Day 14: Spinner

Today we will take a look at Spinner control in Xamarin.Android. We will explore two different ways to create a spinner control in today’s blog post.

1. Simple Spinner with data coming from a hard-coded list of string values
2. A slightly more customized spinner where the data is coming for a repository (can be any repository)

# Simple Spinner with Array Adapter

Let’s get started with Simple Spinner. Open Main.axml Layout file in your Xamarin.Android project and let’s drag and drop a Spinner control on to the layout. If there is a button already on the layout, feel free to delete it. The Main.axml file would look like below –

|  |
| --- |
| ?xml version="1.0" encoding="utf-8"?>  <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  android:orientation="vertical"  android:layout\_width="fill\_parent"  android:layout\_height="fill\_parent">  <Spinner  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content"  android:id="@+id/moviesSpinner"  android:prompt="@string/movie\_prompt" />  </LinearLayout> |

Gist file link: <https://gist.github.com/vkoppaka/ec43c8260b993b930b02>

Next step is to define the data source that will be bound to the Spinner control. As I mentioned above, the data source for this simple spinner control is going to be a hard-code list of string values. For this purpose, we will utilize the **Strings.xml** file that we learned in the blog series earlier and we will define a string array of movies data (movies is what we will showing in the spinner). The array is defined using a **<string-array>** node. The code for the string array will look like the following –

|  |
| --- |
| <?xml version="1.0" encoding="utf-8"?>  <resources>  <string name="ApplicationName">SimpleSpinner</string>  <string name="movie\_prompt">Pick a movie</string>  <string-array name="movies\_array">  <item>A New Hope</item>  <item>Attack of the Clones</item>  <item>The Phantom Menace</item>  <item>Revenge of the Sith</item>  <item>Return of the Jedi</item>  <item>The Empre Strikes Back</item>  <item>The Force Awakens</item>  </string-array>  </resources> |

Gist file link: <https://gist.github.com/vkoppaka/ec1fc7f90a2c8b2090c7>

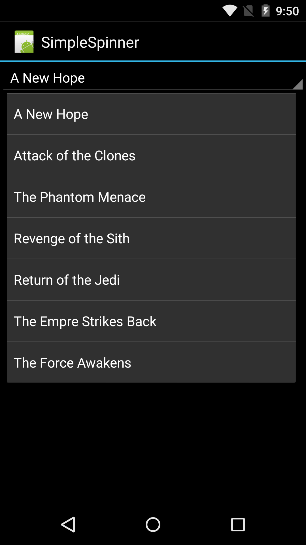
Now that we have both Layout and the Data needed to show in the spinner, the last task is to actually go ahead and bind the data to the spinner control. In our **MainActivity.cs**  we will Find the Spinner by Id using the FindViewById<T> method. We will then create an Array Adapter to host the movies data. This Array Adapter is created using the ArrayAdapter.CreateFromResource() method in the Android API. To this method, we will be passing the current context, the string array information and a simple spinner layout item that is built into Android. We will then set the Adapter’s DropDown View Resource to be SimpleSpinnerDropDownItem. Finally, we will assign this adapter to the Spinner’s Adapter method.

Let’s look how it all comes together –

|  |
| --- |
| using Android.App;  using Android.OS;  using Android.Widget;  namespace SimpleSpinner  {  [Activity(Label = "SimpleSpinner", MainLauncher = true, Icon = "@drawable/icon")]  public class MainActivity : Activity  {  protected override void OnCreate(Bundle bundle)  {  base.OnCreate(bundle);  // Set our view from the "main" layout resource  SetContentView(Resource.Layout.Main);  var moviesSpinner = FindViewById<Spinner>(Resource.Id.moviesSpinner);  var moviesAdapter = ArrayAdapter.CreateFromResource(this, Resource.Array.movies\_array, Android.Resource.Layout.SimpleSpinnerItem);  moviesAdapter.SetDropDownViewResource(Android.Resource.Layout.SimpleSpinnerDropDownItem);  moviesSpinner.Adapter = moviesAdapter;  }  }  } |

Gist file link: <https://gist.github.com/vkoppaka/c32af4fb0e0a9cf6ed54>

And when you run the application, here is how it’s going to look like.



# Spinner with Custom Adapter

The Simple Spinner we saw above is very powerful, but what if the data that we want to bind to the Spinner is coming from a remote Repository or a database? For this scenario, we will explore how to create a Spinner with a Custom Adapter. Open Main.axml Layout file in your Xamarin.Android project and let’s drag and drop a Spinner control on to the layout. If there is a button already on the layout, feel free to delete it. The Main.axml file would look like below –

|  |
| --- |
| <?xml version="1.0" encoding="utf-8"?>  <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  android:orientation="vertical"  android:layout\_width="fill\_parent"  android:layout\_height="fill\_parent">  <Spinner  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content"  android:id="@+id/moviesSpinner"  android:prompt="@string/movie\_prompt" />  </LinearLayout> |

Gist file link: <https://gist.github.com/vkoppaka/478b8124d59f6a710c17>

Since we will be binding data coming from a repository, we would need a POCO(Plain Old CLR Object) which defines the data structure of the data that gets returned. Let’s call that class as Movie –

|  |
| --- |
| using System;  namespace SpinnerWithCustomAdapter  {  public class Movie  {  public string Title { get; set; }  public string Episode { get; set; }  public string Director { get; set; }  public DateTime ReleaseDate { get; set; }  public override string ToString()  {  return Title + " by " + Director;  }  }  } |

Gist file link: <https://gist.github.com/vkoppaka/7b5bd3909772d571a296>

As a simple demonstration of a Repository, I have below, an in-memory repository. But don’t let this stop you from swapping it out to a remote HTTP repository or any other data source. Here is how my simple in-memory repository is going to look like –

|  |
| --- |
| using System;  using System.Collections.Generic;  namespace SpinnerWithCustomAdapter  {  public static class MoviesRepository  {  public static List<Movie> Movies { get; private set; }  static MoviesRepository()  {  Movies = new List<Movie>();  AddMovies();  }  private static void AddMovies()  {  Movies.Add(new Movie  {  Title = "A New Hope",  Director = "George Lucas",  Episode = "IV",  ReleaseDate = new DateTime(1977, 05, 25)  });  Movies.Add(new Movie  {  Title = "The Empire Strikes Back",  Director = "George Lucas",  Episode = "V",  ReleaseDate = new DateTime(1980, 05, 17)  });  Movies.Add(new Movie  {  Title = "Return of the Jedi",  Director = "George Lucas",  Episode = "VI",  ReleaseDate = new DateTime(1983, 05, 25)  });  Movies.Add(new Movie  {  Title = "The Phantom Menace",  Director = "George Lucas",  Episode = "I",  ReleaseDate = new DateTime(1999, 05, 19)  });  Movies.Add(new Movie  {  Title = "Revenge of the Sith",  Director = "George Lucas",  Episode = "III",  ReleaseDate = new DateTime(2005, 05, 19)  });  Movies.Add(new Movie  {  Title = "The Force Awakens",  Director = "J.J. Abrams",  Episode = "VII",  ReleaseDate = new DateTime(2015, 12, 11)  });  }  }  } |

Gist file link: <https://gist.github.com/vkoppaka/dd2e82ac73c61f69c196>

Finally, we will be creating an Adapter (don’t worry if Adapter feels a bit foreign, we will explore that in the coming days in more detail). For now, just know that we will be using an adapter to tell the spinner how to inflate given data into a View Cell. The code for that will look like –

|  |
| --- |
| using System.Collections.Generic;  using Android.App;  using Android.Views;  using Android.Widget;  namespace SpinnerWithCustomAdapter  {  public class MoviesAdapter : BaseAdapter<Movie>  {  private readonly Activity context;  private readonly List<Movie> movies;  public MoviesAdapter(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)  {  View view = convertView;  if (view == null) // otherwise create a new one  {  view = context.LayoutInflater.Inflate(Android.Resource.Layout.SimpleListItem1, null);  }  view.FindViewById<TextView>(Android.Resource.Id.Text1).Text = movies[position].ToString();  return view;  }  }  } |

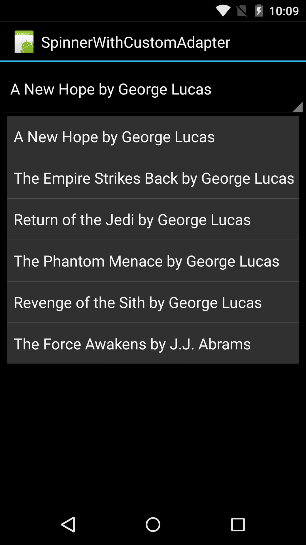
Gist file link: <https://gist.github.com/vkoppaka/ebd3ef117be319164595>

And finally, to tie it all together, our MainActivity.cs file where we set the Spinner’s Adapter property to the Movies Adapter we just created –

|  |
| --- |
| using Android.App;  using Android.OS;  using Android.Widget;  namespace SpinnerWithCustomAdapter  {  [Activity(Label = "SpinnerWithCustomAdapter", MainLauncher = true, Icon = "@drawable/icon")]  public class MainActivity : Activity  {    protected override void OnCreate(Bundle bundle)  {  base.OnCreate(bundle);  // Set our view from the "main" layout resource  SetContentView(Resource.Layout.Main);  var moviesSpinner = FindViewById<Spinner>(Resource.Id.moviesSpinner);  moviesSpinner.Adapter = new MoviesAdapter(this, MoviesRepository.Movies);  }  }  } |

Gist file link: <https://gist.github.com/vkoppaka/3b61291dde27bda7f228>

And if you run it and see, you should see all the movies from the repository being populated.



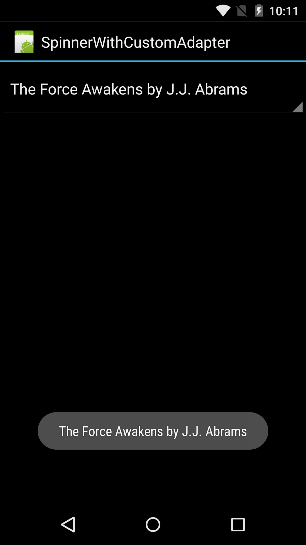
# Item Selection

Now, let’s explore how we can create an event to trigger when an item in the Spinner list is clicked. For this, we will subscribe to the ItemSelected Event and in the event handler, we will get the item that is currently selected using the spinner.GetItemAtPosition API and we will be passing e.Position to that method.

|  |
| --- |
| moviesSpinner.ItemSelected += moviesSpinner\_ItemSelected;  }  void moviesSpinner\_ItemSelected(object sender, AdapterView.ItemSelectedEventArgs e)  {  Spinner spinner = (Spinner)sender;  string toast = string.Format(spinner.GetItemAtPosition(e.Position).ToString());  Toast.MakeText(this, toast, ToastLength.Long).Show();  } |

Gist file link: <https://gist.github.com/vkoppaka/dedc09e9b0fd376120c2>

And voila, our toast with the currently selected Spinner item will show up as below.



So, that is it for today. Tomorrow, we will explore even more Controls in Xamarin.Android.

Venkata