Create a RecyclerView

November 16, 2017

Contents

1 Introduction

How to create a RecyclerView.

2 Add RecyclerView dependency

Add RecyclerView dependency to the app grandle:

```
compile 'com.android.support:recyclerview-v7:25.1.0'
```

3 Create a layout with RecyclerView

```
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="match_parent"
   android:layout_height="match_parent">

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

   <TextView
      android:id="@+id/tv_weather_data"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:padding="16dp"
      android:textSize="20sp" />

   <TextView
      android:id="@+id/tv_error_message_display"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:padding="16dp"
      android:text="@string/error_message"
      android:textSize="20sp"
      android:visibility="invisible" />

   <ProgressBar
```

```
27      android:id="@+id/pb_loading_indicator"
28      android:layout_height="42dp"
29      android:layout_width="42dp"
30      android:layout_gravity="center"
31      android:visibility="invisible" />
32
33  </FrameLayout>
```

4 Create item Layout

Create a generic Layout for items that will be storaged in the RecyclerView Example:

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3  android:layout_width="match_parent"
4  android:layout_height="wrap_content"
5  android:orientation="vertical">
6
7  <TextView
8  android:layout_width="wrap_content"
9  android:layout_height="wrap_content"
10 android:id="@+id/tv_weather_data"
11 android:textSize="22sp"
12 android:padding="16dp"/>
13
14 <View
15 android:layout_width="match_parent"
16 android:layout_height="1dp"
17 android:background="#dadada"
18 android:layout_marginLeft="8dp"
19 android:layout_marginRight="8dp">
20
21 </View>
22
23 </LinearLayout>
```

5 Create the adapter

1. Create a class for the adapter.
2. Extend it to extends RecyclerView.Adapter¡ForecastAdapter.ForecastAdapterViewHolder¿.
3. Create an array where you will storage the data for each item.
4. Create an empty constructor.
5. Create a class inside your class which extends RecyclerView.ViewHolder.
6. Inside your RecyclerView.ViewHolder class create the variables for your TextView or whatever you have in your RecyclerView Layout.
7. Inside your RecyclerView.ViewHolder class create a constructor with View as a parameter.
8. Inside your RecyclerView.ViewHolder class constructor get the references of your RecyclerView Layout TextViews.
9. Inside your adapter class override onCreateViewHolder and inflate your RecyclerView Layout into a view.
10. Return ForecastAdapterViewHolder with the inflated view.
11. Override onBindViewHolder and set the text of the TextView.
12. Override getItemCount and return 0 if the array which we created in the beginning is null or the length if not.
13. create a method to set data in your array.

```
1  /*
2  * Copyright (C) 2016 The Android Open Source Project
3  *
4  * Licensed under the Apache License, Version 2.0 (the "License");
5  * you may not use this file except in compliance with the License.
6  * You may obtain a copy of the License at
7  *
```

```
8  *     http://www.apache.org/licenses/LICENSE-2.0
9  *
10 * Unless required by applicable law or agreed to in writing, software
11 * distributed under the License is distributed on an "AS IS" BASIS,
12 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
       implied.
13 * See the License for the specific language governing permissions and
14 * limitations under the License.
15 */
16 package com.example.android.sunshine;
17
18 import android.content.Context;
19 import android.support.v7.widget.RecyclerView;
20 import android.view.LayoutInflater;
21 import android.view.View;
22 import android.view.ViewGroup;
23 import android.widget.TextView;
24
25 /**
26 * {@link ForecastAdapter} exposes a list of weather forecasts to a
27 * {@link android.support.v7.widget.RecyclerView}
28 */
29 public class ForecastAdapter extends
       RecyclerView.Adapter<ForecastAdapter.ForecastAdapterViewHolder> {
30
31    private String[] mWeatherData;
32
33    public ForecastAdapter() {
34
35    }
36
37    /**
38    * Cache of the children views for a forecast list item.
39    */
40    public class ForecastAdapterViewHolder extends
          RecyclerView.ViewHolder {
```

```
41
42      // Within ForecastAdapterViewHolder
            ////////////////////////////////////////////////////
43      public final TextView mWeatherTextView;
44
45      public ForecastAdapterViewHolder(View view) {
46          super(view);
47          mWeatherTextView = (TextView)
                view.findViewById(R.id.tv_weather_data);
48      }
49      // Within ForecastAdapterViewHolder
            ////////////////////////////////////////////////////
50  }
51
52  /**
53  * This gets called when each new ViewHolder is created. This
        happens when the RecyclerView
54  * is laid out. Enough ViewHolders will be created to fill the
        screen and allow for scrolling.
55  *
56  * @param viewGroup The ViewGroup that these ViewHolders are
        contained within.
57  * @param viewType If your RecyclerView has more than one type of
        item (which ours doesn't) you
58  *                 can use this viewType integer to provide a
        different layout. See
59  *                 {@link
        android.support.v7.widget.RecyclerView.Adapter#getItemViewType(int)}
60  *                 for more details.
61  * @return A new ForecastAdapterViewHolder that holds the View for
        each list item
62  */
63  @Override
64  public ForecastAdapterViewHolder onCreateViewHolder(ViewGroup
        viewGroup, int viewType) {
65      Context context = viewGroup.getContext();
```

```
66      int layoutIdForListItem = R.layout.forecast_list_item;
67      LayoutInflater inflater = LayoutInflater.from(context);
68      boolean shouldAttachToParentImmediately = false;
69
70      View view = inflater.inflate(layoutIdForListItem, viewGroup,
            shouldAttachToParentImmediately);
71      return new ForecastAdapterViewHolder(view);
72  }
73
74  /**
75  * OnBindViewHolder is called by the RecyclerView to display the
        data at the specified
76  * position. In this method, we update the contents of the
        ViewHolder to display the weather
77  * details for this particular position, using the "position"
        argument that is conveniently
78  * passed into us.
79  *
80  * @param forecastAdapterViewHolder The ViewHolder which should be
        updated to represent the
81  *                                  contents of the item at the given
        position in the data set.
82  * @param position                  The position of the item within
        the adapter's data set.
83  */
84  @Override
85  public void onBindViewHolder(ForecastAdapterViewHolder
        forecastAdapterViewHolder, int position) {
86      String weatherForThisDay = mWeatherData[position];
87      forecastAdapterViewHolder.mWeatherTextView.setText(weatherForThisDay);
88  }
89
90  /**
91  * This method simply returns the number of items to display. It
        is used behind the scenes
92  * to help layout our Views and for animations.
```

```
93      *
94      * @return The number of items available in our forecast
95      */
96     @Override
97     public int getItemCount() {
98        if (null == mWeatherData) return 0;
99        return mWeatherData.length;
100    }
101
102    /**
103    * This method is used to set the weather forecast on a
           ForecastAdapter if we've already
104    * created one. This is handy when we get new data from the web
           but don't want to create a
105    * new ForecastAdapter to display it.
106    *
107    * @param weatherData The new weather data to be displayed.
108    */
109    public void setWeatherData(String[] weatherData) {
110       mWeatherData = weatherData;
111       notifyDataSetChanged();
112    }
113 }
```

6 Edit your MainActivity

1. Add a RecyclerView variable.
2. Add your new class adapter variable.
3. Get a reference from your RecyclerView in the variable.
4. Create a LinearLayoutManager and set context, oritentation and shouldReverseLayout.
5. Set your LinearLayoutManager in your RecyclerView

6. set setHasFixedSize in your RecyclerView if you want to keep all the components with the same size.
7. Initialize your new class adapter variable.
8. Use setAdapter to set your adapter to you RecyclerView.
9. Set the data to your adapter.

Example:

```
1
2  /*
3  * Copyright (C) 2016 The Android Open Source Project
4  *
5  * Licensed under the Apache License, Version 2.0 (the "License");
6  * you may not use this file except in compliance with the License.
7  * You may obtain a copy of the License at
8  *
9  *      http://www.apache.org/licenses/LICENSE-2.0
10 *
11 * Unless required by applicable law or agreed to in writing, software
12 * distributed under the License is distributed on an "AS IS" BASIS,
13 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
       implied.
14 * See the License for the specific language governing permissions and
15 * limitations under the License.
16 */
17 package com.example.android.sunshine;
18
19 import android.os.AsyncTask;
20 import android.os.Bundle;
21 import android.support.v7.app.AppCompatActivity;
22 import android.support.v7.widget.LinearLayoutManager;
23 import android.support.v7.widget.RecyclerView;
24 import android.view.Menu;
25 import android.view.MenuInflater;
26 import android.view.MenuItem;
```

```
27 import android.view.View;
28 import android.widget.ProgressBar;
29 import android.widget.TextView;
30
31 import com.example.android.sunshine.data.SunshinePreferences;
32 import com.example.android.sunshine.utilities.NetworkUtils;
33 import com.example.android.sunshine.utilities.OpenWeatherJsonUtils;
34
35 import java.net.URL;
36
37 public class MainActivity extends AppCompatActivity {
38
39    private RecyclerView mRecyclerView;
40
41    private ForecastAdapter mForecastAdapter;
42
43    private TextView mErrorMessageDisplay;
44
45    private ProgressBar mLoadingIndicator;
46
47    @Override
48    protected void onCreate(Bundle savedInstanceState) {
49       super.onCreate(savedInstanceState);
50       setContentView(R.layout.activity_forecast);
51
52
53       /*
54       * Using findViewById, we get a reference to our RecyclerView
             from xml. This allows us to
55       * do things like set the adapter of the RecyclerView and
             toggle the visibility.
56       */
57       mRecyclerView = (RecyclerView)
             findViewById(R.id.recyclerview_forecast);
58
59       /* This TextView is used to display errors and will be hidden
```

```
        if there are no errors */
60      mErrorMessageDisplay = (TextView)
            findViewById(R.id.tv_error_message_display);
61
62      /*
63      * LinearLayoutManager can support HORIZONTAL or VERTICAL
            orientations. The reverse layout
64      * parameter is useful mostly for HORIZONTAL layouts that
            should reverse for right to left
65      * languages.
66      */
67      LinearLayoutManager layoutManager
68      = new LinearLayoutManager(this, LinearLayoutManager.VERTICAL,
            false);
69
70      mRecyclerView.setLayoutManager(layoutManager);
71
72      /*
73      * Use this setting to improve performance if you know that
            changes in content do not
74      * change the child layout size in the RecyclerView
75      */
76      mRecyclerView.setHasFixedSize(true);
77
78      /*
79      * The ForecastAdapter is responsible for linking our weather
            data with the Views that
80      * will end up displaying our weather data.
81      */
82      mForecastAdapter = new ForecastAdapter();
83
84      /* Setting the adapter attaches it to the RecyclerView in our
            layout. */
85      mRecyclerView.setAdapter(mForecastAdapter);
86
87      /*
```

```
88      * The ProgressBar that will indicate to the user that we are
            loading data. It will be
89      * hidden when no data is loading.
90      *
91      * Please note: This so called "ProgressBar" isn't a bar by
            default. It is more of a
92      * circle. We didn't make the rules (or the names of Views), we
            just follow them.
93      */
94      mLoadingIndicator = (ProgressBar)
            findViewById(R.id.pb_loading_indicator);
95
96      /* Once all of our views are setup, we can load the weather
            data. */
97      loadWeatherData();
98  }
99
100 /**
101 * This method will get the user's preferred location for weather,
        and then tell some
102 * background method to get the weather data in the background.
103 */
104 private void loadWeatherData() {
105     showWeatherDataView();
106
107     String location =
            SunshinePreferences.getPreferredWeatherLocation(this);
108     new FetchWeatherTask().execute(location);
109 }
110
111 /**
112 * This method will make the View for the weather data visible and
113 * hide the error message.
114 * <p>
115 * Since it is okay to redundantly set the visibility of a View,
        we don't
```

```
116     * need to check whether each view is currently visible or
            invisible.
117     */
118    private void showWeatherDataView() {
119       /* First, make sure the error is invisible */
120       mErrorMessageDisplay.setVisibility(View.INVISIBLE);
121
122       /* Then, make sure the weather data is visible */
123       mRecyclerView.setVisibility(View.VISIBLE);
124    }
125
126    /**
127    * This method will make the error message visible and hide the
            weather
128    * View.
129    * <p>
130    * Since it is okay to redundantly set the visibility of a View,
            we don't
131    * need to check whether each view is currently visible or
            invisible.
132    */
133    private void showErrorMessage() {
134       /* First, hide the currently visible data */
135       mRecyclerView.setVisibility(View.INVISIBLE);
136       /* Then, show the error */
137       mErrorMessageDisplay.setVisibility(View.VISIBLE);
138    }
139
140    public class FetchWeatherTask extends AsyncTask<String, Void,
            String[]> {
141
142       @Override
143       protected void onPreExecute() {
144          super.onPreExecute();
145          mLoadingIndicator.setVisibility(View.VISIBLE);
146       }
```

```
147
148     @Override
149     protected String[] doInBackground(String... params) {
150
151        /* If there's no zip code, there's nothing to look up. */
152        if (params.length == 0) {
153           return null;
154        }
155
156        String location = params[0];
157        URL weatherRequestUrl = NetworkUtils.buildUrl(location);
158
159        try {
160           String jsonWeatherResponse = NetworkUtils
161           .getResponseFromHttpUrl(weatherRequestUrl);
162
163           String[] simpleJsonWeatherData = OpenWeatherJsonUtils
164           .getSimpleWeatherStringsFromJson(MainActivity.this,
                  jsonWeatherResponse);
165
166           return simpleJsonWeatherData;
167
168        } catch (Exception e) {
169           e.printStackTrace();
170           return null;
171        }
172     }
173
174     @Override
175     protected void onPostExecute(String[] weatherData) {
176        mLoadingIndicator.setVisibility(View.INVISIBLE);
177        if (weatherData != null) {
178           showWeatherDataView();
179           // COMPLETED (45) Instead of iterating through every
                  string, use mForecastAdapter.setWeatherData and pass
                  in the weather data
```

```
180             mForecastAdapter.setWeatherData(weatherData);
181         } else {
182             showErrorMessage();
183         }
184     }
185 }
186
187 @Override
188 public boolean onCreateOptionsMenu(Menu menu) {
189     /* Use AppCompatActivity's method getMenuInflater to get a
            handle on the menu inflater */
190     MenuInflater inflater = getMenuInflater();
191     /* Use the inflater's inflate method to inflate our menu
            layout to this menu */
192     inflater.inflate(R.menu.forecast, menu);
193     /* Return true so that the menu is displayed in the Toolbar */
194     return true;
195 }
196
197 @Override
198 public boolean onOptionsItemSelected(MenuItem item) {
199     int id = item.getItemId();
200
201     if (id == R.id.action_refresh) {
202         mForecastAdapter.setWeatherData(null);
203         loadWeatherData();
204         return true;
205     }
206
207     return super.onOptionsItemSelected(item);
208 }
209 }
```

RecyclerView Resource