Si estás interesado en crear una aplicación en Android que haga uso de listas para desplegar datos, Android Lollipop presenta dos nuevos widgets para hacerte la vida más fácil, RecyclerView y CardView. Usando éstos widgets, es muy fácil dar a tu aplicación una apariencia y un sentido que cumple con las pautas mencionadas en la especificación de material design de Google.

Requisitos Previos

Para continuar, deberás usar la versión más reciente de Android Studio. Puedes obtenerla en el sitio de Android Developer.

1. Soporte para versiones anteriores

Al momento de escribir éste post, menos del 2% de dispositivos Android ejecutan Android Lollipop. Sin embargo, gracias a la libreria de soporte v7Support Library, puedes usar los widgets RecyclerView y CardView en dispositivos que ejecutan versiones anteriores de Android al añadir las siguientes líneas a la seccion de dependencies (dependencias) en el archivo build.grade de tu proyecto:

```
compile 'com.android.support:cardview-v7:21.0.+'
compile 'com.android.support:recyclerview-v7:21.0.+'
```

2. Creando un CardView

Un CardView es un ViewGroup. Como cualquier otro ViewGroup, puede añadirse a tu Activity (Actividad) o Fragment (Fragmento) utilizando un archivo XML que define el layout (maquetado).

Para crear una CardView vacía, habrías de añadir el siguiente código a tu XML Layout como se muestra en el siguiente snippet:

```
<android.support.v7.widget.CardView
        xmlns:card_view="https://schemas.android.com/apk/res-auto"
        android:layout_width="match_parent"
        android:layout_height="wrap_content">

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

Como un ejemplo más realista, vamos a crear un LinearLayout y colocar un CardView dentro de él. El CardView podría representar, por ejemplo, una persona y contener lo siguiente:

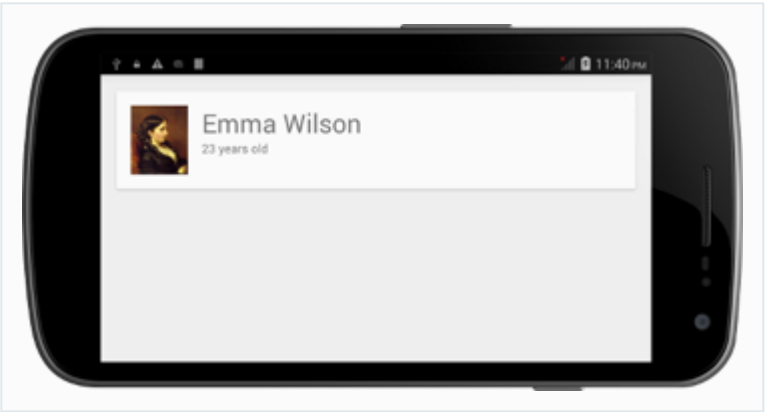
- un TextView para desplegar el nombre de la persona
- un TextView para desplegar la edad de la persona
- un ImageView para mostrar la foto de la persona

Así es como se vería el XML:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent" android:layout_height="match_parent"
    android:padding="16dp"
    >

    <android.support.v7.widget.CardView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/cv"
        >

        <RelativeLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:padding="16dp"
            >

            <ImageView
```

```
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:id="@+id/person_photo"
                android:layout_alignParentLeft="true"
                android:layout_alignParentTop="true"
                android:layout_marginRight="16dp"
                />

            <TextView
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:id="@+id/person_name"
                android:layout_toRightOf="@+id/person_photo"
                android:layout_alignParentTop="true"
                android:textSize="30sp"
                />

            <TextView
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:id="@+id/person_age"
                android:layout_toRightOf="@+id/person_photo"
                android:layout_below="@+id/person_name"
                />

        </RelativeLayout>

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

</LinearLayout>
```

Si éste XML es usado como el layout de una Activity, con los campos delTextView establecidos a valores significativos, entonces así es cómo se renderizaría en un dispositivo Android:



3. Creando un RecyclerView

Paso 1: Definirlo en un Layout

Usando una instancia de RecyclerView es un poco más complicado. Sin embargo definirlo en un archivo XML de Layout es muy sencillo. Puedes definirlo en un layout así:

```
<android.support.v7.widget.RecyclerView
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:id="@+id/rv"
    />
```

Para obtener una forma de manejarlo en tu Activity, usa el siguiente snippet:

```
RecyclerView rv = (RecyclerView)findViewById(R.id.rv);
```

Si estás seguro que el tamaño del RecyclerView no se cambiará, puedes añadirlo lo siguiente para mejorar el desempeño:

```
rv.setHasFixedSize(true);
```

Paso 2: Usando un Layout Manager

A diferencia de un ListView, un RecyclerView necesita un Layout Manager para manejar el posicionamiento de sus elementos. Podrías definir tu propioLayoutManager al extender la clase RecyclerView.LayoutManager. Sin embargo, en la mayoría de los casos, podrías simplemente usar una de las subclases LayoutManager predefinidas:

- `LinearLayoutManager`
- `GridLayoutManager`
- StaggeredGridLayoutManager

En éste tutorial, voy a usar un LinearLayoutManager. Ésta subclase LayoutManager por defecto hará que tu RecyclerView parezca una ListView.

```
LinearLayoutManager llm = new LinearLayoutManager(context);
rv.setLayoutManager(llm);
```

Paso 3: Definiendo los Datos

Al igual que un ListView, un RecyclerView, necesita un adaptador para acceder a sus datos. Pero antes de crear una adaptador, vamos a crear los datos con los que podamos trabajar. Creamos una simple clase para representar una persona y luego escribimos un método para inicializar una List de objetos Person:

```
class Person {
    String name;
    String age;
    int photoId;

    Person(String name, String age, int photoId) {
        this.name = name;
        this.age = age;
        this.photoId = photoId;
    }
}

private List<Person> persons;

// This method creates an ArrayList that has three Person objects
// Checkout the project associated with this tutorial on Github if
// you want to use the same images.
private void initializeData(){
    persons = new ArrayList<>();
    persons.add(new Person("Emma Wilson", "23 years old", R.drawable.emma));
    persons.add(new Person("Lavery Maiss", "25 years old", R.drawable.lavery));
    persons.add(new Person("Lillie Watts", "35 years old", R.drawable.lillie));
}
```

Paso 4: Creando un Adaptador

Para crear un adaptador que una RecyclerView puede usar, debes extender RecyclerView.Adapter. Éste adaptador sigue el patrón de diseño view holder,que significa que define una clase interna que extienda de RecyclerView.ViewHolder. Éste patrón minimiza el número de llamadas al costoso método findViewById.

Anteriormente en éste tutorial, ya definimos el XML Layout para un CardView que representa una persona. Vamos a reutilizar ese layout ahora. Dentro del constructor de nuestro ViewHKolder, inicializa las views (vistas) que pertenezcan a los elementos de nuestro RecyclerView.

```
01  public class RVAdapter extends RecyclerView.Adapter<RVAdapter.PersonViewHolder>{
02
03      public static class PersonViewHolder extends RecyclerView.ViewHolder {
04          CardView cv;
05          TextView personName;
06          TextView personAge;
07          ImageView personPhoto;
08
09          PersonViewHolder(View itemView) {
10              super(itemView);
11              cv = (CardView)itemView.findViewById(R.id.cv);
12              personName = (TextView)itemView.findViewById(R.id.person_name);
13              personAge = (TextView)itemView.findViewById(R.id.person_age);
14              personPhoto = (ImageView)itemView.findViewById(R.id.person_photo);
15          }
16      }
17
18  }
```

Luego, añade un constructor al adaptador para que pueda manejar los datos que muestra el RecyclerView. Como nuestros datos están en forma de List (lista) de objetos Person, usa el siguiente código:

```
1  List<Person> persons;
2
3  RVAdapter(List<Person> persons){
4      this.persons = persons;
5  }
```

ReccylerView.Adapter tiene tres métodos abstractos a los que debemos aplicar el modificador override. Comencemos con el método getItemCount. Éste debería regresar el número de elementos presentes en los datos. Como nuestro datos

están en forma de una List, sólo necesitamos llamar al método size en el objeto List:

```
@Override
public int getItemCount() {
    return persons.size();
}
```

Posteriormente, aplicamos el modificador override al método onCreativeViewHolder. Como sugiere su nombre, éste método es llamado cuando el ViewHolder necesita ser inicializado. Especificamos el layout que cada elemento de RecyclerView debería usar. Ésto se hace al inflar el layout usando LayoutInflater, pasando el resultado al constructor del ViewHolder.

```
@Override
public PersonViewHolder onCreateViewHolder(ViewGroup viewGroup, int i) {
    View v = LayoutInflater.from(viewGroup.getContext()).inflate(R.layout.item, viewGrou
    PersonViewHolder pvh = new PersonViewHolder(v);
    return pvh;
}
```

Aplica el modificador Override a onBindViewHolder para especificar el contenido de cada elemento del RecyclerView. Éste método es muy similar al método getView de un adaptador de ListView. En nuestro ejemplo, aquí es donde tienes que establecer los valores de los campos de nombre, edad y foto del CardView.

```
@Override
public void onBindViewHolder(PersonViewHolder personViewHolder, int i) {
    personViewHolder.personName.setText(persons.get(i).name);
    personViewHolder.personAge.setText(persons.get(i).age);
    personViewHolder.personPhoto.setImageResource(persons.get(i).photoId);
}
```

Finalmente, necesitas hacer el override en el método onAttachedToRecyclerView. Por ahora, simplemente podemos utilizar la implementación de la superclase de éste método como se muestra abajo:

```
@Override
public void onAttachedToRecyclerView(RecyclerView recyclerView) {
    super.onAttachedToRecyclerView(recyclerView);
}
```

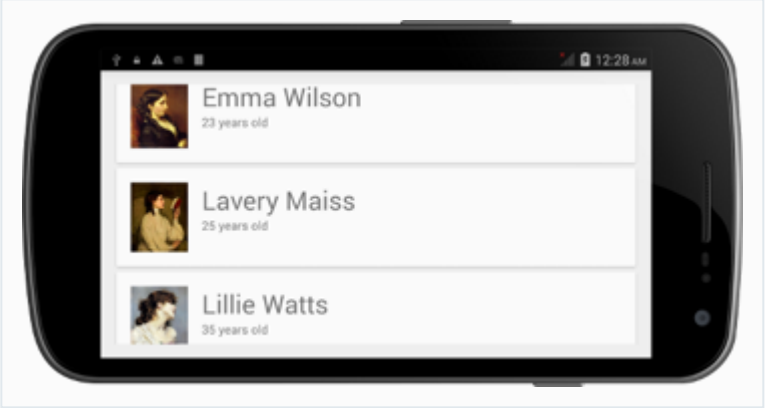
Paso 5: Usando el Adaptador

Ahora que está listo el adaptador, agrega el siguiente código a tu Activity para inicializar y usar el adaptador al llamar al constructor del adaptador y al método setAdapter de RecyclerView:

```
RVAdapter adapter = new RVAdapter(persons);
rv.setAdapter(adapter);
```

Paso 6: Compila y Ejecuta

Cuando ejecutas el ejemplo de RecyclerView en un dispositivo Android, deberíasver algo similar a la siguiente imagen.



Conclusión

En éste tutorial, has aprendido a usar los widgets CardView y RecyclerView que fueron introducidos en Android Lollipop. También has visto ejemplos de como usar éstos widgets en aplicaciones Material Design. Nota que aunque un RecyclerView puede hacer casi lo mismo que un ListView, para pequeños datasets (representaciones de datos), usando un ListView todavía es preferible pues requiere menos líneas de código.

Puedes ver la Guía de Referencia para Desarrolladores de Android para más información sobre las clases de CardView y RecyclerView.