



Android 基础开发

第二章 第三节 Android高级组件



Java与移动智能设备开发



教学目标

- 掌握Android常见AdapterView的使用；
- 掌握Android中常用View的使用。



目录

1 AdapterView类视图控件的使用

2 TabHost视图控件的使用

3 其它常用视图控件的使用



AdapterView简介

- AdapterView：容器控件，其整体效果由每一个**子元素**内容决定，子元素的形式由**Adapter**决定。





AdapterView简介

- AdapterView : 容器控件，其整体效果由每一个子元素内容决定，子元素的形式由Adapter决定。





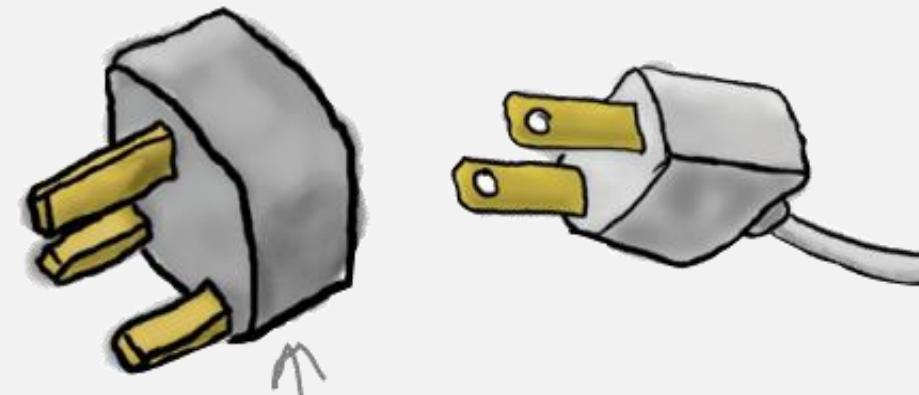
AdapterView简介

- AdapterView的子视图对象：
 - ListView：以垂直滑动列表形式显示一组数据。
 - GridView：以网格形式显示一组数据。
 - Spinner：以下拉列表形式显示一组数据。
 - Gallery：以水平滑动列表形式显示一组数据。
 -



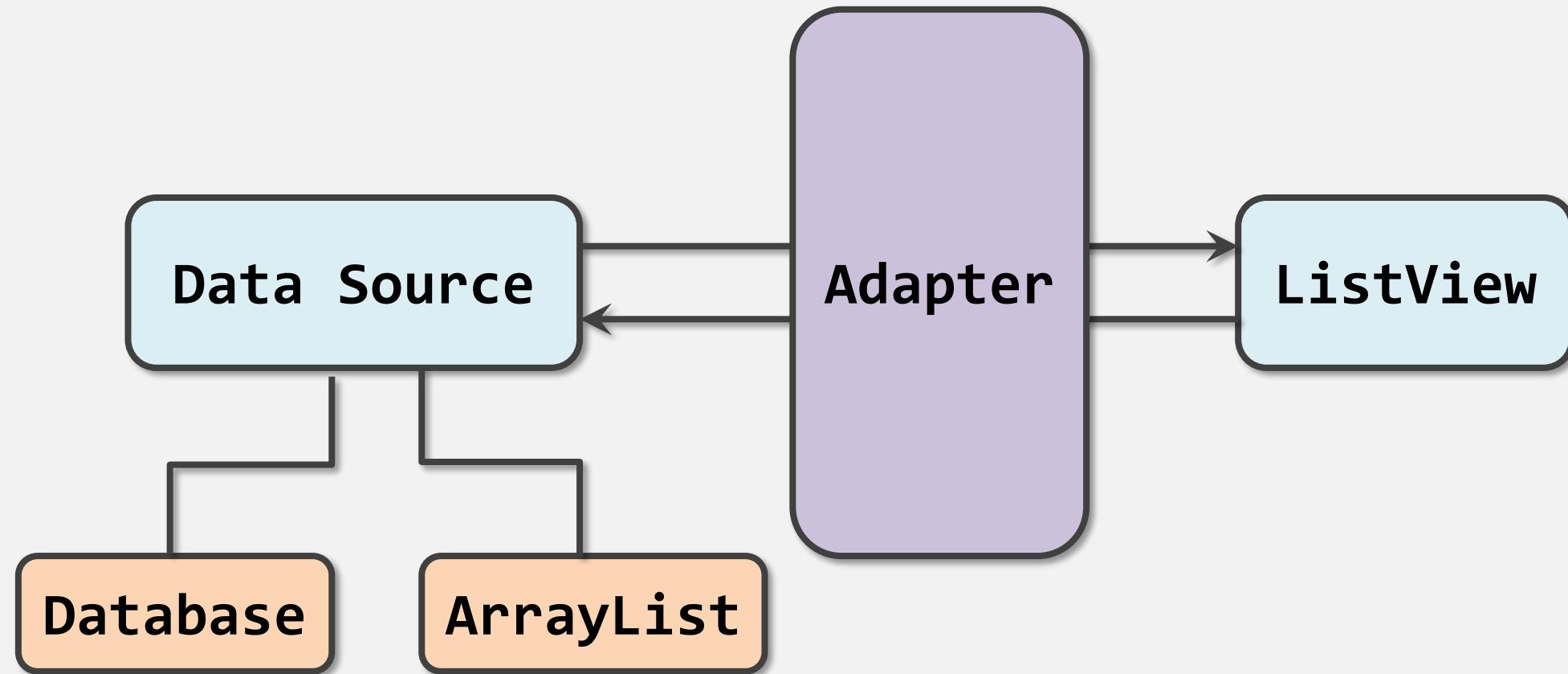
使用AdapterView需要解决的问题

- 待显示的数据如何传递给AdapterView中的Item ?
 - 准备数据（ 数据来源于哪里？数据是什么格式？数据格式和AdapterView视图项匹配吗？）
 - 借助**Adapter**来实现数据与View之间的数据传递。
 - Adapter : 数据和视图之间交互的中介。





使用AdapterView需要解决的问题





Adapter简介

- Adapter : 数据和视图之间交互的中介。
 - 数据改变时，不需要修改视图组件，只需更新Adapter。
 - 对于同一视图组件（ AdapterView子对象 ），数据源可能来自不同形式。
 - 视图组件变化时，不需要修改数据，绑定相同Adapter。
 - 对于同一组数据，可以显示为不同的视图形式（如ListView、GridView或其它）。



常用的Adapter

- Android中的常用Adapter
 - **ArrayAdapter**：最简单的适配器，数据源为文本字符串数组。
 - **SimpleAdapter**：简单适配器，数据源结构比较复杂，一般为List<Map>类型对象。
 - **SimpleCursorAdapter**：游标适配器，数据源一般为数据库中的数据。
 - **自定义适配器**：更灵活的适配器，数据源不定（由用户自行指定），需要继承BaseAdapter抽象类。



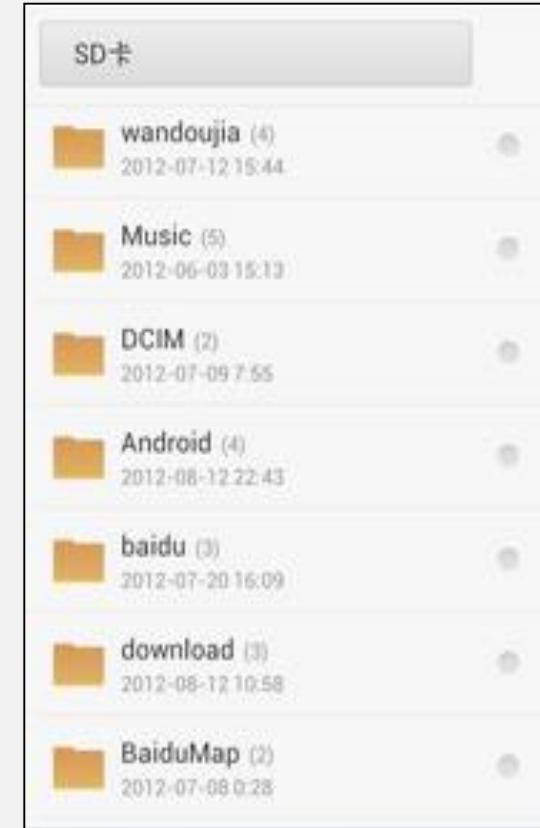
使用AdapterView

- AdapterView使用的基本流程：
 1. 准备数据源（现在为本地数据源，今后扩展为网络）。
 2. 准备AdapterView每一个子项的视图布局。
 3. 创建Adapter（连接数据源和视图布局）。
 4. 为指定AdapterView视图组件绑定适配器。
 5. 为AdapterView绑定事件监听器。



使用ListView

- ListView：以垂直可滑动列表形式显示子项目的视图容器，是一种AdapterView。





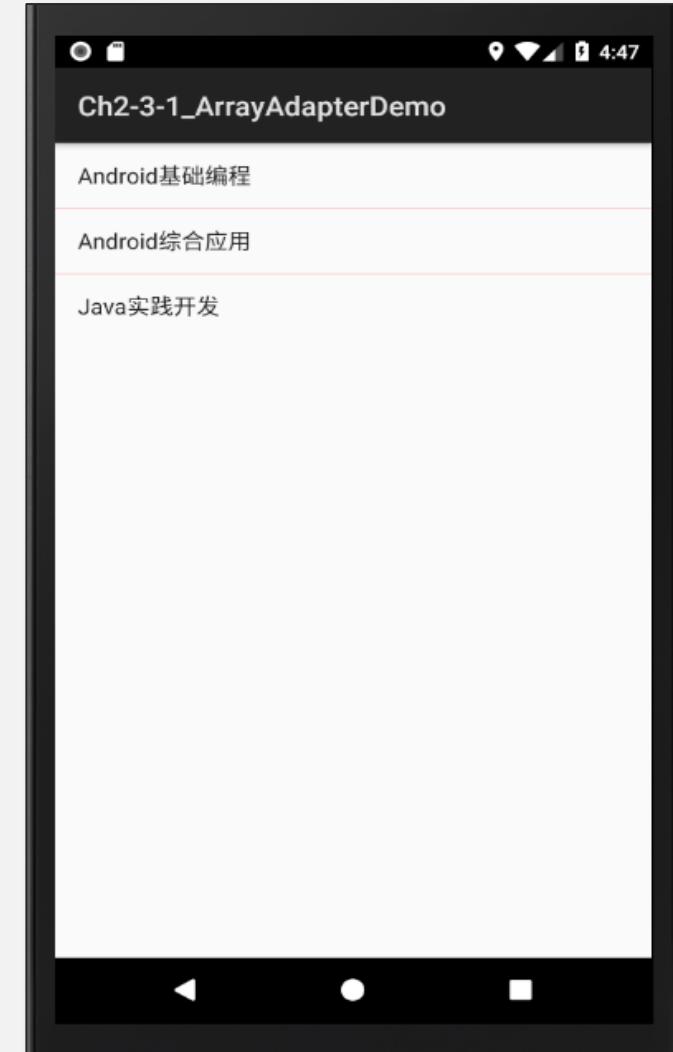
使用ListView

- ListView的应用十分广泛，在各种"设置"界面、列表显示界面都有它的身影。
- ListView使用的基本流程（同AdapterView）：
 1. 准备ListView每一个子项的视图布局。
 - 可以使用内置的布局，也可以用户自定义布局。
 2. 创建Adapter（连接数据源和视图布局）。
 3. 为ListView绑定Adapter。
 4. 为ListView绑定事件监听器。



使用ListView

- 简单ListView视图，要求如下：
 - Item布局使用内置视图布局；
 - 数据源为字符串数组；
 - 使用ArrayAdapter；
 - 绑定子项item的选择事件。





Step1：准备ListView子项视图布局

- 使用内置的视图布局
 - 内置的视图布局文件位于 "SDK目录 \platforms\android-XX\data\res\layout" 目录下。
 - 内置的视图布局文件在Activity中可以使用 "**android.R.layout.*****" 方式引用。



Step2：创建Adapter

- Activity文件中：

```
// 定义数据源数组
final String[] roles = {"Android基础编程", "Android综合应用",
                        "Java实践开发"};
// 定义Adapter
final ArrayAdapter<String> rolesAdapter = new ArrayAdapter<String>(
    this, // 上下文环境
    android.R.layout.simple_list_item_1, // 内置布局视图
    roles // 数据源
);
```

– ArrayAdapter是最简单的适配器，构造方法有很多，参考：

<https://developer.android.com/reference/android/widget/ArrayAdapter.html#pubctors>



Step3：创建Adapter

- Activity文件中：
 - 为AdapterView类对象绑定Adapter很简单，只需找到该对象，直接使用**setAdapter()**方法即可。

```
// 获取ListView控件  
lv_subjects = findViewById(R.id.lv_subjects);  
  
// 给AdapterView ( ListView控件)设置Adapter  
lv_subjects.setAdapter(rolesAdapter);
```



Step4：为ListView绑定事件监听器

- Activity文件：
 - 当ListView每一个子选项被点击时，将触发该事件监听器

```
// 为ListView的每一项绑定选择事件监听器
lv_roles.setOnItemClickListener(new AdapterView.OnItemClickListener() {
    @Override
    public void onItemClick(AdapterView<?> parent, View view,
                           int position, long id) {
        // parent : 该项目父适配器的引用
        // view : 当前项目视图控件的引用
        // position: 当前项目在ListView中的位置序号，序号从0开始
        // id : 当前项目在ListView中的行号
        Log.i("position", position+"");
        Log.i("item",rolesAdapter.getItem(position));
    }
});
```



使用ListView

- 自定义布局的ListView，要求如下：
 - Item布局使用自定义布局；
 - 使用自定义适配器。





Step1：准备ListView子视图布局

- res/layout/activity_main.xml

```
<LinearLayout  
    xmlns:android="http://schemas.android.com/apk/res/android"  
        android:layout_width="match_parent"  
        android:layout_height="match_parent"  
        android:orientation="vertical">  
    <ListView  
        android:id="@+id/lv_persons"  
        android:layout_width="match_parent"  
        android:layout_height="wrap_content">  
    </ListView>  
</LinearLayout>
```



Step1：准备ListView子视图布局

- res/layout/list_item.xml : 自定义布局

```
1. <LinearLayout  
2.     xmlns:android="http://schemas.android.com/apk/res/android"  
3.     android:layout_width="match_parent"  
4.     android:layout_height="match_parent"  
5.     android:orientation="horizontal">  
6.         <ImageView android:id="@+id/img_header"  
7.             android:layout_width="0dp"  
8.             android:layout_height="wrap_content"  
9.             android:layout_weight="2"  
10.            android:adjustViewBounds="true" />  
11.         <LinearLayout android:layout_width="0dp"  
12.             android:layout_height="match_parent"  
13.             android:layout_weight="8"  
14.             android:orientation="vertical">
```



Step1：准备ListView子视图布局

```
15.      <TextView android:id="@+id/txt_name"  
16.          android:layout_width="match_parent"  
17.          android:layout_height="wrap_content"  
18.          android:layout_weight="2"  
19.          android:textSize="24dp" />  
20.      <TextView android:id="@+id/txt_desc"  
21.          android:layout_width="match_parent"  
22.          android:layout_height="wrap_content"  
23.          android:layout_weight="1"  
24.          android:textSize="16dp" />  
25.  </LinearLayout>  
26. </LinearLayout>
```



Step2：创建Adapter

- Activity文件中：

```
CustomAdapter customAdapter = new CustomAdapter(  
    this,           // 上下文环境  
    dataSource,   // 数据源  
    R.layout.list_item // 列表项布局文件  
);
```

- 其中CustomAdapter为用户自定义的Adapter类。
- 该类需要继承**BaseAdapter**父类。



Step2：创建Adapter

- 自定义的BaseAdapter文件 (CustomAdapter)

```
1. public class CustomAdapter extends BaseAdapter {  
2.     private Context context;      // 上下文环境  
3.     private List<Map<String, Object>> dataSource; // 声明数据源  
4.     private int item_layout_id; // 声明列表项的布局  
5.     // 声明列表项中的控件  
6.     public CustomAdapter(Context context, List<Map<String, Object>> dataSource,  
7.                           int item_layout_id) {  
8.         this.context = context;      // 上下文环境  
9.         this.dataSource = dataSource; // 数据源  
10.        this.item_layout_id = item_layout_id; // 列表项布局文件ID  
11.    }  
12.    public int getCount() {return dataSource.size();}  
13.    public Object getItem(int position) { return dataSource.get(position); }  
14.    public long getItemId(int position) { return position; }  
15.    public View getView(final int position, View convertView, ViewGroup parent) {
```



Step2：创建Adapter

```
16.     if(null == convertView){ // 加载列表项布局文件
17.         LayoutInflater mInflater = LayoutInflater.from(context);
18.         convertView = mInflater.inflate(item_layout_id, null);
19.     } // 接下来先获取列表项中的控件对象
20.     ImageView img_header = convertView.findViewById(R.id.img_header);
21.     TextView txt_name = convertView.findViewById(R.id.txt_name);
22.     TextView txt_desc = convertView.findViewById(R.id.txt_desc);
23.     // 给数据项填充数据
24.     final Map<String, Object> mItemData = dataSource.get(position);
25.     img_header.setImageResource((int)mItemData.get("header"));
26.     txt_name.setText(mItemData.get("name").toString());
27.     txt_desc.setText(mItemData.get("desc").toString());
28.     // 给列表项中的控件注册事件监听器
29.     img_header.setOnClickListener(new View.OnClickListener() {.....});
30.     return convertView; // 返回列表项
31. }
32. }
```



Step3、4：绑定Adapter并添加事件监听器

- Activity文件：
 - 准备ListView控件中显示的数据。

```
// 准备数据源
List<Map<String, Object>> dataSource = new ArrayList<>();
Map<String, Object> itemData1 = new HashMap<>();
// 定义第1个数据项
itemData1.put("header", R.mipmap.header_flower);
itemData1.put("name", "校花");
itemData1.put("desc", "女神太累");
dataSource.add(itemData1);
// 第2个数据项
Map<String, Object> itemData2 = new HashMap<>();
.....
dataSource.add(itemData2);
.....
```



Step3、4：绑定Adapter并添加事件监听器

- Activity文件：
 - 获取ListView控件对象。
 - 创建Adapter适配器对象。
 - 给ListView控件设置适配器。
 - 给ListView的列表项注册点击事件监听器。



Step3、4：绑定Adapter并添加事件监听器



ListView小结

- ListView使用方法小结：
 - 准备ListView每一个子项的视图布局。
 - 可以使用内置的布局，也可以用户自定义布局。
 - 创建Adapter（连接数据源和视图布局）。
 - 为ListView绑定Adapter。
 - 为ListView绑定事件监听器。



ListView小结

- 若一个应用屏幕只有ListView列表，则可以令该Activity继承**ListActivity**类。
 - 在布局文件中，不需要添加<ListView>元素。
 - 在Activity中，直接使用**getListView()方法**，即获得ListView对象，其它方法相同。



练习

The screenshot shows a ListViewCon application interface. At the top, there is a navigation bar with icons for back, forward, search, and others. The title bar says "ListViewCon". The main content area displays five items, each consisting of a small profile picture and the celebrity's name followed by a brief biography.

- 范冰冰**
1981年9月16日生于山东青岛，毕业于上海师范大学谢晋影视艺术学院，中国女演员。
- 杨幂**
中国女演员、歌手、电视剧制片人。出生于北京。毕业于北京电影学院表演系。
- 张歆艺**
中国内地女演员，出生于1981年5月29日，2005年毕业于中央戏剧学院表演系本科班。
- 艾薇儿**
1984年9月27日出生于加拿大安大略省，加拿大女歌手、词曲创作者、演员。
- 刘诗诗**
原名刘诗施，中国内地影视女演员

The screenshot shows a WeChat inbox screen with 14 unread messages. The top status bar indicates the time is 10:43 AM, signal strength, battery level at 100%, and other connectivity details. The inbox lists messages from various sources, each with a timestamp and a preview of the message content.

- 腾讯新闻 (2 messages) - 5岁女孩举行婚礼6小时后去世 (早上10:43)
- 订阅号 (1 message) - 河北师大青年:师大最新脱单方式 | 各... (早上7:32)
- 我爱我家 (9 messages) - 陈老师: [链接]凌晨刚发生，再忙也要看... (9月11日)
- 欢乐你我他巫山烤全鱼 (1 message) - 你我他巫山烤鱼周六周日会员日开始啦 (9月10日)
- 离子逗比群 (2 messages) - 玩偶: 连书记都罩不住我们么么哒 (9月7日)
- 妈妈的妈妈 (0 messages)
- 浦发银行信用卡中心 (0 messages)
- 冀海跃 (0 messages)

At the bottom, there are four navigation icons: 微信 (14), 通讯录, 发现, and 我.



练习





练习

中国移动 10:51 61%

综合排序 销量优先 筛选

天猫 【苏宁直发】Apple/苹果 iPhone6s 64G 全网通4G智能
南京
包邮
¥4788.00 2686人付款 ...

顺丰包邮
赠送礼包
全国联保
闪电发货
闪电配送 卖点
4.7寸屏 A9处理器
移动电信联通 指纹识别
0首付分期购物

HOT iPhone 5c
正品苹果5c全网通手机移动联通电信4G
上海
广告
¥739.00 253人付款 ...

美版:三网通
移动2G联通4G电信4G
韩版:支持移动4G联通4G

中国移动 4G 10:50 62%

返回 edu2act.org 编辑

搜索

大赛组委会 8月23日 >
(9月10日截止提交作品) 西门子Wi...
2016年西门子WinCC创新创意设计大
赛 如果邮件无法正常显示, 请点击...

Pu 8月21日 >
【2016-8-30】【EI/SCI Journal P...
【EI/SCI Journal Publication(JA)】
【2016830】尊敬的老师, 你好! ...

劳动报酬、薪酬福利... 8月20日 >
劳动报酬、薪酬福利
新《劳动合同法》、《社会保险
法》、《工伤保险条例》实操应对策...

飘遥 8月19日 >
求《计算机导论》课件
武老师: 我正在学习您在51CTO学院
的《计算机导论》课程, 觉得您讲得...

技术走向管理先生 8月18日 >
从技术走向管理 ----- 技术...
从技术走向管理——课程简介 【时间
地点】 8月2526上海 8月2223深圳...

凡事辛力辛为 忙得佳 8月18日 >



练习

中国移动 10:53 60%

新房 二手房 搜索 地图

附近 总价 户型 更多

富强沁雅园 南北通透 两室两厅 双教育房 随时看房
77m² 2室2厅 南北 **108万**
沁雅园 14025元/m²
自营 钥匙 业主自评 优质教育

Fang.com

裕华 翰林雅筑 2室1厅 全款更名
93m² 2室1厅 南北 **62万**
翰林雅筑 6666元/m²
自营 业主自评

Fang.com

华兴小区 三室两厅 南北通透
豪华装修 华兴小学 看房方便
144m² 3室2厅 南北 **159万**
华兴小区 11041元/m²
自营 优质教育

Fang.com

青南小区 八中斜对面 交通便利 位置好 诚心出售
54.18m² 2室1厅 南 **65万**
青南小区 11997元/m²
自营 优质教育

Fang.com

新上蓝湾家园,低楼层,南北通

中国移动 10:52 61%

桃李阁 搜索

五味拌桃李阁 581m
★★★★★ 月售707单 46分钟
起售价 ¥8 配送费 ¥0
新用户立减5元

桃李阁脆皮鸡拌饭 581m
★★★★★ 月售914单 42分钟
起售价 ¥0 配送费 ¥0
满10减1;满15减3

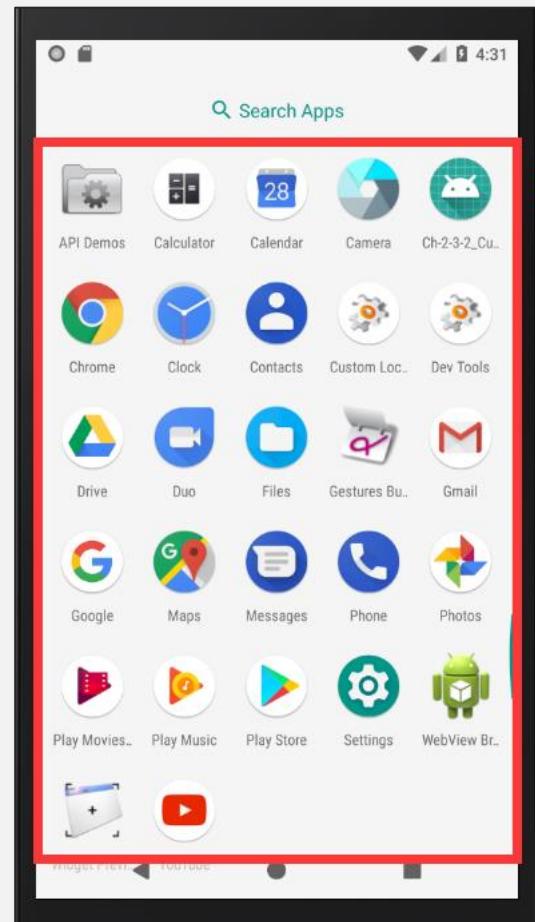
重庆小炒盖浇饭 (桃... 497m
★★★★★ 月售626单 38分钟
起售价 ¥0 配送费 ¥0
新用户立减5元

老干妈特色炒米 (桃... 540m
★★★★★ 月售282单 57分钟
起售价 ¥0 配送费 ¥0
满12减1



使用GridView

- GridView：以网格列表形式显示子项目的视图容器。





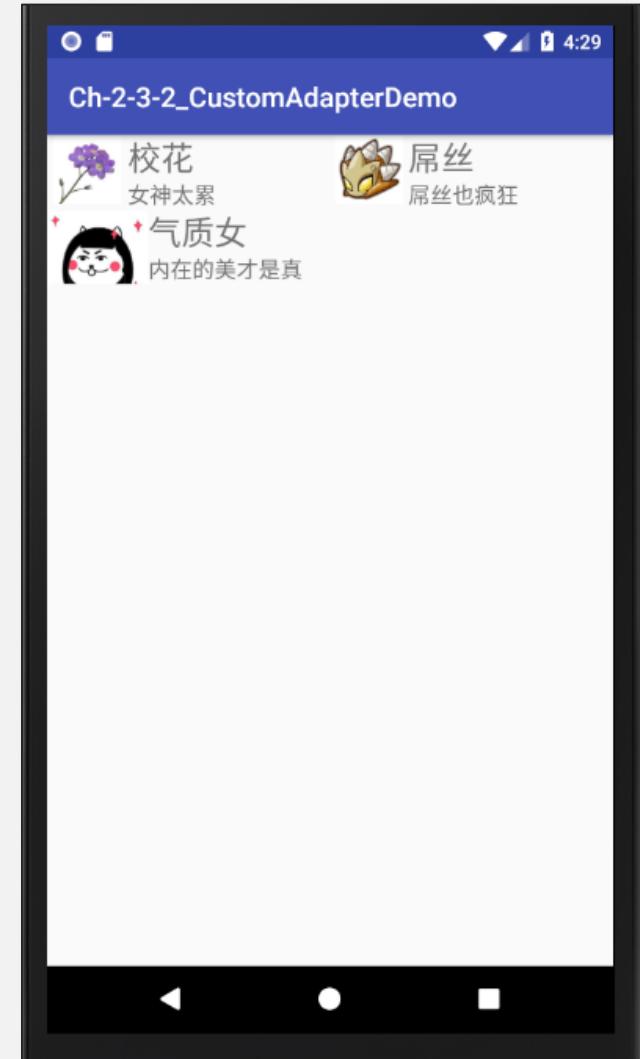
使用GridView

- GridView使用的基本流程（同ListView）：
 1. 准备GridView每一个子项的视图布局。
 - 可以使用内置的布局，也可以用户自定义布局。
 2. 创建Adapter（连接数据源和视图布局）。
 3. 为GridView绑定Adapter。
 4. 为GridView绑定事件监听器。



使用GridView

- 实现简单GridView，要求如下：
 - Item布局使用自定义布局。
 - 使用自定义适配器。





Step1：准备GridView子视图布局

- res/layout/main.xml : 建立GridView控件。

```
<GridView  
    android:id="@+id/lv_persons"  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    android:numColumns="auto_fit"  
    app:layout_constraintLeft_toLeftOf="parent"  
    app:layout_constraintRight_toRightOf="parent"  
    app:layout_constraintTop_toTopOf="parent" />
```

– 属性参考 :

- <http://developer.android.com/reference/android/widget/GridView.html#lattrs>



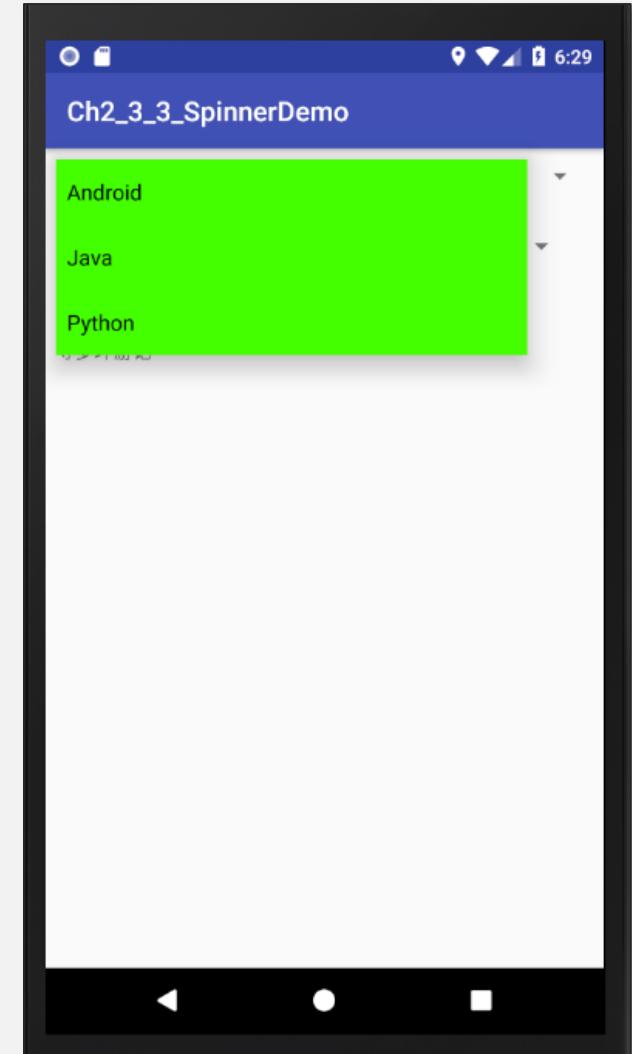
Step2：创建Adapter

- res/layout/list_item.xml：不变。
- Activity文件：
 1. 获取GridView控件；
 2. 创建Adapter对象；
 3. 给GridView控件设置适配器；
 4. 给GridView控件的列表项设置事件监听器。
- 自定义BaseAdapter适配器：不变。



使用Spinner

- Spinner：下拉列表。
- 使用基本流程：
 1. 建立子项目布局文件；
 2. 创建Adapter；
 3. 为视图控件绑定Adapter；
 4. 绑定事件监听器。





使用Gallary

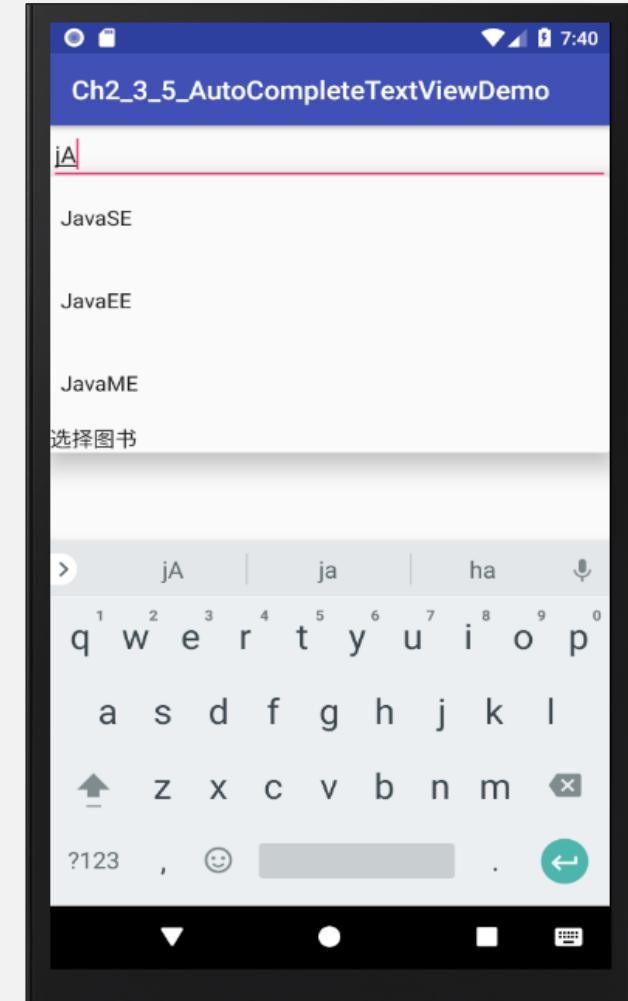
- Gallary : 水平滑动列表。
- 使用基本流程：
 1. 建立子项目布局文件；
 2. 创建Adapter；
 3. 为视图控件绑定Adapter；
 4. 绑定事件监听器。





使用AutocompleteTextView

- AutocompleteTextView。
 - 自动完成输入框
- 使用基本流程：
 1. 建立子项目布局文件；
 2. 创建Adapter；
 3. 为视图控件绑定Adapter；
 4. 绑定事件监听器。





AdapterView小结

- AdapterView：以列表形式一组数据，显示外观由不同的AdapterView子对象决定。
- AdapterView类对象使用的基本流程：
 1. 准备AdapterView每一个子项的视图布局；
 2. 创建Adapter（连接数据源和视图布局）；
 3. 为指定AdapterView视图控件绑定适配器；
 4. 为AdapterView绑定事件监听器。



目录

1 AdapterView类视图控件的使用

2 TabHost视图控件的使用

3 其它常用视图控件的使用



TabHost简介

- TabHost：选项卡或标签页，每一部分显示不同的布局页面。





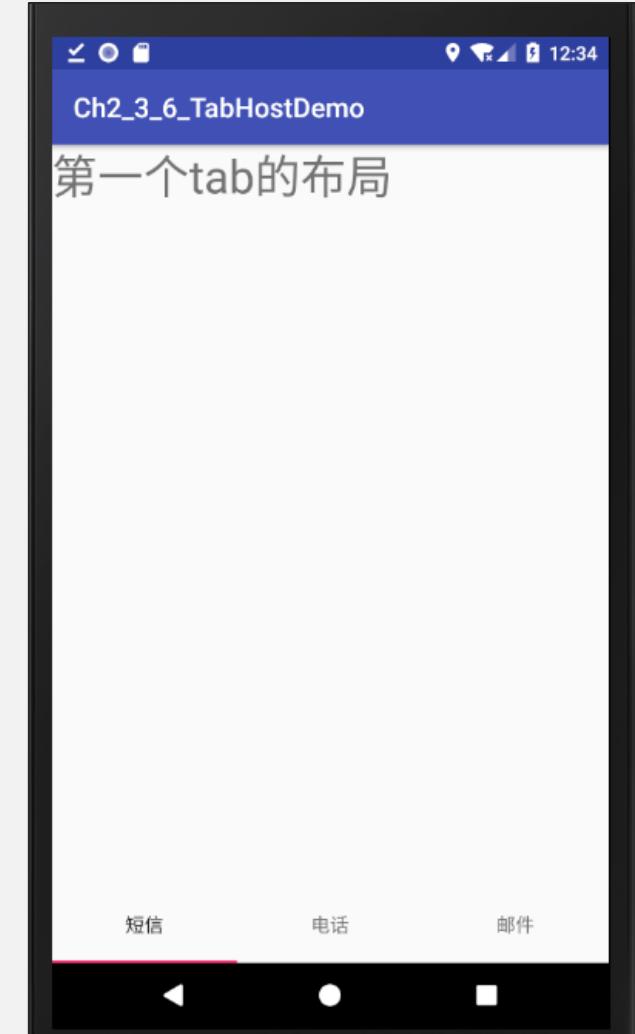
TabHost简介

- TabHost应用也是十分广泛，应用在要快速在多个界面之间切换功能。
- TabHost使用方法：
 1. 建立主布局页面和各个选项页面（使用XML方式）；
 2. 得到Tabhost组件；
 - 在主布局中包含tabhost视图。
 - 主Activity继承TabActivity。
 3. 通过tabhost.add添加子页面。



使用TabHost

- 实现一个简单的TabHost视图
 - 实现基本的TabHost（底部显示选项）。
 - 不同选项界面，布局页面不同。
 - 主Activity继承TabActivity（可选）。





使用TabHost

- TabHost实现步骤：
 1. 定义布局：在XML文件中使用TabHost组件，并在其中定义TabWidget、FrameLayout；
 2. 在Activity中获取TabHost对象；
 3. 创建选项卡：通过TabHost创建添加选项卡。
 - 指定选项卡的标题。
 - 设置选项卡对应的页面。
 - 添加选项卡。



Step1：建立主布局文件和选项界面

- res/layout/main.xml

主页面

```
<!-- TabHost组件id值不可变，需要引用 android的自带id -->
<TabHost
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/tabhost"      ....>
    <RelativeLayout      ....>
        <!-- TabWidget代表选项卡按钮，组件id值不可变，与FrameLayout构成Tabhost的两个必备组件 -->
        <TabWidget
            android:id="@+id/tabs"      ....>
            </TabWidget>
            <!-- FrameLayout组件代表页面，id值不可变 -->
            <FrameLayout
                android:id="@+id/tabcontent"  ...>
                </FrameLayout>
            </RelativeLayout>
        </TabHost>
```

选项菜单

选项页面



Step1：建立主布局文件和选项界面

- res/layout/main.xml

选项页

```
<FrameLayout  
    android:id="@+id/tabcontent" .....>  
    <!-- 第一个tab的布局 -->  
    <LinearLayout  
        android:id="@+id/tab1" .....>  
        <TextView  
            android:layout_width="wrap_content"  
            android:layout_height="wrap_content"  
            android:text="第一个tab的布局"  
            android:textSize="34dp"/>  
    </LinearLayout>  
    <!-- 第二个tab的布局 -->  
    <!-- 第三个tab的布局 -->  
</FrameLayout>
```



Step2：建立选项与主界面关系

- Activity页面

```
public class MainActivity extends AppCompatActivity {  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
        // 获取TabHost控件  
        TabHost tabHost = findViewById(android.R.id.tabhost);  
        // 初始化TabHost容器  
        tabHost.setup();  
        // 创建第1个Tab页面  
        TabHost.TabSpec tab1 = tabHost.newTabSpec("tab1")  
            .setIndicator("短信") // 设置选项卡名称称  
            .setContent(R.id.tab1); // 设置页面视图组件，可以是简单的  
View，也可以是Activity或Fragment  
        // 添加第1个Tab页面  
        tabHost.addTab(tab1); // 添加选项卡
```



Step2：建立选项与主界面关系

TabHost对象常用属性或方法，参考：

<http://developer.android.com/reference/android/widget/TabHost.html#pubmethods>



TabHost小结

- TabHost：选项卡或标签页，每一部分显示不同的布局页面。
- TabHost使用方法：
 1. 建立主布局页面和各个选项页面（使用XML方式）；
 2. 得到Tabhost组件；
 - 在主布局中包含tabhost视图。
 3. 通过tabhost.add添加子页面。
- TabHost应用十分广泛，结合多个Activity页面和Intent使用，功能更加丰富。



FragmentTabHost

- FragmentTabHost是TabHost的替代品。
- Tabhost已经不推荐使用了。

TabHost

命名空间:

```
android.widget.TabHost
```

初始化函数（必须在addTab之前调用）：

```
setup();
```

包含两个子元素:

1. Tab标签容器TabWidget (@android:id/tabs)
2. Tab内容容器FrameLayout (@android:id/tabcontent)

FragmentTabHost

命名空间:

```
android.support.v4.app.FragmentTabHost
```

```
android.support.v13.app.FragmentTabHost
```

初始化函数（必须在addTab之前调用）：

```
setup(this, getSupportFragmentManager(), R.id.realtabcontent);
```

包含三个子元素:

1. Tab标签 容器TabWidget (@android:id/tabs)
2. Tab 内容 容器FrameLayout [宽高皆为0dp] (@android:id/tabcontent)
3. Tab 内容 容器FrameLayout [真正的容器] (@+id/RealTabContent)

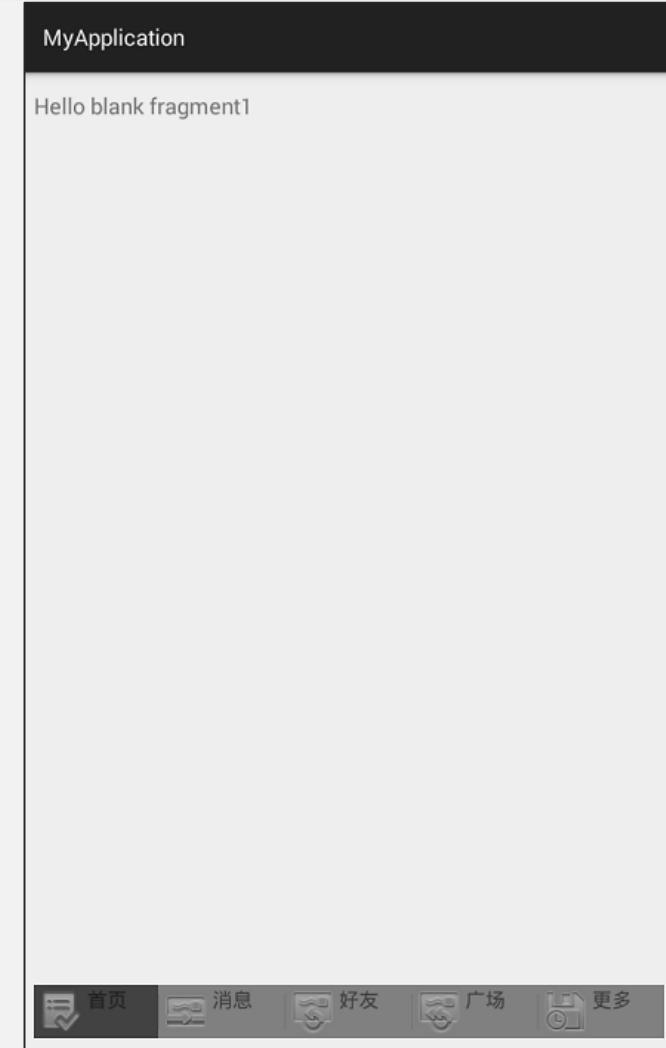


Fragment

- Fragment是为了解决Android APP运行的设备大小不一而提出来的，我们可以把Fragment当成Activity界面的一个组成部分，一个Activity中可以包含很多Fragment。
- Fragment拥有自己的生命周期，可接收、处理用户的事
件。
- Activity中可以动态的添加、替换和移除某个Fragment。



FragmentTabHost示例





FragmentTabHost的使用

1. 创建包含FragmentTabHost组件的布局文件（ XML文件 ）；
2. 设置标签项布局文件 ；
3. 创建标签项对应的页面布局文件；
4. 创建FragmentTabHost的标签页（ 若干Fragment ）；
5. 在Activity中将FragmentTabHost的标签项和标签页绑定在一起；
6. 给FragmentTabHost设置选项改变事件监听器。



FragmentTabHost的使用

- 1. 创建包含FragmentTabHost组件的布局文件（XML文件）。

```
<LinearLayout .....>
    <android.support.v4.app.FragmentTabHost
        android:id="@+id/tabhost"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" >
        <TabWidget android:id="@+id/tabs"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_gravity="bottom" />
        <FrameLayout android:id="@+id/tabcontent"
            android:layout_width="0dp"
            android:layout_height="0dp" />
    </android.support.v4.app.FragmentTabHost>
</LinearLayout>
```

底部标签条

标签项

标签页



FragmentTabHost的使用

- 2.设置标签项布局文件(fragment_tab.xml)。

```
<RelativeLayout .....>
    <LinearLayout android:layout_width="wrap_content"
                  android:layout_height="wrap_content"
                  android:layout_centerHorizontal="true"
                  android:layout_alignParentBottom="true"
                  android:gravity="center" >
        <TextView android:id="@+id/txt_tab_pic"
                  android:layout_width="wrap_content"
                  android:layout_height="wrap_content"
                  android:text="邮件"
                  android:drawableTop="@mipmap/mail"/>
    </LinearLayout>
</RelativeLayout>
```



FragmentTabHost的使用

- 3. 创建标签项对应的页面布局文件（fragment_page.xml）。

```
<RelativeLayout .....>
    <LinearLayout android:layout_width="match_parent"
                  android:layout_height="match_parent"
                  android:layout_alignParentTop="true"
                  android:layout_centerHorizontal="true">
        <TextView android:id="@+id/txt_page_content"
                  android:layout_width="match_parent"
                  android:layout_height="wrap_content"
                  android:text="模拟内容页面"
                  android:textSize="34dp"/>
    </LinearLayout>
</RelativeLayout>
```



FragmentTabHost的使用

- 4. 创建FragmentTabHost的标签页（若干Fragment）。
 - 以一个Fragment为例。

```
public class FragmentTab1 extends Fragment{  
    @Nullable  
    @Override  
    public View onCreateView(LayoutInflater inflater, @Nullable  
    ViewGroup container, Bundle savedInstanceState) {  
        View relativeLayout = inflater.inflate(R.layout.fragment_page,  
                                            container, false);  
        TextView contentPage  
            = relativeLayout.findViewById(R.id.txt_page_content);  
        contentPage.setText("模拟电话页面");  
        return relativeLayout;  
    }  
}
```



FragmentTabHost的使用

- 在Activity中将FragmentTabHost的标签项和标签页绑定在一起。
 - 定义标签项数据。
 - 定义标签项对应的页面。
 - 创建选项卡（关联标签与标签项对应的页面）。
 - 给FragmentTabHost注册选项改变事件监听器。

```
// 标签项图片
private int mImages[] = {R.mipmap.call, R.mipmap.message, R.mipmap.mail};
// 标签项数据
private String mFragmentTags[] = {"电话", "短信", "邮件"};
```



FragmentTabHost的使用

```
// 加载Fragment页面  
private Class mFragment[] = {  
    FragmentTab1.class, FragmentTab2.class, FragmentTab3.class  
};
```

```
// 声明FragmentTabhost控件  
private FragmentTabHost myTabhost;
```

```
@Override  
protected void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.activity_main);  
    initTabHost(); // 初始化FragmentTabHost，并创建选项卡  
}
```



FragmentTabHost的使用

```
private void initTabHost() {  
    myTabhost = (FragmentTabHost) findViewById(android.R.id.tabhost);  
    myTabhost.setup(this, getSupportFragmentManager(), android.R.id.tabhost);  
    // 去掉分割线  
    myTabhost.getTabWidget().setDividerDrawable(null);  
    for (int i = 0; i < mFragmentTags.length; i++) {  
        // 对Tab按钮添加标记和图片，getTextView(i)方法是获取了一个标签项  
        TabHost.TabSpec tabSpec = myTabhost  
            .newTabSpec(mFragmentTags[i]).setIndicator(getTextView(i));  
        // 添加Fragment  
        myTabhost.addTab(tabSpec, mFragment[i], null);  
    }  
    // 设置默认选中的选项卡  
    myTabhost.setCurrentTab(0);
```



FragmentTabHost的使用

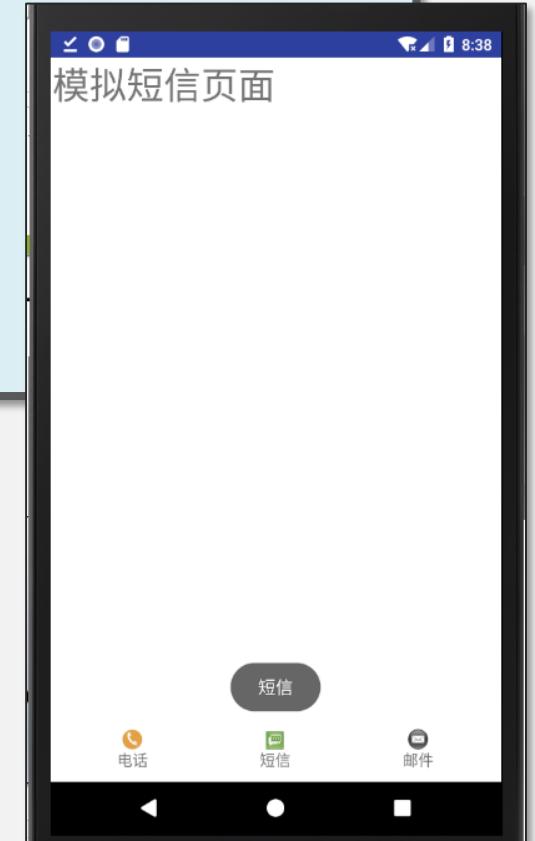
- 创建一个标签项View。

```
private View getTextView(int index){  
    View view = getLayoutInflater().inflate(R.layout.fragment_tab, null);  
    TextView textView = view.findViewById(R.id.txt_tab_pic);  
    textView.setText(mFragmentTags[index]);  
    textView.setCompoundDrawablesWithIntrinsicBounds(null,  
        this.getResources().getDrawable(mImages[index], null),  
        null, null);  
    return view;  
}
```



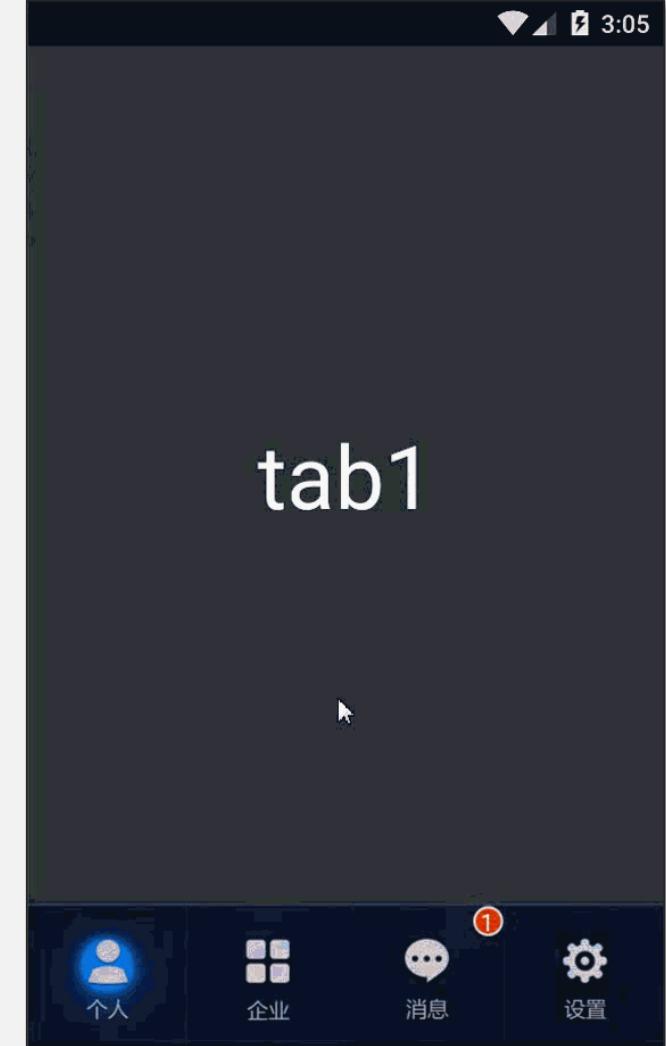
FragmentTabHost的使用

```
// 在Activity的onCreate方法中给FragmentTabHost控件注册选项改变事件监听器  
// 注册事件监听器  
myTabhost.setOnTabChangedListener(new TabHost.OnTabChangeListener() {  
    @Override  
    public void onTabChanged(String tabId) {  
        Toast.makeText(getApplicationContext(), tabId,  
Toast.LENGTH_SHORT).show();  
    }  
});
```





练习





练习



- 扩展
 - 如果不使用FragmentTabHost+FragmentActivity呢？
 - 如果修改标签项对应的页面？
 - 使用Fragment+FragmentManager如何实现同样的效果？



目录

1 AdapterView类视图控件的使用

2 TabHost视图控件的使用

3 其它常用视图控件的使用



其它常用UI组件

- Android中提供了十分丰富的UI组件供用户选择，在此简单罗列出常用的UI组件，具体使用方法，参考Android在线文档。

- Android UI组件参考文档地址：

- 使用示例：

<http://developer.android.com/guide/topics/ui/index.html>

- 文档：

<http://developer.android.com/reference/android/widget/package-summary.html>



其它常用UI组件

- ProgressBar : 进度条。



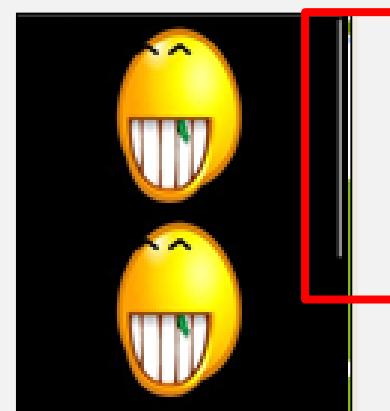
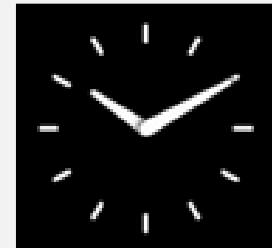
- DatePicker和TimePicker : 日期/时间选择。





其它常用UI组件

- SeekBar : 拖动条。
- RatingBar : 星级评分条。
- AnalogClock : 电子时钟。
- ScrollView : 滚动视图。
- 实现滚动视图。





目录

1 AdapterView类视图控件的使用

2 TabHost视图控件的使用

3 其它常用视图控件的使用



Thank you!

