链滴

Android 教程 2020 - RecyclerView 响应点击

作者：RustFisher

原文链接：https://ld246.com/article/1580780003872

来源网站：链滴

本文介绍 RecyclerView 设置点击的方法。这里给出比较常见的使用方式。

Android教程2020 - 系列总览

本文链接

前面我们已经知道如何用RecyclerView显示一列数据。

用户点击某个 item 时，app 可以做出相应的反应。这里我们使用添加点击监听器的方式来实现这个果。

Android 开发中，监听器模式使用十分广泛。最先被开发者认识到的应该是 Button 的点击事件监听。

设计并添加监听器

首先设计监听器。OnItemClickListener1接口。实际开发中，接口的名字可以定义的更有含义一些。

```
public interface OnItemClickListener {

    void onItemClick(Character c);

    void onItemLongClick(Character c);

}
```

这个接口里我们放置了 2 个方法。分别用来响应点击与长按事件。

Adapter 持有监听器

首先修改一下 VH 类。我们希望整个 item 来接受点击。

```
    private class VH extends RecyclerView.ViewHolder {
        View item; // 我们希望拿到整个item的view
        TextView tv1;
        TextView tv2;

        public VH(@NonNull View itemView) {
            super(itemView);
            item = itemView;
            tv1 = itemView.findViewById(R.id.tv1);
            tv2 = itemView.findViewById(R.id.tv2);
        }
    }
```

接下来修改前面的 Adapter 类。在适配器中持有监听器对象onItemClickListener。

在onBindViewHolder方法中，给 item 设置监听。

```
holder.item.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        if (onItemClickListener != null) {
            onItemClickListener.onItemClick(c);
        }
    }
```

```
});
holder.item.setOnLongClickListener(new View.OnLongClickListener() {
    @Override
    public boolean onLongClick(View v) {
        if (onItemClickListener != null) {
            onItemClickListener.onItemLongClick(c);
        }
        return true;
    }
});
```

这里加上判空处理。防止空指针。

实际上，是 item 接受到了点击事件，再通过我们设计的监听器把事件传出去。

setOnLongClickListener这里返回 true。把这个 long click 事件消费掉。

此时的 LetterAdapter 完整代码如下。

```
    private class LetterAdapter extends RecyclerView.Adapter<VH> {

        private List<Character> dataList;
        private OnItemClickListener onItemClickListener;

        public LetterAdapter(List<Character> dataList) {
            this.dataList = dataList;
        }

        @NonNull
        @Override
        public VH onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
            return new VH(LayoutInflater.from(parent.getContext()).inflate(R.layout.item_letter, pa
ent, false));
        }

        @Override
        public void onBindViewHolder(@NonNull VH holder, int position) {
            final Character c = dataList.get(position);
            holder.tv1.setText(c.toString());
            holder.tv2.setText(String.valueOf(Integer.valueOf(c)));
            holder.item.setOnClickListener(new View.OnClickListener() {
                @Override
                public void onClick(View v) {
                    if (onItemClickListener != null) {
                        onItemClickListener.onItemClick(c);
                    }
                }
            });
            holder.item.setOnLongClickListener(new View.OnLongClickListener() {
                @Override
                public boolean onLongClick(View v) {
                    if (onItemClickListener != null) {
                        onItemClickListener.onItemLongClick(c);
                    }
                    return true;
```

```
            }
        });
    }

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


    public void setOnItemClickListener(OnItemClickListener onItemClickListener) {
        this.onItemClickListener = onItemClickListener;
    }

}
```

使用监听器

经过上面的努力，我们的 Adapter 有了监听器的功能。现在在 activity 中为列表设置监听。

```
    mLetterAdapter.setOnItemClickListener(new OnItemClickListener() {
        @Override
        public void onItemClick(Character c) {
            Toast.makeText(getApplicationContext(), "Click " + c, Toast.LENGTH_SHORT).show();
        }

        @Override
        public void onItemLongClick(Character c) {
            Toast.makeText(getApplicationContext(), "Long click " + c, Toast.LENGTH_LONG).show
);
        }
    });
```

这里可以看出，设置监听器的是 Adapter，而不是 recyclerView。

运行起来，点击列表看看效果。

设计监听器的另一个方案

上面我们通过接口（interface）来给设计了点击监听器。
我们可以试试不用接口，改用抽象类（abstract class）来设计监听器。

新建抽象类AbsOnItemClick.java。

```
public abstract class AbsOnItemClick {

    public abstract void onClick(char c);

    public void onLongClick(char c) {

    }
}
```

里面 1 个抽象方法，1 个普通 public 方法。

依葫芦画瓢，在LetterAdapter类中添加这个监听器。

```
private AbsOnItemClick absOnItemClick;

// onBindViewHolder方法中设置监听

    holder.item.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            if (absOnItemClick != null) {
                absOnItemClick.onClick(c);
            }
        }
    });
    holder.item.setOnLongClickListener(new View.OnLongClickListener() {
        @Override
        public boolean onLongClick(View v) {
            if (absOnItemClick != null) {
                absOnItemClick.onLongClick(c);
            }
            return true;
        }
    });

// setter方法 设置监听器
public void setAbsOnItemClick(AbsOnItemClick absOnItemClick) {
    this.absOnItemClick = absOnItemClick;
}
```

在 activity 中设置监听器。

```
mLetterAdapter.setAbsOnItemClick(new AbsOnItemClick() {
    @Override
    public void onClick(char c) {
        Log.d("rustApp", "[abs] onClick: " + c);
    }
});
```

运行起来看看效果。可以看到打出了 log。

```
rustApp: [abs] onClick: h
```

对比接口我们可体会到，抽象类有自己独特的地方。抽象类强制我们实现了它的抽象方法。而普通的
法是由我们自己选择是否重写。

在实际开发中，我们可以根据需要选择是用接口还是抽象类。

工程放这里： https://github.com/AnRFDev/Tutorial2020

相关阅读

RecyclerView - 使用入门

RecyclerView点击事件 - 如何设置点击事件

RecyclerView示例 - 实际使用

RecyclerView获取滑动距离

RecyclerView显示多种item

原文链接：Android 教程 2020 - RecyclerView 响应点击