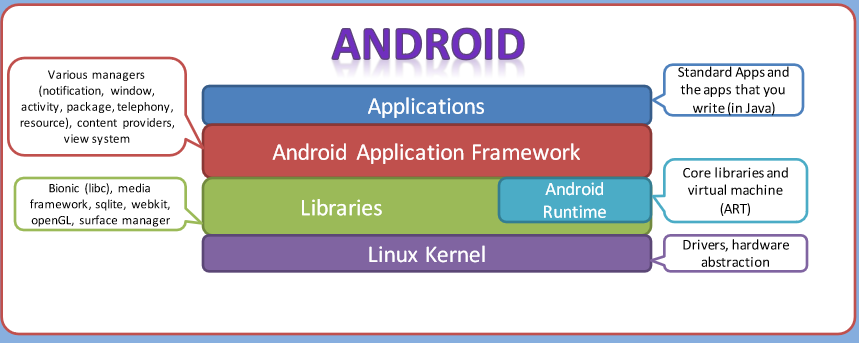
# Kiến trúc của Android

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.



* Applications : các ứng dụng bạn tạo ra sẽ được cài đặt trên lớp này
* Android application 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

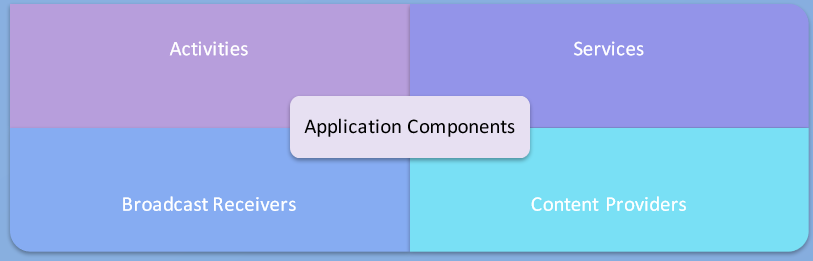
Android Framework bao gồm các dịch vụ chính sau :

* + Activity 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 sẻ 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 được sử dụng để tạo giao diện người dùng.
* Android Runtime : phần này cung cấp 1 bộ phận quan trọng là Dalvik Virtual Machine – là 1 loại java Virtual Machine được thiết kế đặc biệt để tối ưu cho Android.

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

* Libraries : tập các thư viện 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 viện chơi và ghi âm audio, video, hay các thư viện SSL chịu trách nhiệm bảo mật Internet…
* Linux kernel : lớp này cung 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 thành phần cơ bản của ứng dụng.



* Activities : giúp người dùng tương tác với hệ thống, thực hiện các chức năng cần thiết trên đó, chuyển đổi qua lại giữa các màn hình giao diện/chức năng. Thường thường khi sử dụng activity chung ta sẽ kế thừa từ lớp cha của nó là Activity.
* Services : giúp ứng dụng vẫn chạy được, nhưng không cần hiển thị trên giao diện ( chạy ngầm bên dưới). chúng ta cso thể liên kết/kết nối giữa một Activity với một service (vd khi download một file từ trên mạng, việc download thực hiện ở service, sau đó sẽ trả kết quả phần tram download trên activity để hiển thị cho người dùng biết.
* Content provider : quản lý các cách để ứng dụng có thể lưu trữ dữ liệu trên hệ thống. chúng ta se biết cụ thể về thành phần này khi xây dựng các ứng dụng cần lưu trữ vào SQLite. Vidu : các ứng dụng từ điển, các bạn sẽ thấy dữ liệu, từ vựng chung ta tra hiển thị, thì dữ lieu hiển thị đó được lưu trữ trong Slite và Content provider gọi để lấy ra cho người dùng xem. Ngoài ra thành này còn thực hiện các chức năng thêm, sửa, xóa dữ liệu…
* Broadcast receiver : vd: ta có thể chuyển dữ liệu từ service lên activity chung ta có thể sử dụng broadcast để gửi dữ liệu. hoặc trong các ứng dụng như hẹn giờ, khi đến giờ hẹn, ứng dụng sẽ sử dụng broadcast báo thức, tạo ra notification trên màn hình để báo cho người dùng biết.

# Cho biết các loại đối tượng sử dụng để xây dựng giao diện người dùng và cách làm việc với giao diện người dùng.

Giao diện người dùng và hoạt động

* Trong mỗi hoạt động có giao diện người dùng (UI) được liên kết.
* Giao diện người dùng được chỉ định trong file layout
* File layout được cung cấp dưới dạng tham số cho lệnh gọi phương thức setContentView() trong onCreate() của hoạt động

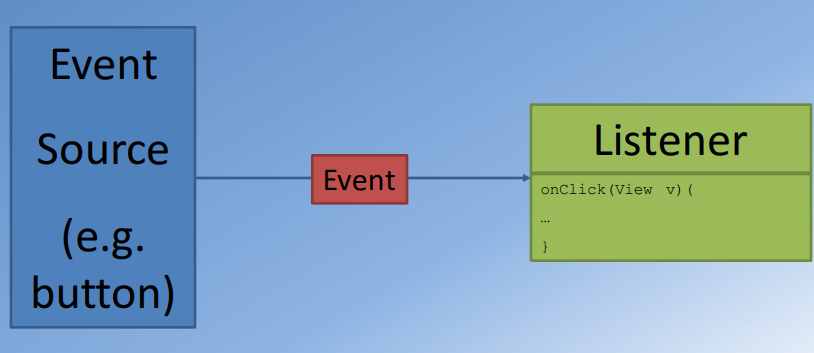
Giao diện người dùng được xây dựng bằng cách sử dụng view và các đối tượng viewgroup

* View :
  + Base class for widgets.
  + TextView, EditText views, Buttons…
* Viewgroud
  + Base class for layout
  + Linear, Relative, Tabular…
* View Hierarchy
  + Hirarchy of views and viewgroups

UI được định nghĩa trong file layout trong thư mục resources

* Hai cách xem layout :
  + Visual view dùng design tab
  + XML code view sử dụng text tab
* Các widget được cung cấp một ID để cho phép chúng tôi tham chiếu chúng trong mã JAVA
  + R.id<widget\_ID>

# Trình bày mô hình lập trình hướng hướng sự kiện



Sự kiên UI là một hành động được kích hoạt bởi người dùng tương tác với UI

**An UI event is an action that is triggered by the user interacting with the UI**

**– A button pressed or released**

**– A key is pressed or released**

**– An area of a touch screen is touched, etc.**

**– Gestures**

* Button hoặc phát hành
* Một phím được nhấn hoặc phát hành
* Một khu vực của màn hình cảm ứng được chạm…
* Cử chỉ

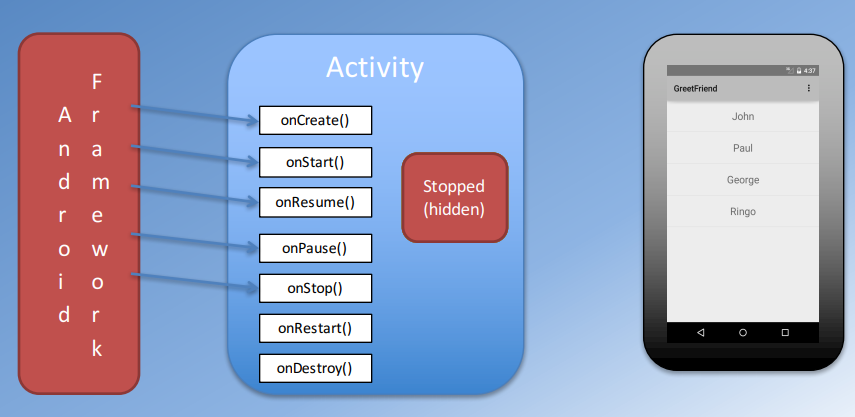
**Event Listeners**

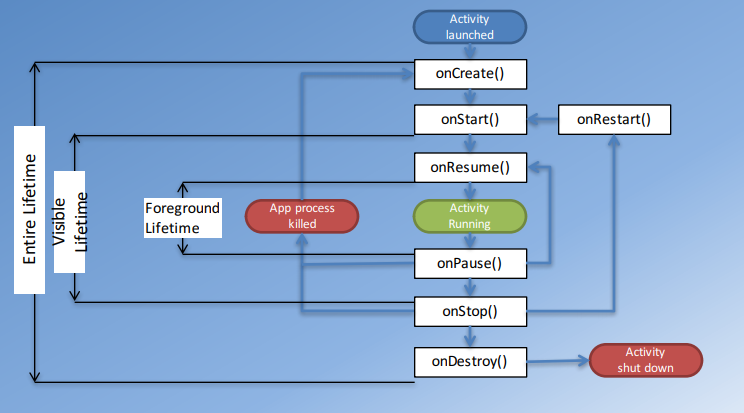
* View class định nghĩa giao diện có chứa một phương thức callback
* Hoạt động thực hiện phương thức gọi lại giao diện
* Gọi bởi Android Framework khi View mà listeners đã đăng ký được kích hoạt bởi sự tương tác của người dùng với mục trong UI
* Một số phương thức callback có sẵn : onClick(), onLongClick(), onFFocusChange(), onKey(), onTouch(), onCreateContextMothy()

**Resource**

Tài nguyên là các tệp và nội dung bổ sung độc lập với mã nguồn • Được sử dụng bởi mã khi triển khai ứng dụng • Tài nguyên bao gồm: bố cục, hình ảnh, menu, chuỗi, biểu tượng trình khởi chạy, hoạt hình • Tách tài nguyên khỏi mã cho phép tùy chỉnh ứng dụng đối với các thiết bị, ngôn ngữ, ngôn ngữ khác nhau - Mã có thể sử dụng các tài nguyên phù hợp dựa trên cấu hình

# Trình bày vòng đời của Activity

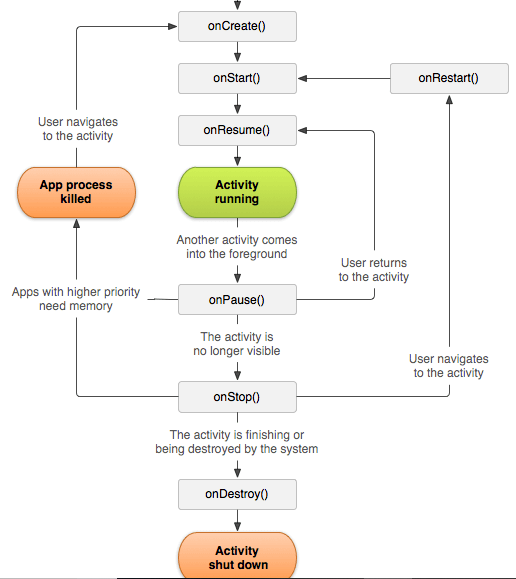




* Activity là một mã Java có hỗ trợ một màn hình hoặc giao diện người dùng. Nói cách khác, xây dựng khối của giao diện người dùng là một Activity.

Activity là một lớp được có sẵn trong Android và tất cả các ứng dụng trong đó có giao diện người dùng phải kế thừa nó để tạo ra cửa sổ

* Vòng đời gồm :
  + Bắt đầu từ lúc Android khởi tạo thành phần để phản hồi ý định đến cuối khi cá thể bị phá hủy
  + Thành phần đôi khi có thể hoạt động hoặc không hoạt động.
  + Các hoạt động có thể được hiển thị cho người dùng hoặc vô hình

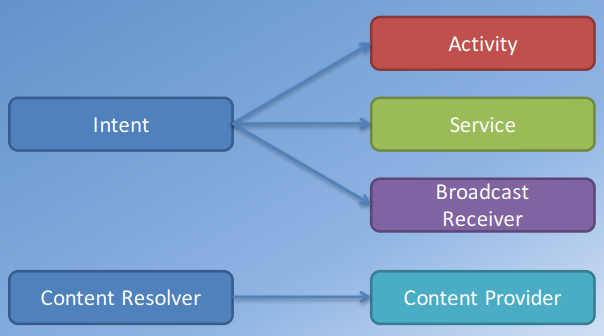


Khi một Activity được khởi tạo nó sẽ gọi hàm OnCreate() đầu tiên, ở đây nó sẽ vẽ lên giao diện của màn hình đó lúc này giao diện vẫn chưa hiển thị ra. Và tiếp sau đó nó se gọi đến hàm OnStart(), lúc này giao diện đã được vẽ ra rồi và bạn có thể nhìn thấy được giao diện đó tuy nhiên vẫn chưa tương tác vào được, ví dụ như chưa click vào button đăng ký chẳng hạn. tiếp theo nó sẽ gọi đến hàm OnResume(), ở đây giao diện đã được khởi tạo xong và người dùng 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 :

* Activity bị che mất 1 phần : 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 gờ chúng ta có thể thao tác, click.. được trên Activity này.
* Activity bị che mất toàn phần : nếu như một cuộc gọi đến chặng 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 sau đó, nghĩa là Activity hiện tại bị che khuất hoàn toàn rồi. sau khi cuộc gọi kết thúc thì hàm OnRestart() sẽ được goi, tiếp đến hàm OnStart() và OnResume() cũng được gọi ngay sau đó luôn và bây giờ lại quay về trạng thái hoạt động bình thường.

Một hàm cuối cùng đó là OnDestroy(), hàm này sẽ được gọi khi bạn bấm nút back trên đt để thoát ứng dụng hoặc tắt đi một màn hình activity nào đó. Sau khi hàm này được gọi thì Activity sẽ bị hủy.

# Intents là gì? Cho biết các loại Intents



* Là những tin nhắn không đồng bộ cho phép các thành phần ứng dụng yêu cầu chức năng từ các thành phần Android khác. Intents cho phép bạn tương tác với các thành phần từ các ứng dụng như nhau cũng như với các thành phần đóng gói trong một ứng dụng khác. vd : một activity co thể gọi một Activity bên ngoài để chụp ảnh.

Intent co thể sử dụng để :

Bắt đầu một Activity, bắt đầu một Activity con, Bắt đầu một dịch vụ (Service)

* Có 2 loại Intent là Intent tường minh (explicit) và Intent không tường minh (implicit)
  + Intent tường minh (explicit ) : là những ý định chỉ định rõ ràng tên của các thành phần mục tiêu để xử lý; trong đó, trường mục tiêu được set một giá trị cụ thể thông qua các phương thức setComponet() hoặc setClass().
  + Intent không tường minh (implicit) : là những ý định không chỉ rõ một mục tiêu thành phần, nhưng bao gồm đầy đủ thong tin cho hệ thống để xác định các thnafh phần có sẵn là tốt nhất để chạy cho mục đích đó. Hãy xem xét một ứng dụng liệt kê các nhà hang có sẵn ở gần bạn. khi bạn bấm vào một tùy chọn nhà hang cụ thể, ứng dụng sẽ hỏi một ứng dụng khác để hiển thị các tuyết đường đến nhà hàng đó. Để đạt được điều này, nó có thể gửi một ý định rõ ràng trực tiếp đến các ứng dụng Google máp, hoặc gửi ý định ngầm, ý định sẽ được chuyển giao cho bất kỳ ứng dụng nào cung cấp các tính năng bản đồ.

# Trình bày cách thức vẽ các đối tượng đồ họa 2D với Canvas

* Thích hợp cho các ứng dụng yêu cầu vẽ and/or điều khiển hoạt hình đồ họa
* Canvas cung cấp giao diện cho bề mặt thực tế mà đồ họa có thể được vẽ
  + Vẽ vào môt bipmap được đặt vào cửa sổ
* Canvas class cung cấp một tập hợp các phương thức vẽ : drawBipmap(), drawRect(), drawText()
* Đối với các ứng dụng không yêu cầu tốc độ xử lý hoặc tốc độ khung hình đáng kể, hãy tào thành phần chế độ xem tùy chỉnh và vẽ lên đó bằng Canvas trong View.onDraw()

2D graphic trên View

Mở rộng lớp View và định nghĩa phương thức gọi lại onDraw()

* Sử dụng phương thức onDraw() để thực hiện tất cả các cuộc gọi để vẽ trên khung vẽ
* Android Framework gọi onDraw() để cập nhật màn hình
* Sử dụng invalidate() để kích hoạt chế độ xem của bạn được vẽ
* Bên trong onDraw() của thành phần chế độ xem của bnaj, sử dụng các phương thức Canvas.draw…() khác hoặc các phương thức class draw() lấy Canvas của bạn làm đối số
* Khi onDraw() của bạn hoàn tất, khung Android sẽ sử dụng Canvas của bạn để vẽ Bipmap do hệ thống xử lý

# Cho biết ưu điểm của SurfaceView và cách thức sử dụng để vẽ các đối tượng đồ họa 2D

SurfaceView là mọt lớp con đặc biệt của View cung cấp một bề mặt vẽ chuyên dụng trong hệ thống phân cấp View

* Cung cấp bề mặt bản vẽ này cho luồng phụ của ứng dụng, do đó ứng dụng không bắt buộc phải đợi cho đến khi hệ thống phân cấp View của hệ thống sẵn sàng để vẽ.
* Thay vào đó, một luồng thứ cấp có tham chiếu đến SurfaceView có thể vẽ lên Canvas của chính nó theo tốc độ của riêng nó.

Trong mã của bạn mở rộng lớp SurfaceView

* Cần triển khai SurfaceHolder.Callback
* Ngoài ra giao diện ueeu cầu ba phương thức bổ sung được triển khai : SurfaceCreated(), SurfaceChanged(), và SurfaceDestroyed()
  + Quan trọng để bạn biết khi nào bạn có thể bắt đầu vẽ, liệu bạn có cần điều chỉnh dựa trên các thuộc tính bề mặt mới hay không và khi nào dừng vẽ và có khả năng giết một số tác vụ.
* Surface Object được xử lý thông qua SurfaceHolder mà bạn nhận được khi khởi tạo SurfaceView, bằng cách gọi getHolder()
* Thông báo cho SurfaceHolder rằng bạn sẽ xử lý các cuộc gọi lại bằng cách gọi Holder.addCallback(this). Sau đó, ghi đè các phương thức gọi lại trong SurfaceView của bạn : SurfaceCreated(), SurfaceChanged() và SurfaceDestroyed()
* Tạo một luồng thức cấp sẽ điều khiển tất cả các bả vẽ lên bề mặt và truyền cho SurfaceHolder
  + Trong chuỗi, giữ khung vẽ : SurfaceHolder.lockCanvas()
  + Sau đó thực hiện sửa đổi cho khung vẽ
  + Cuối cùng, SurfaceHolder.unlockCanvasAndPost(canvas)

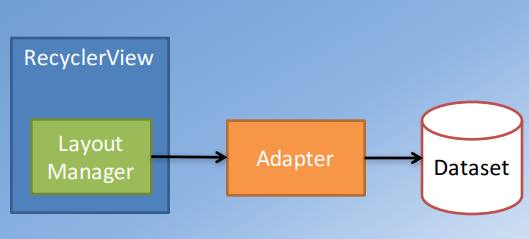
Trong trường hợp chúng ta cần hiển thị một thứ gì đó thay đổi liên tục như màn hình hiển thị camera, đồ họa game… thì chúng ta nên sử dụng lớp SurfaceView, SurfaceView cung cấp cho lập trình viên 1 mặt phẳng để có thể vẽ liên lục lên mặt phẳng đó thông qua việc can thiệp tới từng điểm ảnh trên mặt phẳng.

Surfaceview cung cấp cho lập trình viên 1 SurfaceHolder để tiến hành các thao tác thay đổi trên khung hình. Dù là tiện lợi và đáp ứng được đa số các ứng dụng cần sự thay đổi liên tục trên màn hình như (stream video, custom camera…) nhưng SurfaceView có 1 số hạn chế :

# Trình bày việc sử dụng Drawable để vẽ các đối tượng đồ họa 2D

* Drawables để vẽ hình dạng và hình ảnh cho View
  + Class có thể vẽ
  + Các lớp con : BipmapDrawable, ShapeDrawable, PictureDrawable, LayreDrawable, v.v.
* Drawables có thể được xác định và khởi tạo theo ba cách :
  + Sử dụng một hình ảnh được lưu trong project resource của bạn
  + Sử dụng tệp XML xác định thuộc tính Drawable
  + Sử dụng các hàm tạo lớp bình thường trong code
* Drawable Resources thường được lưu trữ trong thư mục res/drawable của project của bạn.
* Drawables có thể có nhiều hình thức:
  + Bipmap : hình ảnh đơn giản nhất có thể vẽ, hình ảnh PNG hoặc JPEG
  + Nice Patch : một phần mở rộng cho định dạng PNG cho phép nó chỉ định thông tin về cách kéo dài nó và đặt những thứ bên trong nó.
  + Shape : chứa các lệnh vẽ đơn giản thay vì raw bipmap, cho phép nó thay đổi kích thước tốt hơn trong một số trường hợp.
  + Layer : một drawable hợp chất, mà draws nhiều ngăn kéo nằm bên dưới nhau
  + States : một hơp chất drawable chọn một trong các tập hợp drawable dựa trên trạng thái của nó
  + Level : một hợp chất drawable chọn một trong các bộ drawable dựa trên cấp độ của nó
  + Scale : môt hợp chất drawable với một đứa trẻ duy nhất drawable, có kích thước tổng thể được sửa đổi dựa trên cấp độ hiện tại

# Trình bày mô hình sử dụng RecyclerView



Layout Manager để xem định vị

* LinearLayoutManager
* GridLayoutManager
* StaggeredGridLM

Hoạt ảnh mặc định

Triển khai RecyclerView.Adapter

Câu 16 : đặc điểm và phương thức tiêu biểu của Sysnctask

sysncTask đơn giản hóa việc tạo các tác vụ dài hạn cần giao tiếp với UI, quản lý thread, phải được tạo trên thread UI

* Phương thức tiêu biêu của SysncTask :
  + doInBackground() thực thi tự động trên luồng worker
  + onPreExecute(), onPostExecute() và onProgressUpdate() được gọi trên thread UI
  + call publishProgress() bất cứ lúc nào trong doInBackground() để thực hiện onProgressUpdate() trên thread UI
  + giá trị được trả về bởi doInBackground() được gửi đến onPostExecute()
  + bạn có thể hủy tác vụ bất cứ lúc nào, từ bất kỳ threat
* các tính năng :
  + khả năng trả về các giá trị của loại hình tùy chỉnh cho thread UI khi tác vụ kết thúc
  + khả năng thực thi một số mã trong thread UI trước khi tác vụ background bắt đầu thực thi và sau khi hoàn thành
  + khả năng push update lên thread UI trong khi thực hiện tác vụ background
  + quản lý luồng tự động under-the-hood

Phần thực hành

Bài 3 :

* Lớp data, lớp này sẽ được sử dụng để gán giá trị cho Recyclerview
* Lớp Adapter là lớp đóng vai trò quan trọng trong việc xử lý dữ liệu trong Recyclerview.RecyclerView.Adapter
* Đảm bảo thay đổi giá trị trả về bên trong phương thức ghi đè (getItemCount) thành kích thước danh sách thức tế, bởi vì nếu bạn không làm điều đó, bạn sẽ không thể thấy bất kỳ dữ liệu nào trong Android Recyclerview.
* Android linearLayoutManager, trình quản lý bố cục này được sử dụng cùng với Recyclerview của android để xác định xem bạn muốn cho phép người dùng cuộn théo hướng dọc hay ngang.
* Android DividerItemDecor, trang trí vật phẩm này được sửa dụng để vẽ bên dưới mỗi mục trong Recyclerview

# Trình bày các hỗ trợ đa phương tiện

Android cung cấp chức năng đa phương tiện toàn diện :

* Âm thanh : tất cả các định dạng chuẩn bao gồm MP3, Ogg, Midi,..
* Video: MPEG-4, H.236, H.264
* Hình ảnh : PNG(ưu tiên), JPEG, GIF (không được đề xuất)

Một số định dạng phương tiện và codecs khác nhau được hỗ trợ

Bạn có thể phát âm thanh hoặc video từ :

* Các media files được lữ trữ trong resources của ứng dụng (raw resources)
* Media file trong files system (thẻ SD)
* Luồng dữ liệu đến qua kết nối mạng

MediaPlayer class

* Phát âm thanh/video lâu dài
* API chính cho âm thanh và video

SoundPool class

* Những tiếng nổ ngắn

Quản lý âm thanh

* Quản lý đầu ra âm thanh và nguồn âm thanh

# Các phương thức và đặc điểm tiêu biểu của lớp MediaPlayer

* Create a media player :

MediaPlayer player = new MediaPlayer();

* Chỉ định nguồn của media:
  + Nếu file trong thư mục raw resources : player = MediaPlayer.create(context, R.raw.music\_file);
  + Nếu file hoặc stream : player.setDataSource(URL);

Player.prepare(); hoặc Player.prepareAsync();

* + - Sử dụng readyAsync() khi media cần được prepare hoặc buffered offline
    - Sẽ return khi media sẵn sàng
    - Yêu càu triển khai giao diện OnPreparedListener():phương thức onPrepared() cần được thực hiện.

API phong phú được thiết kế để phát các luồng phương tiện dài, chẳng hạn như bản ghi âm thanh, nhạc và video

* Tìm kiếm hoạt động được hỗ trợ
* Source buffering (bộ đệm nguồn)

Heavyweight resource-wise

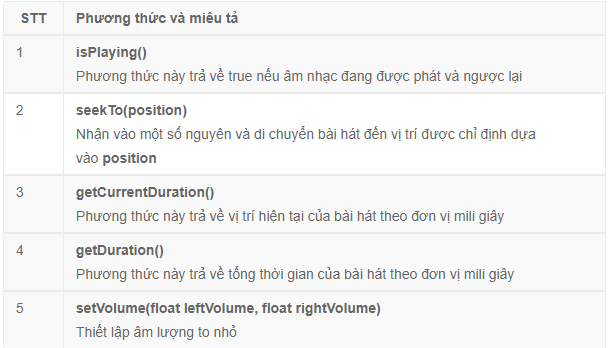
Slow to initialize (khởi tạo chậm)

Không phù hợp với các tình huống có độ trễ thấp như mẫu âm thanh ngắn cho hiệu ứng âm thanh, chẳng hạn như trong các trò chơi.

Được thiết kế cho các tình huốn không quá một hoặc hai MediaPlayer làm việc cùng một lúc.

Android cung cấp nhiều cách để điều khiển tập tin autio hoặc video. Một trong những cách này là sử dụng lớp MediaPlayer. Xử lý media player trong android chính là việc phát các tập tin audio/video… Để sử dụng lớp MediaLayer trong android, chúng ta phải gọi phương thưc create(). Phương thức này giúp chúng ta tạo một thể hiện của lớp MediaPlayer.

Phương thức start() sẽ được phát audio ngay từ đầu khi gọi phương thức này. Nếu phương thức start() được gọi sau phương thức pause() thì audio sẽ được phát từ chỗ nó bị dừng chứ không phải được phát từ đầu. để phát audio ngay từ đầu, chúng ta phải sử dụng phương thức reset(). Ngoài ra MediaPlayer còn cung cấp các phuơng thức khác nhau như :



<https://o7planning.org/vi/10487/huong-dan-android-mediaplayer-va-videoview#a1247663>

# Các đặc điểm của lớp SoundPool

* SoundPool class quản lý và phát tài nguyên âm thanh(resource audio) cho các ứng dụng
* soundPool là tập hợp các mẫu có thể được tải vào bộ nhớ từ tài nguyên bên trong APK hoặc từ file trong file system
* soundPool libarary sử dụng MediaPlayer service để giải mã âm thanh thành luồng âm thanh đơn hoặc âm thanh PCM 16 raw
* điều này cho phép các ứng dụng vân chuyển với các luồng nén mà không phải chịu tải CPU và độ trễ giải nén trong khi phát lại
* kịch bản sử dụng điển hình bao gồm các trò chơi cso âm thanh trong đó một số âm thanh được sử dụng ở cấp độ và đôi khi có thể phải chơi với sự chồng chéo.

# Trình bày định dạng dữ liệu JSON

JSON – javaScript Object Notation là một chuẩn để định dạng dữ liệu. JSON cũng như XML, nó chỉ mô tả dữ liệu, không có khả năng trình bày, không có khả năng tự chạy, nói chung là thụ động. ta chỉ có thể nhận JSON rồi làm gì đó, hoặc ta có gì đó rồi chuyển thành JSON để gửi đi, chứ ta không thể chạy JSON.

Json có 5 kiểu dữ liệu chính

* Number : kiểu số bao gồm số nguyên và số thực.
* String : kiểu chuỗi, nội dung bao bởi cặp dấu nháy kép “, những ký tự đặc biệt được escape bởi dấu \. Theo chuẩn JSON thì không sd dấu nháy đơn để bọc chuỗi.
* Boolean : kiểu luận lý bao gồm giá trị true, false
* Array : kiểu mảng, gồm các phần tử phân cách nhau bởi dấu phẩy ‘,’ và mảng được bao bởi dấu []
* Object : kiểu đối tượng, gồm những cặp giá trị đi cùng nhau, mỗi cặp phân cách bởi dấy ‘,’ đối tượng được bao bởi cặp dấu {}, cặp giá trị bao gồm tên, giá trị được phân cách bởi dấu ‘;’
* Null : giá trị null.

Ưu điểm :

* Có thể đọc hiểu được
* Là kiểu dữ liệu trên nền cơ sở Javascript nên dễ dàng tiếp cận
* Dữ liệu truyền tải ngắn gọn so với những định dạng dữ liệu khác như xml, html…
* Dễ dàng chuyển đổi dữ liệu từ dạng chuỗi sang dữ liệu có thể sử dụng được.
* Dễ truy cập nội dung

So sánh với XML

* Cú pháp ngắn gọn, đơn giản , tiết kiệm hơn XML
* Có cách phân tích xử lý đơn giản, chỉ qua 1 bước (chuyển chuỗi nhận được từ máy chủ sang dữ liệu có thể sử dụng được)

# Trình bày mô hình thực hiện công việc ngầm

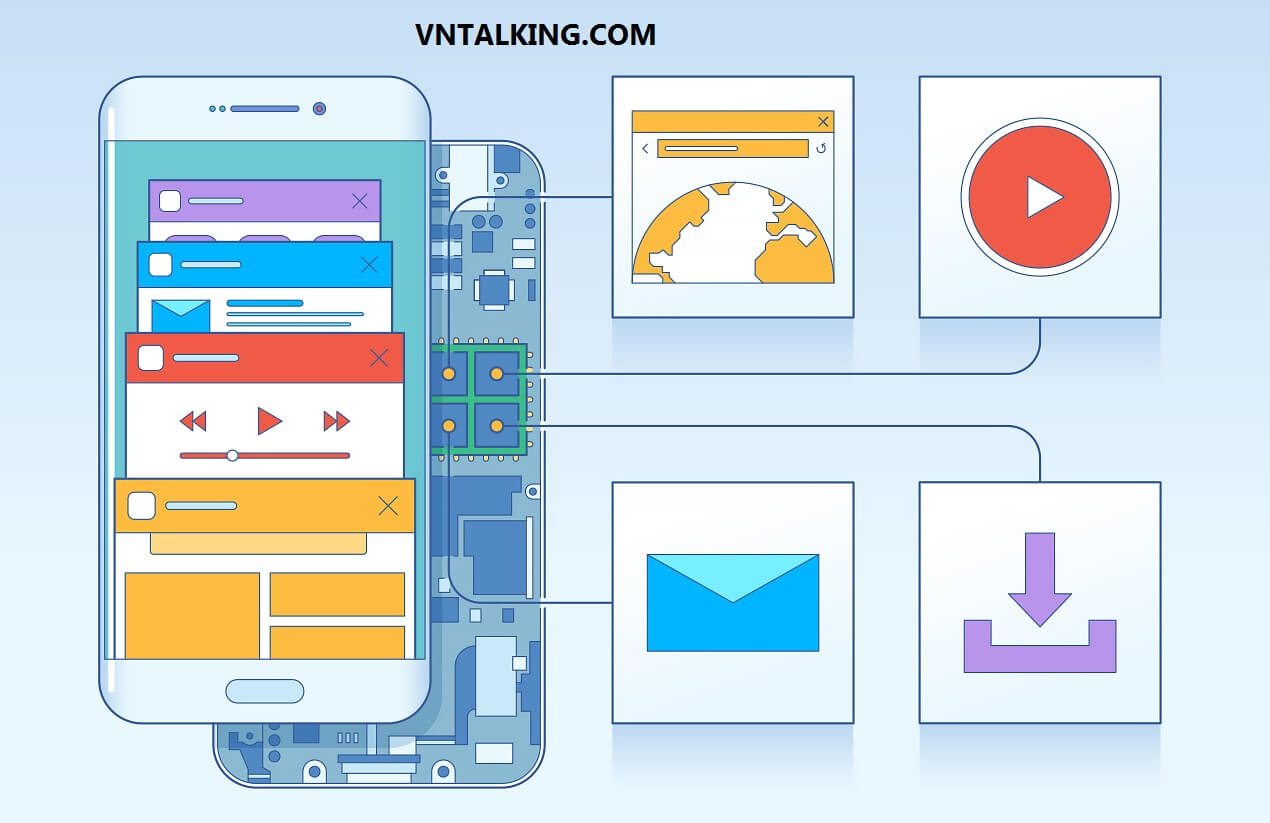
# Trình bày các đặc điểm và phương thức tiêu biểu của AsyncTask

Xử lý đa nhiệm trong Android

Nếu các sự kiện hoặc một task nào đó không được xử lý đồng thời. Thì toàn bộ mã của ứng dụng Android sẽ chạy trên luồng chính và code sẽ được thực hiện tuần tự từng dòng một.

Giả sử nếu bạn thực hiện một công việc/ tác vụ cần thời gian xử lý như tải nhạc từ Internet, ứng dụng sẽ hiển thị trạng thái treo cho đến khi tải xong.

Để mang lại trải nghiệm người dùng tốt, tất cả tác vụ có khả năng chạy chậm đều phải chạy không đồng bộ.



Mình có thể tạm liệt kê một số tác vụ cần thời gian xử lý như:

Truy cập tài nguyên (như MP3, JSON, Hình ảnh) từ Internet

Thao tác với cơ sở dữ liệu

Tương tác với webserice như [RESTFUL](https://vntalking.com/xay-dung-nodejs-restful-apis-2018.html), SOAP…

Các Logic phức tạp mất khá nhiều thời gian như: Nén/giải nén file, sao chép/di chuyển file trong bộ nhớ…

Và còn rất nhiều các trường hợp khác cần phải xử lý bất đồng bộ, đa nhiệm khác nữa. Tùy vào ứng dụng của bạn như thế nào mà ứng biến cho phù hợp.

Bài viết này mình sẽ tập trung giới thiệu cho các bạn các sử dụng AsyncTask trong Android, một khái niệm của Android và cũng được khuyên dùng khi cần xử lý tác vụ nặng, cần thời gian.

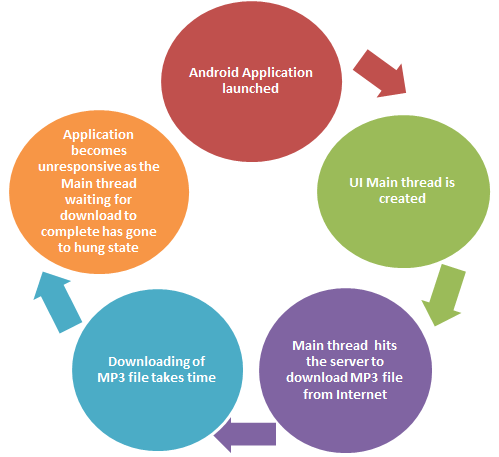
AsyncTask trong Android là gì?

AsyncTask là một abstract Android class, giúp ứng dụng Android xử lý main UI thread hiệu quả hơn. AsyncTask trong Android cho phép chúng ta thực hiện những tác vụ dài mà không ảnh hưởng đến main thread

Khi nào thì sử dụng AsyncTask?

Để dễ hình dung, mình giả sử bạn tạo một ứng dụng Android để tải xuống tệp MP3 từ Internet.

Sơ đồ trạng thái dưới đây cho thấy một loạt các hoạt động sẽ diễn ra khi bạn chạy ứng dụng



Trong khi chờ nhận file MP3 từ máy chủ, ứng dụng sẽ bị treo vì main thread vẫn đang chờ tác vụ tải xuống hoàn tất.

Để khắc phục điều này, chúng ta có thể tạo thread mới và thực hiện các tác vụ trên thread mới đó. Do đó giao diện người dùng sẽ không bị ảnh hưởng và treo nữa

Nhưng việc xử lý với thread riêng biệt có thể tạo ra một số vấn đề như việc cập nhật giao diện người dùng. Bạn sẽ cần phải cập nhập trạng thái download được bao nhiêu % file đó, và khi kết  thúc tải thì cũng phải cập nhập cho người dùng biết. Nếu bạn sử dụng Thread đơn giản của java thì việc cập nhập này sẽ khá phức tạp.

Android đã xem xét tất cả các vấn đề này và tạo một lớp chuyên dụng có tên là **AsyncTask**.

Cách triển khai AsyncTask trong Android?

Tạo một class mới bên trong Activity và kế thừa từ AsyncTask như dưới đây

private class DownloadMp3Task extends AsyncTask<URL, Integer, Long> {

protected Long **doInBackground**(URL... urls) {

//Yet to code

}

protected void **onProgressUpdate**(Integer... progress) {

//Yet to code

}

protected void **onPostExecute**(Long result) {

//Yet to code

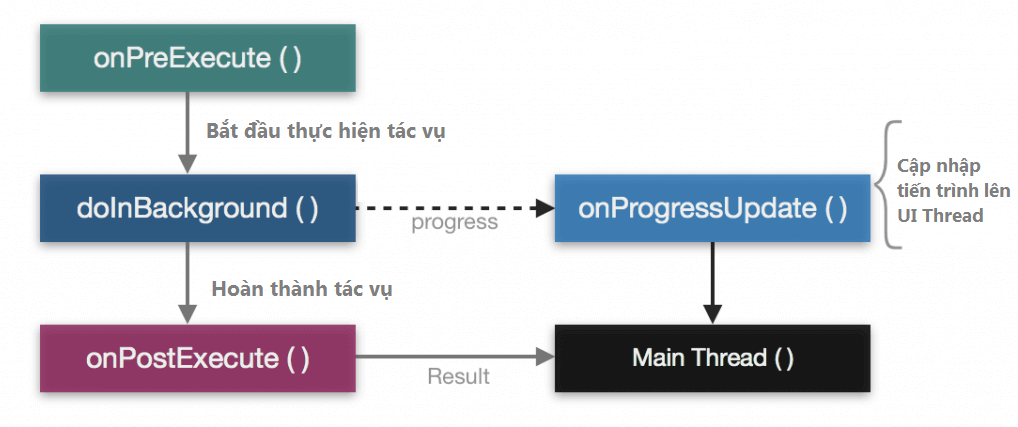
}

}

Để thực thi tác vụ, đơn giản bằng cách gọi phương thức execute

**new** **DownloadMp3Task**().execute(mp3URL);

Bản chất **Asynctask** gồm có 4 bước:



**Bước 1:** **onPreExecute**()

Được thực hiện trước khi bắt đầu thực hiện tác vụ. Hàm được gọi trước phương thức doInBackground() và được gọi trên UI thread.

Thông thường, hàm này được dùng để hiển thị thanh progressbar thông báo cho người dùng biết tác vụ bắt đầu thực hiện

**Bước 2**: **doInBackground**()

Tất cả code mà cần thời gian thực hiện sẽ được đặt trong hàm này.  Vì hàm này được thực hiện ở một thread hoàn toàn riêng biệt với UI thread nên bạn không được phép cập nhật giao diện ở đây.

Để có thể cập nhập giao diện khi tác vụ đang thực hiện. Ví dụ như cập nhập trạng thái % file đã download được, chúng ta sẽ phải sử dụng đến hàm bên dưới onProgressUpdate()

**Bước 3:** **onProgressUpdate**()

Hàm này được gọi khi trong hàm **doInBackground**()gọi đến hàm **publishProgress**()

**Bước 4**: **onPostExecute**()

Hàm này được gọi khi doInBackground hàm thành công việc. Kết quả của doInBackground() sẽ được trả cho hàm này để hiển thị lên giao diện người dùng.

Trong quá trình Asynctask thực hiện tác vụ, bạn hoàn toàn có thể tạm dừng bất kể lúc nào mà không cần phải đợi AsyncTask làm xong. Đơn giản là bạn gọi hàm cancel(boolean)

**Một số lưu ý về các sử dụng AsynctTask**

Lớp AsyncTask phải được thực hiện trên UI Thread

Hàm execute(Params…) phải được gọi trên UI Thread

Không nên gọi onPreExecute (), onPostExecute(), doInBackground (Params…), onProgressUpdate (Progress…) theo cách thủ công.

Task chỉ được thực thi một lần tại một thời điểm (Exception sẽ được throw nếu thực hiện lần thứ hai).

# Cách thức lập trình mạng