhưng lần này nó được trình bày trên bản đồ.

Ở đây đi.

Và như bạn có thể thấy, thay vì chế độ xem danh sách, hiển thị nhiều văn bản, dữ liệu động đất giờ đây xuất hiện dưới dạng tập hợp các điểm đánh dấu trên bản đồ thế giới.

Vị trí của điểm đánh dấu cho chúng ta biết nơi xảy ra trận động đất.

Và màu sắc của điểm đánh dấu cho thấy cường độ của trận động đất. Các điểm đánh dấu có màu đỏ hơn cho thấy các trận động đất cường độ cao hơn. Các điểm đánh dấu màu xanh lam nhiều hơn cho thấy các trận động đất cường độ thấp hơn.

Và ngoài ra, nếu tôi chạm vào một điểm đánh dấu, một cửa sổ bật lên sẽ xuất hiện cho thấy cường độ của trận động đất đó.

Bây giờ hãy để tôi phóng to và chạm vào một vài điểm đánh dấu.

Được rồi, chúng ta hãy xem mã nguồn cho ứng dụng này.

Đây là ứng dụng mở trong IDE.

Bây giờ, tôi sẽ mở hoạt động chính. Trong OnCreate, ứng dụng đặt chế độ xem nội dung thành tệp main.xml trong thư mục res / layout. Hãy mở tập tin đó ngay bây giờ.

Và ở đây bạn có thể thấy rằng toàn bộ bố cục bao gồm một đoạn duy nhất được cung cấp bởi lớp com.google.android.gms.maps.MapFragment.

Bây giờ quay trở lại hoạt động chính, ứng dụng tạo và bắt đầu một tác vụ không đồng bộ, thu nhận dữ liệu động đất theo phương thức nền.

Và sau đó nó phân tích cú pháp và sau đó cập nhật bản đồ trong phương thức onPostExecute. Vì vậy, hãy cuộn xuống phương thức onPostExecute.

Phương pháp này bắt đầu bằng cách lấy tham chiếu đến bản đồ Google bên dưới MapFragment. Tiếp theo, nó lặp lại thông qua dữ liệu kết quả và với mỗi bản ghi động đất trong dữ liệu kết quả đó, mã này sẽ thêm một điểm đánh dấu mới.

Và mỗi điểm đánh dấu được tạo bằng cách tạo một đối tượng tùy chọn đánh dấu và sau đó chuyển đối tượng đó sang phương thức addMarker của Google maps.

Như chúng ta đã thấy với các API khác, lớp MarkerOptions sử dụng giao diện lưu loát và do đó, mã trước tiên tạo một đối tượng tùy chọn đánh dấu trống mới và sau đó đặt vị trí đánh dấu bằng cách gọi phương thức vị trí. Và sau đó, nó xử lý văn bản sẽ xuất hiện trong cửa sổ thông tin bật lên khi người dùng chạm vào điểm đánh dấu bằng cách gọi phương thức tiêu đề.

Và trong trường hợp này, tiêu đề chỉ đơn giản là hiển thị cường độ của trận động đất. Bây giờ sau đó, mã đặt màu của điểm đánh dấu bằng cách thực hiện cuộc gọi đến phương thức biểu tượng đi qua Marker mặc định, nhưng đặt màu của nó để phản ánh cường độ của trận động đất.

Và cuối cùng, mã được gọi là phương thức moveCamera, để định tâm bản đồ tại một vị trí cụ thể.

Bây giờ, để giữ cho ví dụ này đơn giản, tôi đã tính toán trước trung tâm bản đồ. Sau đó, nhưng tất nhiên sẽ tốt hơn nhiều và mạnh mẽ hơn khi tính toán trung tâm dựa trên dữ liệu thực tế được trả về bởi dịch vụ web.

# Data Management - Part 1

**1. Giới thiệu**

Hệ thống cầm tay có thể tạo và thao tác một lượng lớn dữ liệu. Vì vậy, Android cung cấp một số lớp hỗ trợ cho phép bạn quản lý dữ liệu qua nhiều phiên ứng dụng.

Trong bài học hôm nay, tôi sẽ nói về một vài trong số các lớp hỗ trợ này. Tôi sẽ bắt đầu bằng cách nói về lớp SharedPreferences, cho phép các ứng dụng để lưu trữ và quản lý một lượng nhỏ dữ liệu nguyên thủy. Tiếp theo, tôi sẽ nói về việc ghi tập tin vào cả bộ nhớ trong và bộ nhớ ngoài. Và cuối cùng, tôi sẽ kết thúc bằng cách thảo luận về việc tạo và sử dụng các cơ sở dữ liệu SQLite phức tạp.

Chúng ta hãy thảo luận ngắn gọn về từng tùy chọn lưu trữ này cùng một lúc.

a. Shared preferences

Các ứng dụng của bạn thường sẽ sử dụng lớp shared preferences khi bạn muốn lưu trữ một lượng nhỏ dữ liệu nguyên thủy, chẳng hạn như tên người dùng.

b. Internal storage

Các ứng dụng của bạn thường sẽ sử dụng lưu trữ thiết bị nội bộ khi bạn cần quản lý lượng dữ liệu nhỏ đến trung bình nên giữ riêng tư cho ứng dụng, chẳng hạn như các tệp tạm thời được ứng dụng sử dụng.

c. External storage.

Các ứng dụng của bạn thường sẽ sử dụng bộ nhớ ngoài khi bạn muốn lưu trữ lượng lớn dữ liệu không riêng tư, chẳng hạn như các bài hát hoặc tệp video.

d. Database

Và ứng dụng của bạn thường sẽ sử dụng cơ sở dữ liệu khi bạn có ý định lưu trữ một lượng nhỏ đến lớn dữ liệu riêng tư, có cấu trúc.

SharedPreferences về cơ bản là các map liên tục. Và giống như bất map nào, chúng giữ các cặp khóa-giá trị của các loại dữ liệu đơn giản, những thứ như chuỗi và số float.

Và SharedPreferences được tự động duy trì trong các phiên ứng dụng. Và điều này cho phép người dùng tạo thông tin trong một lần sử dụng ứng dụng, thoát khỏi ứng dụng, khởi động lại sau và vẫn có quyền truy cập vào thông tin họ đã tạo trước đó.

SharedPreferences thường được sử dụng để lưu trữ lâu dài dữ liệu ứng dụng có thể tùy chỉnh, bao gồm những thứ như tên người dùng, mạng WiFi yêu thích hoặc tùy chọn hoặc tùy chọn người dùng cụ thể.

Activity sharedpreferences

Để có được một đối tượng SharedPreferences được liên kết với một hoạt động nhất định, bạn có thể sử dụng phương thức Activity.getPreferences, chuyển qua chế độ truy cập làm tham số. Ví dụ: chế độ này có thể là MODE\_PRIVATE, chỉ ra rằng dữ liệu là riêng tư đối với ứng dụng gọi điện.

**Activity.getPreferences (int mode)**

**MODE\_PRIVATE**

Nếu bạn muốn một đối tượng SharedPreferences không được liên kết với một hoạt động cụ thể, thì bạn có thể sử dụng phương thức Context.getSharedPreferences để nhận một đối tượng SharedPreferences có tên. Và với phương thức này, bạn truyền vào tên cho đối tượng SharedPreferences và truyền vào chế độ truy cập, chẳng hạn như MODE\_PRIVATE mà chúng ta đã thấy trước đó.

**Context.getSharedPreferences(String name, int mode)**

Khi bạn đã có được một đối tượng SharedPreferences, bạn có thể chỉnh sửa đối tượng đó bằng cách gọi SharedPreferences.edit và phương thức này trả về một thể hiện của **SharedPreferences.edito**r. Sau đó, bạn có thể thêm hoặc thay đổi các giá trị của đối tượng SharedPreferences bằng các phương thức như putInt, putString và remove.

**putInt(String key,int value)**

**putString(String key,String value)**

**remove(String key)**

Và sau khi bạn thực hiện bất kỳ thay đổi nào bạn muốn, bạn có thể thực hiện những thay đổi đó vĩnh viễn bằng cách gọi phương thức **SharedPreferences.Editor.commit**. Và tại thời điểm này, đối tượng SharedPreferences được lưu và các ứng dụng có thể thoát khi biết rằng dữ liệu của chúng có thể được truy xuất trong các phiên ứng dụng sau này.

Nếu một ứng dụng sau đó muốn đọc các giá trị này, nó có thể lấy lại đối tượng SharedPreferences và sau đó sử dụng các phương thức khác nhau để đọc các giá trị được lưu trữ trong đối tượng. Chẳng hạn, nó có thể gọi **getAll** để nhận các giá trị SharedPrefences. Nó có thể gọi **getBoolean** để lấy một giá trị Boolean cụ thể hoặc nó có thể gọi **getString** để lấy một giá trị chuỗi cụ thể.

**2. ứng dụng mẫu SharedPreferences**

Ứng dụng ví dụ đầu tiên của chúng tôi trong bài học này được gọi là DataMan Quản lýSharedPreferences.

Ứng dụng này có một nút có nhãn Play và khi người dùng nhấn nút Play, ứng dụng sẽ hiển thị một số ngẫu nhiên.

Ứng dụng theo dõi số lượng cao nhất được thấy cho đến nay và lưu số đó qua các phiên người dùng khác nhau. Hãy chạy ứng dụng này ngay bây giờ.

Bây giờ, tôi sẽ bắt đầu ứng dụng DataMan Quản lýSharedPreferences.

Khi ứng dụng này bắt đầu, nó cho thấy điểm cao hiện tại bằng không. Khi tôi nhấn nút Play, bạn sẽ thấy rằng một số mới hiện được hiển thị ở giữa màn hình. Và vì con số đó cao hơn điểm số cao trước đó, màn hình điểm cao đã được cập nhật để hiển thị số mới. Bây giờ, hãy nhớ điểm số cao này.

Bây giờ tôi sẽ thoát khỏi ứng dụng này.

Và bây giờ, tôi sẽ khởi động lại nó.

Và như bạn có thể thấy, điểm số cao hiện tại là điểm số cao tương tự mà chúng ta đã thấy từ lần cuối cùng chúng ta chạy ứng dụng. Bây giờ, hãy để tôi nhấn nút Play thêm một vài lần nữa để tôi tăng điểm cao. Chúng tôi đi đây. Và

Bây giờ, tôi sẽ dừng ứng dụng một lần nữa và khởi động lại nó.

Và một lần nữa, bạn có thể thấy ứng dụng hiển thị điểm số cao đã được đặt trong lần sử dụng trước của ứng dụng.

Bây giờ, tôi cũng có thể nhấn nút Đặt lại để đặt lại điểm số cao đó về không.

Hãy xem mã nguồn của ứng dụng này.

Vì vậy, đây là ứng dụng mở trong IDE.

Bây giờ, tôi sẽ mở hoạt động chính.

Trong onCreate, mã thu được đối tượng SharedPreferences cho hoạt động này.

Tiếp theo, khi người dùng nhấp vào nút Play, mã sẽ tạo một điểm mới, nó sẽ lưu trong một biến gọi là val.

Sau đó, mã gọi getInt trên đối tượng ưu tiên để lấy điểm cao hiện tại, nếu val lớn hơn điểm cao hiện tại, thì chúng ta cần cập nhật điểm cao. Vì vậy, mã gọi phương thức chỉnh sửa trên đối tượng tùy chọn, trả về một đối tượng SharedPreferences.Editor.

Tiếp theo, mã gọi, gọi putInt trên đối tượng soạn thảo để cập nhật điểm cao thành giá trị hiện tại.

Cuối cùng, mã gọi cam kết trên trình chỉnh sửa để lưu điểm cao hiện tại.

**3. PreferencesFragment**

Như tôi đã nói trước đó, lớp SharedPreferences thường được sử dụng để lưu trữ tùy chọn người dùng của ứng dụng.

Để làm điều này dễ dàng hơn, Android cung cấp một lớp có tên PreferenceFragment mà các ứng dụng có thể sử dụng để hiển thị và sửa đổi tùy chọn của người dùng.

Ứng dụng ví dụ tiếp theo của chúng tôi được gọi là DataManloymentPreferenceFragment. Ứng dụng này sử dụng PreferenceFragment để hiển thị và sửa đổi tùy chọn người dùng của ứng dụng. Trong trường hợp này, ưu tiên là tên mà ứng dụng sẽ sử dụng khi đánh địa chỉ người dùng. Chúng ta hãy xem ứng dụng đó.

**4. ứng dụng mẫu PreferencesFragment**

Bây giờ, tôi sẽ khởi động ứng dụng DataManloymentPreferenceFragment.

Khi ứng dụng này khởi động, nó sẽ hiển thị một nút có nhãn Xem tên người dùng.

Khi tôi nhấp vào nút này, PreferenceFragment sẽ xuất hiện, cho phép tôi xem tên người dùng hiện tại của mình và thay đổi nó nếu tôi muốn. Hãy để tôi làm điều đó bây giờ. Và như bạn có thể thấy, tên người dùng của tôi chưa được đặt, vì vậy tôi sẽ nhấp vào khu vực này. Và bây giờ một hộp thoại bật lên, yêu cầu tôi nhập tên người dùng của mình.

Hãy để tôi làm điều đó bây giờ.

Và bây giờ, tôi sẽ nhấp vào nút Gửi để chấp nhận tên người dùng mới này. Hộp thoại đóng lại.

Và ở đây bạn có thể thấy rằng tên người dùng của tôi bây giờ là tên mà tôi vừa nhập.

Bây giờ tôi sẽ đóng ứng dụng và khởi động lại nó.

Tôi sẽ nhấp lại vào nút Xem Tên người dùng.

Và như bạn có thể thấy, tên người dùng của tôi vẫn là Adam, vì vậy thông tin thực sự vẫn tồn tại trong các phiên của người dùng.

Hãy xem mã nguồn cho ứng dụng này.

Vì vậy, đây là ứng dụng mở trong IDE.

Bây giờ, tôi sẽ mở hoạt động ViewAndUpdatePreferences. Hoạt động này là hoạt động được bắt đầu khi người dùng nhấp vào nút Xem Tên người dùng.

Phương thức onCreate trước tiên gọi setContentView, truyền vào một tệp XML có tên user\_prefs\_fragment.xml. Và tệp bố cục này khởi tạo và hiển thị một thể hiện của lớp UserPreferenceFragment, được định nghĩa trong tệp này.

Bây giờ chúng ta hãy nhìn vào lớp học đó. Phương thức onCreate của lớp này trước tiên gọi phương thức addPreferencesFromResource, truyền vào tệp user\_prefs.xml trong thư mục res / xml. Hãy mở tập tin đó.

Như bạn có thể thấy, tệp này xác định tài nguyên PreferenceScreen. PreferenceScreen này chứa một tùy chọn được hiển thị trong hộp văn bản chỉnh sửa và tùy chọn đó có một khóa, chuỗi uname và được hiển thị với tiêu đề Tên người dùng.

Khi người dùng nhấp vào hộp văn bản Chỉnh sửa để thay đổi tên người dùng, một hộp thoại sẽ bật lên với tiêu đề Thay đổi tên người dùng,

một phụ đề hoặc tin nhắn có nội dung, Nhập Tên người dùng của bạn và hai nút, được gắn nhãn Hủy và Gửi.

Ngoài các tùy chọn, Android hỗ trợ sử dụng các tệp.

**5.File**

Tệp là một lớp đại diện cho một thực thể hệ thống tệp được xác định bởi một tên đường dẫn.

Trong Android, các khu vực lưu trữ được phân loại là nội bộ hoặc bên ngoài.

Trong lịch sử, điều này phân biệt giữa bộ nhớ flash bên trong trên thiết bị và thẻ nhớ ngoài có thể tháo rời được gắn vào thiết bị.

Tuy nhiên, ngày nay, không phải tất cả bộ nhớ ngoài đều có thể tháo rời.

Bộ nhớ trong thường được sử dụng cho các tập dữ liệu nhỏ hơn riêng tư cho một ứng dụng.

Ngược lại, bộ nhớ ngoài thường được dành cho một tập dữ liệu lớn hơn, không riêng tư, những thứ như tệp nhạc và hình ảnh.

**6. File API**

Các ứng dụng ví dụ mà chúng ta sẽ xem xét chỉ trong lần truy cập thứ hai và sử dụng các tệp để lưu trữ thông tin.

Hai trong số các phương thức họ sử dụng để làm điều này là phương thức openFileOutput, mở một tệp riêng để ghi. Và phương pháp này sẽ tạo một tệp vật lý nếu nó không tồn tại.

**FileOutStream openFileOutput (String name, int Mode)**

Và cũng có phương thức openFileInput, mở một tệp riêng để đọc. Và dĩ nhiên, có nhiều phương pháp liên quan đến tập tin khác, vì vậy vui lòng xem tài liệu để biết thêm thông tin.

**7. Ứng dụng mẫu DataManloymentFileIternalMemory**

Ứng dụng ví dụ tiếp theo được gọi là DataManloymentFileIternalMemory. Khi ứng dụng này khởi động, nó sẽ kiểm tra xem một tệp văn bản cụ thể có tồn tại hay không và nếu không, ứng dụng sẽ tạo tệp đó và ghi một số văn bản vào đó.

Sau đó, ứng dụng sẽ mở tập tin đó, đọc văn bản từ nó và hiển thị văn bản đó trên màn hình. Chúng ta hãy xem ứng dụng này.

Vì vậy, đây là thiết bị của tôi. Bây giờ, tôi sẽ mở ứng dụng Data ManagementFileI InternalalMemory. Như bạn có thể thấy, ứng dụng đã hiển thị một số văn bản trên màn hình.

Hãy xem mã nguồn của ứng dụng này để tìm hiểu văn bản đó đến từ đâu.

Vì vậy, đây là ứng dụng mở trong IDE.

Bây giờ, tôi sẽ mở hoạt động chính cho ứng dụng này.

Và ở đây trong phương thức onCreate, trước tiên mã sẽ lấy FileStreamPath được liên kết với tên tệp, testfile.txt. Và nếu tệp đó không tồn tại, thì mã sẽ tiếp tục gọi phương thức writeFile.

Hãy cuộn xuống và xem phương pháp đó.

Bây giờ, phương thức này trước tiên gọi phương thức openFileOutput, trả về FileOutputSteam.

Đoạn mã tiếp tục bằng cách viết ba dòng văn bản vào tệp văn bản đó và cuối cùng, nó đóng tệp.

Bây giờ, hãy cuộn lại vào phương thức onCreate.

Bây giờ mã tiếp tục bằng cách gọi phương thức readFile, chuyển qua một bố cục tuyến tính để hiển thị văn bản. Phương thức readFile hiện mở tệp văn bản cho đầu vào. Và sau đó bắt đầu đọc từng dòng trong tệp. Và mỗi dòng này sau đó được đặt vào TextView và thêm vào bố cục tuyến tính

# Data Management- Part 2

**1. Giới thiệu chung**

Như tôi đã nói trước đây, Android cũng cho phép các ứng dụng ghi vào bộ nhớ ngoài.

Một thay đổi nữa mà bạn sẽ cần xem xét, tuy nhiên, khi bạn làm điều này là bộ nhớ ngoài có thể tháo rời. Ví dụ: khi được cung cấp bởi thẻ SD và vì điều này, phương tiện đó có thể xuất hiện hoặc biến mất mà không có cảnh báo.

Và do đó, khi ứng dụng của bạn sẽ ghi vào bộ nhớ ngoài, trước tiên bạn cần xác định trạng thái của bộ nhớ ngoài.

Bây giờ, một cách để làm điều đó là bằng cách sử dụng phương thức getExternalStorageState của lớp Môi trường.

Phương pháp đó sẽ trả về một giá trị cho biết trạng thái hiện tại của bộ nhớ ngoài của thiết bị và một số giá trị được hiển thị ở đây.

**String Enviroment.getExternalStrogeState()**

Ví dụ: có MEDIA\_MOUNTED, có nghĩa là phương tiện được trình bày và nó được gắn với quyền truy cập đọc và ghi. Có MEDIA\_MOUNTED\_READ\_ONLY, có nghĩa là phương tiện có mặt và được gắn kết, nhưng có quyền truy cập chỉ đọc. Và có MEDIA\_REMOVED, chỉ ra rằng bộ nhớ ngoài không có. Và, tất nhiên, có một số giá trị khác là tốt.

Nếu ứng dụng của bạn muốn ghi vào bộ nhớ ngoài, thì nó sẽ cần yêu cầu sự cho phép thích hợp. Cụ thể, bạn sẽ cần thêm quyền WRITE\_EXTERNAL\_STORAGE trong tệp AndroidManifest.xml của ứng dụng.

**2. Ứng dụng mẫu DataManagementFileExternalMemory**

Ứng dụng ví dụ tiếp theo của chúng tôi được gọi là DataMan Quản lýFileExternalMemory.

Ứng dụng này đọc một tập tin hình ảnh từ thư mục res / raw.

Nó sao chép tập tin đó vào bộ nhớ ngoài. Nó đọc dữ liệu hình ảnh trở lại từ tập tin bên ngoài và sau đó nó sẽ hiển thị hình ảnh trên màn hình. Chúng ta hãy xem ứng dụng đó ngay bây giờ.

Khi tôi khởi động ứng dụng DataMan Quản lýFileExternalMemory, bạn sẽ thấy một hình ảnh được hiển thị trên màn hình. Ở đây đi.

Hãy xem mã nguồn để hiểu rõ hơn dữ liệu hình ảnh này đến từ đâu.

Vì vậy, đây là ứng dụng mở trong IDE.

Bây giờ tôi sẽ mở hoạt động chính. Trong phương thức onCreate, mã kiểm tra trạng thái lưu trữ bên ngoài để đảm bảo rằng phương tiện được gắn kết, và nó có thể đọc và ghi được.

Sau đó, mã nhận được thư mục tệp bên ngoài nơi hình ảnh thường được lưu trữ.

Và nó xây dựng một đối tượng File mới trỏ trong thư mục đó.

Tiếp theo, nó kiểm tra xem một tập tin thực sự tồn tại trên bộ nhớ ngoài. Đoạn mã sau đó gọi phương thức copyImageDataToMemory.

Hãy xem phương pháp đó.

Bây giờ, phương thức này bắt đầu bằng cách tạo một luồng đầu ra mới sẽ được sử dụng để ghi vào một tệp trên bộ nhớ ngoài.

Tiếp theo, nó tạo ra một luồng đầu vào để nó có thể đọc dữ liệu hình ảnh từ thư mục res / raw.

Và cuối cùng, nó sao chép dữ liệu từ luồng đầu vào sang luồng đầu ra.

Khi phương thức này hoàn thành, sẽ có một tệp mới trong bộ nhớ ngoài.

Bây giờ, quay trở lại phương thức onCreate, mã sẽ tham chiếu đến ImageView và sau đó đặt URI hình ảnh thành tệp vừa tạo trong bộ nhớ ngoài.

**3. Cache Files**

Nếu ứng dụng của bạn tạo các tệp tạm thời, thì bạn có thể xem xét ghi chúng vào các thư mục bộ đệm. Tệp bộ nhớ cache là các tệp tạm thời có thể bị hệ thống xóa khi dung lượng lưu trữ thấp. Và các tệp này được gỡ bỏ khi gỡ cài đặt ứng dụng.

Và bạn có thể truy cập vào thư mục bộ đệm bằng cách sử dụng phương thức getCacheDir của lớp Context.

**File Context.getCacheDir()**

Phương thức này trả về đường dẫn tuyệt đối đến một thư mục dành riêng cho ứng dụng có thể được sử dụng cho các tệp tạm thời.

**4. Saving Cache Files**

Ngoài ra, bạn cũng có thể sử dụng phương thức getExternalCacheDir của lớp ngữ cảnh, trả về một đối tượng Tệp đại diện cho một thư mục cho các tệp bộ đệm trong bộ nhớ ngoài.

**Context.getExternalCacheDir()**

**5. SQL LITE**

Khi ứng dụng của bạn đọc và sử dụng một lượng lớn dữ liệu có cấu trúc và phức tạp, bạn có thể muốn đưa dữ liệu đó vào cơ sở dữ liệu. Android cung cấp triển khai SQLite, cho phép các ứng dụng tạo và sử dụng cơ sở dữ liệu quan hệ trong bộ nhớ.

SQLite được thiết kế để hoạt động trong một dấu chân rất nhỏ, giả sử, dưới 300 kilobyte lưu trữ.

SQLite, tuy nhiên, là một cơ sở dữ liệu quan hệ chính thức. Nó thực hiện hầu hết các tiêu chuẩn SQL92 và nó hỗ trợ các giao dịch ACID có độ tin cậy cao, có nghĩa là các giao dịch là nguyên tử, nhất quán, tách biệt và bền.

Để sử dụng cơ sở dữ liệu SQL trong các ứng dụng của bạn, Android khuyên bạn nên sử dụng lớp helper có tên SQLiteOpenHelper.

Và những gì bạn sẽ làm là bạn sẽ tạo một lớp con của lớp SQLiteOpenHelper này. Và trong hàm tạo cho lớp đó, bạn sẽ cần gọi constructor của lớp cha, truyền một số thông tin về cơ sở dữ liệu bạn muốn tạo.

Tiếp theo, bạn cũng sẽ ghi đè phương thức **onCreate**, phương thức **onUpgrad**.

Thông thường, trong onCreate, bạn sẽ thực thi một hoặc nhiều lệnh CREATE TABLE, xác định cấu trúc và bố cục của cơ sở dữ liệu.

Và sau đó, bạn sẽ sử dụng các phương thức SQLiteOpenHelper, ví dụ, để mở và trả về cơ sở dữ liệu SQLite bên dưới. Và sau đó, bạn sẽ thực hiện các hoạt động trên cơ sở dữ liệu cơ bản.

**EXECTU CREATE TABLE commands.**

**6. Ứng dụng mẫu**

Ứng dụng ví dụ cuối cùng của chúng tôi được gọi là DataMan Quản lý.

Ứng dụng này tạo ra một cơ sở dữ liệu SQLite và sau đó chèn một số bản ghi, một số có lỗi vào cơ sở dữ liệu.

Ứng dụng cũng sẽ hiển thị một nút có nhãn Fix.

Và khi người dùng nhấn nút Fix, ứng dụng sẽ xóa và cập nhật một số bản ghi vừa được chèn. Và sau đó nó sẽ hiển thị lại các bản ghi cơ sở dữ liệu được cập nhật trên màn hình. Bây giờ chúng ta hãy xem ứng dụng đó.

Bây giờ, tôi sẽ khởi động ứng dụng DataMan Quản lý.

Như bạn có thể thấy, ứng dụng hiện hiển thị bốn bản ghi cơ sở dữ liệu, mỗi bản ghi chứa ID bản ghi và tên của một nghệ sĩ.

Ở phía dưới màn hình, cũng có nút Fix mà tôi đã đề cập trước đó.

Khi tôi nhấn nút này, bản ghi số hai, Lady Gaga, sẽ bị xóa.

Và kỷ lục số ba, Jawny Cash, sẽ được cập nhật để đánh vần chính xác tên của nghệ sĩ.

Hãy để tôi nhấn nút Fix ngay bây giờ. Và như bạn có thể thấy, bản ghi số hai đã bị xóa và bản ghi số ba hiện hiển thị đúng chính tả tên của Johnny Cash.

Hãy xem mã nguồn của ứng dụng này.

Đây là ứng dụng mở trong IDE.

Bây giờ tôi sẽ mở ra hoạt động chính.

Ở đây trong phương thức onCreate, chúng ta bắt đầu bằng cách tạo một cá thể DatabaseOpenHelper mới. Bây giờ, lớp đó là một lớp con của SQLiteOpenHelper.

Hãy xem lớp học đó.

Hàm tạo cho lớp này gọi hàm tạo của siêu lớp, truyền thông tin như tên của cơ sở dữ liệu và số phiên bản.

Bây giờ, lớp này là phương thức onCreate nhận một đối tượng cơ sở dữ liệu SQLite và sau đó gọi phương thức execQuery của nó, truyền vào một chuỗi tương ứng với một lệnh SQL thực tế sẽ tạo một bảng có tên là các nghệ sĩ. Và bảng đó sẽ chứa hai trường, ID số nguyên và chuỗi cho tên nghệ sĩ.

Lớp này cũng có một phương thức xóaDatabase, đơn giản là xóa cơ sở dữ liệu.

Bây giờ, quay trở lại hoạt động chính, phương thức onCreate tiếp tục bằng cách tham chiếu đến cơ sở dữ liệu gạch chân có thể được sử dụng để đọc và viết.

Tiếp theo, nó gọi phương thức clearAll, chỉ xóa mọi bản ghi trong cơ sở dữ liệu.

Sau đó, mã gọi phương thức insertArtist, chèn một số bản ghi vào cơ sở dữ liệu.

Bây giờ, phương thức này trước tiên tạo một đối tượng ContentValues, sau đó đưa thông tin vào đối tượng đó tương ứng, trong trường hợp này, với tên của nghệ sĩ, Frank Sinatra.

Tiếp theo, nó chèn bản ghi vào bảng nghệ sĩ bằng cách gọi phương thức chèn.

Bây giờ, do trường ID được cơ sở dữ liệu tạo tự động, nên ứng dụng không cần phải đưa nó vào đây.

Tiếp theo, mã xóa đối tượng giá trị, và sau đó thêm bản ghi thứ hai cho Lady Gaga, bản ghi thứ ba cho Johnny Cash và bản ghi thứ tư cho Ludwig van Beethoven. Bây giờ, quay trở lại phương thức onCreate, mã gọi phương thức readArtists, đọc tất cả các bản ghi trong cơ sở dữ liệu và trả về một đối tượng Con trỏ. Con trỏ về cơ bản là một trình vòng lặp qua một tập hợp các bản ghi được trả về bởi một thao tác truy vấn. Và Con trỏ này sau đó được sử dụng để tạo Bộ điều hợp cho chế độ xem danh sách sẽ hiển thị các bản ghi này trên màn hình. Và cuối cùng, mã đặt một trình nghe trên nút Fix. Khi người dùng nhấn nút này, mã sẽ gọi phương thức sửa lỗi. Hãy cuộn xuống phương pháp đó.

Phương thức này trước tiên gọi phương thức xóa, tìm bản ghi với tên nghệ sĩ Lady Gaga, sau đó xóa nó.

Và sau đó, phương thức tạo ra một đối tượng ContentValues ​​với cách viết đúng tên của Johnny.

Sau đó, nó thực hiện một hoạt động cập nhật đầu tiên tìm thấy bản ghi với tên sai chính tả, và sau đó thay thế nó bằng chính tả.