项目七 电器清单

项目介绍

本项目与项目六家电列表作用类似，都是用来将各类家用电器的操作入口汇集在一个Android界面上，方便用户快速选择。两个项目的不同之处为：项目六使用了列表形式呈现家电清单，而本项目则使用网格形式设计界面。本项目的任务实践也分为两个目标：一个是先完成一个仅显示家电品类的简单网格界面，如图7-1(a)所示；另一个是在此基础上进一步修改，设计完成一个包含图文自定义网格的电器清单界面，如图7-1(b)所示。

(a) 简单网格界面　　(b) 自定义网格界面

图7-1　网格界面

相关知识

一、GridView

GridView(网格视图)用于在界面上按行、列分布的方式显示多个组件。GridView和ListView有相同的父类，因此它们具有相似的特性。它们的主要区别在于：ListView是在

一个方向上分布，而 GridView 是在两个方向上分布。当设计九宫格界面时，GridView 是首选，也是最简单的。

一个 GridView 的创建需要以下 3 个元素（这与 ListView 相似）。

（1）GridView 中的每一个 View。

（2）填入 View 的数据或者图片等。

（3）连接数据与 GridView 的适配器。

要使用 GridView，重点就是要掌握适配器的使用。与 ListView 中适配器的使用类似，GridView 中也是使用 ArrayAdapter、SimpleAdapter 等适配器实现数据与 View 间的连接与显示。

GridView 的属性如表 7-1 所示。

表 7-1　GridView 的属性

属　性	描　述
android:numColumns	设置内容显示的列数
Android:columnWidth	设置列的宽度
android:horizontalSpacing	设置两列之间的间距
android:verticalSpacing	设置两行之间的间距
android:stretchMode	设置缩放模式时多余空间的填充方式。 值选项： • none：缩放时元素左对齐，右方填充多余空间。 • columnWidth：缩放与列宽大小同步。 • spacingWidth：缩放时列宽不变，多余空间填充在列间。 • spacingWidthUniform：缩放时列宽不变，多余空间填充在列间与第 1 列的左边和最后 1 列的右边
android:gravity	设置内容的对齐方式。 值选项： • center_horizontal：水平居中。 • center_vertical：垂直居中。 • center：水平与垂直均居中。 • fill_horizontal：拉伸对象的水平尺寸以完全填满其所在容器（垂直尺寸不变）。 • fill_vertical：拉伸对象的垂直尺寸以完全填满其所在容器（水平尺寸不变）。 • fill：拉伸对象完全填满水平与垂直两个方向。 • clip_vertical：垂直方向裁剪溢出内容（当对象边缘超出容器时，将上、下边缘超出的部分剪切掉）。 该选项可以多选，用“|”分开

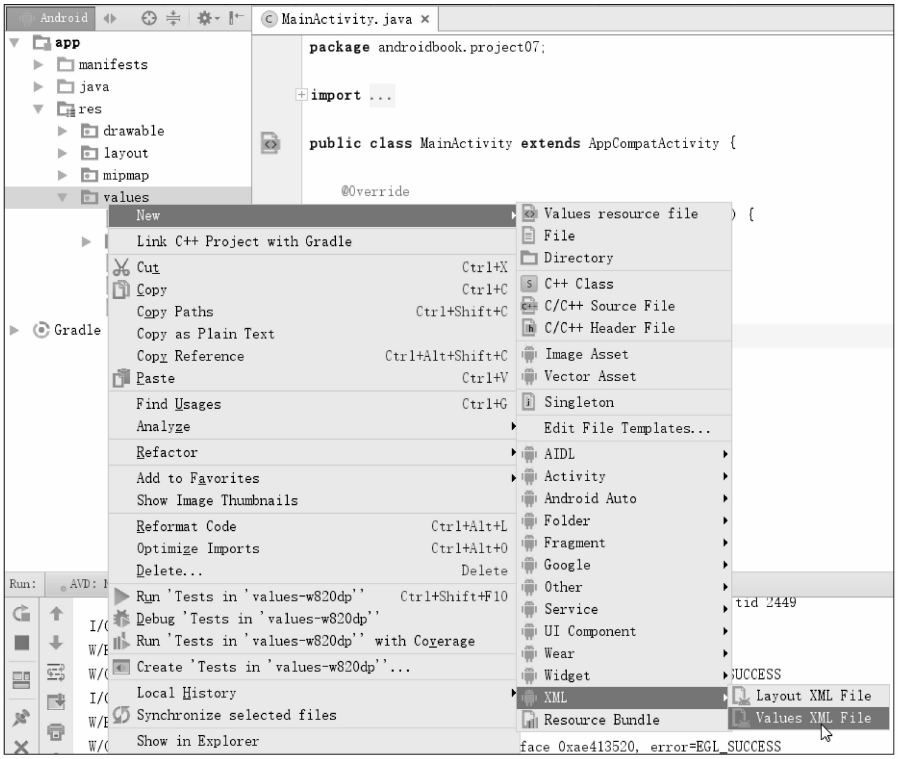
二、XML 格式数组数据的设计与提取

在 Android 工程的“res>values”文件夹中，可以创建一个用来存放各种数组数据的文件，文件以 XML 格式组织内容。其数据可以是字符串数组、整型数组等，数组中的数据可以是具体的值，也可以是对资源数据的引用。它的使用分两个部分：一是设计，二是提取。下

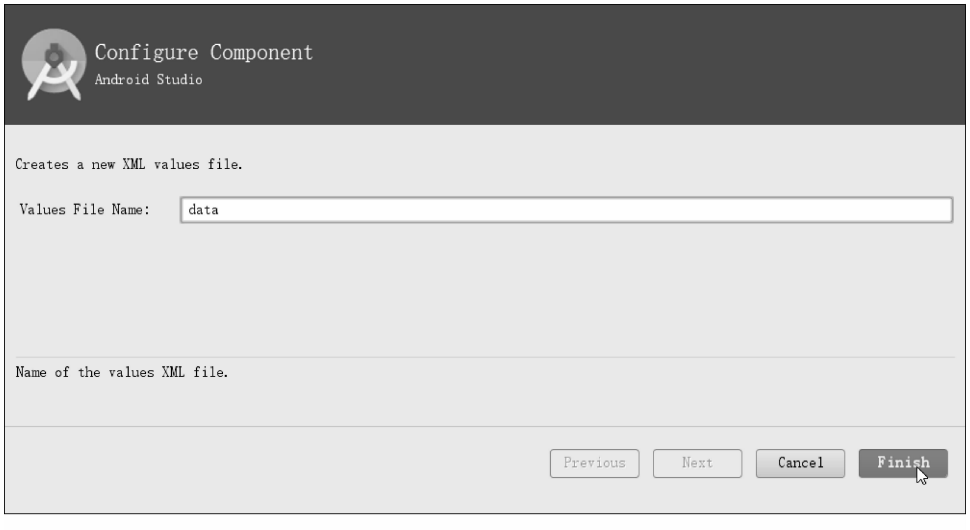
面分别说明。

1. 设计 XML 格式的数组数据

(1)第一步:在“res>values”文件夹中新建“data. xml”文件,如图 7-2 所示。



(a)



(b)

图 7-2　新建 XML 格式的数据文件

(2)第二步:在“data. xml”文件中编辑数据。

①当数组中的数据为具体的值时,其代码结构与内容如下。

```
<string-array name="types">
    <item>冰箱</item>
    <item>彩电</item>
    <item>洗衣机</item>
    <item>空调</item>
</string-array>
```

②当数组中的数据是对资源数据的引用时,其代码结构与内容如下。

```
<string-array name="images">
    <item>@drawable/fridge</item>
    <item>@drawable/tv</item>
    <item>@drawable/airconditioner</item>
    <item>@drawable/airconditioner</item>
</string-array>
```

2. 在代码中获取数组中的数据

(1)当数组中的数据为具体的值时,其获取数据的代码如下。

```
String[] types= getResources().getStringArray(R.array.types);
```

若数据是其他类型的数组,也可以通过 Resources 类中相应的方法获取,例如,获取整型数组的数据方法为 getIntArray()。

(2)当数组中的数据是对资源数据的引用时,其获取数据的代码如下。

```
TypedArray images= getResources().obtainTypedArray(R.array.images);
for (int i = 0; i < images.length(); i++) {
    Bitmap bitmap = BitmapFactory.decodeResource(this.getResources(),
images.getResourceId(i,0));
}
```

获取“data. xml”中数据项为引用资源数据的数组时,首先需要通过 Resources 类中的 obtainTypedArray 方法获取到 TypedArray 实例,然后通过 TypedArray 方法中的 getResourcesId 方法获取数组中每一项的资源 id,这样就能顺利地引用到数组中的资源了。

通过代码获取“data. xml”中的数组资源时,数组中的元素项不宜过多,特别是一次性获取时,有可能你在使用时它还没有获取到你需要使用的数组项。

三、自定义适配器

在开发 GridView、ListView 等集合类控件时，可以直接使用 Android API 封装好的适配器，如 ArrayAdapter、SimpleAdapter 等。但这些适配器绑定的 View 对象在处理事件的响应方面，通常只能局限在一个网格或一个行单位上，若一个网格里面有文本、按钮和图片等多个控件，而不同的控件希望得到不一样的响应操作，那就需要采用自定义适配器的解法方案了。

自定义适配器需要继承自某个 Adapter 类，BaseAdapter 是所有适配器类的基类，但由于没有实现绑定数据的功能，就必须重写 getCount、getItem、getItemId 和 getView 方法。自定义适配器也可以根据数据类型选择继承自 ArrayAdapter 或 SimpleAdapter 等类，由于这些派生类实现过基本控件的绑定，因而在方法的重写上可以根据需要进行选择。

如果适配的数据只是简单的数组类型的数据，可以使用继承 ArrayAdapter 类，其语法结构如下。

```
public class GridViewAdapter extends ArrayAdapter {
//定义类成员、重写方法等
}
```

在整个 Adapter 类需要重写的四个方法中，getView 方法是最重要的，它的作用是得到一个显示在数据列表中的 View。在 getView 方法中重点需要实现：加载指定 XML 布局文件，设置每个 Item 显示内容，绑定 Item 中各个控件的监听事件等功能。其核心代码如下。

```
1    @Override
2    public View getView(int position, View convertView, ViewGroup parent) {
3    View item = convertView;
4    TextView type;
5    //加载指定 XML 布局文件
6    LayoutInflater inflater = ((Activity) context).getLayoutInflater();
7    item = inflater.inflate(layoutResource, parent, false);
8    //设置每个 Item 显示内容
9    type = (TextView) item.findViewById(R.id.type);
10    item.setTag(type);
11    type.setText(gridItem.getType());
12    //返回 view
```

```
13      return item;
14      }
```

注意：上述第 11 行代码中 setText 的参数 gridItem. getType()，其作用是为文本对象传入对应的数据。

项目开发

一、任务分析

本项目需要学习者从简单到复杂、循序渐进地实现两个 GridView 设计的页面。第一个页面设计使用 Android API 自带的适配器，实现简单网格界面；第二个页面设计则通过自定义适配器，实现自定义网格显示界面。

1. 简单网格界面

简单网格界面元素分解如图 7-3 所示。整个界面只有一个 GridView 控件，网格中的数据来源于 XML 格式的字符串数组，使用 ArrayAdapter 做数据适配器。

图 7-3　简单网格界面元素分解

2. 自定义内容网格界面

自定义内容网格界面元素分解如图 7-4 所示。整个主界面仍然是 1 个 GridView，GirdView 中每一个 Item 的布局是通过自定义的 Layout XML 文件来设计的，数据来自 XML 格式封装的 Values XML 文件，最后通过自定义的适配器作为桥梁将数据显示在 APP 界面上。

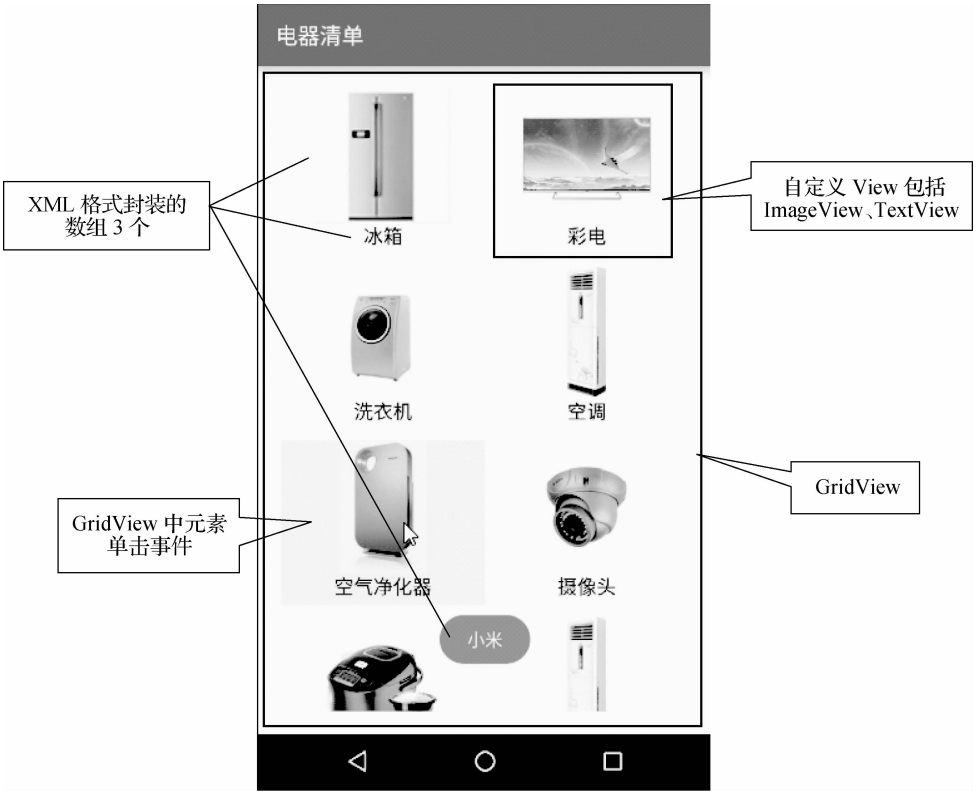


图 7-4　自定义内容网格界面元素分解

GirdView 的 Item 中包含 1 个 ImageView 和 2 个 TextView。

3 类数据分别由 3 个字符串型数组提供，其中，图像数据是对资源的引用。

二、任务实施

1. 简单网格界面

(1)“activity_appliances_grid. xml”界面布局文件。“activity_appliances_grid. xml”界面布局文件内容如下。

```
1   <? xml version="1.0" encoding="utf-8"? >
2   <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/
    android"
3   xmlns:tools="http://schemas.android.com/tools"
4   android:layout_width="match_parent"
5   android:layout_height="match_parent"
6   android:paddingBottom="@dimen/activity_vertical_margin"
7   android:paddingLeft="@dimen/activity_horizontal_margin"
8   android:paddingRight="@dimen/activity_horizontal_margin"
9   android:paddingTop="@dimen/activity_vertical_margin"
10   tools:context="androidbook.project07.AppliancesGridActivity">
11
12   <GridView
13       android:id="@+id/gvAppliances"
14       android:layout_width="match_parent"
15       android:layout_height="match_parent"
16       android:numColumns="2"
17       android:columnWidth="100dp"
18       android:layout_margin="5dp"
19       android:verticalSpacing="10dp"
20       android:gravity="center"
21       android:stretchMode="columnWidth"/>
22       </RelativeLayout>
```

关键代码解析如下。

①第 2～10 行代码：RelativeLayout（相对布局）及其属性设置（容器宽、高、内部边距等）。

②第 13～15 行代码：GridView（列表视图），其 id 属性为 gvAppliances。

③第 16 行代码：numColumns 属性，用来设置网格对象显示的列数，默认也为 2 列。

④第 17 行代码：columnWidth 属性，为每列的宽度。

⑤第 21 行代码：当界面缩放时列宽也同比例缩放。

GridView 布局设计视图如图 7-5 所示。

图 7-5　GridView 布局设计视图

(2)“griddata.xml”数据文件。“griddata.xml”数据文件内容如下。

```
<? xml version="1.0" encoding="utf-8"? >
<resources>
  <string-array name="types">
      <item>冰箱</item>
      <item>彩电</item>
      <item>洗衣机</item>
      <item>空调</item>
      <item>空气净化器</item>
      <item>摄像头</item>
       <item>电饭煲</item>
       <item>豆浆机</item>
       <item>面包机</item>
```

```
13    </string-array>
14    <string-array name="brands">
15        <item>海尔</item>
16        <item>创维</item>
17        <item>小天鹅</item>
18        <item>格力</item>
19        <item>小米</item>
20        <item>海康威视</item>
21        <item>苏泊尔</item>
22        <item>九阳</item>
23        <item>东菱</item>
24    </string-array>
25    <string-array name="images">
26        <item>@drawable/fridge</item>
27        <item>@drawable/tv</item>
28        <item>@drawable/airconditioner</item>
29        <item>@drawable/aircleaner</item>
30        <item>@drawable/camera</item>
31        <item>@drawable/electriccooker</item>
32        <item>@drawable/airconditioner</item>
33        <item>@drawable/breadmachine</item>
34    </string-array>
35 </resources>
```

关键代码解析如下。

①第 3～13 行代码:家电类型数组数据。

②第 14～24 行代码:家电品牌数组数据。

③第 25～34 行代码:家电图像数组数据,其资源引用自 drawable 目录下的图像文件。

(3)“AppliancesGridActivity.java”文件。“AppliancesGridActivity.java”文件内容如下。

```
1  import android.support.v7.app.AppCompatActivity;
2  import android.os.Bundle;
3  import android.widget.GridView;
4  import android.widget.ArrayAdapter;
5
```

```
6  public class AppliancesGridActivity extends AppCompatActivity {
7    private GridView gvAppliances;
8
9    @Override
10    protected void onCreate(Bundle savedInstanceState) {
11        super.onCreate(savedInstanceState);
12        setContentView(R.layout.activity_appliances_grid);
13
14        gvAppliances = (GridView) findViewById(R.id.gvAppliances);
15
16        String[] types = getResources().getStringArray(R.array.types);
17        /*简单网格*/
18         ArrayAdapter<String> adapter = new ArrayAdapter<String>
          (this, android.R.layout.simple_list_item_1, types);
19        gvAppliances.setAdapter(adapter);
20  }
```

关键代码解析如下。

①第 7 行代码:声明界面中的一个 GridView 对象。

②第 14 行代码:与第 7 行定义的 GridView 变量关联并实例化。

③第 16 行代码:从“griddata.xml”文件中提取 types 数组数据。

④第 18 行代码:实例并配置 arrayAdapter(数据适配器),显示样式选用了 layout 模板中的 simple_list_item_1 设置。

⑤第 19 行代码:为 GridView 对象绑定 arrayAdapter 适配器,显示于设备的屏幕上。

2. 自定义内容网格界面

(1)新建布局文件“grid_applicances.xml”。新建布局文件“grid_applicances.xml”的内容如下。

```
1  <? xml version="1.0" encoding="utf-8"? >
2  <LinearLayout xmlns:android="http://schemas.android.com/apk/res/
     android"
3    android:layout_width="match_parent"
4    android:layout_height="match_parent"
5    android:orientation="vertical">
6
```

```
7    <ImageView
8        android:id="@+id/image"
9        android:layout_width="120dp"
10       android:layout_height="120dp"
11       android:layout_gravity="center_horizontal" />
12
13   <TextView
14       android:id="@+id/type"
15       android:layout_width="wrap_content"
16       android:layout_height="wrap_content"
17       android:textColor="#000"
18       android:textSize="8pt"
19       android:layout_gravity="center_horizontal"
20       android:layout_marginBottom="2pt" />
21
22   <TextView
23       android:id="@+id/brand"
24       android:layout_width="wrap_content"
25       android:layout_height="0dp"/>
26
27 </LinearLayout>
```

关键代码解析如下。

①第 2～5 行，第 27 行代码：网格元素的容器，使用垂直线性布局。

②第 7～11 行代码：ImageView 控件，id 值为 image，在容器中高、宽均为 120 dp，水平居中。

③第 13～25 行代码：2 个 TextView，一个用来显示家电类型，id 值为 type，另一个用来显示家电品牌，id 值为 brand，高度为 0 dp(初始不显示在界面上)。

(2)新建网格元素实体类“GridItem. java”文件。新建网格元素实体类“GridItem. java”文件内容如下。

```
1  import android.graphics.Bitmap;
2
3  public class GridItem {
4    private Bitmap image;
```

```
5    private String brand;
6    private String type;
7    //构造函数
8    public GridItem() {   }
9
10    public GridItem(Bitmap image, String brand, String type) {
11        super();
12        this.image = image;
13        this.brand = brand;
14        this.type = type;
15    }
16
17    public String getType() {
18        return type;
19    }
20    public void setType(String type) {
21        this.type = type;
22    }
23
24    public Bitmap getImage() {
25        return image;
26    }
27    public void setImage(Bitmap image) {
28        this.image = image;
29    }
28
31    public String getBrand() {
32        return brand;
33    }
34    public void setBrand(String brand) {
35        this.brand = brand;
36    }
37 }
```

该实体类主要用于对网格元素中的 3 类数据进行封装，方便写入和读取元素中的数据。

（3）新建自定义适配器“GridViewAdapter. java”文件。新建自定义适配器“GridViewAdapter. java“文件内容如下。

```
1  import android.app.Activity;
2  import android.content.Context;
3  import android.support.annotation.NonNull;
4  import android.view.LayoutInflater;
5  import android.view.View;
6  import android.view.ViewGroup;
7  import android.widget.ArrayAdapter;
8  import android.widget.ImageView;
9  import android.widget.TextView;
10
11  import java.util.ArrayList;
12
13  public class GridViewAdapter extends ArrayAdapter {
14     private Context context;
15     private int layoutResource;
16     private ArrayList<GridItem> data = new ArrayList<GridItem>();
17
18      public GridViewAdapter(Context context, int resource, ArrayList<
        GridItem> objects) {
19         super(context, resource, objects);
20         this.layoutResource = resource;
21         this.context = context;
22         this.data = objects;
23     }
24
25     @NonNull
26     @Override
27     public View getView(int position, View convertView, ViewGroup parent) {
28         View item = convertView;
29         ViewHolder holder = null;
30
```

```
31          if (item == null) {
32                  LayoutInflater inflater = ((Activity) context).
getLayoutInflater();
33              item = inflater.inflate(layoutResource, parent, false);
34              holder = new ViewHolder();
35              holder.image = (ImageView) item.findViewById(R.id.image);
36              holder.brand = (TextView) item.findViewById(R.id.brand);
37              holder.type = (TextView) item.findViewById(R.id.type);
38              item.setTag(holder);
39          } else {
40              holder = (ViewHolder) item.getTag();
41          }
42          GridItem gridItem = data.get(position);
43          holder.image.setImageBitmap(gridItem.getImage());
44          holder.brand.setText(gridItem.getBrand());
45          holder.type.setText(gridItem.getType());
46          return item;
47      }
48
49      static class ViewHolder {
50          ImageView image;
51          TextView brand;
52          TextView type;
53      }
54  }
```

关键代码解析如下。

①第 13 行代码：定义 GridViewAdapter 类，继承自 ArrayAdapter 类。

②第 14～16 行代码：定义成员变量，用于类体内数据存取。

③第 18～23 行代码：声明一个基于 ArrayAdapter 的构造器(必需)。3 个参数分别为上下文对象、布局资源文件和数据列表。

④第 27～47 行代码：重写 getView 方法。

⑤第 32～38 行代码：加载布局文件，且与 ViewHolder 持有者类中定义的 3 个变量关联并实例化。

⑥第 42～45 行代码：设置每个网格元素的显示内容。

⑦第 49～53 行代码：定义 ViewHolder 持有者类，是一个临时的储存器，可以把 getView 方法中每次返回的 View 存起来，下次再用。这样做的好处是不必每次都到布局文件中去获取 View，提高了效率。

(4)“AppliancesGridActivity.java”文件程序改进。“AppliancesGridActivity.java”文件程序改进的内容如下。

```
1  import android.support.v7.app.AppCompatActivity;
2  import android.os.Bundle;
3  import android.view.View;
4  import android.widget.AdapterView;
5  import android.widget.GridView;
6  import android.widget.TextView;
7  import android.widget.Toast;
8  import android.content.res.TypedArray;
9  import android.graphics.Bitmap;
10  import android.graphics.BitmapFactory;
11
12  import java.util.ArrayList;
13
14  public class AppliancesGridActivity extends AppCompatActivity {
15    private GridView gvAppliances;
16    private GridViewAdapter customGridAdapter;
17
18    @Override
19    protected void onCreate(Bundle savedInstanceState) {
20        super.onCreate(savedInstanceState);
21        setContentView(R.layout.activity_appliances_grid);
22
23        gvAppliances = (GridView) findViewById(R.id.gvAppliances);
24
25        /* 自定义内容网格 */
26         customGridAdapter = new GridViewAdapter(this, R.layout.grid_
           appliances, getData());
27        gvAppliances.setAdapter(customGridAdapter);
28
```

```
29              gvAppliances.  setOnItemClickListener  ( new  AdapterView.
                OnItemClickListener() {
30          @Override
31          public void onItemClick(AdapterView<? > parent, View view,
            int position, long id) {
32                TextView text = (TextView) view.findViewById(R.id.
                  brand);
33                Toast.makeText(AppliancesGridActivity.this, text.
                  getText(), Toast.LENGTH_SHORT).show();
34          }
35      });
36  }
37
38  private ArrayList<GridItem> getData() {
39      final ArrayList<GridItem> gridItems = new ArrayList<GridItem
        >();
40       TypedArray images = getResources().obtainTypedArray(R.array.
         images);
41        String[] brands = getResources().getStringArray(R.array.
          brands);
42      String[] types = getResources().getStringArray(R.array.types);
43      for (int i = 0; i < images.length(); i++) {
44              Bitmap bitmap = BitmapFactory.decodeResource(this.
                getResources(), images.getResourceId(i, 0));
45          String brand = brands[i];
46          String type = types[i];
47          gridItems.add(new GridItem(bitmap, brand, type));
48      }
49      return gridItems;
50  }
51 }
```

关键代码解析如下。

①第 16、26 行代码：声明自定义适配器变量，并实例化。参数 this 为当前文档，参数 R.layout.grid_appliances 是网格元素的布局文件，参数 getData()是调用第 38～50 行的函

数，得到用来显示的数据。

②第 27 行代码：为 gvAppliances 的网格视图对象设置适配器，实现界面内容的显示。

③第 29～35 行代码：为网格元素设计单击事件的侦听功能，实现元素被点击后弹出“品牌”信息。

④第 40～42 行代码：从“griddata. xml”数据文件中提取 3 组数据。

⑤第 43～48 行代码：利用循环将数组中的数据分别取出，并添加至以网格元素为单位的数据列表中。