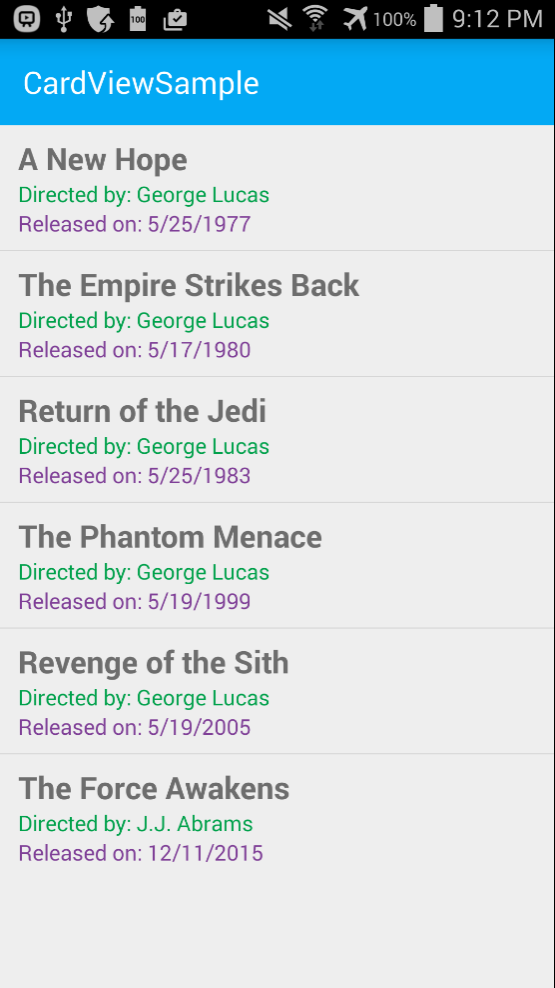
Day 19: CardView

Today, we will explore more about CardView in Xamarin.Android.

CardView was introduced in Android 5.0 (Lollipop). CardView is mainly used to create Card like user interface for showing lists of data in Android Application. CardView can hold just text (basic) and text with images (advanced) in its Layout. CardView’s Layout is based on the FrameLayout widget with rounded corners and shadows.

We will be taking the Star Wars sample app that we have been working on in the last two days and we will be transforming to use CardView. To recap, here is how the Star Wars data App currently looks like –



Please note, couple of fonts and styling might appear a little different than yesterday because I moved the same app into the [Xamarin.Android AppCompat Template](http://blog.falafel.com/31-days-of-xamarin-android-day-3-creating-your-first-android-application-with-xamarin/)

To get started with CardView first we should install the required Nuget Packages. CardView functionality depends on three Nuget packages from Xamarin –

1. Xamarin.Android.Support.v7.AppCompat
2. Xamarin.Android.Support.v4 (this package is a dependency for the v7 package and is automatically brought down when installing the V7 package)
3. Xamarin.Android.Support.v7.CardView

Please be sure to use at least version 22.2.0.0 for these Packages.

Once you bring all the required Nuget Packages down, to the Main.axml (or whichever is your main layout) let’s go and add a ListView to host the Movies data. –

|  |
| --- |
| <?xml version="1.0" encoding="utf-8"?>  <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"  xmlns:local="http://schemas.android.com/apk/res-auto"  android:layout\_width="fill\_parent"  android:layout\_height="fill\_parent">  <include  android:id="@+id/toolbar"  layout="@layout/toolbar" />  <LinearLayout  android:orientation="vertical"  android:layout\_width="fill\_parent"  android:layout\_height="fill\_parent"  android:id="@+id/main\_content"  android:layout\_below="@id/toolbar">  <ListView  android:id="@+id/moviesListView"  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content" />  </LinearLayout>  </RelativeLayout> |

Gist file link: <https://gist.github.com/vkoppaka/f4efa07e21c5a4a6d51b>

And the Activity that drives this layout would look similar to the activity we saw in the last tutorial with couple of changes –

|  |
| --- |
| using Android.App;  using Android.Widget;  using Android.OS;  namespace CardViewSample  {  [Activity(Label = "CardViewSample", MainLauncher = true, Icon = "@drawable/icon")]  public class MainActivity : BaseActivity  {  protected override int LayoutResource  {  get { return Resource.Layout.main; }  }  protected override void OnCreate(Bundle bundle)  {  base.OnCreate(bundle);  var moviesListView = FindViewById<ListView>(Resource.Id.moviesListView);  moviesListView.ItemClick += moviesListView\_ItemClick;  var moviesAdapter = new MovieAdapter(this, MoviesRepository.Movies);  moviesListView.Adapter = moviesAdapter;  SupportActionBar.SetDisplayHomeAsUpEnabled(false);  SupportActionBar.SetHomeButtonEnabled(false);  }  void moviesListView\_ItemClick(object sender, AdapterView.ItemClickEventArgs e)  {  Toast.MakeText(this, MoviesRepository.Movies[e.Position].ToString(), ToastLength.Long).Show();  }  }  } |

Gist file link: <https://gist.github.com/vkoppaka/c19f5b509e3528c825f9>

The only thing new in the above snippet is that we are not inheriting from a “BaseActivity”. This BaseActivity gives us an easy way to in turn inherit from AppCompatActivity in the V7 Nuget Package –

|  |
| --- |
| using Android.OS;  using Android.Support.V7.App;  using Android.Support.V7.Widget;    namespace CardViewSample  {  public abstract class BaseActivity : AppCompatActivity  {  public Toolbar Toolbar  {  get;  set;  }  protected override void OnCreate(Bundle bundle)  {  base.OnCreate(bundle);  SetContentView(LayoutResource);  Toolbar = FindViewById<Toolbar>(Resource.Id.toolbar);  if (Toolbar != null)  {  SetSupportActionBar(Toolbar);  SupportActionBar.SetDisplayHomeAsUpEnabled(true);  SupportActionBar.SetHomeButtonEnabled(true);  }  }  protected abstract int LayoutResource  {  get;  }  protected int ActionBarIcon  {  set { Toolbar.SetNavigationIcon(value); }  }  }  } |

Gist file link: <https://gist.github.com/vkoppaka/4363f27e1d5fe1e65266>

All that the BaseActivity is doing here is that it’s setting the LayoutResource as the ContentView to show.

And now, let’s switch our attention to how the individual row is inflated where we get to use the CardView. The AXML for the MovieRow Layout which has CardView looks like –

|  |
| --- |
| <?xml version="1.0" encoding="utf-8"?>  <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  xmlns:cardview="http://schemas.android.com/apk/res-auto"  android:orientation="vertical"  android:layout\_width="fill\_parent"  android:layout\_height="fill\_parent"  android:gravity="center\_horizontal"  android:padding="5dp">  <android.support.v7.widget.CardView  android:layout\_width="fill\_parent"  android:layout\_height="245dp"  android:layout\_gravity="center\_horizontal"  cardview:cardElevation="4dp"  cardview:cardCornerRadius="5dp">  <LinearLayout  android:layout\_width="fill\_parent"  android:layout\_height="240dp"  android:orientation="vertical"  android:padding="8dp">  <TextView  android:layout\_width="fill\_parent"  android:layout\_height="wrap\_content"  android:textAppearance="?android:attr/textAppearanceMedium"  android:textColor="#333333"  android:text="Photo Title"  android:id="@+id/cardViewText"  android:layout\_gravity="center\_horizontal"  android:layout\_marginLeft="5dp" />  </LinearLayout>  </android.support.v7.widget.CardView>  </LinearLayout> |

Gist file link: <https://gist.github.com/vkoppaka/8e5f040f1648622fcf9f>

In the above snippet, you can see we now have a CardView tag which contains a LinearLayout and a TextView. This TextView is used to show Text data in the CardView. Also, if you notice carefully, we are using a “cardview” xml namespace to define CardView specific properties.

|  |
| --- |
| xmlns:cardview="http://schemas.android.com/apk/res-auto" |

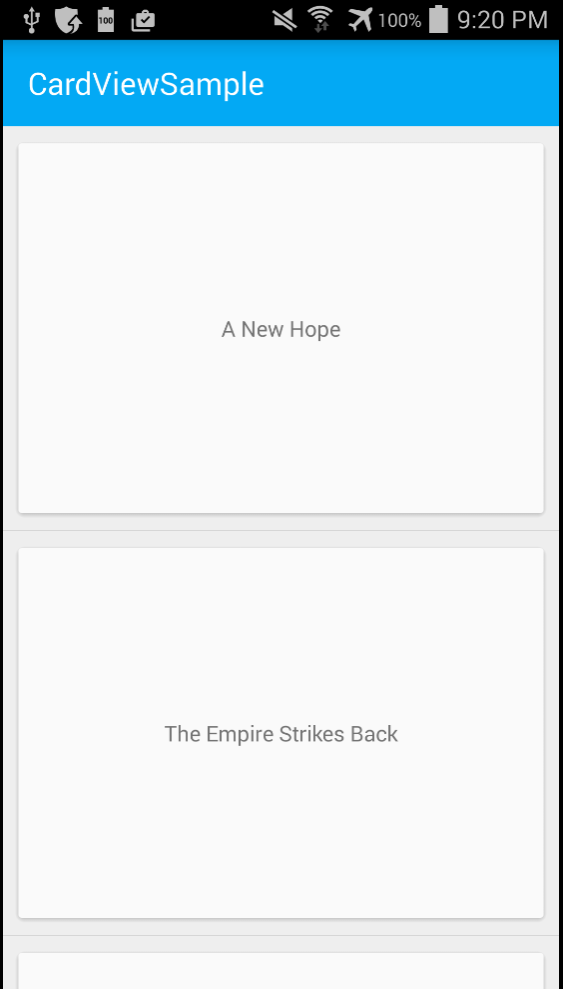
The reason for this is CardView is provided by a Nuget Package (V7 support library) so the attributes of CardView like cardElevation and cardCornerRadius are not available under the regular android: namespace.

And finally, let’s look at the Adapter that inflates the MovieRow CardView Layout –

|  |
| --- |
| using System.Collections.Generic;  using Android.App;  using Android.Views;  using Android.Widget;  namespace CardViewSample  {  public class MovieAdapter : BaseAdapter<Movie>  {  private readonly Activity context;  private readonly List<Movie> movies;  public MovieAdapter(Activity context, List<Movie> movies)  {  this.context = context;  this.movies = movies;  }  public override Movie this[int position]  {  get  {  return movies[position];  }  }  public override int Count  {  get  {  return movies.Count;  }  }  public override long GetItemId(int position)  {  return position;  }  public override View GetView(int position, View convertView, ViewGroup parent)  {  var view = convertView;  if (view == null)  {  view = context.LayoutInflater.Inflate(Resource.Layout.MovieRow, parent, false);  }  var titleTextView = view.FindViewById<TextView>(Resource.Id.cardViewText);  titleTextView.Text = movies[position].Title;  return view;  }  }  } |

There is not much different with the adapter when you compare it with the adapter that powers the ListView. All we are doing here is we are finding the TextView from the MovieRow and setting its Text property to the Title of the Movie.

If you run the application now, you will now see a CardView Layout of your data –



Let’s take this a notch further, and let’s add Image to the CardView. In the process, our MovieRow will change ever so slightly to include an ImageView –

|  |
| --- |
| <?xml version="1.0" encoding="utf-8"?>  <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  xmlns:cardview="http://schemas.android.com/apk/res-auto"  android:orientation="vertical"  android:layout\_width="fill\_parent"  android:layout\_height="fill\_parent"  android:gravity="center\_horizontal"  android:padding="5dp">  <android.support.v7.widget.CardView  android:layout\_width="fill\_parent"  android:layout\_height="245dp"  android:layout\_gravity="center\_horizontal"  cardview:cardElevation="4dp"  cardview:cardCornerRadius="5dp">  <LinearLayout  android:layout\_width="fill\_parent"  android:layout\_height="240dp"  android:orientation="vertical"  android:padding="8dp">  <ImageView  android:layout\_width="fill\_parent"  android:layout\_height="190dp"  android:id="@+id/imageView"  android:scaleType="centerCrop" />  <TextView  android:layout\_width="fill\_parent"  android:layout\_height="wrap\_content"  android:textAppearance="?android:attr/textAppearanceMedium"  android:textColor="#333333"  android:text="Photo Title"  android:id="@+id/cardViewText"  android:layout\_gravity="center\_horizontal"  android:layout\_marginLeft="5dp" />  </LinearLayout>  </android.support.v7.widget.CardView>  </LinearLayout> |

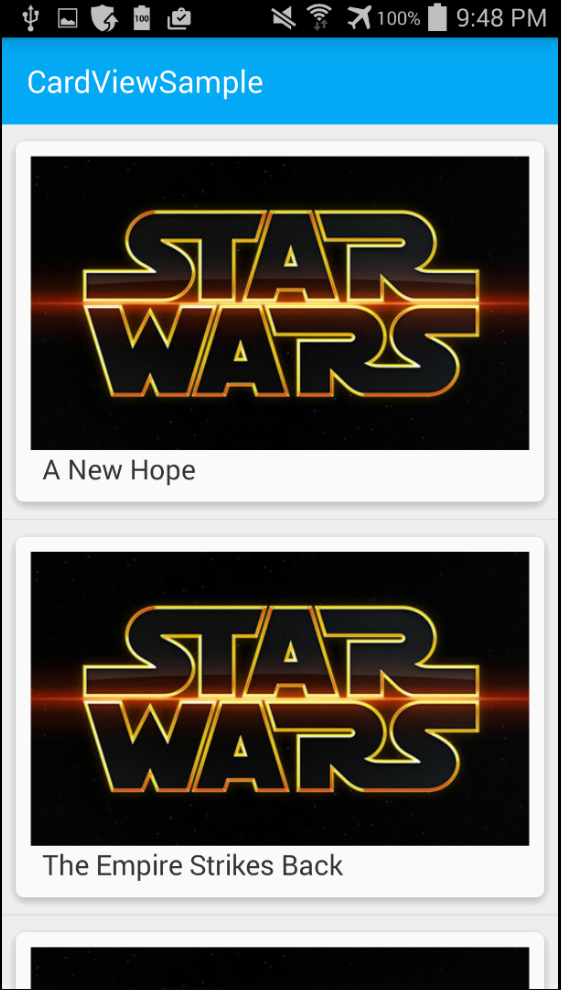
Gist file link: <https://gist.github.com/vkoppaka/4a1e6256b33bd5efa2ea>

And our Adapter also changes, to accommodate the ImageView –

|  |
| --- |
| using System.Collections.Generic;  using Android.App;  using Android.Views;  using Android.Widget;  namespace CardViewSample  {  public class MovieAdapter : BaseAdapter<Movie>  {  private readonly Activity context;  private readonly List<Movie> movies;  public MovieAdapter(Activity context, List<Movie> movies)  {  this.context = context;  this.movies = movies;  }  public override Movie this[int position]  {  get  {  return movies[position];  }  }  public override int Count  {  get  {  return movies.Count;  }  }  public override long GetItemId(int position)  {  return position;  }  public override View GetView(int position, View convertView, ViewGroup parent)  {  var view = convertView;  if (view == null)  {  view = context.LayoutInflater.Inflate(Resource.Layout.MovieRow, parent, false);  }  var titleTextView = view.FindViewById<TextView>(Resource.Id.cardViewText);  var imageView = view.FindViewById<ImageView>(Resource.Id.imageView);  titleTextView.Text = movies[position].Title;  imageView.SetImageResource(Resource.Drawable.starwarslogo);  return view;  }  }  } |

Gist file link: <https://gist.github.com/vkoppaka/ad8a22f67f8268dab202>

Let’s run the updated sample and let’s take a look at how the app looks –



And voila, we now have a CardView layout showing up with Images and Text data.

That’s it for today, tomorrow, we will explore what RecyclerView in Xamarin.Android is.