Create a ClickHandler for RecyclerView items

November 17, 2017

Contents

1 Introduction

How to add a ClickHandler for your items in a RecyclerView Layout.

2 Edit your Adapter

1. Add an Interface.
2. Within that interface, define a void method that access a variable as a parameter.
3. Create a final private yourInterface attribute.
4. Add yourInterface as a parameter to the constructor and store it in the attribute created before.
5. Implement OnClickListener in your class AdapterViewHolder class.
6. Override onClick, passing what you want to yourInterface via its method.
7. Call setOnClickListener on the view passed into the constructor

```
1  private final ForecastAdapterOnClickHandler mClickHandler;
2
3  public interface ForecastAdapterOnClickHandler{
4     public void onClick(String s);
5  }
6
7  public ForecastAdapter(ForecastAdapterOnClickHandler mClickHandler) {
8     this.mClickHandler = mClickHandler;
9
10 }
11
12 /**
13 * Cache of the children views for a forecast list item.
14 */
15 public class ForecastAdapterViewHolder extends
      RecyclerView.ViewHolder implements View.OnClickListener{
```

```
16    public final TextView mWeatherTextView;
17
18    public ForecastAdapterViewHolder(View view) {
19       super(view);
20       mWeatherTextView = (TextView)
             view.findViewById(R.id.tv_weather_data);
21       view.setOnClickListener(this);
22
23    }
24
25    @Override
26    public void onClick(View v) {
27       int adapterPosition = getAdapterPosition();
28       String weatherForTodaz = mWeatherData[adapterPosition];
29       mClickHandler.onClick(weatherForTodaz);
30    }
31
32
33 }
```

3 Edit your Activity

1. Implement your interface from the MainActivity.
2. Override interface method.
3. Pass in 'this' as your Adapter constructor.

```
1  public class MainActivity extends AppCompatActivity implements
       ForecastAdapter.ForecastAdapterOnClickHandler{
2
3     private RecyclerView mRecyclerView;
4     private ForecastAdapter mForecastAdapter;
5
6     private TextView mErrorMessageDisplay;
```

```
7
8   private ProgressBar mLoadingIndicator;
9
10  @Override
11  protected void onCreate(Bundle savedInstanceState) {
12     super.onCreate(savedInstanceState);
13     setContentView(R.layout.activity_forecast);
14
15     /*
16     * Using findViewById, we get a reference to our RecyclerView
           from xml. This allows us to
17     * do things like set the adapter of the RecyclerView and
           toggle the visibility.
18     */
19     mRecyclerView = (RecyclerView)
           findViewById(R.id.recyclerview_forecast);
20
21     /* This TextView is used to display errors and will be hidden
           if there are no errors */
22     mErrorMessageDisplay = (TextView)
           findViewById(R.id.tv_error_message_display);
23
24     /*
25     * LinearLayoutManager can support HORIZONTAL or VERTICAL
           orientations. The reverse layout
26     * parameter is useful mostly for HORIZONTAL layouts that
           should reverse for right to left
27     * languages.
28     */
29     LinearLayoutManager layoutManager
30     = new LinearLayoutManager(this, LinearLayoutManager.VERTICAL,
           false);
31
32     mRecyclerView.setLayoutManager(layoutManager);
33
34     /*
```

```
35      * Use this setting to improve performance if you know that
            changes in content do not
36      * change the child layout size in the RecyclerView
37      */
38      mRecyclerView.setHasFixedSize(true);
39
40      /*
41      * The ForecastAdapter is responsible for linking our weather
            data with the Views that
42      * will end up displaying our weather data.
43      */
44      mForecastAdapter = new ForecastAdapter(this);
45
46      /* Setting the adapter attaches it to the RecyclerView in our
            layout. */
47      mRecyclerView.setAdapter(mForecastAdapter);
48
49      /*
50      * The ProgressBar that will indicate to the user that we are
            loading data. It will be
51      * hidden when no data is loading.
52      *
53      * Please note: This so called "ProgressBar" isn't a bar by
            default. It is more of a
54      * circle. We didn't make the rules (or the names of Views), we
            just follow them.
55      */
56      mLoadingIndicator = (ProgressBar)
            findViewById(R.id.pb_loading_indicator);
57
58      /* Once all of our views are setup, we can load the weather
            data. */
59      loadWeatherData();
60  }
61
62  /**
```

```
63    * This method will get the user's preferred location for weather,
          and then tell some
64    * background method to get the weather data in the background.
65    */
66   private void loadWeatherData() {
67       showWeatherDataView();
68
69       String location =
             SunshinePreferences.getPreferredWeatherLocation(this);
70       new FetchWeatherTask().execute(location);
71   }
72
73   @Override
74   public void onClick(String s) {
75       Toast.makeText(this,s,Toast.LENGTH_LONG).show();
76   }
```