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:

```
    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

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    >

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

```
public class Fruit {

    private String name;
    private int imageId;

    public Fruit(String name, int imageId){
        this.name = name;
        this.imageId = imageId;

    }

    public String getName() {
        return name;
    }

    public int getImageId() {
        return imageId;
    }
}
```

新建 fruit_item.xml

创建ImageView来显示水果图片,TextView来显示水果名字。

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

    >
    <ImageView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/fruit_image"/>

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/fruitname"
        android:layout_gravity="center_vertical"
        android:layout_marginLeft="10dp"/>

</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` 。

```
public class FruitAdapter extends RecyclerView.Adapter<FruitAdapter.ViewHolder> {

    private  List<Fruit> mFruitList;
    static class ViewHolder extends RecyclerView.ViewHolder{
        ImageView fruitImage;
        TextView fruitName;

        public ViewHolder (View view)
        {
            super(view);
            fruitImage = (ImageView) view.findViewById(R.id.fruit_image);
            fruitName = (TextView) view.findViewById(R.id.fruitname);
        }

    }

    public  FruitAdapter (List <Fruit> fruitList){
        mFruitList = fruitList;
    }

    @Override

    public ViewHolder onCreateViewHolder(ViewGroup parent, int viewType){
        View view = LayoutInflater.from(parent.getContext()).inflate(R.layout.fruit_it
        ViewHolder holder = new ViewHolder(view);
        return holder;
    }

    @Override
    public void onBindViewHolder(ViewHolder holder, int position){

        Fruit fruit = mFruitList.get(position);
        holder.fruitImage.setImageResource(fruit.getImageId());
        holder.fruitName.setText(fruit.getName());
    }

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

- 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。

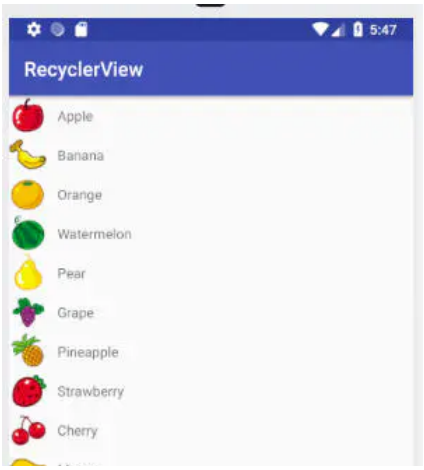
- FruitAdapter构造函数,用于把要展示的数据源传入,并赋予值给全局变量mFruitList。
- FruitAdapter继承RecyclerView.Adapter。因为必须重写 `onCreateViewHolder()` , `onBindViewHolder()` 和 `getItemCount()` 三个方法
  - `onCreateViewHolder()` 用于创建ViewHolder实例,并把加载的布局传入到构造函数去,再把ViewHolder实例返回。
  - `onBindViewHolder()` 则是用于对子项的数据进行赋值,会在每个子项被滚动到屏幕内时执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

修改 MainActivity.java

```
public class MainActivity extends AppCompatActivity {

    private List<Fruit> fruitList = new ArrayList<>();

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        initFruits();
        RecyclerView recyclerView = (RecyclerView) findViewById(R.id.recycler_view);
        LinearLayoutManager layoutManager = new LinearLayoutManager(this);
        recyclerView.setLayoutManager(layoutManager);
        FruitAdapter adapter = new FruitAdapter(fruitList);
        recyclerView.setAdapter(adapter);
    }

    private void initFruits() {
        for (int i = 0; i < 2; i++) {
            Fruit apple = new Fruit("Apple", R.drawable.apple_pic);
            fruitList.add(apple);
            Fruit banana = new Fruit("Banana", R.drawable.banana_pic);
            fruitList.add(banana);
            Fruit orange = new Fruit("Orange", R.drawable.orange_pic);
            fruitList.add(orange);
            Fruit watermelon = new Fruit("Watermelon", R.drawable.watermelon_pic);
            fruitList.add(watermelon);
            Fruit pear = new Fruit("Pear", R.drawable.pear_pic);
            fruitList.add(pear);
            Fruit grape = new Fruit("Grape", R.drawable.grape_pic);
            fruitList.add(grape);
            Fruit pineapple = new Fruit("Pineapple", R.drawable.pineapple_pic);
            fruitList.add(pineapple);
            Fruit strawberry = new Fruit("Strawberry", R.drawable.strawberry_pic);
            fruitList.add(strawberry);
            Fruit cherry = new Fruit("Cherry", R.drawable.cherry_pic);
            fruitList.add(cherry);
            Fruit mango = new Fruit("Mango", R.drawable.mango_pic);
            fruitList.add(mango);

        }
    }
}
```

`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

```
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="100dp"
    android:layout_height="wrap_content"
    android:orientation="vertical"

    >
    <ImageView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/fruit_image"
        android:layout_gravity="center_horizontal"/>

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/fruitname"
        android:layout_gravity="center_horizontal"
        android:layout_marginTop="10dp"/>
</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

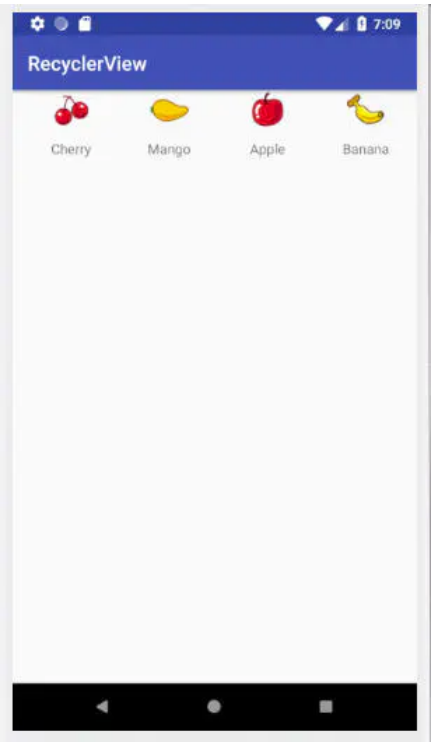
修改MainActivity.java

```
@Override
   protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        initFruits();
        RecyclerView recyclerView = (RecyclerView) findViewById(R.id.recycler_view);
        LinearLayoutManager layoutManager = new LinearLayoutManager(this);
        layoutManager.setOrientation(LinearLayoutManager.HORIZONTAL);
        recyclerView.setLayoutManager(layoutManager);
        FruitAdapter adapter = new FruitAdapter(fruitList);
        recyclerView.setAdapter(adapter);
    }
```

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

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

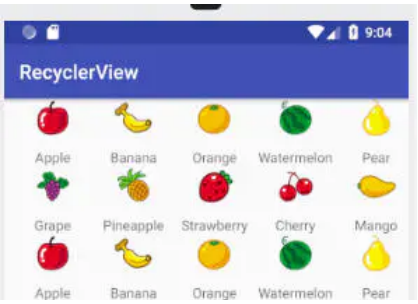
换成

```
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:



image

StaggeredGridLayoutManager

StaggeredGridLayoutManager(瀑布流布局)

修改fruit_item.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_margin="5dp"

    >
    <ImageView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/fruit_image"
        android:layout_gravity="center_horizontal"/>

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/fruitname"
        android:layout_gravity="left"
        android:layout_marginTop="10dp"/>

</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

```
public class MainActivity extends AppCompatActivity {

    private List<Fruit> fruitList = new ArrayList<>();

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        initFruits();
        RecyclerView recyclerView = (RecyclerView) findViewById(R.id.recycler_view);
        StaggeredGridLayoutManager layoutManager = new StaggeredGridLayoutManager(3,St
        recyclerView.setLayoutManager(layoutManager);
        FruitAdapter adapter = new FruitAdapter(fruitList);
        recyclerView.setAdapter(adapter);
    }
    private void initFruits() {
        for (int i = 0; i < 2; i++) {
            Fruit apple = new Fruit(getRandomLengthName("Apple"), R.drawable.apple_pic
            fruitList.add(apple);
            Fruit banana = new Fruit(getRandomLengthName("Banana"), R.drawable.banana_
            fruitList.add(banana);
            Fruit orange = new Fruit(getRandomLengthName("Orange"), R.drawable.orange_
            fruitList.add(orange);
            Fruit watermelon = new Fruit(getRandomLengthName("Watermelon"), R.drawable
            fruitList.add(watermelon);
            Fruit pear = new Fruit(getRandomLengthName("Pear"), R.drawable.pear_pic);
```

```
            fruitList.add(pear);
            Fruit grape = new Fruit(getRandomLengthName("Grape"), R.drawable.grape_pic
            fruitList.add(grape);
            Fruit pineapple = new Fruit(getRandomLengthName("Pineapple"), R.drawable.p
            fruitList.add(pineapple);
            Fruit strawberry = new Fruit(getRandomLengthName("Strawberry"), R.drawable
            fruitList.add(strawberry);
            Fruit cherry = new Fruit(getRandomLengthName("Cherry"), R.drawable.cherry_
            fruitList.add(cherry);
            Fruit mango = new Fruit(getRandomLengthName("Mango"), R.drawable.mango_pic
            fruitList.add(mango);
        }
    }
    private String getRandomLengthName(String name){
        Random random = new Random();
        int length= random.nextInt(20)+1;  // 产生1-20的随机数
        StringBuilder builder = new StringBuilder();
        for (int i =0;i<length;i++){
            builder.append(name);
        }
        return  builder.function function toString() { [native code] }() { [native cod
    }
}
```

```
StaggeredGridLayoutManager layoutManager = new
StaggeredGridLayoutManager(3,StaggeredGridLayoutManager.VERTICAL);
```

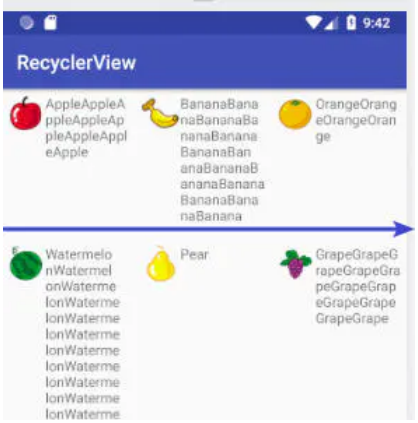
StaggeredGridLayoutManager传入2个参数,第一个是布局的列数,第二个是布局的排列方向。

`random.nextInt(20)+1` 产生1-20的随机数

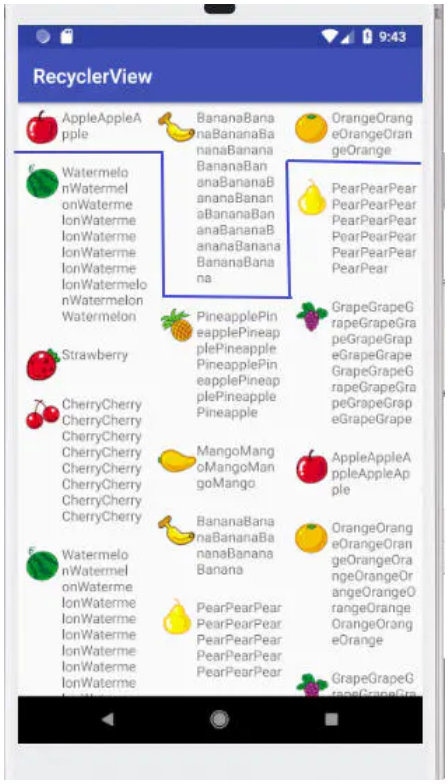
运行效果:

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

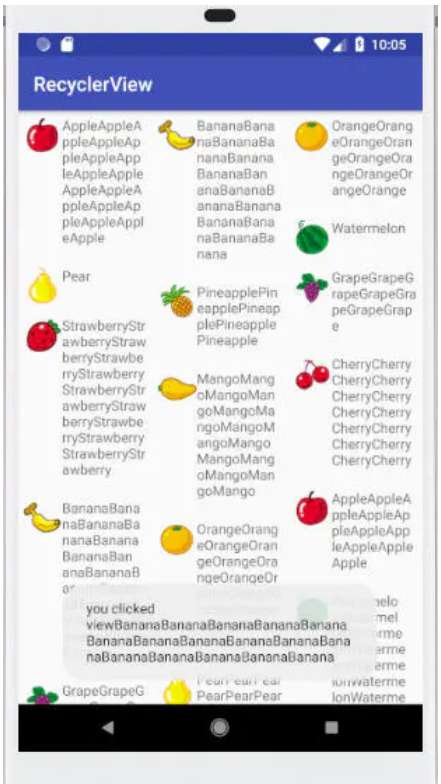
修改 FruitAdapter.java

```
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);
```

```
            fruitView = view;
            fruitImage = (ImageView) view.findViewById(R.id.fruit_image);
            fruitName = (TextView) view.findViewById(R.id.fruitname);
        }

    }

    public  FruitAdapter (List <Fruit> fruitList){
        mFruitList = fruitList;
    }

    @Override

    public ViewHolder onCreateViewHolder(ViewGroup parent, int viewType){
        View view = LayoutInflater.from(parent.getContext()).inflate(R.layout.fruit_it
        final ViewHolder holder = new ViewHolder(view);
        holder.fruitView.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                int position = holder.getAdapterPosition();
                Fruit fruit = mFruitList.get(position);
                Toast.makeText(view.getContext(), "you clicked view" + fruit.getName()
            }
        });

        holder.fruitImage.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                int position = holder.getAdapterPosition();
                Fruit fruit = mFruitList.get(position);
                Toast.makeText(view.getContext(), "you clicked image" + fruit.getName(
            }
        });
        return holder;
    }

  ...
}
```

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