ANDROID NÂNG CAO

BÀI 6: Animations & Widgets

Nội dung

- **Animations**
  - Nguyên tắc
  - XML vs Code
  - Interpolator
  - Drawable Animation
- **Widgets**
  - Khái niệm
  - Các bước thực hiện
  - Các chú ý
  - Ví dụ về widget có tương tác

Phần 1

Animations

Animations – nguyên tắc

- **Animation: tập hợp các API cho phép biến đổi các view trong một thời gian ngắn (từ android 3.0)**
- **Tạo animation theo nhiều cách**
  - Định nghĩa trong XML (folder "res/anim/") và nạp bởi câu lệnh `AnimationUtils.loadAnimation`
  - Tự tạo bằng cách new các đối tượng phù hợp
- **Các animation có thể ghép với nhau thành để thực hiện nhiều hiệu ứng (gọi là `AnimationSet`)**
- **(advanced) Có thể tự tạo custom animation bằng cách kế thừa class `Animation` và viết lại phương thức `applyTransformation`**

Animations – nguyên tắc

- **Các animation được cung cấp bởi Android**
  - AlphaAnimation: Thay đổi độ trong suốt của đối tượng
    - Fade In / Fade Out
    - Blink
  - RotateAnimation: Xoay đối tượng
  - ScaleAnimation: Thay đổi kích thước
    - Zoom In / Zoom Out
    - Slide Up / Slide Down
    - Bounce
  - TranslateAnimation: Dịch chuyển đối tượng
- **Bằng cách điều chỉnh các tham số ta có thể tạo các animation khác nhau**

Animations – XML vs Code

- **Ví dụ về XML: res/animator/fadein.xml**

```
<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android" >
<alpha android:duration="1000"
       android:interpolator="@android:anim/accelerate_interpolator"
       android:fromAlpha="0.0"
       android:toAlpha="1.0" />
</set>
// nạp animation từ XML để sử dụng
Animation ani = AnimationUtils.loadAnimation(this, R.animator.fadein);
```

- **Ví dụ tạo animation bằng code:**

```
Animation ani = new AlphaAnimation(1, 0);
ani.setInterpolator(new AccelerateInterpolator());
ani.setDuration(1000);
```

- **Sử dụng animation:**

```
myView.startAnimation(ani);
```

Animations – XML vs Code

Một số thuộc tính quan trọng

- android:duration: thời gian chạy hiệu ứng (ms)
- android:startOffset: thời điểm bắt đầu chạy (ms)
- android:fillAfter: có giữ lại trạng thái sau hiệu ứng không (true/false)
- android:repeatCount: số lần lặp lại hiệu ứng
- android:repeatMode: chế độ lặp (RESTART | REVERSE)
- android:interpolator: kiểu diễn biến của hiệu ứng
  - *"@android:anim/accelerate_interpolator"*
  - *"@android:anim/ accelerate_decelerate_interpolator"*
  - …

Animations – XML vs Code

Có thể xử lý sự kiện khi hiệu ứng bắt đầu, kết thúc hoặc lặp lại bằng AnimationListener (3 phương thức cho giai đoạn)

```
rotAni.setAnimationListener(new AnimationListener() {
    public void onAnimationEnd(Animation arg0) {
        layerImage.setVisibility(View.INVISIBLE);
        layerButtons.setVisibility(View.VISIBLE);
    }
    public void onAnimationRepeat(Animation arg0) { }
    public void onAnimationStart(Animation arg0) { }
});
```

Animations – Interpolator

- **Interpolator: trình điều khiển quá trình thực hiện hiệu ứng**
- **Một số Interpolator thông dụng:**
  - AccelerateDecelerateInterpolator: chậm-nhanh-chậm
  - AccelerateInterpolator: nhanh dần
  - DecelerateInterpolator: chậm dần
  - CycleInterpolator: lặp lại animation theo chu kì
  - LinearInterpolator: tuyến tính, không đổi
- **Có thể tự viết interpolator của mình nếu thích (phải tìm hiểu thêm về Interpolator.Result)**

Animations – set of animations

```
<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android"
  android:interpolator="@android:anim/accelerate_interpolator"
  android:shareInterpolator="true"
  android:duration="1000"
  android:fillAfter="false" >

  <alpha android:fromAlpha="1.0" android:toAlpha="0.0" />

  <rotate android:fromDegrees="0“ android:toDegrees="360"
    android:pivotX="50%" android:pivotY="50%" />

  <scale android:fillAfter="true"
    android:fromXScale="1.0" android:fromYScale="1.0"
    android:toXScale="0.0" android:toYScale="0.0"
    android:pivotX="50%" android:pivotY="50%" />

</set>
```

Animations – example

```
public class MainActivity extends Activity {
    Animation x;

    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        x = AnimationUtils.loadAnimation(this, R.animator.abc);
    }

    public void btnAni(View v) {
        View t = findViewById(R.id.editText1);
        t.startAnimation(x);
    }
}
```

Drawable Animation

- **3 kiểu animations trong Android:**
  - Property animation: gốc của mọi animation
  - View animation: animation trên một view và sử dụng một image
  - Drawable animation: animation trên một view và sử dụng dãy image
- **Drawable Animation**

```
<animation-list xmlns:android="http://schemas.android.com/apk/res/android"
   android:oneshot="true">
   <item android:drawable="@drawable/rocket_thrust1" android:duration="200" />
   <item android:drawable="@drawable/rocket_thrust2" android:duration="200" />
   <item android:drawable="@drawable/rocket_thrust3" android:duration="200" />
</animation-list>
```

Drawable Animation

```
AnimationDrawable rocketAnimation;

public void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);
  setContentView(R.layout.main);

  ImageView rocketImage = (ImageView) findViewById(R.id.rocket_image);
  rocketImage.setBackgroundResource(R.drawable.rocket_thrust);
  rocketAnimation = (AnimationDrawable) rocketImage.getBackground();
}

public boolean onTouchEvent(MotionEvent event) {
  if (event.getAction() == MotionEvent.ACTION_DOWN) {
    rocketAnimation.start();
    return true;
  }
  return super.onTouchEvent(event);
}
```

Phần 2

Widgets

Widgets – khái niệm

- Widget là phần màn hình nhỏ đặt lên home-screen hoặc lock-screen của thiết bị Android
- Widget chạy như một phần của ứng dụng chủ vì thế cần chú ý việc widget sẽ bị hạn chế bởi các quyền cấp cho ứng dụng chủ
- Widget sử dụng RemoteView để tạo giao diện
- Widget sử dụng BroadcastReceiver để liên lạc
- Widget thường cung cấp thông tin hoặc tương tác tối thiểu, tránh lạm dụng widget
- Ứng dụng có chứa widget không thể chuyển sang SD card

Widgets – các bước thực hiện

1. Tạo file layout phù hợp cho màn hình của widget
2. Tạo class kế thừa từ AppWidgetProvider
3. Viết các phương thức cần thiết của widget
4. Tạo file XML mô tả về widget (type = AppWidget Provider)
5. Cập nhật thông tin trong AndroidManifest.xml

Widgets: layout

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/LinearLayout01"
    android:layout_height="200dp"
    android:layout_width="160dp"
    android:orientation="horizontal">
    <TextView
        android:id="@+id/textview1"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:gravity="center_vertical|center_horizontal"
        android:text="00:00:00 AM"
        android:textColor="@android:color/black"
        android:textSize="8pt" />
</LinearLayout>
```

Widgets: code

```
public class MyWatchWidget extends AppWidgetProvider {
    public void onUpdate(Context context,
        AppWidgetManager appWidgetManager, int[] appWidgetIds) {
        DateFormat format = SimpleDateFormat.getTimeInstance(
                        SimpleDateFormat.MEDIUM, Locale.getDefault());

        RemoteViews remoteViews = new RemoteViews(
                        context.getPackageName(), R.layout.mywatch);
        ComponentName watchWidget = new ComponentName(context,
                                                MyWatchWidget.class);
        remoteViews.setTextViewText(R.id.textview1,
                                "Time = " + format.format(new Date()));
        appWidgetManager.updateAppWidget(watchWidget, remoteViews);
    }
}
```

Widgets: provider

```
<?xml version="1.0" encoding="utf-8" ?>
<appwidget-provider
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:minWidth="146dp" android:minHeight="144dp"
    android:initialLayout="@layout/mywatch"
    android:updatePeriodMillis="1000" />
```

Chú ý:

- File được đặt trong "res/xml"
- Trong file đã quy định sẵn sẽ sử dụng layout nào thông qua thuộc tính "android:initialLayout"
- Các thuộc tính "android:minWidth" và "android:minHeight" quy định chiều rộng và chiều cao tối thiểu của widget

Widgets: AndroidManifest.xml

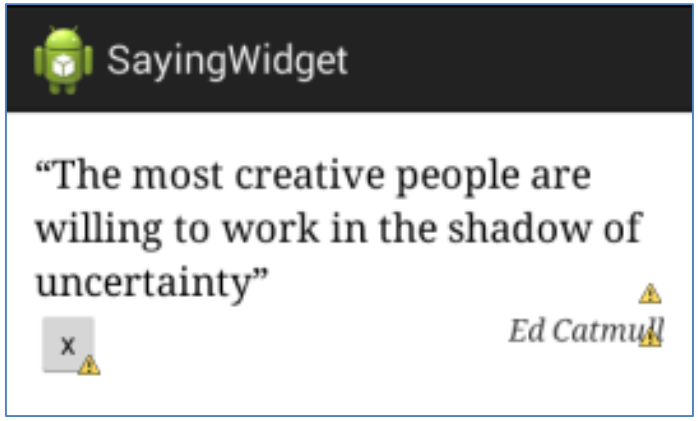
```
<receiver android:name=".MyWatchWidget" android:label="MyWatchWidget">
  <intent-filter>
   <action android:name="android.appwidget.action.APPWIDGET_UPDATE" />
  </intent-filter>
  <meta-data
     android:name="android.appwidget.provider"
     android:resource="@xml/mywatch" />
</receiver>
```

Widgets – các chú ý

- Thuộc tính android:updatePeriodMillis trong provider nhỏ nhất là 30 phút
- Trong trường hợp cần update thường xuyên hơn, sử dụng AlarmManager hoặc service để xử lý background
- Sử dụng: android:widgetCategory="keyguard|home_screen" để widget có thể đặt ở cả lock screen và home screen
- Chú ý kích thước layout của widget khi nhắm tới các thiết bị đa dạng về độ phân giải (chỉ cung cấp minResizeHeight và minResizeWidth là đủ)

Ví dụ về widget có tương tác

- **Hiển thị câu châm ngôn**
- **Refresh bằng nút bấm**
- **Mỗi khi refresh:**
  - Hiển thị câu châm ngôn khác
- **Cách làm:**
  - Sử dụng BroadcastReceiver để yêu cầu cập nhật
  - Có thể tách riêng thành 2 class hoặc dùng chung 1 class cho cả Widget và Receiver
  - Biến dùng chung cần được chia sẻ (vì có thể thuộc các instance khác nhau)



AndroidManifest.xml

```
<receiver
   android:name="SayingWidget“
   android:label="Saying Widget">
   <intent-filter>
      <action
         android:name="android.appwidget.action.APPWIDGET_UPDATE" />
      <action
         android:name="txnam.sayingwidget.intent.action.REFRESH" />
   </intent-filter>
   <meta-data
      android:name="android.appwidget.provider“
      android:resource="@xml/saying1" />
</receiver>
```

Viết mã cho widget (1)

```
static final String ACTION_REFRESH = "txnam.sayingwidget.intent.action.REFRESH";
static String[] sayings = {...};
static String[] authors = {...};
static int position = 0;

// xử lý khi nhận được PendingIntent từ Widget
@Override
public void onReceive(Context context, Intent intent) {
    if (intent.getAction().equals(ACTION_REFRESH)) {
        updateContent(context, AppWidgetManager.getInstance(context));
        return;
    }
    super.onReceive(context, intent);
}
```

Viết mã cho widget (2)

```
// xử lý khi cập nhật Widget
@Override
public void onUpdate(Context context, AppWidgetManager app, int[] Ids) {
    updateContent(context, app);
}

// tạo PendingIntent với tên action
PendingIntent buildPendingIntent(Context context, String action) {
    Intent intent = new Intent(action);
    return PendingIntent.getBroadcast(context, 0, intent,
        PendingIntent.FLAG_UPDATE_CURRENT);
}
```

Viết mã cho widget (3)

```
// cập nhật hình ảnh trên Widget
void updateContent(Context ct, AppWidgetManager app) {
    RemoteViews remoteViews =
        new RemoteViews(ct.getPackageName(), R.layout.saying);
    position++;
    if (position >= sayings.length) position = 0;
    remoteViews.setTextViewText(R.id.textView1, sayings[position]);
    remoteViews.setTextViewText(R.id.textView2, authors[position]);
    remoteViews.setOnClickPendingIntent(R.id.button1,
        buildPendingIntent(ct, ACTION_REFRESH));
    ComponentName widget = new ComponentName(ct, SayingWidget.class);
    app.updateAppWidget(widget, remoteViews);
}
```