1.Activity

Một Activity được hiểu như là một màn hình thể hiện những gì đang diễn ra đang thao tác với giao diện người dùng (user interface).

1.1. Avtivity Lifecycle



- onCreate() được gọi lần đầu tiên khi activity bắt đầu tạo

- onStart() được gọi khi activity đắt đầu thực hiện cho người dùng thấy

- onResum() khi activity bắt đầu có sự tương tác với người dùng

- onPause() được gọi khi activity tạm dừng và trước đó là đã được resumed

- onStop() được gọi khi activity không còn hiện cho người dùng, tức là việc tương tác của người dùng với chương trình sẽ không còn (chuyển activity khác, nhấn phím home)

- onDestroy() khi activity bị hủy

- onRestart() khi activity bị hủy và được gọi lại restartung again

1.2. Styles and Themes

Styles

Android Style làm việc tương tự như CSS. Một Style xác định định dạng một user interface, nó có thể được áp dụng cho một individual View(từ bên trong một file Layout) hoặc một Activity, hoặc úng dụng (từ bên trong file Manifest).

Định dạng Styles: Một Style được định nghĩa trong file styles.xml (res/values/styles.xml).

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

<resources>

<style name="CustomFontStyle">

<item name="android:layout\_width">fill\_parent</item>

<item name="android:layout\_height">wrap\_content</item>

<item name="android:capitalize">characters</item>

<item name="android:typeface">monospace</item>

<item name="android:textSize">12pt</item>

<item name="android:textColor">#00FF00</item>/>

</style>

</resources>

hoặc ta có thể kế thừa từ Styles khác

<resources>

...

<style name="MyCustomTheme" parent="android:style/Theme">

<item name="android:textColorPrimary">#ffff0000</item>

</style>

...

</resources>

Themes

Theme là style nhưng được áp dụng cho toàn bộ một avtivity hay cho cả ứng dụng.

<application android:theme="@style/CustomFontStyle">

<activity android:theme="@style/CustomFontStyle">

2. Fragment

Fragment là một thành phần giao diện người dùng hoặc một hành vi của ứng dụng. Fragment có thể được cài đặt trong Activity, nó có thể cho phép thiết kế activity với nhiều module. Có thể nói fragment là một loại sub-Activity.

- Fragment cũng có layout riêng, có các hành vi và vòng đời riêng

- Chúng ta có thể thêm hoặc xóa Fragment trong một Activity trong khi Activity này đang chạy

- Có thể kết hợp nhiều Fragment trong một Activity để xây dựng giao diện người dùng đa khung

- Một Fragment có thể được sử dụng trong nhiều Activities

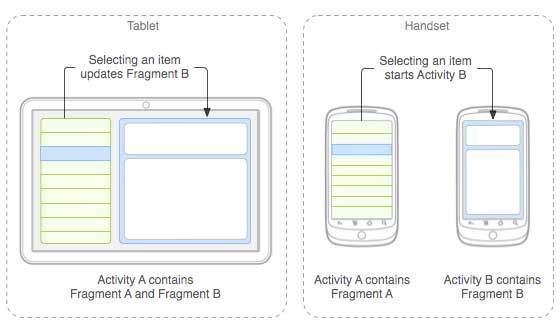
- Vòng đời của Fragment có quan hệ chặt chẽ với vòng đời của Activity đang dùng nói, điều này có nghĩa là khi Activity bị tạm dừng thì các Fragment sẽ dừng lại

- Fragment có thể thực hiện một hành vi mà không có trong thành phần user interface

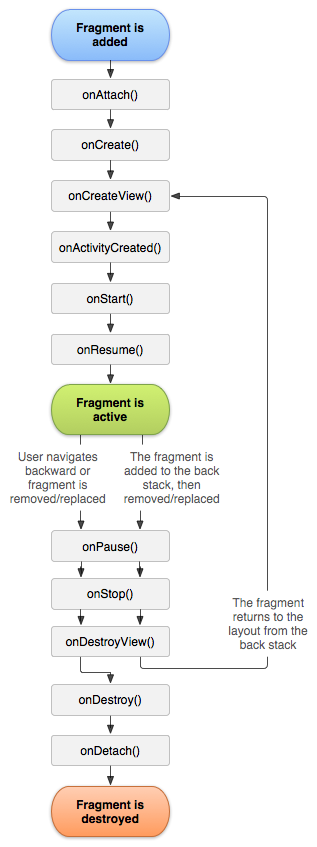
- Fragment được thêm vào từ API11 trở lên

- Có thể tạo các Fragment bằng cách kế thừa lớp Fragment và Fragment được thêm vào layout bời thẻ <fragment>

Cùng một thời điểm chúng ta chỉ có thể hiển thị một Activity duy nhất trên màn hình, vì vậy ta không thể chia màn hình thiết bị ra thành nhiều phần và kiểm soát các thành phần khác nhau này một cách riêng biệt. Nhưng với Fragment thì màn hình được linh hoạt hơn, xóa bỏ việc chỉ có duy nhất một Activity duy nhất, nhưng Activity này có thể chứa nhiều Fragment với layout, event, và lifecycle riêng.



2.1. Life Cycle

Fragment có vòng đời rất giống với vòng đời của một Activity.

Giai đoạn 1: Fragment được khởi tạo

1. onAttach()

2. onCreate()

3.onCreateView()

4.onActivityCreated()

Giai đoạn 2: Fragment được hiển thị

1. onStart()

2. onResum()

Giai đoạn 3: Fragment chạy ẩn (background mode)

1. onPause()

2. onStop()

Giai đoạn 4: Fragment bị destroy

1. onPause()

2. onStop()

3. onDestroyView()

4. onDestroy()

5. onDectach()

2.2. Methods

- onCreate(): hệ thống gọi phương thức này khi tạo Fragment. Bạn nên khởi tạo các thành phần thiết yếu của Fragment mà bạn muốn giữ lại khi Fragment được tạm dừng hoặc sau đó được tiếp tục.

- onCreateView(): hệ thống gọi phương thức này khi Fragment vẽ giao diện của chính nó lần đầu tiên. Để vẽ giao diện cho Fragment bạn cần phải trả về một thành phần View từ phương thức này. Trả về null nếu Fragment không cung cấp giao diện.

- onPause(): hệ thống gọi phương thức này như là để đánh dấu lần đầu người dùng rời Fragment.

3. Services

Service là một trong bốn compoment cơ bản của android (services, antivities, content providers, broadcast receivers).

Service chạy ẩn ở dưới để thực hiện các thao tác mà không cần tương tác người dùng.

Một service có 2 loại:

- Started: một Service được bắt đầu giống như compoment khác như activities, bắt đầu service bằng cashc gọi phương thức startService(). Thông thường, một “start service” thực hiện một hành động đơn lẻ và không trả về kết quả cho đối tượng.

- Bound: một service được ràng buộc khi các component ràng buộc thông qua phương thức bindService(). Service ràng buộc thường là kiểu giao diện client-server, nó cho phép các component tương tác với service, gửi yêu cầu, nhận kết quả trả về. Một service ràng buộc có thể chạy với nhiều component ràng buộc đến nó khi chạy lần đầu tiên, khi tất cả các component của chúng không còn ràng buộc nữa thì service sẽ bị hệ thống hủy.



Để tạo một service bạn cần tạo một class kế thừa từ class Service hoặc một subclass. Một số phương thức quan trọng:

- onStartCommand(): hệ thống sẽ gọi hàm này khi một component khác hoặc một activity yêu cầu bắt đầu service bằng gọi phương thức startService(). Nếu bạn chỉ muốn sử dụng service kiểu ràng buộc thì không cần phải cài đặt phương thức này.

- onBind(): hệ thống sẽ gọi hàm này khi một component muốn ràng buộc mới service bằng cách gọi phương thức bindService(). Khi cài đặt hàm này bạn phải cung cấp giao tiếp giữa client và service bằng cách trả về iBinder. Bạn luôn luôn phải cài đặt hàm này, nếu không muốn ràng buộc thì có thể trả về null.

- onDestroy(): hệ thống gọi hàm này khi service không được sử dụng nữa và cần phải gọi phương thức này để giải phóng tài nguyên cần thiết liên quan đến service.

4. UI Layout