Android App的开发
基本组件

基本视图

- 基本组件
- AdapterView类视图控件的使用
- 提示框

Android常用的View（一）

- TextView：显示一段文本内容
- EditText：显示接收用户输入的输入框

属性	说明
gravity	TextView内文本对齐方式
text	TextView内文本显示的内容
hint	EditText内默认显示的提示文本
inputType	EditText内文本的格式
ellipsize	如果TextView中文本太长可以设置中间文本用省略号取代，取值center
autoLink	取值email、phone等，给文本中的email或者电话增加链接

Android常用的View（二）

•RadioButton：单选按钮，用户只能在一组单选按钮中选择一个；使用时需要借助RadioGroup一起使用。

•CheckBox：多选框。

属性名	描述
orientation	RadioGroup的属性，设置其内部的RadioButton排列方式（水平或者垂直）
Checked	RadioButton或者CheckBox的属性，设置此项是否为选中状态

RadioButton/CheckBox

```
    CheckBox cb1= (CheckBox) findViewById(R.id.cb1);
    cb1.setOnCheckedChangeListener(new CompoundButton.OnCheckedChangeListener() {
        @Override
        public void onCheckedChanged(CompoundButton buttonView, boolean isChecked) {
            if(isChecked)
                Toast.makeText(MainActivity.this,"你选择的是苹果",Toast.LENGTH_LONG).show();
            else
                Toast.makeText(MainActivity.this,"你取消了苹果的选择",Toast.LENGTH_LONG).show();
```

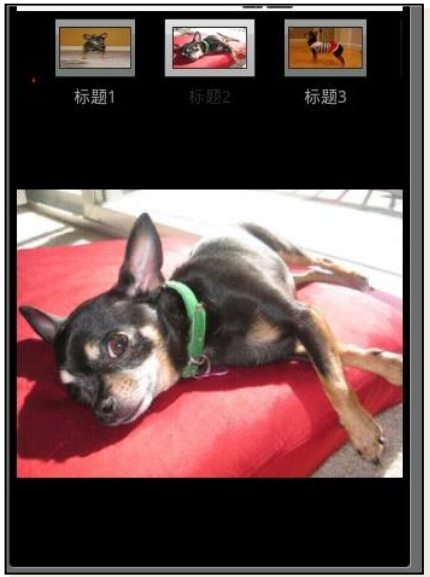
```
RadioButton rb1 = (RadioButton) findViewById(R.id.rb1);
rb1.setOnCheckedChangeListener(new CompoundButton.OnCheckedChangeListener() {
    @Override
    public void onCheckedChanged(CompoundButton buttonView, boolean isChecked) {
        if(isChecked)
            Toast.makeText(MainActivity.this,"你选择的是男",Toast.LENGTH_LONG).show();
        else
            Toast.makeText(MainActivity.this,"你选择的是女",Toast.LENGTH_LONG).show();
    }

});
```

基本视图

- 基本组件
- AdapterView类视图控件的使用
- 提示框

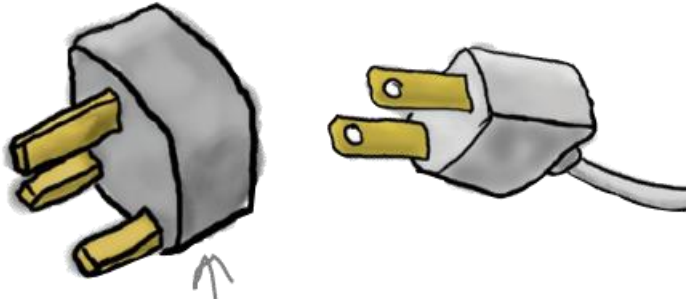
AdapterView简介

- 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。

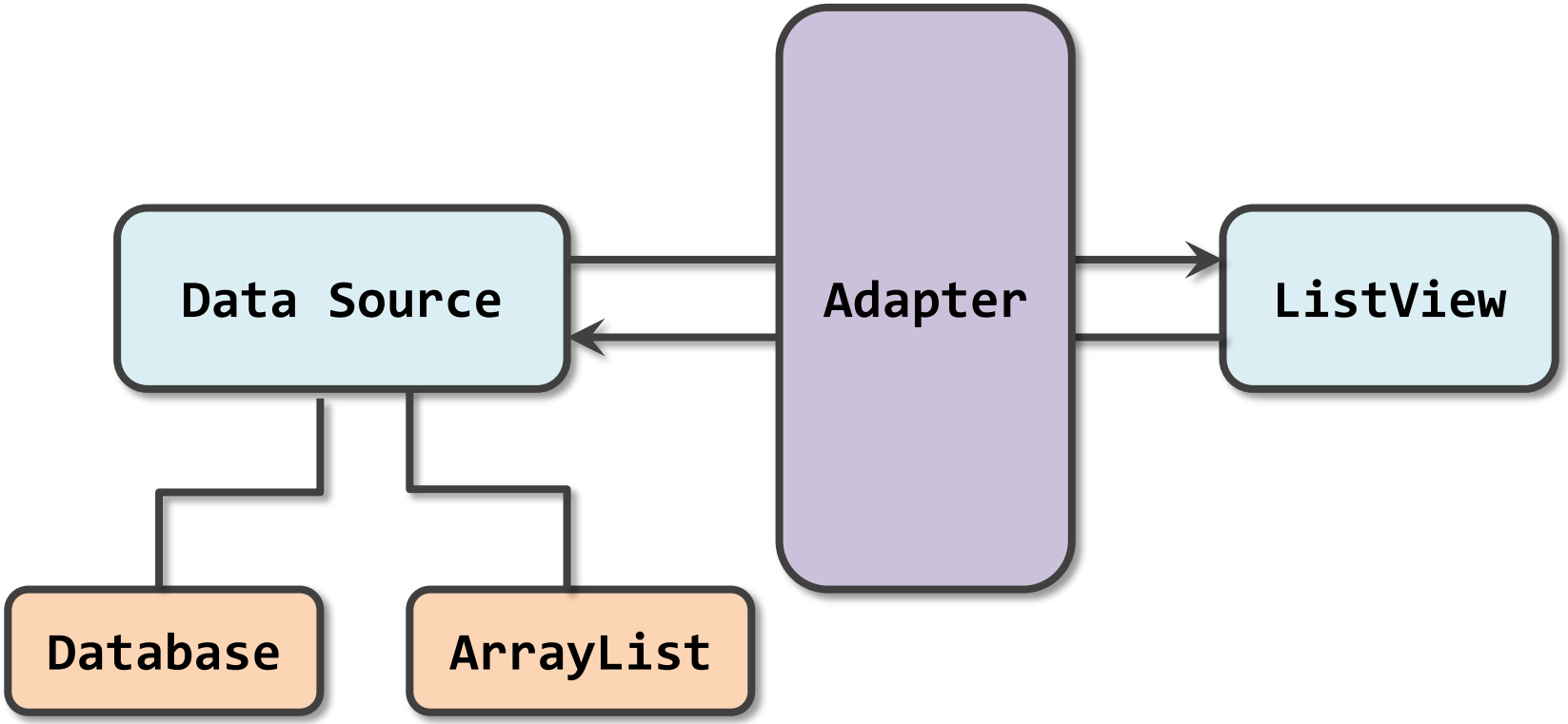


使用AdapterView需要解决的问题

- 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



AdapterView介绍

- 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。

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。

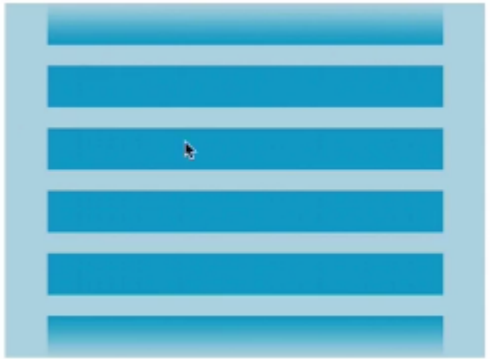
AdapterView简介

- AdapterView的子视图对象：
  - GridView：以网格形式显示一组数据。
  - ListView：以垂直滑动列表形式显示一组数据。
  - 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。
- ListView使用的基本流程：

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

- 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。

使用ListView

- Step2：创建Adapter：Activity文件中

```
// 定义数据源数组
final String[] course = {"NoSQL", "性能测试", "移动端测试"};
// 定义Adapter
final  ArrayAdapter<String> adapter = new ArrayAdapter<String>
        (this, //上下文
android.R.layout.simple_list_item_1,//内置视图样式
 course);//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

使用ListView

- Step3：为ListView绑定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 = findViewById(R.id.ListView1);
// 给AdapterView（ListView控件)设置Adapter
lv.setAdapter(adapter);
```

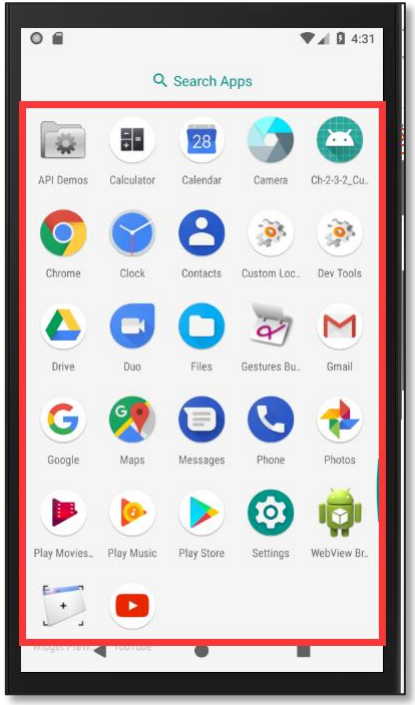
使用ListView

- 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.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",adapter.getItem(position));     }
});
```

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

自定义适配器

1)创建一个类继承BaseAdapter

2)实现4个方法

getCount：获取要显示的选项总数

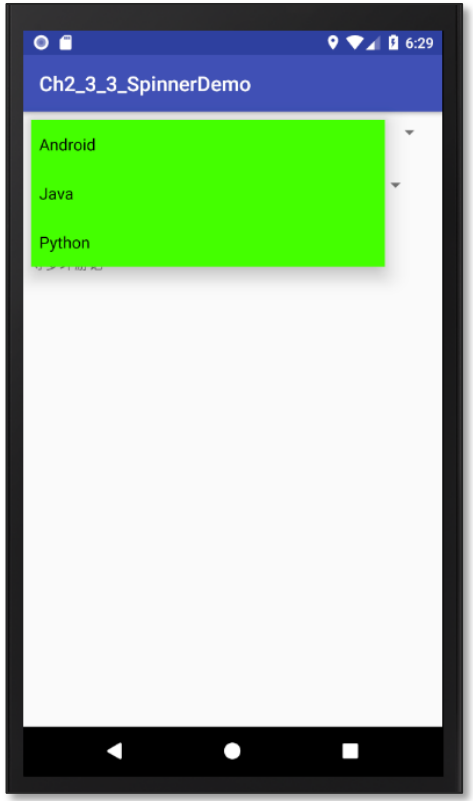
getItem：获取每一个选项

getItemId：选项ID

getView：该方法用来为每一个选项生成视图

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Spinner

绑定数据源：android:entries=″@array/week“

或者使用代码绑定

```
//通过适配器进行数据的绑定
String[] roles={"金牌会员","银牌会员","普通"};
ArrayAdapter<String> adapter =new ArrayAdapter<String>
        ( context: this,android.R.layout.simple_dropdown_item_1line,roles);
spinner.setAdapter(adapter);
```

基本视图

- 基本组件
- AdapterView类视图控件的使用
- 提示框

Toasts显示文本

- 提示信息Toast：当用户执行某个操作后，自动显示，且显示时间较短，会自动消失
- Toast一般使用在用户信息合法性校验、**关闭应用时的提示**等场合。

使用Toast

- Toast使用的基本流程
  - 创建Toast

```
Toast toastTip = Toast.makeText(MyActivity.this, “提示字符串”, Toast.LENGTH_LONG);
```

  - 设置Toast基本属性
    - toastTip.setGravity(Gravity.CENTER, 0, 0);

    //设置Toast信息的显示位置
  - 显示Toast

```
toastTip.show( );
```

或者：Toast.makeText(this,“显示的字符串”,Toast.LENGTH_LONG).show()

简单对话框实现

- Step1：创建对话框创建器
  - 使用AlertDialog.Builder类创建
    - AlertDialog.Builder AdBuilder = new AlertDialog.Builder(MyActivity.this);
- Step2：设置对话框属性
  - 设置基本属性

```
AdBuilder.setTitle("温馨提示");
AdBuilder.setMessage("您确定要退出安智市场吗？");
```

  - 添加按钮

```
AdBuilder.setPositiveButton("确定", new DialogInterface.OnClickListener() {
        public void onClick(DialogInterface dialog, int which) {
            // 退出当前应用
            MyActivity.this.finish();
        }
});
AdBuilder.setNegativeButton("取消", null);
```

简单对话框实现

- Step3：创建对话框

```
AdBuilder.create( );
```

- Step4：显示对话框

```
AdBuilder.show( );
```

菜单简介

- 菜单：显示一个应用程序的主界面中不直接可见额外选项的视图组件。

菜单简介

- 在Android中支持3种菜单形式
- 选项菜单：当用户按下“Menu”键时，弹出的菜单。
  - Android主窗口点击“Menu”弹出的菜单
- 子菜单：当用户点击“选项菜单”中的某一项时，弹出的附加菜单。
  - 在选项菜单中点击某一个选项时，弹出。
- 上下文菜单：当用户长按某个视图元素时，弹出的菜单（相当于电脑中的右键菜单）。
  - 文本元素长按时，会出现“复制”类菜单。

使用选项菜单和子菜单

- 选项菜单/子菜单使用的基本流程
  - 创建菜单项XML布局文件（或由Java代码生成）
  - 在Activity中创建菜单（onCreateOptionsMenu方法）
  - 绑定菜单项选择事件（onOptionsItemSelected方法）

使用选项菜单和子菜单

- Step1：使用XML文件创建菜单资源
  - 在res/menu/menu_options.xml

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android" >
    <item android:icon="@android:drawable/ic_menu_view"
        android:id="@+id/menu_item1"
        android:title="查看">
        <menu>
            <item android:id="@+id/menu_item8_submenu1"
                android:title="子菜单1" />
            <item android:id="@+id/menu_item8_submenu2"
                android:title="子菜单2" />
        </menu>
    </item>
    <item android:icon="@android:drawable/ic_menu_add"
        android:id="@+id/menu_item2"
        android:title="添加" />
    <item android:icon="@android:drawable/ic_menu_edit"
        android:id="@+id/menu_item3"
        android:title="编辑" />
</menu>
```

使用选项菜单和子菜单

- Step1：使用XML文件创建菜单资源
  - 在res/menu/menu_options.xml
    - 该文件中根节点必须是<menu>元素
    - 子元素可以是<item>或<group>元素
    - 具体节点属性参考：

      http://developer.android.com/guide/topics/resources/menu-resource.html

使用选项菜单和子菜单

- Step2：在Activity中加载菜单资源

```
public boolean onCreateOptionsMenu(Menu menu) {
    //加载菜单资源
    getMenuInflater().inflate(R.menu.menu, menu);
    return super.onCreateOptionsMenu(menu);
}
```

  - 加载XML文件方式的菜单项
  - 若使用Java代码方式加载菜单项，需要调用menu对象的add方法为其依次添加菜单项
  - 具体查看：

    http://developer.android.com/guide/topics/ui/menus.html#options-menu

使用选项菜单和子菜单

- Step3：绑定菜单项选择事件

```
public boolean onOptionsItemSelected(MenuItem item) {
    //依次处理每一个菜单项
    switch (item.getItemId()) {
    case R.id.menu_item1:
        Toast.makeText(getApplicationContext(), "菜单项1", 1000).show();
        return true;
    case R.id.menu_item2:
        startActivity(new Intent());
        return true;
    }
    return super.onOptionsItemSelected(item);
}
```

  - 其它菜单事件监听器，参考：

    http://developer.android.com/reference/android/app/Activity.html#pubmethods

使用选项菜单和子菜单

```
    //创建菜单
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.contextmenu1, menu);
//          menu.add(0,100,1,"菜单1");
//          menu.add(0,200,2,"菜单2");
        return super.onCreateOptionsMenu(menu);
    }
    @Override
//    绑定菜单栏事件
    public boolean onOptionsItemSelected(MenuItem item) {
        switch (item.getItemId()) {
            case R.id.m1:
                Toast.makeText(MainActivity.this, item.getTitle(), Toast.LENGTH_LONG).show();
                break;
            case R.id.m2:
                Toast.makeText(MainActivity.this, item.getTitle(), Toast.LENGTH_LONG).show();
                break;
        }
        return super.onOptionsItemSelected(item);
    }
}
```

使用上下文菜单

- 上下文菜单使用的基本流程
  - 创建菜单项XML布局文件（或由Java代码生成）
  - 在Activity中创建上下文菜单
  - 为视图元素绑定上下文菜单
  - 绑定菜单项选择事件

使用上下文菜单

```
Button btn;
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    btn = (Button) findViewById(R.id.btn1);
    registerForContextMenu(btn);

@Override
public boolean onContextItemSelected(MenuItem item) {
    switch (item.getItemId()) {
        case R.id.m11:
            Toast.makeText(MainActivity.this, "m11", Toast.LENGTH_LONG).show();
            break;
        case R.id.m3:
            Toast.makeText(this,"m3", Toast.LENGTH_LONG).show();
            break;
    }
    return super.onContextItemSelected(item);
}
```

使用上下文菜单

```
@Override
public boolean onContextItemSelected(MenuItem item) {
    switch (item.getItemId()) {
        case R.id.green:
            btn.setBackgroundColor(Color.GREEN);
            return true;
        case R.id.blue:
            btn.setBackgroundColor(Color.BLUE);
            return true;
        case R.id.red:
            btn.setBackgroundColor(Color.RED);
            return true;
    }
    return super.onContextItemSelected(item);
}
```

使用上下文菜单

1、在布局文件创建菜单

2、在onCreateOptionsMenu装载菜单

```
public boolean onCreateOptionsMenu(Menu menu) {
    getMenuInflater().inflate(R.menu.menu1, menu);
    return super.onCreateOptionsMenu(menu);
}
```

3、实现菜单的事件操作