- https://www.youtube.com/@LapTrinhJava/playlist

Bài 3: Giao diện người dùng

LẬP TRÌNH ANDROID CƠ BẢN

Link videos:
https://www.youtube.com/playlist?list=PL0YqB-plTBzuSdqePxDnKQfoSZpoPKhzw

Nội dung bài học

- Process/Thread trong Android
- Vòng đời của Service
- Thiết kế giao diện Mobile
- LinearLayout
- XML Layout

- https://www.youtube.com/@LapTrinhJava/playlist

Ôn lại bài trước

- Task = Danh sách hàng đợi các Activity
- Vòng đời Activity
  - Active: chế độ nền, gọi onResume()
  - Paused: bị che mờ, gọi onPause()
  - Stopped: không hiển thị, gọi onStop()
- Tài nguyên – phân tách logic của chương trình với các phần khác
  - String, ảnh, giao diện UI
- AndroidManifest.xml – kết nối các thành phần với nhau

Process/Thread trong Android

- Mặc định: Một ứng dụng = một process
  - Tất cả thành phần được khởi tạo trong phương thức main
  - Không nên thực hiện các thao tác tốn nhiều thời gian
- Định nghĩa ứng dụng: tất cả thành phần được nhóm trong tag <application>, trong file AndroidManifest

```
<application
    android:allowBackup="true"
    android:icon="@drawable/ic_launcher"
    android:label="@string/app_name"
    android:theme="@style/AppTheme" >
    <activity
        android:name="fptpolytechnic.MOB202.examples.ActivityLifecycle"
        android:label="@string/app_name" >
        <intent-filter>
            <action android:name="android.intent.action.MAIN" />

            <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
    </activity>
</application>
```

Vòng đời của Service

- Service có các phương thức onCreate, onStart, và onDestroy (không có pause/resume)
- startService (giống startActivity) sẽ khởi tạo service nếu service chưa chạy
- Sau đó, gọi onStart
  - Nếu service đang chạy, chỉ gọi hàm onStart
- Nên sinh ra Thread mới để điều khiển công việc

Service.stopSelf

- onStart(Intent i, int startId)
- Nên dừng Service khi tất cả lệnh được xử lý
  - Vì đa tiến trình nên không có trật tự thực hiện nào được đảm bảo
- stopSelf(startId): sẽ dừng Service nếu startId tương ứng với lệnh cuối cùng
- Cấu trúc dữ liệu gì nên được sử dụng để lưu trữ startId?

Service

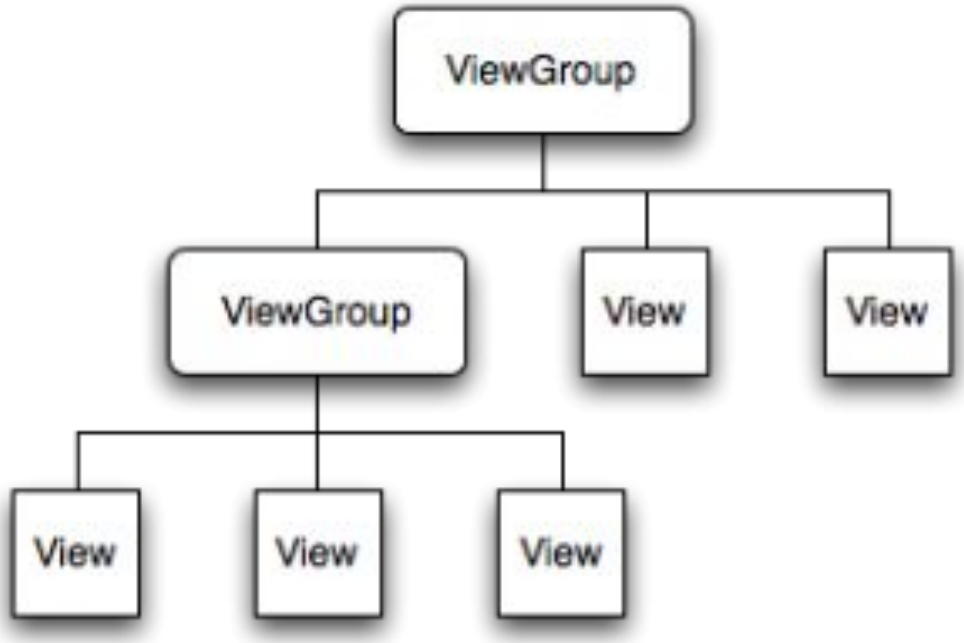
- Tham khảo thêm http://developer.android.com/reference/android/app/Service.html
- Tìm hiểu sâu hơn trong bài Background Task

Thiết kế giao diện người dùng

- Càng đơn giản càng tốt
- Dành nhiều thời gian tìm hiểu nhu cầu của khách hàng về giao diện
- Sử dụng điều khiển giao diện chuẩn

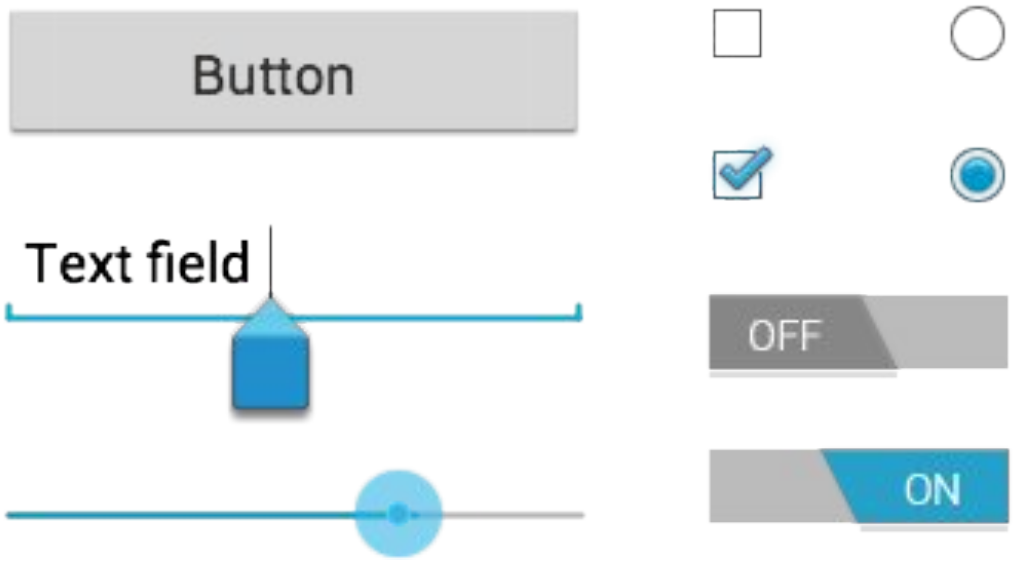
Cây phân cấp View (View Hierarchy)

- View: đơn vị cơ bản của giao diện người dùng
  - Widgets: android.widget.*
  - Là lá của cây phân cấp View
- ViewGroup: định nghĩa layout
  - Nằm trong android.widget.*
  - Định nghĩa nơi chứa các Views (hoặc View Group) con



Ví dụ Widget

- Button
- EditText
- CheckBox và RadioButton
- Spinner
- Others: TextView, ImageView
- Có thể kế thừa một widget hoặc tạo mới

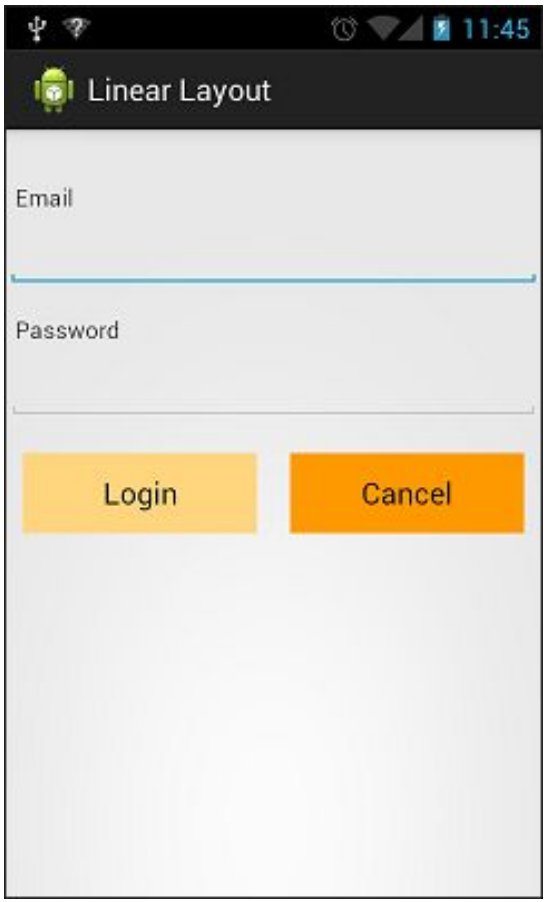


Ví dụ về Layout

- Ghi nhớ: ViewGroup là một lớp con của View
- Đơn giản nhất: FrameLayout
  - Khoảng trắng được lấp đầy với một đối tượng đơn
  - Gắn đối tượng vào góc trái trên
  - Nếu nó chứa nhiều hơn một đối tượng, đơn giản vẽ chúng chồng lên nhau
- Đây là lớp gốc của Activity

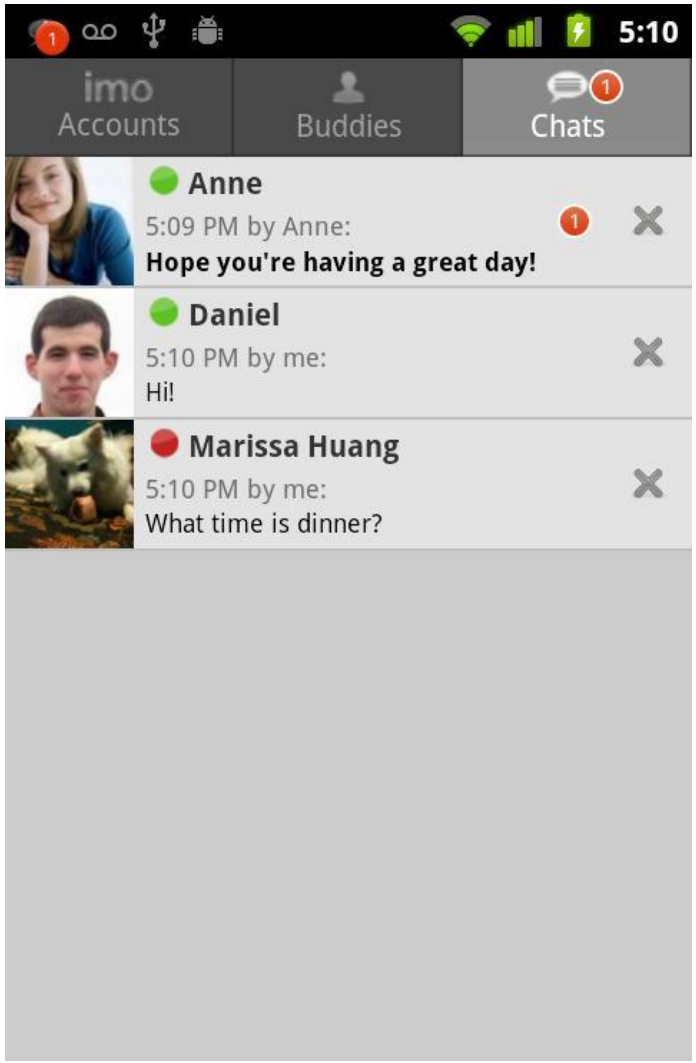
LinearLayout

- Là layout sắp xếp các View con trong nó theo duy nhất một chiều, ngang hoặc dọc theo giá trị của thuộc tính android:orientation
  - Orientation = vertical hoặc horizontal
- Có thể lồng nhiều layout phức tạp
  - Thông thường sử dụng cho form nhỏ



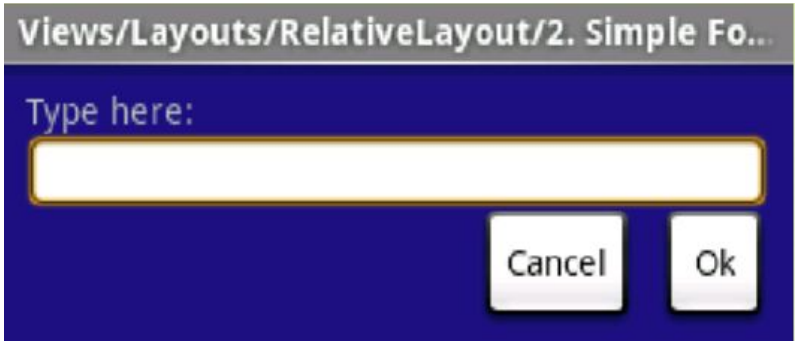
TabLayout

- Gồm 2 phần chia ra riêng biệt, phần nhỏ hiển thị tổng quát các chức năng và phần lớn hiển thị nội dung của mỗi chức năng
- Thuận tiện và dễ quản lý hơn menu
- 2 phương pháp để lấp đầy Tab
  - Đổi View: lý tưởng cho các chức năng tương tự nhau được nhóm vào một group chức năng
  - Đổi Activity: lý tưởng cho quản lý công việc tách rời nhau, thay vì sử dụng một activity và layout lớn

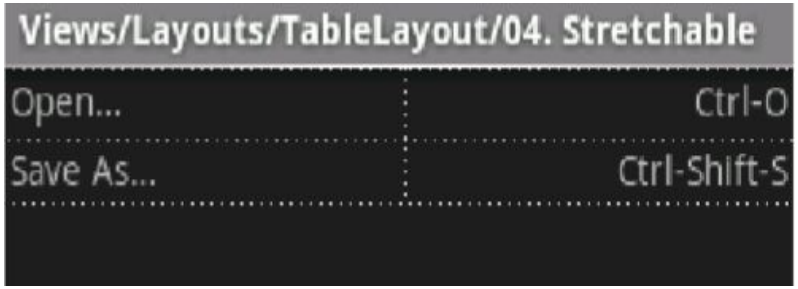


Layout khác

- RelativeLayout
  - Là một view group hiển thị các thành phần con dựa vào mối quan hệ vị trí giữa chúng với nhau hoặc giữa chúng với thành phần cha chứa nó.
  - EditText: nằm dưới TextView
  - Nút OK: dưới EditText, căn phải với phần tử cha (screen)
  - Nút Cancel: căn trái nút OK, có khoảng cách nhỏ với bên phải



- TableLayout



LinearLayout lồng nhau

- LinearLayout lồng nhau là một cách để dễ dàng tạo các giao diện chung
- Chú ý: nếu lồng nhau mà số cấp lớn hơn hoặc bằng 5 sẽ làm cho việc tải giao diện chậm hơn
- Công cụ để phát hiện vấn đề:
  http://android-developers.blogspot.com/2009/11/optimize-your-layouts.html

LinearLayout -> RelativeLayout

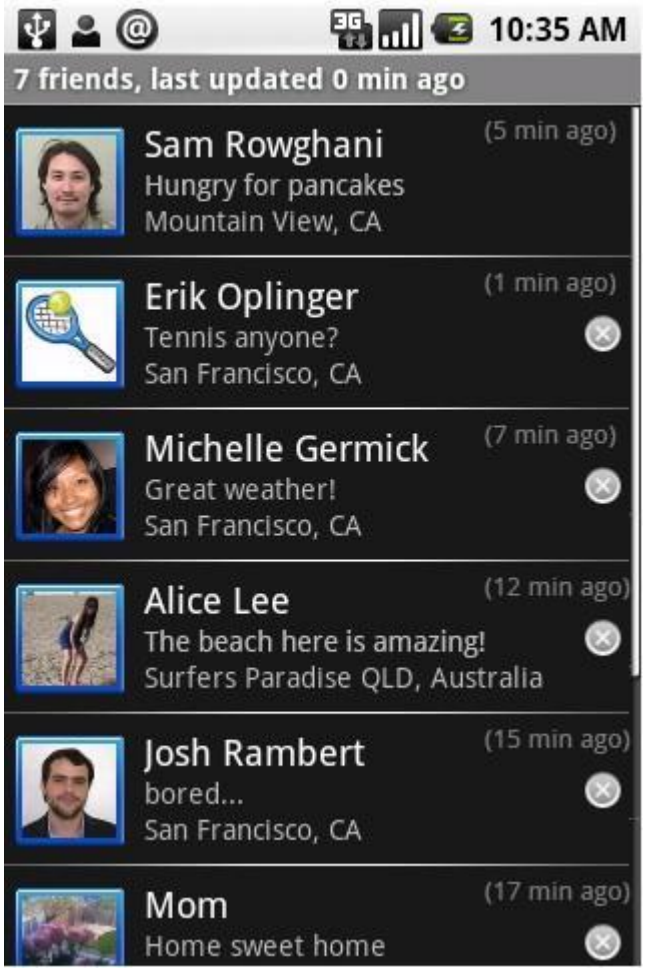
- Refactor sang RelativeLayout

  Ví dụ:

  http://android-developers.blogspot.com/2009/11/optimize-your-layouts.html

ListView

- ViewGroup chứa danh sách các View
- Có thể định nghĩa một View để hiển thị khi List rỗng sử dụng setEmptyView
- Mỗi dòng mặc định là TextView, có thể tùy biến
- Thông thường được load dữ liệu động



ListView Adapter

- Adapter – ràng buộc nội dung động vào View trong ListView

  Ví dụ ArrayAdapter đối với mảng
- Đơn giản – ràng buộc giá trị text vào text field trong ListView
- Phức tạp hơn – tùy biến ListView row, đối tượng tùy biến được ràng buộc vào View

Trọng lượng của Layout (layout weight)

- Trọng lượng cho phép tạo LinearLayout với cỡ cân đối
- Default = 0 – không gian tối thiểu để hiển thị tất cả nội dung

Ví dụ

- Chúng ta sẽ định nghĩa layout cho giao diện sau như thế nào?

Định nghĩa Layout

- Phương pháp phổ biến – định nghĩa thông qua file XML
- Nằm trong thư mục res/layout/<file>.xml – có thể truy cập theo R.layout.<file> từ code
- Tất cả các file đều chứa
  - XML version

    <?xml version="1.0" encoding="utf-8"?>
  - xmlns:android tag trong phần tử gốc

  <LinearLayout

  xmlns:android=“http://schemas.android.com/apk/res/android">
- Từ onCreate, gọi setContentView(R.layout.<file>) để thiết lập phần tử layout gốc cho một activity

XML Layout

- Tất cả thuộc tính có tiền tố “android:”
- Các thuộc tính được áp dụng cho mọi View
  - Id: không bắt buộc. Giá trị duy nhất cho đối tượng do đó có thể sử dụng để truy cập đối tượng từ code
    - android:id=“@+id/<name>”
    - Trong Java: (Button) b = (Button) findViewById(R.id.<name>);
  - layout_width/layout_height – chiều của đối tượng
    - Xác định cỡ (pixel hoặc dips – density independent pixcels)
    - fill_parent: chiếm toàn bộ cỡ của ViewGroup cha
    - wrap_parent:chiếm không gian cần thiết để hiển thị View

Styles

- Bản chất là CSS của Android UI
- Nằm trong res/values/<anything>.xml
  - Có thể đặt tên là styles.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <style name="BigText">
        <item name="android:textSize">30dip</item>
</style>
</resources>
```

- Áp dụng cho một view trong XML với style="@style/BigText"

Theme

- Style áp dụng cho toàn bộ Activity hoặc ứng dụng
- Thay đổi trong file AndroidManifest.xml
- Ví dụ:
  - Thay đổi Activity giống dialog box

    <activity android:theme="@android:style/Theme.Dialog">
  - Bỏ title bar

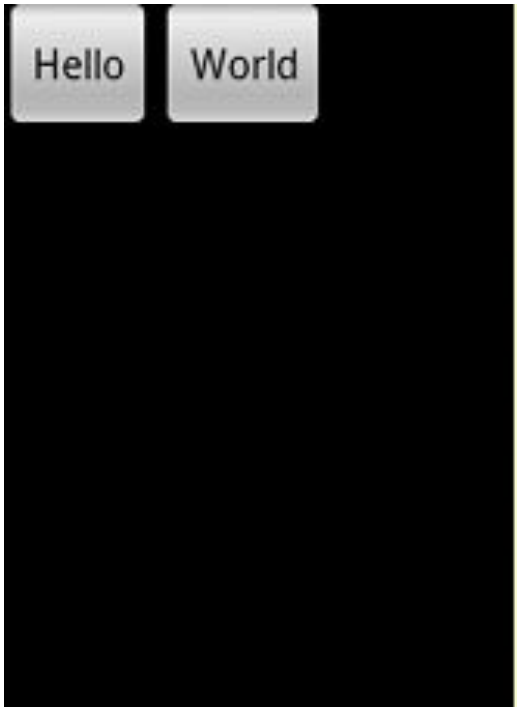
    @android:style/Theme.NoTitleBar
- Thông tin thêm về Styles/Themes

http://developer.android.com/guide/topics/ui/themes.html

Tạo layout bằng code

- Style áp dụng cho toàn bộ Activity hoặc ứng dụng

```
public void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);
  LinearLayout ll = new LinearLayout(this);
  Button button1 = new Button(this);
  button1.setText("Hello");
  Button button2 = new Button(this);
  button2.setText("World");
  ll.addView(button1);
  ll.addView(button2);
  setContentView(ll);
}
```



- Trong thực tế, tạo XML layout dễ hơn nhiều
  - Nhưng cần Java để nhận sự kiện từ Views

UI trong Java

- Định nghĩa layout trong XML, vậy làm thế nào để xử lý sự kiện trên layout?
- Chúng ta cần biết làm thế nào để:
  - Nhận dữ liệu và truyền dữ liệu tới widget
    - Làm thế nào để truy cập text trong EditText?
  - Nhận sự kiện từ widget
    - Chúng ta sẽ làm gì khi một button được click?

Nhận tham chiếu đến đối tượng

- Có view với android:id="@+id/widget";
- <Class> widget = (<Class>) findViewById(R.id.widget);
  - <Class> là lớp của View, ví dụ Button hoặc EditText
  - Đây là đối tượng chúng ta cần
    - Get/set fields
    - Thiết lập event handlers

Nhận tham chiếu đến đối tượng

- Nếu chúng ta cần truy cập một view trong nhiều hơn một phương thức
  - Có một biến trong lớp
    - Private ListView mList;
  - Ràng buộc biến trong phương thức onCreate()
- Câu lệnh sau sai ở đâu:

```
public void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState); Button b = (Button)
findViewById(R.id.button);
    setContentView(R.layout.main);
}
```

Nhận tham chiếu đến đối tượng

- Nếu chúng ta cần truy cập một view trong nhiều hơn một phương thức
  - Có một biến trong lớp
    - Private ListView mList;
  - Ràng buộc biến trong phương thức onCreate()
- Câu lệnh sau sai ở đâu:

```
public void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState); Button b = (Button)
findViewById(R.id.button);
    setContentView(R.layout.main);
}
```

Bạn phải gọi hàm setContentView trước khi gọi hàm findViewById

Getting/setting field

- Khi chúng ta có tham chiếu tới widget, truy cập tới các đối tượng khá đơn giản

```
EditTexttextField= …
String text = textField.getText().toString();
textField.setText(“Hello”);
textField.setTextColor(Color.RED);
```

- https://www.youtube.com/@LapTrinhJava/playlist

Tổng kết nội dung bài học

- Process/Thread trong Android
- Vòng đời của Service
- Thiết kế giao diện Mobile
- LinearLayout
- XML Layout

- https://www.youtube.com/@LapTrinhJava/playlist