OnClickListener in Android RecyclerView: Example

November 24, 2020 Techenum

Recently, I have written a post about RecyclerView: **Android RecyclerView: How to Insert, Update and Delete Item**. It is rather long and includes the basics of RecyclerView. But, here I have tried to show you how can you use OnClickListener in Android RecyclerView.

Quick Navigation

How to add OnClickListener in Android RecyclerView ?

We are going to follow these steps to add a click listener from our Activity or Fragment to the `RecyclerView.Adapter`.

Step 1: Create an interface in the `Adapter` class. And,

Step 2: Create member variable `mMyClickListener` to hold the instance of interface implementation. Also,

Step 3: Implement the `MyClickListener` in our `Fragment` or `Activity` and pass it to `Adapter`'s constructor. And,

Step 4: Set click listener in `onCreateViewHolder()` invoking the method of our custom interface. Also,

Step 5: Pass in the required data `position` and `data item`.

Let us look at each of these steps one by one.

Creating an interface to handle clicks

Not sure what a callback is ? Read : **Interface in OOP: Guide to Polymorphism and Callbacks**.

We have to create an interface that we will be using. Choose a suitable and descriptive identifier (name) for the interface. You can create this interface in the `Adapter` itself or somewhere else.

I like to keep related things together. So, I will have the interface in `Adapter`.

```
/**
* This interface can be named anything you like.
*/
public interface MyClickListener {
    void onItemClick(int position);
}
```
Code language: Java (java)

Add the code above in your class that extends `RecyclerView.Adapter` and continue.

Passing the listener in constructor

Now that we have create an interface. We have to create an instance of the interface and pass it to the class that extends `RecyclerView.Adapter`.

Let us also create a variable and assign it using the constructor.

```
private final MyClickListener mMyClickListener;

public MyAdapter(MyClickListener myClickListener) {
    mMyClickListener = myClickListener;
}
```
Code language: Java (java)

Handling Click in RecyclerView.Adapter

Now that we have created everything in the adapter class. Let us modify the the `onCreateViewHolder()` to set click listener in our root view i.e. `itemView` (by default).

Modify the `onCreateViewHolder()` according to the code below.

```
public Vh onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
    // this is just an example to show the click listener only
    // everything else is just garbage code
    Vh viewHOlder = new Vh(null); // don't pass in null this is just for
demonstration
    viewHOlder.itemView.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            // get the position of this Vh
            int position = viewHOlder.getAdapterPosition();
            if (mMyClickListener != null)
mMyClickListener.onItemClick(position);
        }
    });
    return viewHOlder;
}
```
Code language: PHP (php)

The Final RecyclerView Adapter code

Here is the final code for our hypothetical RecyclerView. Just place follow the code below and use it in your project. Also copy pasting it will not work.

```
public class MyAdapter extends RecyclerView.Adapter<MyAdapter.Vh> {

    private final MyClickListener mMyClickListener;

    public MyAdapter(MyClickListener myClickListener) {
        mMyClickListener = myClickListener;
    }

    @NonNull
    @Override

    public Vh onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
        // this is just an example to show the click listener only
        // everything else is just garbage code
        Vh viewHOlder = new Vh(null); // don't pass in null this is just
for demonstration
        viewHOlder.itemView.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                // get the position of this Vh
                int position = viewHOlder.getAdapterPosition();
                if (mMyClickListener != null)
mMyClickListener.onItemClick(position);
            }
        });
        return viewHOlder;
    }

    @Override
    public void onBindViewHolder(@NonNull Vh holder, int position) {
        // never set click listener here for performance reason
    }

    @Override
    public int getItemCount() {
        return 0;
    }

    /**
     * Modify according to the project.
     *
     * @param position the data at this position.
     */
    public void getItemAt(int position) {
        // return here the value in the list
    }

    static class Vh extends RecyclerView.ViewHolder {
        public Vh(@NonNull View itemView) {
            super(itemView);
        }
    }

    public interface MyClickListener {
        void onItemClick(int position);
    }

}
```
Code language: Java (java)

Using the RecyclerView Adapter

This is the final step we need to do. We will have to create an implementation of `MyClickListener` where we have used the `RecyclerView`.

An example implementation will look something like this.

```
final MyClickListener mcl = new MyClickListener() {
    @Override
    public void onItemClick(int position) {
        // your code here when the user clicks the row
    }
};
MyAdapter myAdapter = new MyAdapter(mcl);
// recyclerView.setAdapter(myAdapter)
```

Code language: Java (java)

The above code is the sample usage which typically goes in the `Fragment` or `Activity` class.

Why setting listener in onBindViewHolder() is bad ?

If you are not familiar with the `RecyclerView` you might not know that. The system calls the `onBindViewHolder()` each time the user does some scrolling interaction.

It is not good practice to create a new instance of listener each time the user scrolls. Because we are not really achieving anything extra by doing this.

The device is only creating a new instance of the object each time with exact same details. And why should the device do repeated thing when it can be done once ?

If you have any queries or if you are stuck feel free to drop a comment below.