

RecyclerView



_iamjerry

关注



5 2019.03.30 22:18:13 字数 867 阅读 164,594

RecyclerView是Android一个更强大的控件,其不仅可以实现和ListView同样的效果,还有优化了ListView中的各种不足。其可以实现数据纵向滚动,也可以实现横向滚动(ListView做不到横向滚动)。接下来讲解RecyclerView的用法。

RecyclerView 基本用法

因为 **RecyclerView** 属于新增的控件,Android将RecyclerView定义在support库里。若要使用RecyclerView,第一步是要在 **build.gradle** 中添加对应的依赖库。

添加RecyclerView 依赖库

在 **app/build.gradle** 中的 **dependencies**闭包 添加以下内容:

```
1 | implementation 'com.android.support:recyclerview-v7:27.1.1'
```

然后点击顶部的Sync Now进行同步

修改 activity_main.xml

```
1 | <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
2 |     xmlns:app="http://schemas.android.com/apk/res-auto"
3 |     xmlns:tools="http://schemas.android.com/tools"
4 |     android:layout_width="match_parent"
5 |     android:layout_height="match_parent"
6 |     >
7 |
8 |     <android.support.v7.widget.RecyclerView
9 |         android:id="@+id/recycler_view"
10 |         android:layout_width="match_parent"
11 |         android:layout_height="match_parent"
12 |     />
13 | </LinearLayout>
```

由于 **RecyclerView** 不是内置在系统SDK中,需要把其完整的包名路径写出来

新建 Fruit.java

```
1 | public class Fruit {
2 |
3 |     private String name;
4 |     private int imageId;
5 |
6 |     public Fruit(String name, int imageId){
7 |         this.name = name;
8 |         this.imageId = imageId;
9 |
10 |    }
11 |
12 |    public String getName() {
13 |        return name;
14 |    }
15 |
16 |    public int getImageId() {
17 |        return imageId;
18 |    }
19 | }
```

新建 fruit_item.xml

创建ImageView来显示水果图片,TextView来显示水果名字。

```

1  <LinearLayout
2      xmlns:android="http://schemas.android.com/apk/res/android"
3      android:layout_width="match_parent"
4      android:layout_height="wrap_content"
5
6      >
7      <ImageView
8          android:layout_width="wrap_content"
9          android:layout_height="wrap_content"
10         android:id="@+id/fruit_image"/>
11
12         <TextView
13             android:layout_width="wrap_content"
14             android:layout_height="wrap_content"
15             android:id="@+id/fruitname"
16             android:layout_gravity="center_vertical"
17             android:layout_marginLeft="10dp"/>
18
19  </LinearLayout>

```

新增适配器 FruitAdapter

为 RecyclerView 新增适配器 FruitAdapter ,并让其继承于 RecyclerView.Adapter ,把泛型指定为 FruitAdapter.ViewHolder 。

```

1  public class FruitAdapter extends RecyclerView.Adapter<FruitAdapter.ViewHolder> {
2
3      private List<Fruit> mFruitList;
4      static class ViewHolder extends RecyclerView.ViewHolder{
5          ImageView fruitImage;
6          TextView fruitName;
7
8          public ViewHolder (View view)
9          {
10             super(view);
11             fruitImage = (ImageView) view.findViewById(R.id.fruit_image);
12             fruitName = (TextView) view.findViewById(R.id.fruitname);
13         }
14     }
15
16     public FruitAdapter (List <Fruit> fruitList){
17         mFruitList = fruitList;
18     }
19
20     @Override
21
22     public ViewHolder onCreateViewHolder(ViewGroup parent, int viewType){
23         View view = LayoutInflater.from(parent.getContext()).inflate(R.layout.fruit_it
24         ViewHolder holder = new ViewHolder(view);
25         return holder;
26     }
27
28     @Override
29     public void onBindViewHolder(ViewHolder holder, int position){
30
31         Fruit fruit = mFruitList.get(position);
32         holder.fruitImage.setImageResource(fruit.getImageId());
33         holder.fruitName.setText(fruit.getName());
34     }
35
36     @Override
37     public int getItemCount(){
38         return mFruitList.size();
39     }
40 }

```

- 定义内部类 ViewHolder ,并继承 RecyclerView.ViewHolder 。传入的View参数通常是 RecyclerView子项的最外层布局。

- FruitAdapter构造函数,用于把要展示的数据源传入,并赋予值给全局变量mFruitList。
- FruitAdapter继承RecyclerView.Adapter。因为必须重写 `onCreateViewHolder()` , `onBindViewHolder()` 和 `getItemCount()` 三个方法
 - `onCreateViewHolder()` 用于创建ViewHolder实例,并把加载的布局传入到构造函数去,再把ViewHolder实例返回。
 - `onBindViewHolder()` 则是用于对子项的数据进行赋值,会在每个子项被滚动到屏幕内时执行。 `position` 得到当前项的Fruit实例。
 - `getItemCount()` 返回RecyclerView的子项数目。

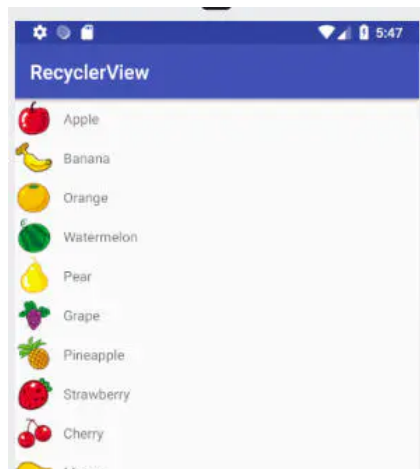
修改 MainActivity.java

```

1 public class MainActivity extends AppCompatActivity {
2
3     private List<Fruit> fruitList = new ArrayList<>();
4
5     @Override
6     protected void onCreate(Bundle savedInstanceState) {
7         super.onCreate(savedInstanceState);
8         setContentView(R.layout.activity_main);
9         initFruits();
10        RecyclerView recyclerView = (RecyclerView) findViewById(R.id.recycler_view);
11        LinearLayoutManager layoutManager = new LinearLayoutManager(this);
12        recyclerView.setLayoutManager(layoutManager);
13        FruitAdapter adapter = new FruitAdapter(fruitList);
14        recyclerView.setAdapter(adapter);
15    }
16
17    private void initFruits() {
18        for (int i = 0; i < 2; i++) {
19            Fruit apple = new Fruit("Apple", R.drawable.apple_pic);
20            fruitList.add(apple);
21            Fruit banana = new Fruit("Banana", R.drawable.banana_pic);
22            fruitList.add(banana);
23            Fruit orange = new Fruit("Orange", R.drawable.orange_pic);
24            fruitList.add(orange);
25            Fruit watermelon = new Fruit("Watermelon", R.drawable.watermelon_pic);
26            fruitList.add(watermelon);
27            Fruit pear = new Fruit("Pear", R.drawable.pear_pic);
28            fruitList.add(pear);
29            Fruit grape = new Fruit("Grape", R.drawable.grape_pic);
30            fruitList.add(grape);
31            Fruit pineapple = new Fruit("Pineapple", R.drawable.pineapple_pic);
32            fruitList.add(pineapple);
33            Fruit strawberry = new Fruit("Strawberry", R.drawable.strawberry_pic);
34            fruitList.add(strawberry);
35            Fruit cherry = new Fruit("Cherry", R.drawable.cherry_pic);
36            fruitList.add(cherry);
37            Fruit mango = new Fruit("Mango", R.drawable.mango_pic);
38            fruitList.add(mango);
39        }
40    }
41 }
42 }
```

`LayoutManager` 用于指定RecyclerView的布局方式。 `LinearLayoutManager` 指的是线性布局。

运行效果：



image

修改RecyclerView 显示效果

横向滚动

修改 fruit_item.xml

```
1 <LinearLayout
2     xmlns:android="http://schemas.android.com/apk/res/android"
3     android:layout_width="100dp"
4     android:layout_height="wrap_content"
5     android:orientation="vertical"
6
7     >
8     <ImageView
9         android:layout_width="wrap_content"
10        android:layout_height="wrap_content"
11        android:id="@+id/fruit_image"
12        android:layout_gravity="center_horizontal"/>
13
14    <TextView
15        android:layout_width="wrap_content"
16        android:layout_height="wrap_content"
17        android:id="@+id/fruitname"
18        android:layout_gravity="center_horizontal"
19        android:layout_marginTop="10dp"/>
20 </LinearLayout>
```

把LinearLayout改成垂直排列,因为水果名字长度不一样,把宽度改为100dp。

ImageView和TextView都改为水平居中

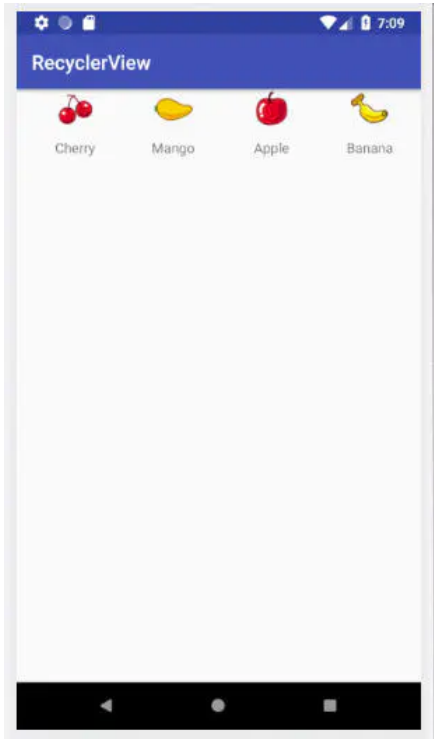
修改MainActivity.java

```
1 @Override
2     protected void onCreate(Bundle savedInstanceState) {
3         super.onCreate(savedInstanceState);
4         setContentView(R.layout.activity_main);
5         initFruits();
6         RecyclerView recyclerView = (RecyclerView) findViewById(R.id.recycler_view);
7         LinearLayoutManager layoutManager = new LinearLayoutManager(this);
8         layoutManager.setOrientation(LinearLayoutManager.HORIZONTAL);
9         recyclerView.setLayoutManager(layoutManager);
10        FruitAdapter adapter = new FruitAdapter(fruitList);
11        recyclerView.setAdapter(adapter);
12    }
```

通过调用 `setOrientation()` 把布局的排列方向改为水平排列。

得益于RecyclerView的设计,我们可以通过LayoutManager实现各种不同的排列方式的布局。

运行结果:



image

除了 `LinearLayoutManager` , `RecyclerView` 还提供了 `GridLayoutManager`(网格布局) 和 `StaggeredGridLayoutManager`(瀑布流布局)

GridLayoutManager

`GridLayoutManager`(网格布局)

修改MainActivity.java

修改 `MainActivity.java` ,把

```
1 | LinearLayoutManager layoutManager = new LinearLayoutManager(this);
2 | layoutManager.setOrientation(LinearLayoutManager.HORIZONTAL);
```

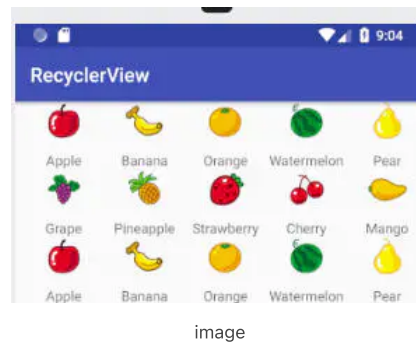
换成

```
1 | GridLayoutManager layoutManager = new GridLayoutManager(this,5);
```

`GridLayoutManager (Context context, int spanCount)`

- Context: Current context, will be used to access resources.
- spanCount int: The number of columns in the grid(网格的列数)

运行结果:



StaggeredGridLayoutManager

StaggeredGridLayoutManager(瀑布流布局)

修改fruit_item.xml

```

1  <?xml version="1.0" encoding="utf-8"?>
2  <LinearLayout
3      xmlns:android="http://schemas.android.com/apk/res/android"
4      android:layout_width="match_parent"
5      android:layout_height="wrap_content"
6      android:layout_margin="5dp"
7
8      >
9      <ImageView
10         android:layout_width="wrap_content"
11         android:layout_height="wrap_content"
12         android:id="@+id/fruit_image"
13         android:layout_gravity="center_horizontal"/>
14
15         <TextView
16             android:layout_width="wrap_content"
17             android:layout_height="wrap_content"
18             android:id="@+id/fruitname"
19             android:layout_gravity="left"
20             android:layout_marginTop="10dp"/>
21
22  </LinearLayout>

```

把LinearLayout的宽度设为 `match_parent` 是因为瀑布流的宽度是 根据布局的列数来自动适配的, 而不是固定值。(GridLayoutManager也是 根据布局的列数来自动适配的)

修改 MainActivity.java

```

1  public class MainActivity extends AppCompatActivity {
2
3      private List<Fruit> fruitList = new ArrayList<>();
4
5      @Override
6      protected void onCreate(Bundle savedInstanceState) {
7          super.onCreate(savedInstanceState);
8          setContentView(R.layout.activity_main);
9          initFruits();
10         RecyclerView recyclerView = (RecyclerView) findViewById(R.id.recycler_view);
11         StaggeredGridLayoutManager layoutManager = new StaggeredGridLayoutManager(3, StaggeredGridLayoutManager.VERTICAL);
12         recyclerView.setLayoutManager(layoutManager);
13         FruitAdapter adapter = new FruitAdapter(fruitList);
14         recyclerView.setAdapter(adapter);
15     }
16     private void initFruits() {
17         for (int i = 0; i < 20; i++) {
18             Fruit apple = new Fruit(getRandomLengthName("Apple"), R.drawable.apple_pic);
19             fruitList.add(apple);
20             Fruit banana = new Fruit(getRandomLengthName("Banana"), R.drawable.banana_pic);
21             fruitList.add(banana);
22             Fruit orange = new Fruit(getRandomLengthName("Orange"), R.drawable.orange_pic);
23             fruitList.add(orange);
24             Fruit watermelon = new Fruit(getRandomLengthName("Watermelon"), R.drawable.watermelon_pic);
25             fruitList.add(watermelon);
26             Fruit pear = new Fruit(getRandomLengthName("Pear"), R.drawable.pear_pic);

```

```

27      fruitList.add(pear);
28      Fruit grape = new Fruit(getRandomLengthName("Grape"), R.drawable.grape_pic);
29      fruitList.add(grape);
30      Fruit pineapple = new Fruit(getRandomLengthName("Pineapple"), R.drawable.pineapple);
31      fruitList.add(pineapple);
32      Fruit strawberry = new Fruit(getRandomLengthName("Strawberry"), R.drawable.strawberry);
33      fruitList.add(strawberry);
34      Fruit cherry = new Fruit(getRandomLengthName("Cherry"), R.drawable.cherry_pic);
35      fruitList.add(cherry);
36      Fruit mango = new Fruit(getRandomLengthName("Mango"), R.drawable.mango_pic);
37      fruitList.add(mango);
38  }
39  }
40  private String getRandomLengthName(String name){
41      Random random = new Random();
42      int length= random.nextInt(20)+1; // 产生1-20的随机数
43      StringBuilder builder = new StringBuilder();
44      for (int i =0;i<length;i++){
45          builder.append(name);
46      }
47      return builder.toString();
48  }
49  }

```

```
StaggeredGridLayoutManager layoutManager = new
```

```
StaggeredGridLayoutManager(3,StaggeredGridLayoutManager.VERTICAL);
```

StaggeredGridLayoutManager传入2个参数,第一个是布局的列数,第二个是布局的排列方向。

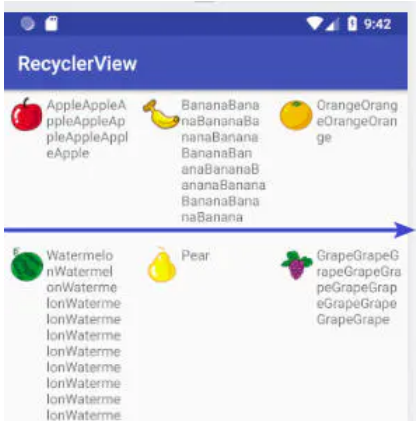
```
random.nextInt(20)+1 产生1-20的随机数
```

运行效果:

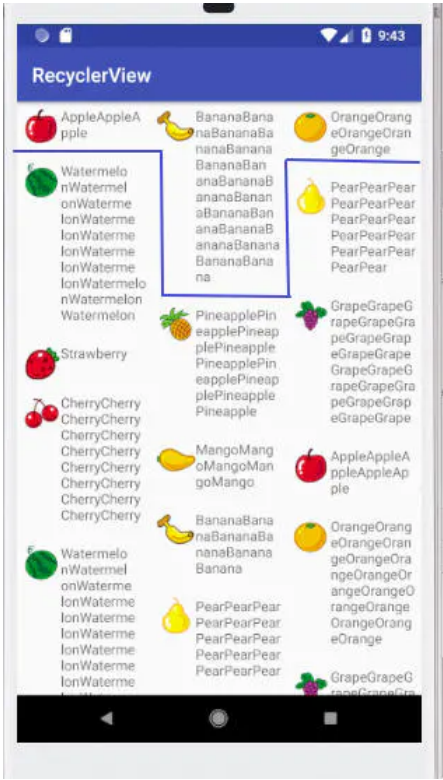


image

GridLayoutManager和StaggeredGridLayout的区别



image



image

上图是GridLayoutManager,下图是StaggeredGridLayout。
当从显示效果来看,已经一目了然。
GridLayoutManager是会固定高度的,所以会留下很多空白区域。
相反,StaggeredGridLayout并不会固定高度,以至于就算子项的高度不一致,下一行的会自动靠拢上一行。

RecyclerView 的点击事件

修改 FruitAdapter.java

```
1 public class FruitAdapter extends RecyclerView.Adapter<FruitAdapter.ViewHolder> {
2
3     private List<Fruit> mFruitList;
4     static class ViewHolder extends RecyclerView.ViewHolder{
5         View fruitView;
6         ImageView fruitImage;
7         TextView fruitName;
8
9         public ViewHolder (View view)
10        {
11            super(view);
```



```

12         fruitView = view;
13         fruitImage = (ImageView) view.findViewById(R.id.fruit_image);
14         fruitName = (TextView) view.findViewById(R.id.fruitname);
15     }
16
17 }
18
19 public FruitAdapter (List <Fruit> fruitList){
20     mFruitList = fruitList;
21 }
22
23 @Override
24
25 public ViewHolder onCreateViewHolder(ViewGroup parent, int viewType){
26     View view = LayoutInflater.from(parent.getContext()).inflate(R.layout.fruit_item, parent, false);
27     final ViewHolder holder = new ViewHolder(view);
28     holder.fruitView.setOnClickListener(new View.OnClickListener() {
29         @Override
30         public void onClick(View view) {
31             int position = holder.getAdapterPosition();
32             Fruit fruit = mFruitList.get(position);
33             Toast.makeText(view.getContext(), "you clicked view" + fruit.getName(), Toast.LENGTH_SHORT).show();
34         }
35     });
36
37     holder.fruitImage.setOnClickListener(new View.OnClickListener() {
38         @Override
39         public void onClick(View view) {
40             int position = holder.getAdapterPosition();
41             Fruit fruit = mFruitList.get(position);
42             Toast.makeText(view.getContext(), "you clicked image" + fruit.getName(), Toast.LENGTH_SHORT).show();
43         }
44     });
45     return holder;
46 }
47
48 ...
49 }

```

修改ViewHolder,添加fruitView变量来保存子项最外层布局的实例。

运行效果：



image

