AndroidManifest.xml

Giao tiếp giữa app và hđh

Android App

Res/menu: \*.xml

Chứa các file giao diện menu

Res/values

Chứa các tập mã nguồn

java : \*.java

Thư mục chứa các file nguồn, file Activity

Res/layout :\*.xml

Chứa các file giao diện

Res/mipmap

Chứa ảnh

Yêu cầu cài font trong cái thư mục font nhé, cho đẹp màu mè tí.

I. Ánh xạ cho giống với web tí nhé:

1. **AndroidManifest** ⬄ Web.config
2. **Java** :

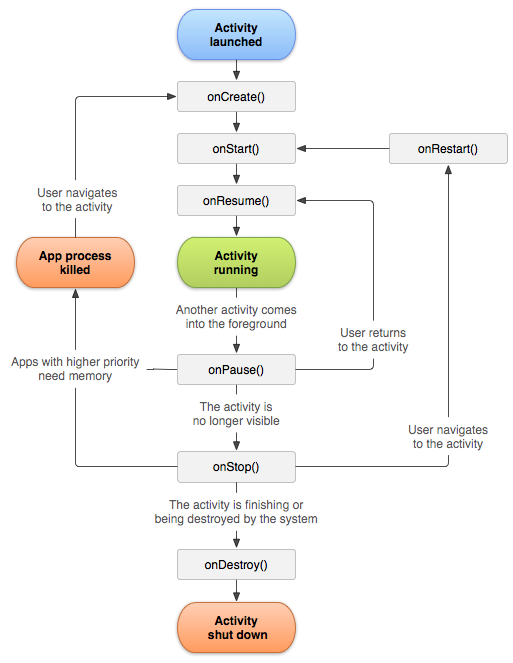
* \*Activity.java ⬄ Controller
* \*.java ⬄ Các class, code khác vv

1. **Layout** ⬄ View
2. **Menu** ⬄ View
3. **Mimmap** ⬄ Content (chứa image)
4. **Values** ⬄ Style.css

II. Một **Activity** khi tạo ra là một class và phải thừa kế những gì nó có nó để tạo ra giao diện sử dụng nó. Mỗi activity sẽ có 1 vòng đời từ khi bắt đầu đến khi kết thúc.

Mỗi quá trình của nó cần xem để code vào từng quá trình một cách hợp lý cho mỗi mục đích ứng dụng.

Tạm thời lấy hình internet để khỏi vẽ lại



Khi tạo ra theo mặc định có file **MainActivity**.**java** và file **activity**\_**main**.**xml** tương ứng là layout của nó.

Khi chạy tới **MainActivity**.**java**, activity này là một đối tượng, và đã có sự kiện **onCreate**() của chính nó. Để **MainActivity**.**java** khi khởi tạo có thể trả về một màn hình như đã thiết kế trong **activity**\_**main**.**xml** thì phải ghi đè lên hàm onCreate, sẽ dùng **@Override**

Đây là code:

@Override  
protected void onCreate(Bundle saveInstanceState){

super.onCreate(saveInstanceState);  
setContentView(R.layout.activity\_main);

}

Có thể thấy là code trong sự kiện **onCreate** đã chạy lệnh tạo một **ContentView** (giao diện đi cho dễ hiểu) là từ file **activity\_main.xml**. Như vậy có thể hiểu là, không nhất thiết **MainActivity.java** phải có file **activity\_main.xml** mới chạy được mà có thể có một file **abc.xml** chúng ta cũng có thể **setContentView(R.layout.abc)** hoặc cũng có thể viết code tạo giao diện trong chính **MainActivity.java**!

* Tạo giao diện trên file xml, code thao tác trên file java. Đặt tên 2 file đó giống nhau đi cho dễ quản lý. Đừng đặt A.java mà gọi tới B.xml chi cho khổ.
* Khuyến mại cuốn sách CleanCode, mọi người cùng đọc, code thì nên đọc, sắp code cũng nên đọc, quản lý cũng nên đọc mà không code cũng nên đọc, rất có ích cho việc đặt tên con cái sau này.  
  **Ấn vô, không phải cái hình đâu!**

\* Nói thêm về việc khi tạo ra một Activity:

Khi tạo ra nó thì cái view, cái layout .xml đấy nó đã được **đăng ký** vào trong file **AndroidManifest.xml**, chưa nói tới những cái khác trong AndroidManifest nó làm thì ví dụ tạo ra 1 Activity là **Sanpham.java** và **DanhSachSanPham.xml** thì trong **AndroidManifest** sẽ có <activity android:name=”.DanhSachSanPham” /> … thử xóa nó xem, chạy tới cái nào gọi DanhSachSanPham là lỗi liền.

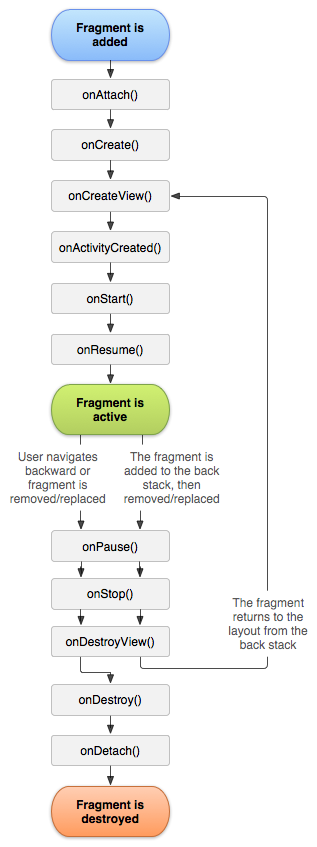
III. Nói về **Fragment** nhé:

**Fragment** nó cũng giống như **Usercontrol, PartialView** vv đại loại thế và nó có thể sử dụng trong một Activity, đọc tài liệu thì người ta viết là “có thể hiểu là sub-activity” => nó có thể sử dụng như một Activity!

Copy-Paste:

* Fragment cũng có layout của riêng của nó, cũng 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 nó 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 giao diện người dùng.
* Fragment được thêm vào API 11 trở lên.
* Bạn có thể tạo các Fragments bằng cách kế thừa lớp Fragment và Fragment được thêm vào layout bởi thẻ <fragment>

Thêm cái ảnh vòng đời của một Fragment vào, khá dài :



IV. Intent, Shared Preferences

Sau khi tạo ra MainActivity, SecondActivity, ThirdActivity vv… Giờ muốn tương tác với nhau, trao đổi dữ liệu qua lại với nhau thì sao?

Chưa tìm hiểu nhiều là có cách nào để có thể truyền dữ liệu từ Activity này sang Activity khác để tương tác với nhau một cách hiệu quả và nhanh chóng, không mất công phải xuống database ví dụ như khi ấn vào Nút A ở Trang1 sẽ chạy Trang2 và có dữ liệu từ Trang1. Thì đã xem qua về 2 thằng là Intent và Shared Preferences, 2 thằng này khác nhau và sử dụng khác nhau luôn.

* Intent: Là một tin nhắn được gửi boardcast trong app của chúng ta hay với những thành phần trong hệ điều hành, background app, service vv.
* SharedPreferences: Là một kiểu lưu trữ dữ liệu theo dạng key, value. Nếu xem thì cũng na ná như cách dùng Session, Cookies vậy.

**Intent:**

Intent có **2 loại Intent**. Một cái là **tường minh (explicit intent)** là nó sẽ được chỉ định rõ ràng các mục tiêu mà nó gửi dữ liệu còn cái còn lại là **không tường minh (implicit intent)** nó sẽ không chỉ định rõ mục tiêu mà nó gửi dữ liệu tới.

Để cho dễ hiểu, trong câu hỏi đặt ra ở đầu khi muốn gửi một tin nhắn chứa dữ liệu từ MainActivity sang SecondActivity thì đã xác định mục tiêu gửi dữ liệu, sẽ dùng Intent tường minh còn không tường minh ví dụ khi ấn vào nút Gửi mail chẳng hạn, nó sẽ đưa dữ liệu là nội dung email nhưng mục tiêu của nó không xác định, có thể là dùng Gmail, Outlook vv và sẽ hiện lên cho mình chọn.

Chưa nghiên cứu nhiều về cái không tường minh, còn cái tường minh thì để sử dụng thì làm như sau:

Trên **MainActivity.java**

//Intent tenbien = new Intent(MainActivity, SecondActivity);

Intent sendIntent = **new** Intent(**MainActivity**.**this**, **SecondActivity**.**class**);

Intent này chưa có bất cứ dữ liệu nào, để truyền dữ liệu vào nó thì trước tiên chúng ta sẽ biết là Intent có thể đưa dữ liệu vào theo kiểu **key, value** hoặc một **Bundle** (cứ hiểu như ngày xưa là Session chứa được dữ liệu kiểu int, boolean hoặc một Datatable :v )

String ten = “Park Chym Sung”

int tuoi = 18;

String gioiTinh = “FyMeu”;

String ngaySinh = “01/02/2012”;

sendIntent.**putExtra**(“HoTen”, ten);

sendIntent.**putExtra**(“Tuoi”, tuoi);

Bundle duLieuCuaDoiTuong = **new** Bundle();

duLieuCuaDoiTuong.putString(“NgaySinh”, ngaySinh);

duLieuCuaDoiTuong.putString(“GioiTinh”, gioiTinh);

sendIntent.**putExtra**(duLieuCuaDoiTuong);

Để khởi chạy cái intent này thì gọi như sau:

MainActivity.this.**startActivity**(sendIntent);

Trên SecondActivity.java, là activity mà ta gửi dữ liệu tới

Intent getIntent = **getIntent**();

String ten = getIntent.**getStringExtra**(“HoTen”);

int tuoi = getIntent.**getIntExtra**(“Tuoi”);

Bundle duLieuTrongCaiBundle = **getIntent**().**getExtras**();

String ngaySinh = duLieuTrongCaiBundle.**getString**("NgaySinh ");

String gioiTinh = duLieuTrongCaiBundle.**getString**("GioiTinh ");

Còn về Intent không tường minh thì chỉ biết cơ bản như sau vì chưa viết nhiều liên quan tới nó:

Intent intent = **new** Intent(**Intent**.**ACTION**\_VIEW, Uri.parse(“http://thi\*\*\*\*\*.com”));

MainActivity.this.**startActivity**(intent);

Thì trong cái sự kiện gọi hàm chứa code trên nó sẽ chạy đến cái app nào sử dụng cái url được, ở đây là một trình duyệt web mặc định có sẵn, nếu xài 2 cái app duyệt web thì nó sẽ hỏi.

**SharedPreferences:**

Trước khi sử dụng cần hiểu thêm 1 chút về cái **SharedPreferences** này, ngay từ cái tên của nó là nghe cũng hiểu được đôi phần là gì rồi. Vậy sao cần dùng nó?

Nếu như một dữ liệu gì đó như thông tin người dùng, sản phẩm, nhân vật cần lưu xuống database thì một số như tổng 2 số, đơn giá vv thì chỉ cần dùng biến tạm tính toán ngay lúc cần là được rồi. Nhưng có một số biến cần phải lưu lâu dài, kể cả khi tắt app đi rồi mở lên vẫn còn lưu trữ và cũng không nhất thiết phải lưu xuống database như nhớ đăng nhập (remember me :v ), username, vài cái cấu hình game như độ sáng, âm lượng vv thì chỉ cần lưu vào **SharedPreferences** là được rồi và khác **Intent** khi có thể đưa một số kiểu dữ liệu phức tạp thì **SharedPreferences** chỉ lưu những dữ liệu đơn giản thôi.

Cũng đơn giản, ví dụ một cái cấu hình trò chơi đi cho nhiều biến lưu, cách sử dụng như sau:

//SharedPreferences tenbienshare = getSharedPreferences(TenPreferences, Mode);

SharedPreferences shareData = getSharedPreferences(“CAUHINH”, Context.MODE\_PRIVATE);

Hoặc:

SharedPreferences shareData = getSharedPreferences(“CAUHINH”, MODE\_PRIVATE);

Ở đây **Mode** là các mode lưu trữ của SharedPreferences, thường là để MODE\_PRIVATE cho bảo mật, ngoài ra còn nhiều mode khác mà muốn thì phải tìm hiểu từng mode 1 thôi.

Tên của cái cấu hình này sẽ đặt tạm là **CAUHINH** đi.

Để **đưa dữ liệu vào** nó, phải qua một cái **Editor**:

SharedPreferences.**Editor** editor = shareData.**edit**();

editor.**putString**(“TenNguoiChoi”, “HuongHana”);

editor.**putInt**(“Tuoi”, 20);

editor.**putInt**(“DoSang”, 90);

editor.**putBoolean**(“TrongTrang”, false);

editor.**commit**(); /\*hoặc dùng \*/ editor.**apply**();

Có thể thấy là ban đầu phải tạo cái editor, sau đó đưa dữ liệu vào rồi xác nhập lưu những dữ liệu theo các khóa (key : TenNguoiChoi, Tuoi, DoSang …) đó bằng commit hoặc apply. Vậy commit và apply khác nhau điều gì?

**apply**() được thêm vào từ bản 2.3, nó cũng là commit nhưng không có dữ liệu trả về.

**commit**() là thực hiện lưu dữ liệu nhưng có kết quả trả về là true hoặc false;

Một cái là void còn cái kia là return value, có thể xài commit như sau:

boolean luuThanhCong1 = editor.commit(); //trả về true hoặc false

boolean luuThanhCong2 = editor.apply(); //Báo lỗi

Theo tìm hiểu thì chả ai xài cái biến trả về cả nên nhà phát triển cho thêm cái apply vô xài cho lẹ.

Để **sử dụng dữ liệu** từ SharedPreferences làm như sau:

SharedPreferences getCauHinh = **getSharedPreferences**(“CAUHINH” , Context.MODE\_PRIVATE);

String ten = getCauHinh.**getString**(“TenNguoiChoi”, “”);

String tuoi = getCauHinh.**getInt**(“Tuoi”, 0);

String doSang = getCauHinh. **getInt** (“DoSang”, 0);

String trongTrang = getCauHinh.**getBoolean**(“TrongTrang”, false);

Trong các hàm **getXXX** đó thì cái đối số thứ nhất là key còn đối số thứ hai là dữ liệu mặc định nếu như key đó không có dữ liệu.

Để có thể **clear** dữ liệu trong một SharedPreferences nào đó thì làm như sau

SharedPreferences getCauHinh = getSharedPreferences(“CAUHINH” , Context.MODE\_PRIVATE);

SharedPreferences.**Editor** editor = shareData.**edit**();

editor.**clear**();

editor.**commit**();

\* Mẹo:

Vì tính chất của SharedPreferences nên dùng lưu mấy cái cấu hình trong setting của app, cấu hình profile sẽ tiện hơn nhiều so với lưu xuống database, đỡ mất một bảng, và xài chủ yếu dữ liệu thô thôi mà.

Trong một vòng đời của Activity có **onPause** và **onResme** vậy 2 cái sự kiện này nó chạy thế nào?

a) Mở Activity mới lên làm che khuất toàn bộ Activity cũ (không nhìn thấy Activity cũ): sảy ra sự kiện **onPause** rồi **onStop** đối với Activity cũ

b) Mở Activity mới lên làm che khuất một phần Activity cũ (vẫn nhìn thấy Activity cũ): Sảy ra sự kiện **onPause** với Activity cũ.

– Khi quay trở về Activity cũ thì sau khi thực hiện xong các hàm cần thiết, chắc chắn nó phải gọi hàm **onResume** để phục hồi lại trạng thái ứng dụng

– Như vậy ta thường lưu lại trạng thái của ứng dụng trong sự kiện **onPause** và đọc lại trạng thái ứng dụng trong sự kiện **onResume**

V. Tạo thông báo Toast

Toast là cái thông báo hiện trên màn hình của app ấy. Gọi cũng dễ, cứ như là msgbox.show ấy.

Toast.makeText(getApplicationContext(), “Lời nhắn ghi vô đây” ,Toast.LENGTH\_SHORT).show();

Trong đó đối số thứ 3 là thời gian hiển thị, thì có 2 loại là **Toast.LENGTH\_SHORT** và **Toast.LENGTH\_LONG**.