Empty View for Android's RecyclerView

27 November 2015 on Android

We are developing the next killer app and the pride of place in our main activity is the shiny new `RecyclerView`. But when the app is first installed and run, the user is confronted with a vast expanse of nothingness. That's because there is no data in our app yet and the `RecyclerView` has nothing to show.

This makes for a very poor user experience and could put off users from using our app. The ideal way to avoid showing an empty `RecyclerView` when there is no data is to show an alternate view in its place. This alternate view would probably contain a message which explains to the users that there is nothing to see at the moment and guides them on how they can add new content. What the alternate view shows is up to us and can be customized based on the needs of the app. For example, the alternate view could contain a screenshot of how the `RecyclerView` would look with data.

In Android's `ListView` terminology this alternate view is called an *empty view*. `ListView` has a `setEmptyView()` method that makes the task of showing an alternate view trivial.

`RecyclerView` is a fantastic replacement for `ListView`. It is powerful, flexible, efficient and all together a major enhancement over `ListView`. `RecyclerView` is based on a modular design and it delegates a lot of its responsibilities to various components. So many things that the `ListView` does on its own are now handled by different components in the `RecyclerView` world. Due to this modular design, `RecyclerView` does not have a `setEmptyView()` method or any other built-in way of showing an alternate view when there is no data.

In this post I will show how to add a `setEmptyView()` method to `RecyclerView` to achieve the same functionality as in `ListView`.

RecyclerView without an Empty View

The killer app we are building is a Todo List app *(Why not? There's no such thing as too many todo list apps. The more the merrier.)*

Our app has an activity called `TodoListActivity` that displays a list of todo items. This activity has a layout file *(activity_todo_list.xml)* that looks like this.

```
<?xml version="1.0" encoding="utf-8"?>

<android.support.v7.widget.RecyclerView
    xmlns:android="http://schemas.android.com/apk/res/androi
    android:id="@+id/todo_list_recycler_view"
    android:layout_width="match_parent"
    android:layout_height="match_parent" />
```

Our model is a class called `Todo`.

```
public class Todo {
    private String title;

    public String getTitle() {
        return title;
    }

    public void setTitle(String title) {
        this.title = title;
    }
}
```

In `TodoListActivity`'s `onCreate()` method, the `RecyclerView` will be fetched and initialized in the following manner.

```
public class TodoListActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_todo_list);

        RecyclerView recyclerView =
            (RecyclerView)findViewById(R.id.todo_list_recycl
        recyclerView.setLayoutManager(new LinearLayoutManage

        List<Todo> todos = // Fetch list of todos from the d
        TodoAdapter dataAdapter = new TodoAdapter(todos);
        recyclerView.setAdapter(adapter);
    }
```

```
    private class TodoHolder extends RecyclerView.ViewHolder
        private Todo todo;
        private TextView titleTextView;

        public TodoHolder(View itemView) {
            super(itemView);
            titleTextView = (TextView)itemView;
        }

        public void bindTodo(Todo todo) {
            todo = todo;
            titleTextView.setText(todo.getTitle());
        }
    }

    private class TodoAdapter extends RecyclerView.Adapter<T
        private List<Todo> todos;

        public TodoAdapter(List<Todo> todos) {
            this.todos = todos;
        }

        @Override
        public TodoHolder onCreateViewHolder(ViewGroup paren
                                             int viewType) {
            LayoutInflater layoutInflater =
                                    LayoutInflater.from(TodoList
            View view = layoutInflater.inflate(
                                    android.R.layout.simple_list_ite
                                    parent, false);
            return new TodoHolder(view);
        }

        @Override
        public void onBindViewHolder(TodoHolder holder, int
            Todo todo = todos.get(position);
            holder.bindTodo(todo);
        }

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

    // More activity code here...
}
```

This implementation works fine and displays a list of todos in the `RecyclerView` when the activity launches. But it suffers from the problem described at the beginning of this post i.e. it displays an empty view when there are no todo items.

RecyclerView with an Empty View

Instead of showing an empty `RecyclerView` we would like to show the user an alternate view that displays a message along the lines of *'There are no todo items yet. Start adding todo items by clicking on the **New Todo** button below.'*. The alternate view also has a ***New Todo*** button that takes the user to another activity where they can add a new todo item. After the user adds a todo item and comes back to the `TodoListActivity`, we would like to hide the alternate view and show the `RecyclerView`.

To provide empty view support for `RecyclerView` we will create a subclass that contains all the logic required to show or hide an empty view based on whether the adapter provided to the `RecyclerView` has data or not.

The `RecyclerView` subclass is shown below in its entirety.

```
package com.akbaribrahim;

import android.content.Context;
import android.support.v7.widget.RecyclerView;
import android.util.AttributeSet;
import android.view.View;

public class EmptyRecyclerView extends RecyclerView {

    private View emptyView;

    final private AdapterDataObserver observer = new Adapter
        @Override
        public void onChanged() {
            checkIfEmpty();
        }

        @Override
        public void onItemRangeInserted(int positionStart, i
            checkIfEmpty();
        }

        @Override
        public void onItemRangeRemoved(int positionStart, in
            checkIfEmpty();
        }
    };

    public EmptyRecyclerView(Context context) {
        super(context);
    }

    public EmptyRecyclerView(Context context, AttributeSet a
        super(context, attrs);
```

```
    }

    public EmptyRecyclerView(Context context, AttributeSet a
                             int defStyle) {
        super(context, attrs, defStyle);
    }

    void checkIfEmpty() {
        if (emptyView != null && getAdapter() != null) {
            final boolean emptyViewVisible =
                                    getAdapter().getItemCoun
            emptyView.setVisibility(emptyViewVisible ? VISIB
            setVisibility(emptyViewVisible ? GONE : VISIBLE)
        }
    }

    @Override
    public void setAdapter(Adapter adapter) {
        final Adapter oldAdapter = getAdapter();
        if (oldAdapter != null) {
            oldAdapter.unregisterAdapterDataObserver(observe
        }
        super.setAdapter(adapter);
        if (adapter != null) {
            adapter.registerAdapterDataObserver(observer);
        }

        checkIfEmpty();
    }

    public void setEmptyView(View emptyView) {
        this.emptyView = emptyView;
        checkIfEmpty();
    }
}
```

The `checkIfEmpty()` method in `EmptyRecyclerView` checks if both the empty view and adapter are not null. Then if the item count provided by the adapter is equal to zero the empty view is shown and the `EmptyRecyclerView` is hidden. If the item count provided by the adapter is not zero then the empty view is hidden and the `EmptyRecyclerView` is shown.

The `EmptyRecyclerView` overrides the `setAdapter()` method of its superclass and registers an `AdapterObserver` whenever an adapter is set. It also unregisters the observer whenever the adapter is changed or unset. The `AdapterObserver` calls `checkIfEmpty()` every time it observes an event that changes the content of the adapter. `checkIfEmpty()` is also called when the adapter or empty view are set.

To use this subclass of `RecyclerView`, we replace all occurrences of `RecyclerView` with `EmptyRecyclerView` in our layout and activity source code. We also create a view in the layout which will serve as the empty view. This empty view will be set as the empty view on the `EmptyRecyclerView`.

Here's the updated activity_todo_list.xml.

```
<?xml version="1.0" encoding="utf-8"?>

<!-- Added a LinearLayout to enclose the EmptyRecyclerView a
     the empty view -->
<LinearLayout xmlns:android="http://schemas.android.com/apk/
              android:orientation="vertical"
              android:layout_width="match_parent"
              android:layout_height="match_parent">

    <!-- Replaced android.support.v7.widget.RecyclerView wit
         new EmptyRecyclerView -->
    <com.akbaribrahim.EmptyRecyclerView
              android:id="@+id/todo_list_recycler_view"
              android:layout_width="match_parent"
              android:layout_height="match_parent" />

    <!-- Added an empty view which will be shown when the Em
         is empty -->
    <LinearLayout
        android:id="@+id/todo_list_empty_view"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical"
        android:textAlignment="center">

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:padding="16dp"
            android:text="@string/no_todos"/>

        <Button
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:padding="16dp"
            android:text="@string/new_todo"/>

    </LinearLayout>

</LinearLayout>
```

The changes to the `TodoListActivity` are shown below. The rest of the

activity including the `TodoHolder` and `TodoAdapter` remain unchanged.

```
public class TodoListActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_todo_list);

        // Replaced RecyclerView with EmptyRecyclerView
        EmptyRecyclerView recyclerView =
            (EmptyRecyclerView)findViewById(R.id.todo_list_r
        recyclerView.setLayoutManager(new LinearLayoutManage

        // Fetch the empty view from the layout and set it c
        // the new recycler view
        View emptyView = v.findViewById(R.id.todo_list_empty
        recyclerView.setEmptyView(emptyView);

        List<Todo> todos = // Fetch list of todos from the d
        TodoAdapter dataAdapter = new TodoAdapter(todos);
        recyclerView.setAdapter(adapter);
    }

    // Rest of the activity code...
}
```

That's it! We have added the missing empty view functionality to `RecyclerView` and our users will forever marvel at our wondrous new empty view whenever they launch the app with no todo items.

References

The source code for `EmptyRecyclerView` has been taken as is from an answer, by user Kernald on StackOverflow, to a question asking if there was an equivalent of `ListView.setEmptyView()` for `RecyclerView`. That answer in turn is based on this gist.

SEE ALL POSTS »