Lezione 22 – Esempi personalizzazione View e introduzione Intenti

Esempio

Mettendo insieme tutti i file xml visti finora (manifesto, string, layout) possiamo scrivere la seguente attività che simula l'analoga applet vista a lezione.

```
package it.unive.dsi.android;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.net.HttpURLConnection;
import java.net.MalformedURLException;
import java.net.URL;
import java.net.URLConnection;
import java.util.ArrayList;
import android.app.Activity;
import android.os.Bundle;
import android.view.KeyEvent;
import android.view.View;
import android.widget.*;
public class Prova extends Activity {
  /** Called when the activity is first created. */
  @Override
  public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
    ListView list = (ListView) findViewById(R.id.listView1);
    final EditText text = (EditText) findViewById(R.id.editText1);
    final ArrayList<String> strings = new ArrayList<String>();
    final ArrayAdapter<String> aa =
       new ArrayAdapter<String>(this,
                    android.R.layout.simple_list_item_1, strings);
    list.setAdapter(aa);
    text.setOnKeyListener(new View.OnKeyListener() {
      public boolean onKey(View v, int keyCode, KeyEvent event) {
        if (event.getAction() == KeyEvent.ACTION_DOWN)
          if (keyCode == KeyEvent.KEYCODE_DPAD_CENTER
              || keyCode == KeyEvent.KEYCODE_ENTER) {
            try {
              URL url = new URL(text.getText().toString());
              URLConnection connection = url.openConnection();
              HttpURLConnection httpConnection = (HttpURLConnection)
connection;
              int responseCode = httpConnection.getResponseCode();
              if (responseCode == HttpURLConnection.HTTP_OK) {
                InputStream in = httpConnection.getInputStream();
                BufferedReader d = new BufferedReader(new
InputStreamReader(in));
                String res = d.readLine();
                while (res != null) {
```

```
                strings.add(res);
                res = d.readLine();
              }
            }
            else stringa.add("NO HTTP_OK");
          } catch (MalformedURLException e) {
            strings.add(e.getMessage());
          } catch (IOException e) {
            strings.add(e.getMessage());
          }
          if (strings.size() > 5) strings.remove(0);
          strings.notifyDataSetChanged();
          text.setText("");
          return true;
        }
        return false;
      }
    });
  }
```

Esempio personalizzazione Componenti

E' possibile personalizzare i componenti :

1. **estendendo** la classe del componente che li definisce (es. `TestView`)
2. **raggrupare** più controlli in modo che si comportino come un'unico nuovo controllo estendendo una classe dei `Layout`
3. anche partendo da zero **estendendo** la classe `View`. I metodi che devono essere riscritti sono:

- costruttore con parametro `Context` : chiamato dal codice Java.
- Costruttore con parametri `Context` e `Attributes`: richiesto dal gonfiaggio dell'interfaccia dal file di risorse. Anche il costruttore con parametri `Context`, `Attributes` e `int` con lo stile di default.
- `onMeasure` per calcolare le dimensoni del componente e impostarle tramite chiamata al metodo `setMeasuredDimension`.
- `OnDraw` per disegnare l'aspetto del componente;

Esempio personalizzazione con estensione TextView

```
public class MyTextView extends TextView {

Paint paint = new Paint(Paint.ANTI_ALIAS_FLAG);
//creazione da codice
public MyTextView (Context context) {
super(context);
}
//gonfiaggio da resource file
public MyTextView (Context context, AttributeSet ats, int defStyle)
{
super(context, ats, defStyle);
}
```

```
//gonfiaggio da resource file
public MyTextView (Context context, AttributeSet attrs) {
super(context, attrs);
}

@Override
public void onDraw(Canvas canvas) {
// disegni che vanno sotto il testo
canvas.drawText("sotto", 0, 0, paint);
// onDraw della classe TextView disegna il testo
super.onDraw(canvas);
//disegni che vanno sopra il testo
canvas.drawText("sopra", 0, 0, paint);
}

@Override
public boolean onKeyDown(int keyCode, KeyEvent keyEvent) {
//personalizzare il comportamento del controllo
if (keyEvent.getEventTime()%2==0)
return true;

// e poi chiamare il metodo della superclasse
return super.onKeyDown(keyCode, keyEvent);
}
}
```

Esempio personalizzazione con estensione da View

```
public class MyView extends View {

Paint paint = new Paint(Paint.ANTI_ALIAS_FLAG);

// necessario per la crezione da codice
public MyView(Context context) {
super(context);
}

// necessario per il gonfiaggio da resource file
public MyView (Context c, AttributeSet ats, int defStyle) {
super(c, ats, defStyle );
}

// necessario per il gonfiaggio da resource file
public MyView (Context context, AttributeSet attrs) {
super(context, attrs);
}

@Override
protected void onMeasure(int wMeasureSpec, int hMeasureSpec) {
int measuredHeight = measureHeight(hMeasureSpec);
int measuredWidth = measureWidth(wMeasureSpec);


// in questo metodo bisogna chiamare setMeasuredDimension
// altrimenti viene sollevata un'eccezione durante il
// layout del componete
```

```
setMeasuredDimension(measuredHeight, measuredWidth);
}

private int measureHeight(int measureSpec) {
int specMode = MeasureSpec.getMode(measureSpec);
int specSize = MeasureSpec.getSize(measureSpec);

// dimesione di default

int result = 250;
if (specMode == MeasureSpec.AT_MOST) {
// calcolare la dimensione ideale per
// il componente
int mySize=....
// ritornalre la dimensione minima
// tra quella ideale e quella massima
result = Math.min(specSize,mySize);
}
else if (specMode == MeasureSpec.EXACTLY)
{
// nel caso il controllo sia contenuto
// nella dimensione, ritornare tale valore
result = specSize;
}//else MeasureSpec.UNSPECIFIED si usa il default
return result;
}

private int measureWidth(int measureSpec) {
int specMode = MeasureSpec.getMode(measureSpec);
int specSize = MeasureSpec.getSize(measureSpec);
//... calcolo analogo al precedente ... ]
}

@Override
protected void onDraw(Canvas canvas) {
// recuperare le dimensioni ritornate da onMeasure.
int height = getMeasuredHeight();
int width = getMeasuredWidth();
// troviamo il centro
int px = width/2;
int py = height/2;
mTextPaint.setColor(Color.WHITE);
String displayText = "Hello World!";
// troviamo la dimensione del testo da scrivere
float textWidth = mTextPaint.measureText(displayText);
// disegnamo il testo nel centro del controllo
canvas.drawText(displayText, px-textWidth/2, py, mTextPaint);
}
@Override
public boolean onKeyDown(int keyCode, KeyEvent keyEvent) {
// Return true if the event was handled.
return true;
}
@Override
public boolean onKeyUp(int keyCode, KeyEvent keyEvent) {
// Return true if the event was handled.
return true;
```

```
}

@Override
public boolean onTrackballEvent(MotionEvent event ) {
// Get the type of action this event represents
int actionPerformed = event.getAction();
// Return true if the event was handled.
return true;
}
}
```

Eventi per l'interazione dei componenti con l'utente

Per fare in modo che l'estensione della View sia interattiva essa deve rispondere a tutta una serie di eventi generati dall'utente quali:

- `onKeyDown`
- `onKeyUp`
- `onTrackballEvent`
- `onTouchEvent`

Questi metodi ritornano true se l'evento è stato gestito dal metodo stesso

Menu

Non vedremo i menu in quanto sono relativamente semplici e non richiedono particolari attenzioni. Una volta che vediamo funzionare il menu stesso abbiamo la ragionevole certezza di non aver fatto errori.

Intenti

Gli intenti sono gli oggetti che Android usa per gestire il suo sistema di message passing. Uno degli usi più comune è quello di utilizzarli per far partire nuove Attività/servizi della stessa applicazione o anche di altre applicazioni. Usare gli intenti per propagare informazioni è uno dei principi di progettazione fondamentali di Android che favorisce il disaccoppiamento dei componenti e il loro riuso.

Nuove Attività

L'utilizzo più comune degli intenti è quello necessario per eseguire una nuova Activity, Service o Broadcast Receiver (della stessa applicazione o di una nuova).

Ci sono due modalità di esecuzione a seconda che vogliamo o meno esaminare il valore di ritorno dell'attività invocata. Ci sono anche due modalità di attivazione esplicita o meno a seconda se vogliamo accoppiare o disaccoppiare le due attività.

1) Esplicita senza esaminare il valore di ritorno:

```
Intent intent new Intent(this,OtherActivity.class);
startActivity(intent);
```

2) Implicita senza esaminare il valore di ritorno:

```
Intent intent new
Intent(Intent.ACTION_DIAL,URI.parse(“tel:0412348457”));
startActivity(intent);
```

Esempio: per far partire il browser web preinstallato nel terminale andorid sostituire il codice in rosso con:

```
Intent intent = new Intent(Intente.ACTION_VIEW,
Uri.parse(text.getText().toString());
startActivity(intent);
```

3) Esplicita con esame del valore di ritorno:

```
    int originalRequestCode;
... onKey(..)
  {..
   originalRequestCode = 2;
    Intent intent new Intent(this,OtherActivity.class);
    startActivityForResult(intent, originalRequestCode);
   …}
   @Override void onActionResult(int requestCode, int resultCode,
Intent data){
  if (requestCode==originalRequestCode){
    if (resultCode==Activity.RESULT_OK)
     {
       Uri uri = data.getData();
       boolean b = data.getBooleanExtra(“b”,false);
       int i = data.getIntExtra(“i”);
       Bundle extra = data.getExtras();
     }
  }
  if (requestCode == SHOW_2) {
   …}
}
```

La nuova attività prima di terminare (chiamando il metodo `finish()`) deve chiamare il metodo `setResult` specificando resultCode e un intento che può contenere i dati di risposta.
Es.

```
Uri data = ...
Intent result = new Intent(null,data);
result.putExtra(“b”,true);
result.putExtra(“i”,4);
result.put...
setResult(Activity.RESULT_OK,result); //Activity.RESULT_CANCELED
finish();
```

4) Implicita con esame del valore di ritorno.

```
private static final int PICK_CONTACT_SUBACTIVITY = 2;
Uri uri = Uri.parse("content://contacts/people");
Intent intent = new Intent(Intent.ACTION_PICK, uri);
startActivityForResult(intent, PICK_CONTACT_SUBACTIVITY);
```

L'Intent contiene informazioni che verranno elaborate del componente che riceve l'intento più informazioni che servono ad Android per selezionare il componente a cui inviare l'intento.
Le informazioni che servono ad Android sono:

1. per gli intenti espliciti il **nome del componente**, ovvero la classe (di solito per i componenti interni all'applicazione);
2. per gli intenti impliciti: action (costruttore, setAction), data (sia URI che il mimetype setUri, setType setData&Type) e le categorie (addCategory).

I dati che servono solo per il componente destinatario sono Extras (putIntExtras, putEstras e Flag) e servono per comunicare parametri tra il componente chiamante e il componente chiamanato.