4-Graphical User Interface - part II

ActionBar

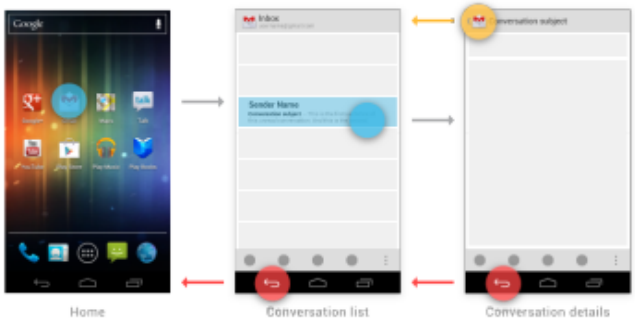
La barra delle azioni (action bar) è una caratteristica della finestra che consente di:

- Identificare la posizione dell'utente
- Fornire azioni per l'utente e modalità di navigazione
- Offrire un'interfaccia familiare tra le applicazioni che il sistema adatta correttamente per diverse configurazioni dello schermo
- Supportare la navigazione coerente e il cambio di visualizzazione all'interno delle applicazioni (con schede o elenchi a discesa)

ActionBar & Navigation

> Nota
> “Consistent navigation is an essential component of the overall user experience” (Una navigazione coerente è un componente essenziale dell'esperienza utente complessiva)

Con Android 2.3 e le versioni precedenti del sistema operativo, la navigazione si basava sul pulsante Indietro di sistema, mentre con l'introduzione delle action bar in Android 3.0, è comparso un secondo meccanismo di navigazione utilizzando il pulsante "Su" (l'icona dell'app e una freccia rivolta verso sinistra).



Action Bar Code

```
<application
                android:allowBackup="true"
                android:icon="@drawable/ic_launcher"
                android:label="@string/app_name"
                android:theme="@style/Theme.AppCompat">

        <activity
                android:name=".MainActivity"
                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>
```

```
public class MainActivity extends ActionBarActivity{
// ...
        public void onCreate(Bundle savedInstanceState) {
                super.onCreate(savedInstanceState);
                setContentView(R.layout.activity_main);

                // Set up the action bar.
                // getSupportActionBar() if we have the support library v7 otherwise just
getActionBar()
                ActionBar actionBar = getSupportActionBar();

                // Specify that the Home/Up button should not be enabled,
                //since there is no hierarchical parent.
                actionBar.setHomeButtonEnabled(false);

                //UnComment the following lines to hide the top part of the action bar
                //actionBar.setDisplayShowTitleEnabled(false);
                //actionBar.setDisplayHomeAsUpEnabled(false);
                //actionBar.setDisplayShowHomeEnabled(false);
        }
}
```

ActionBar Actions

La barra delle azioni fornisce agli utenti accesso agli elementi di azione più importanti relativi al contesto attuale dell'applicazione, quelli che appaiono direttamente nella barra delle azioni con un'icona e/o testo sono conosciute come pulsanti di azione.

> Nota
> Le azioni che non possono adattarsi nella barra delle azioni o non sono abbastanza importanti vengono nascoste nel menu di overflow delle azioni

L'utente può visualizzare un elenco delle altre azioni premendo il pulsante di overflow (o il pulsante Menu del dispositivo, se disponibile) sul lato destro della barra delle azioni.

Quando un'attività viene avviata, il sistema popola gli elementi di azione chiamando il metodo `onCreateOptionsMenu()`, si utilizza questo metodo per "infilare" una risorsa del menu che definisce tutti gli elementi di azione.

file `res/menu/main_activity_actions.xml`

```
<menu xmlns:android="http://schemas.android.com/apk/res/android" >
        <item android:id="@+id/action_search"
                android:icon="@drawable/ic_action_search"
                android:title="@string/action_search"/>

        <item android:id="@+id/action_compose"
                android:icon="@drawable/ic_action_compose"
                android:title="@string/action_compose" />
</menu>
```

XML "infilato" nel metodo `onCreateOptionsMenu(...)`

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu items for use in the action bar
        MenuInflater inflater = getMenuInflater();
        inflater.inflate(R.menu.main_activity_actions, menu);
        return super.onCreateOptionsMenu(menu);
}
```

Per richiedere che un elemento appaia direttamente nella barra delle azioni come pulsante di azione, includi `showAsAction="ifRoom"` nell'elemento `<item>` del file di menu XML, se non c'è abbastanza spazio nella barra delle azioni l'elemento verrà posizionato nel menu di overflow.

```
<menu xmlns:android="http://schemas.android.com/apk/res/android"
                xmlns:yourapp="http://schemas.android.com/apk/res-auto" >

        <item android:id="@+id/action_search"
                        android:icon="@drawable/ic_action_search"
                        android:title="@string/action_search"
                        yourapp:showAsAction="ifRoom" />
        <!-- ... -->
</menu>
```

Se l'elemento di menu utilizza un titolo e un'icona, specificati tramite gli attributi `title` e `icon`, l'elemento di azione mostrerà solo l'icona per impostazione predefinita.
Se si desidera visualizzare anche il testo del titolo, bisogna aggiungere il tag `withText` all'attributo `showAsAction`

```
<item yourapp:showAsAction="ifRoom|withText" ... />
```

Handling Actions

Quando l'utente preme un'azione, il sistema chiama il metodo `onOptionsItemSelected()` della relativa attività, ed utilizzando il `MenuItem` passato, si può identificare l'azione chiamando `getItemId()` che restituisce l'ID univoco fornito dall'attributo `id` dell'elemento `<item>` nel file di menu XML, consentendo di eseguire l'azione desiderata.

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
        // Handle presses on the action bar items
        switch (item.getItemId()) {
                case R.id.action_search:
                        openSearch();
                        return true;

                case R.id.action_compose:
                        composeMessage();
                        return true;

                default:
                        return super.onOptionsItemSelected(item);
        }
}
```

ActionBar & Navigation

Abilitare l'icona dell'app come pulsante "Su" consente all'utente di navigare nell'applicazione in base alle relazioni gerarchiche tra le schermate, ad esempio se la schermata A mostra un elenco di elementi e selezionando un elemento porta alla schermata B, allora la schermata B dovrebbe includere il pulsante "Su", che riporta alla schermata A.

Per abilitare l'icona dell'app come pulsante "Su", chiama il metodo `setDisplayHomeAsUpEnabled()`

```
@Override
protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_details);
        ActionBar actionBar = getSupportActionBar();
        actionBar.setDisplayHomeAsUpEnabled(true);
        // ...
}
```

Ora l'icona nella barra delle azioni appare con la freccia "Su" (Up caret), tuttavia per impostazione predefinita, non farà nulla, quindi per specificare l'attività da aprire quando l'utente preme il pulsante "Su", ci sono 2 opzioni:

- Mediante file XML

- Programmaticamente

ActionBar & Navigation - XML

Questa è la migliore opzione quando l'attività genitore è sempre la stessa, dichiarando nel manifest quale attività è l'attività genitore, la barra delle azioni esegue automaticamente l'azione corretta quando l'utente preme il pulsante "Su".

> Nota
> A partire da Android 4.1 (API livello 16), è possibile dichiarare l'attività genitore utilizzando l'attributo `parentActivityName` nell'elemento `<activity>`.

Per supportare dispositivi più vecchi utilizzando la libreria di supporto, è necessario includere anche un elemento `<meta-data>` che specifica l'attività genitore come valore per `android.support.PARENT_ACTIVITY`.

Una volta specificata l'attività genitore nel manifest in questo modo e abilitato il pulsante "Su" con `setDisplayHomeAsUpEnabled()`, il lavoro è completato e la barra delle azioni esegue correttamente la navigazione "Su"

```
<application ... >

        ...
        <!-- The main/home activity (has no parent activity) -->
        <activity
                        android:name="com.example.myfirstapp.MainActivity" ...>
                ...
        </activity>
        <!-- A child of the main activity -->
        <activity
                        android:name="com.example.myfirstapp.DisplayMessageActivity"
                        android:label="@string/title_activity_display_message"
                        android:parentActivityName="com.example.myfirstapp.MainActivity" >
        <!-- Parent activity meta-data to support API level 7+ -->
        <meta-data
                        android:name="android.support.PARENT_ACTIVITY"
                        android:value="com.example.myfirstapp.MainActivity" />
        </activity>
</application>
```

ActionBar & Navigation - Programmatically

Definire la navigazione attraverso il codice è appropriato quando l'attività genitore può essere diversa a seconda di come l'utente è arrivato alla schermata corrente.
Il sistema chiama `getSupportParentActivityIntent()` quando l'utente preme il pulsante "Su" mentre naviga nell'applicazione, se l'attività che deve essere aperta durante la navigazione "Su" varia a seconda di come l'utente è arrivato alla posizione corrente, allora bisognerebbe sovrascrivere questo metodo per restituire l'intent che avvia l'attività genitore appropriata.

Il sistema chiama `onCreateSupportNavigateUpTaskStack()` per l'attività quando l'utente preme il pulsante "Su" mentre l'attività sta eseguendo in un task che non appartiene alla applicazione. Pertanto, bisogna utilizzare il `TaskStackBuilder` passato a questo metodo per costruire la pila di back appropriata che dovrebbe essere sintetizzata quando l'utente naviga verso l'alto.

Anche se si sovrascrive `getSupportParentActivityIntent()` per specificare la navigazione "Su" mentre l'utente naviga nell'applicazione, si può evitare la necessità di implementare `onCreateSupportNavigateUpTaskStack()` dichiarando attività genitori "predefinite" nel file manifest come mostrato in precedenza.

In questo caso, l'implementazione predefinita di `onCreateSupportNavigateUpTaskStack()` sintetizzerà una pila di back basata sulle attività genitore dichiarate nel manifesto.

```
@Override
public boolean onOptionsItemSelected(MenuItem item){
        switch (item.getItemId()) {

                // Respond to the action bar's Up/Home button
                case android.R.id.home:

                        //NavUtils.navigateUpFromSameTask(this);
                        Intent intent = new Intent(this, MainActivity.class);

                        // svuota lo stak delle attività e ne crea una nuova
                        intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP |
Intent.FLAG_ACTIVITY_NEW_TASK);
                        startActivityForResult(intent, 0);
                        return true;
        }
        return super.onOptionsItemSelected(item);
}
```

ActionBar → ToolBar

Con Android 5.0, Android introduce un nuovo widget chiamato Toolbar, che è una generalizzazione del pattern Action Bar.

La Toolbar offre maggiore controllo e flessibilità, e può essere definito nel file XML, è una vista all'interno della gerarchia come qualsiasi altra vista, il che la rende più facile da inserire tra le altre viste, animarla e reagire agli eventi di scorrimento.

> Nota
> Si può anche impostarla come action bar dell'attività, il che significa che le azioni del menu delle opzioni standard verranno visualizzate all'interno di esso

La Toolbar è completamente supportato in AppCompat, ha le stesse caratteristiche e API del widget di framework, ed è implementata nella classe `android.support.v7.widget.Toolbar`.

La differenza più importante è che questa Toolbar ora fa parte del layout, consentendo di lavorarci in qualsiasi modo si voglia:

- Animazioni
- Overlay del drawer
- ecc...

Tutte le tue attività devono estendere `AppCompatActivity`, che a sua volta estende `FragmentActivity` dalla libreria di supporto v4, così da poter continuare a utilizzare i frammenti.

> Nota
> Tutti i temi (che vogliono una Action Bar/Toolbar) devono ereditare da `Theme.AppCompat`

Quando si infila qualcosa da mostrare nella barra delle azioni (come un `SpinnerAdapter` per la navigazione a elenco nella toolbar), assicurati di utilizzare il contesto tematizzato della barra delle azioni, ottenuto tramite `getSupportActionBar().getThemedContext()`.

Bisogna utilizzare i metodi statici in `MenuItemCompat` per qualsiasi chiamata relativa alle azioni su un `MenuItem`

ActionBar → ToolBar

```
<style name="AppTheme.Base" parent="Theme.AppCompat.Light">
        <item name="colorPrimary">@color/primary</item>
        <item name="colorPrimaryDark">@color/primary_dark</item>
        <item name="android:textColorPrimary">@color/blue_grey_100</item>
        <item name="android:textColor">@color/primary</item>
        <!-- <item name="android:textColorSecondary">@color/primary</item> -->
        <!-- <item name="android:background">@color/primary</item> -->
        <item name="android:windowNoTitle">true</item>
        <item name="windowActionBar">false</item>
</style>

<!--
Disabilitare la ActionBar tradizionale (e vecchia) per aggiungere e gestire manualmente la
nuova Toolbar.
-->
<style name="AppTheme.NoActionBar">
        <item name="windowActionBar">false</item>
        <item name="windowNoTitle">true</item>
</style>
```

Toolbar Layout XML

```
<LinearLayout
                xmlns:android="http://schemas.android.com/apk/res/android"
                xmlns:tools="http://schemas.android.com/tools"
                android:layout_width="match_parent"
                android:layout_height="match_parent"
                android:orientation="vertical" >

        <androidx.appcompat.widget.Toolbar
                        android:id="@+id/my_awesome_toolbar"
                        android:layout_width="match_parent"
                        android:layout_height="wrap_content"
                        android:background="?attr/colorPrimary"
                        android:minHeight="?attr/actionBarSize" />

        <FrameLayout
                        android:id="@+id/container"
                        android:layout_width="match_parent"
                        android:layout_height="match_parent"
```

```
                        tools:context="com.mobdev.multifragment.MainActivity"
                        tools:ignore="MergeRootFrame" >
        </FrameLayout>
</LinearLayout>
```

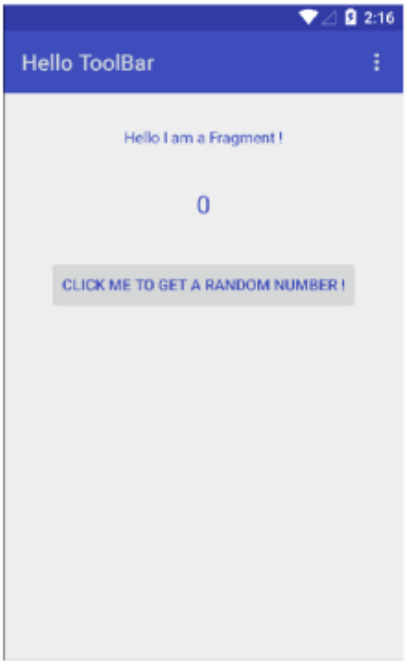
Toolbar Code

```
// importa la classe Toolbar
import androidx.appcompat.widget.Toolbar;

public class InfoActivity extends ActionBarActivity {
        @Override
        protected void onCreate(Bundle savedInstanceState) {

                super.onCreate(savedInstanceState);
                setContentView(R.layout.single_fragment_activity);
                […]

                // Recuperare e configurare la Toolbar dal layout:
                Toolbar toolbar = (Toolbar)findViewById(R.id.my_awesome_toolbar);
                toolbar.setTitle("App Info");
                // Imposta la Toolbar come ActionBar
                setSupportActionBar(toolbar);
                // Utilizza un metodo appartenente alla ActionBar classica
                getSupportActionBar().setHomeButtonEnabled(true);
                getSupportActionBar().setDisplayHomeAsUpEnabled(true);
        }
        […]
}
```

Toolbar Code - Navigation

```
@Override
public boolean onOptionsItemSelected(MenuItem item){
        switch (item.getItemId()) {
                // Respond to the action bar's Up/Home button
                case android.R.id.home:
                        //NavUtils.navigateUpFromSameTask(this);
                        Intent intent = new Intent(this, MainActivity.class);
                        intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP |
Intent.FLAG_ACTIVITY_NEW_TASK);
                        startActivityForResult(intent, 0);
                        return true;
        }
return super.onOptionsItemSelected(item);
}
```

> Nota
> Il codice e il comportamento sono gli stessi della ActionBar tradizionale



Android Application Icons

Icon Design: https://m2.material.io/design/iconography/product-icons.html

Tab Icon Design: http://developer.android.com/guide/practices/ui_guidelines/icon_design_tab.html

Android Icon Web Tool: http://android-ui-utils.googlecode.com/hg/asset-studio/dist/index.html

---

Lists & RecyclerView

Lists

Le liste sono un modo comune e potente per visualizzare contenuti eterogenei all'interno delle applicaizoni, con il material design, si può creare liste complesse utilizzando il RecyclerView.

> Nota
> `ListView` è stato abbandonato a favore di `RecyclerView`, che fa parte della libreria di supporto, e `CardView` che si vedrà successivamente

È satto presente ListView per molto tempo su Android ed il concetto di riciclo delle viste non è nuovo, nelle versioni precedenti (con ListView), l'aspetto, il riciclo e tutto erano strettamente accoppiati.

> Nota
> Il nuovo approccio è più flessibile con il nuovo RecyclerView

RecyclerView

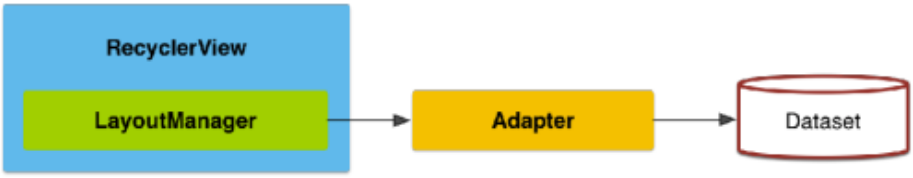
Il widget RecyclerView è una versione più avanzata e flessibile di ListView, è un contenitore per visualizzare grandi set di dati che possono si possini scorrere in modo molto efficiente mantenendo un numero limitato di viste.

> Nota
> È consigliato utilizzare il widget RecyclerView quando si hanno collezioni di dati i cui elementi cambiano durante l'esecuzione in base alle azioni dell'utente o agli eventi di rete

La classe `RecyclerView` semplifica la visualizzazione e la gestione di grandi set di dati fornendo:

- Gestori di layout per posizionare gli elementi
- Animazioni predefinite per operazioni comuni sugli elementi, come la rimozione o l'aggiunta di elementi

> Nota
> C'è anche la flessibilità di definire gestori di layout e animazioni personalizzati per i widget RecyclerView



Il widget RecyclerView si basa su un Adapter e un LayoutManager per il rendering.

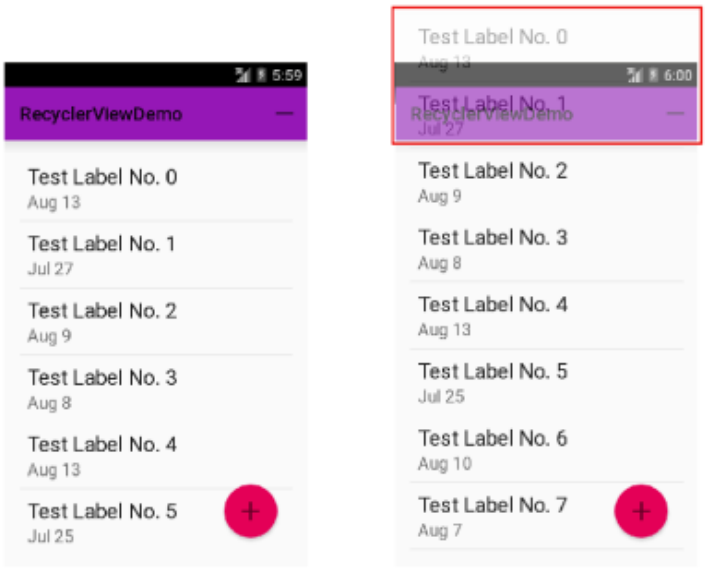
> Nota
> un Adapter è utilizzato per fornire al LayoutManager informazioni su quanti elementi devono essere visualizzati (`getItemCount()`) e la vista effettiva da utilizzare

Per creare un Adapter bisogna estendere la classe `RecyclerView.Adapter`, ed i dettagli implementatici dipendono dalle specificità del dataset e dal tipo di viste.

Layout Manager

Un Layout Manager posiziona le viste degli elementi all'interno di un RecyclerView e determina quando riutilizzare le viste degli elementi che non sono più visibili all'utente, per farlo il gestore può chiedere all'adapter di sostituire i contenuti della vista con un elemento diverso del dataset.

> Nota
> Il riciclo delle viste in questo modo migliora le prestazioni evitando la creazione di viste inutili o eseguendo ricerche costose con `findViewById()`.



Il RecyclerView fornisce questi Layout Manager integrati:

- `LinearLayoutManager`, che mostra gli elementi in un elenco con scorrimento verticale o orizzontale
- `GridLayoutManager`, che mostra gli elementi in una griglia
- `StaggeredGridLayoutManager`, che mostra gli elementi in una griglia sfalsata (griglia con elementi di altezze diverse)

> Nota
> Per creare un gestore di layout personalizzato, bisogna estendere la classe `RecyclerView.LayoutManager`

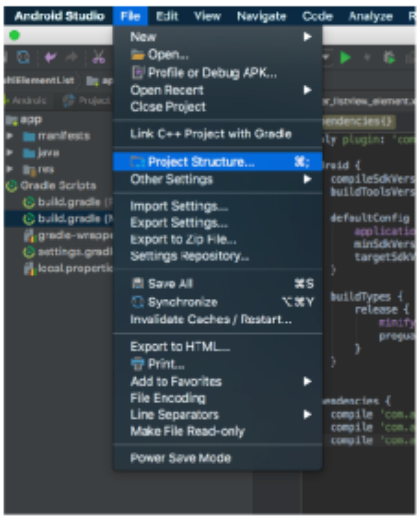
Il riciclo delle viste è un approccio molto utile, risparmia risorse della CPU, poiché non è necessario infilare nuove viste ogni volta, e risparmia memoria.

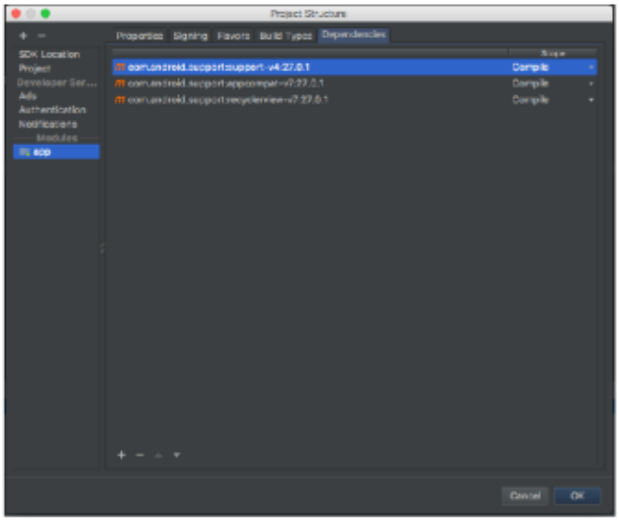
Recycler View Dependencies

Android Studio

```
dependencies {
        ...
```

```
        implementation 'androidx.recyclerview:recyclerview:1.1.0'
}
```





Recycler View XML

```
<androidx.recyclerview.widget.RecyclerView
                android:id="@+id/my_recycler_view"
                android:layout_width="match_parent"
                android:layout_height="match_parent"
                android:scrollbars="vertical" />
```

Quando si ha aggiunto un widget RecyclerView al layout, si può ottenere un riferimento all'oggetto, collegarlo a un gestore di layout e associargli un adapter per i dati da visualizzare.

Recycler View Code

```
mRecyclerView = (RecyclerView)rootView.findViewById(R.id.my_recycler_view);

// use a linear layout manager
mLayoutManager = new LinearLayoutManager(getActivity());
mLayoutManager.setOrientation(LinearLayoutManager.VERTICAL);
mLayoutManager.scrollToPosition(0);

mRecyclerView.setLayoutManager(mLayoutManager);

// use this setting to improve performance if you know that changes
// in content do not change the layout size of the RecyclerView
mRecyclerView.setHasFixedSize(true);

// specify an adapter (next slide)
mAdapter = new MyAdapter(myDataSet);
mRecyclerView.setAdapter(mAdapter);
```

Recycler Adapter

L'adapter fornisce:

- Accesso agli elementi del tuo set di dati
- Crea le viste per gli elementi
- Sostituisce il contenuto di alcune delle viste con nuovi elementi di dati quando l'elemento originale non è più visibile

e richiede un `ViewHolders`, ovvero i container per le viste da visualizzare (celle riciclabili)

```
public class MyAdapter extends RecyclerView.Adapter<MyAdapter.ViewHolder> {

        private ArrayList<Double> mDataset;

        // Provide a reference to the views for each data item
        // Complex data items may need more than one view per item, and
        // you provide access to all the views for a data item in a view holder
        public class ViewHolder extends RecyclerView.ViewHolder {
                // ..
```

```
        }
        // Provide a suitable constructor (depends on the kind of dataset)
        public MyAdapter(ArrayList<Double> myDataset) {
                mDataset = myDataset;
        }

        // Create new views (invoked by the layout manager)
        @Override
        public MyAdapter.ViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
                // ..
        }

        // Replace the contents of a view (invoked by the layout manager)
        @Override
        public void onBindViewHolder(ViewHolder holder, final int position) {
        // ..
        }

        // Return the size of your dataset (invoked by the layout manager)
        @Override
        public int getItemCount() {
                return mDataset.size();
        }
}
```

Creazione nuova vista o rimpiazzo dei contenuti

```
// Create new views (invoked by the layout manager)
@Override
public MyAdapter.ViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
        // create a new view
        View v = LayoutInflater.from(parent.getContext()).inflate(R.layout.listview_element,
parent, false);
        // set the view's size, margins, paddings and layout parameters
        ViewHolder vh = new ViewHolder(v);
        return vh;
}

// Replace the contents of a view (invoked by the layout manager)
@Override
public void onBindViewHolder(ViewHolder holder, final int position) {
        // - get element from your dataset at this position
        // - replace the contents of the view with that element
        holder.setText(""+mDataset.get(position));
}
```

View Holder

I ViewHolder sono essenzialmente delle cache degli oggetti View

```
public class ViewHolder extends RecyclerView.ViewHolder {
        private View v = null;

        public ViewHolder(View v) {
                super(v);
                this.v = v;
        // Click listener per la vista, sia setOnLongClickListener(), sia
setOnClickListener()
        v.setOnLongClickListener(new OnLongClickListener() {
                        @Override
                        public boolean onLongClick(View v) {
                                int position = getPosition();
                                NumberManager.getInstance().removeNumber(position);
                                notifyItemRemoved(position);
                                return false;
                        }
                        setOnLongClickListener()
                        setOnClickListener()
                });
        }
        public void setText(String text){
        TextView tView = (TextView)v.findViewById(R.id.myTextView);
        tView.setText(text);
        }
}
```

Animation

La classe `ItemAnimator` aiuta il RecyclerView ad animare gli elementi individuali, gli ItemAnimator gestiscono tre eventi:

- Un elemento viene aggiunto al set di dati
- Un elemento viene rimosso dal set di dati
- Un elemento si sposta a seguito di una o più delle operazioni precedenti

> Nota
> Esiste un'implementazione predefinita chiamata `DefaultItemAnimator`. Se non si imposta un ItemAnimator personalizzato, RecyclerView utilizza un'istanza di DefaultItemAnimator.

La classe `RecyclerView.Adapter` contiene molti metodi `notifyX()`, i due più specifici sono:

- `public final void notifyItemInserted(int position)`
- `public final void notifyItemRemoved(int position)`

```
addButton.setOnClickListener(new OnClickListener() {
        @Override
        public void onClick(View v) {
                Log.d(MainActivity.TAG,"Add Button Clicked !");

                // Animazione di inserimento
                Random rand = new Random();
                int number = rand.nextInt((1000 - 0) + 1);
                NumberManager.getInstance().addNumberToHead(Double.valueOf(number));

                // Animazione
                mAdapter.notifyItemInserted(0);
                mLayoutManager.scrollToPosition(0);
        }
});


v.setOnLongClickListener(new OnLongClickListener() {
        @Override
        public boolean onLongClick(View v) {
                int position = getPosition();

                // Animazione di rimozione
                NumberManager.getInstance().removeNumber(position);

                // Animazione
                notifyItemRemoved(position);
                return false;
        }
});
```

Handle Multiple View Types

L'Adapter di RecyclerView consente di definire e gestire più tipi di viste eterogenee per una lista, sovrascrivi il metodo `getItemViewType()` per restituire il tipo di vista dell'elemento in una determinata posizione, al fine di riciclarle correttamente.

L'implementazione predefinita di questo metodo restituisce 0, assumendo un singolo tipo di vista per l'adapter, a differenza degli adapter di ListView, i tipi non devono essere contigui.

Considerando l'uso di risorse id per identificare univocamente i tipi di vista degli elementi, la combinazione di `getItemViewType()` con più ViewHolder consente di gestire e creare liste con elementi eterogenei.

```
@Override
public int getItemViewType(int position) {
        // ...
}
```

Multiple View Holders

```
public class MyAdapter extends RecyclerView.Adapter<RecyclerView.ViewHolder> {
        // ...
}
```

Poiché ci sono più tipi di holder nella lista, si utilizza come riferimento un ViewHolder generico

Primo ViewHolder

```
public class FirstViewHolder extends RecyclerView.ViewHolder {
        private View v = null;

        public FirstViewHolder(View v) {
                super(v);
                this.v = v;
                // ...
        }
        public void updateUI(String text){
                // ...
        }
}
```

Secondo ViewHolder_

```
public class SecondViewHolder extends RecyclerView.ViewHolder {
        private View v = null;

        public SecondViewHolder(View v) {
                super(v);
                this.v = v;
                // ...
        }
        public void updateUI(String text){
                // ...
        }
}
```

i-esimo ViewHolder

```
public class iViewHolder extends RecyclerView.ViewHolder {
        private View v = null;

        public iViewHolder(View v) {
                super(v);
                this.v = v;
                // ...
        }
        public void updateUI(String text){
                // ...
        }
}
```

Manage Multiple ViewHolders

```
// Infila la corretta vista conformememnte al parametro viewType
@Override
public RecyclerView.ViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {

        if(viewType == 0){
                View v =
LayoutInflater.from(parent.getContext()).inflate(R.layout.first_listview_element, parent,
false);
                return new FirstViewHolder(v);
        }
        else if(viewType == 1){
                View v =
LayoutInflater.from(parent.getContext()).inflate(R.layout.second_listview_element, parent,
false);
                return new SecondViewHolder(v);
        }
}

// Effettua il cast dell'holder in base al tipo per gestire e aggiornare correttamente
l'interfaccia utente
@Override
public void onBindViewHolder(ViewHolder holder, final int position) {
        if(getItemViewType(position) == 0){
                FirstViewHolder firstViewHolder = (FirstViewHolder)holder;
                firstViewHolder.updateUI(…);
        }
        else if(getItemViewType(position) == 1){
```

```
                SecondViewHolder secondViewHolder = (SecondViewHolder)holder;
                secondViewHolder.updateUI(…);
        }
}
```

Card View

`CardView` estende la classe `FrameLayout` e consente di mostrare informazioni all'interno di schede con un aspetto uniforme su tutte le piattaforme.

> Nota
> I widget CardView possono avere ombre e angoli arrotondati

Per creare una scheda con un'ombra, si utilizza l'attributo `card_view:cardElevation`.

> Nota
> CardView utilizza un'elevazione reale e ombre dinamiche su Android 5.0 (API livello 21) e versioni successive, e ricorre a un'implementazione programmata delle ombre su versioni precedenti

Si utilizzano queste proprietà per personalizzare l'aspetto del widget CardView:

- Per impostare il raggio dell'angolo nelle tue layout, utilizza l'attributo `card_view:cardCornerRadius`
- Per impostare il raggio dell'angolo nel codice, utilizza il metodo `CardView.setRadius`
- Per impostare il colore di sfondo di una scheda, utilizza l'attributo `card_view:cardBackgroundColor`

Card View in your layout

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
                        android:layout_width="fill_parent"
                        android:layout_height="fill_parent"
                        android:layout_marginLeft="10dp"
                        android:layout_marginRight="10dp"
                        android:background="@drawable/listview_selector"
                        android:orientation="vertical" >
        <!--
                estende FrameLayout, rendendo più semplice l'utilizzo di un solo figlio.
                Ad esempio, si può utilizzare un semplice TextView o un layout ViewGroup per
creare schede più complesse.
        -->
        <androidx.cardview.widget.CardView
                        xmlns:card_view="http://schemas.android.com/apk/res-auto"
                        android:id="@+id/card_view"
                        android:layout_width="match_parent"
                        android:layout_height="match_parent"
                        android:layout_margin="10dp"
                        android:layout_gravity="center"
                        android:elevation="10dp"
                        card_view:cardCornerRadius="4dp" >

                <TextView
                                android:id="@+id/myTextView"
                                android:layout_width="match_parent"
                                android:layout_height="wrap_content"
                                android:layout_margin="10dp"
                                android:text="Large Text"
                                android:textAppearance="?android:attr/textAppearanceMedium" />

        </android.support.v7.widget.CardView>
</LinearLayout>
```

> Nota
> GRADLE → implementazione `androidx.cardview:cardview:1.0.0`