Android事件响应

本章概要

- 事件处理原理
- 常用事件

清華大學 Tsinghua University

Android 的事件处理

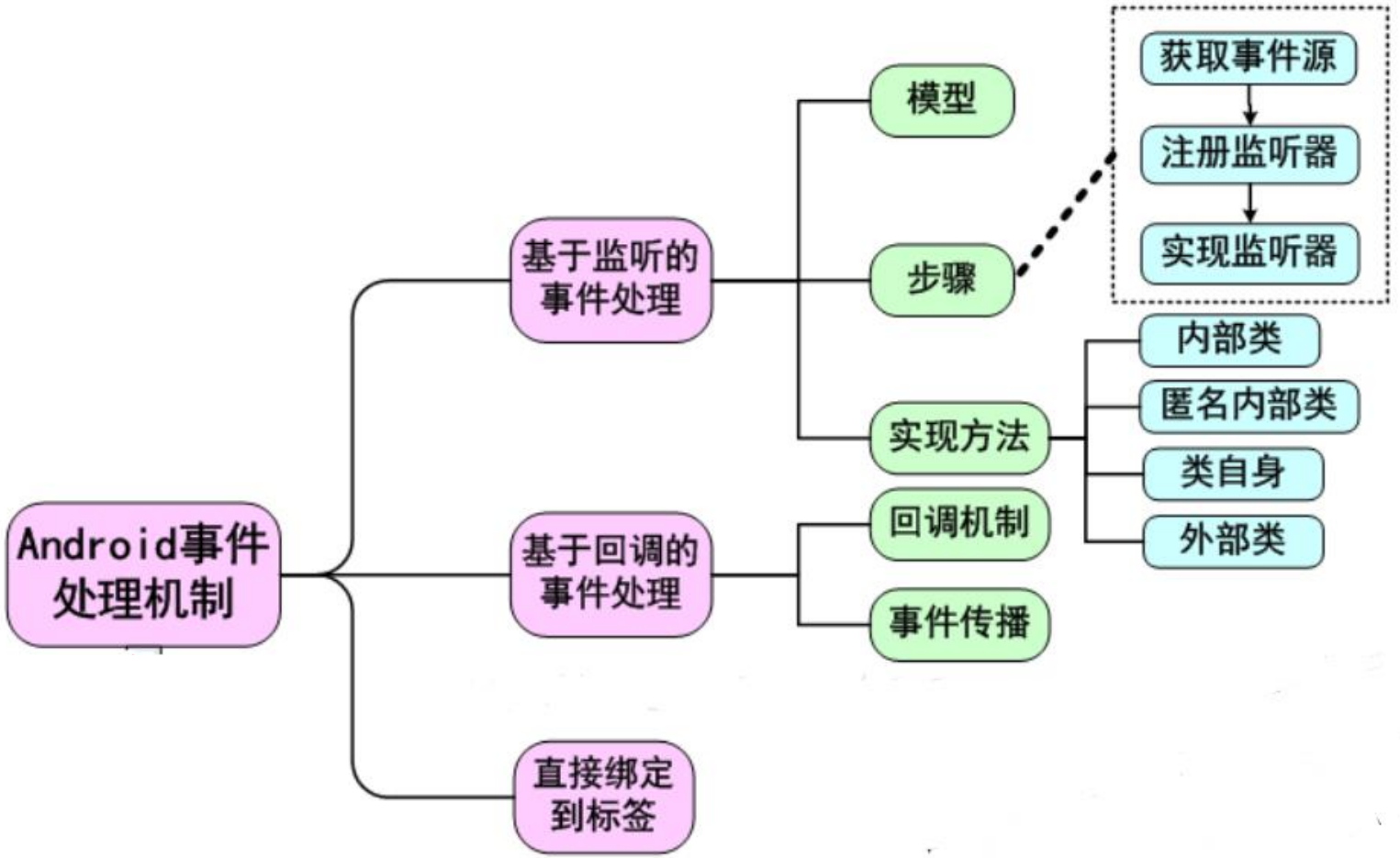
基于监听的事件处理

- 在前面的课程中，我们设计了活动与布局，在屏幕上放置了多个控件
- 如何使这些控件能够与用户进行互动？使用Java 中的事件处理机制
- Android提供了强大的事件处理机制，包括两套事件处理机制：
  1）基于监听的事件处理
  2）基于回调的事件处理
  这里主要介绍基于监听的事件处理
- 每一个事件的触发，都由对应的监听器进行处理

- 什么是事件？用户的操作
  - 单击事件（触屏）
  - 长按事件（长触屏）
  - 键盘事件（点击某个键盘按键）
  - 控件操作

  …

基于监听的事件处理



基于监听的事件处理

◆ 基于监听是一种更“面向对象”的事件处理，主要涉及如下三个对象。

1. **EventSource (事件源)**:

   事件发生的场所，通常就是各个组件，例如窗口、按钮、菜单等

2. **Event (事件)**:

   事件封装了界面组件上发生的特定事情，通常是一次用户操作，如果程序需要获得界面组件上所发生事件的相关信息，一般通过Event对象来取得

3. **EventListener (事件监听器)**:

   负责监听事件源所发生的事件，并对各种事件作出相应的响应。

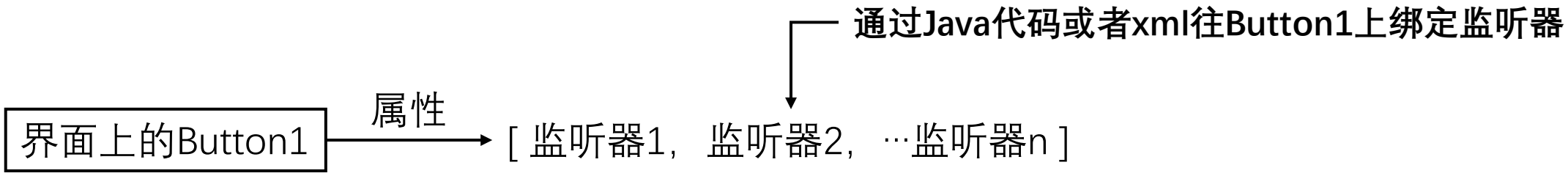
例如，如果用户用触屏单击了按钮对象button，则该按钮button就是事件源，而java运行时系统会生成Event类的对象event，该对象中描述了该单击事件发生时的一些信息，然后，事件处理者对象将接收由java运行时系统传递过来的事件对象event并进行相应的处理

基于监听的事件处理

- **授权处理机制(Delegation Model)**：事件源可以把在其自身所有可能发生的事件分别授权给不同的事件处理者来处理，比如在Canvas对象上既可能发生鼠标事件，也可能发生键盘事件，该Canvas对象就可以授权给事件处理者1来处理鼠标事件，同时授权给事件处理者2来处理键盘事件
- **事件处理者**：监听器，时刻监听着事件源上所有发生的事件类型，一旦该事件类型与自己所负责处理的事件类型一致，就马上进行处理
- 授权模型把事件的处理委托给外部的处理实体进行处理，实现了**将事件源和监听器分开的机制**。事件处理者（监听器）通常是一个类，该类如果要能够处理某种类型的事件，就必须实现与该事件类型相对的接口

基于监听的事件处理

◆每个事件源维护了一个被注册的监听器列表，用于处理事件



◆ 当发生事件时，Button1会遍历它自己的监听器列表，逐一通知它们处理事件

如果列表为空?

◆ 通过这种机制，我们可以往一个事件源上注册多个监听器，也可以用一个监听器监听多个事件源。

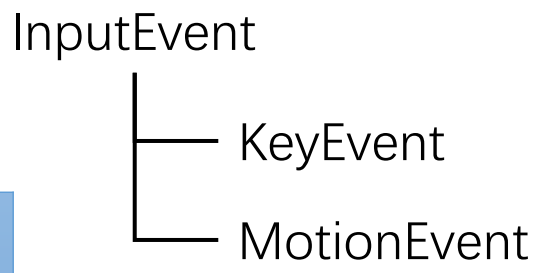
◆ **模块解耦，代码灵活性高**

Android 中的常用事件

◆包 android.view 中包含一个抽象类 InputEvent

public abstract class InputEvent extends Object implements Parcelable

它有两个子类，分别为KeyEvent和MotionEvent



KeyEvent	Object used to report key and button events.
MotionEvent	Object used to report movement (mouse, pen, finger, trackball) events.

更多信息参阅
https://developer.android.com/reference/android/view/InputEvent.html

事件操作流程

事件获取三步骤

通过 id 获取控件

- 为了在Java代码中访问并修改控件的属性，比如添加监听器，更改控件上的文字/样式，我们需要在Java代码中获得这个控件对象。

  前面的内容中，我们使用XML来定义布局中的控件，在Java代码中如何访问它们?

- 控件的 android:id 属性

  这个属性是找到指定界面控件的索引，在XML文件中，我们通过字段android:id来定义

```
<Button
    android:id = "@+id/button1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="按钮1" />
```

  android:id 属性值是一个字符串，由”@+id/”开头，后面跟着的内容即为id的具体内容，在上面的例子中为button1

通过 id 获取控件

- 在XML中引用该控件，使用@id/button1或者@android:id/button1
- 在Java程序代码中，通过R.id.button1来引用该控件
- 在定义好XML中的id之后，我们查看编译生成的R.java文件，可以看见以下的内容

```
public static final class id {
    public static final int action_settings=0x7f080002;
    public static final int button1=0x7f080000;
    public static final int textView1=0x7f080001;
}
```

- Android 提供了一个方法findViewById()，通过控件的数值常量来获得控件对象。

```
Button button1 = (Button) findViewById(R.id.button1);
```

等价于

```
Button button1 = (Button) findViewById(0x7f080000);
```

创建事件处理程序

- 第一种写法：在XML文件中声明onClick属性

在XML文件中显式指定控件的**onClick**属性，点击按钮时会利用反射的方式调用对应Activity中的方法

这种方法在工程中很少使用

创建事件处理程序

◆ 第一种写法：在XML文件中声明onClick属性

XML内容：

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

    <Button
        android:id="@+id/btn1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:onClick="onClick"
        android:text="按钮1" />

    <Button
        android:id="@+id/btn2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:onClick="onClick"
        android:text="按钮2" />

</LinearLayout>
```

创建事件处理程序

- 第一种写法：在XML文件中声明onClick属性

Activity内容：

练一练

```
public class MainActivity extends Activity {
    private Button btn1, btn2;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        initView();
    }
    // 方法：初始化View
    private void initView() {
        btn1 = (Button) findViewById(R.id.btn1);
        btn2 = (Button) findViewById(R.id.btn2);
    }
    //方法：控件View的点击事件
    public void onClick(View v) {
        switch (v.getId()) {
        case R.id.btn1:
            Toast.makeText(MainActivity.this, "btn1", Toast.LENGTH_SHORT).show();
            break;
        case R.id.btn2:
            Toast.makeText(MainActivity.this, "btn2", Toast.LENGTH_SHORT).show();
            break;

        default:
            break;
        }
    }
}
```

参数固定，为触发该事件的View控件对象

switch对不同的触发源采用不同的代码进行运行

创建事件处理程序

- **第二种写法：匿名内部类（**适合场景：测试、或者只有单个button的时候。使用较多**）**

删除android:onClick="onClick"

```
btn1.setOnClickListener(new OnClickListener() {

    @Override
    public void onClick(View v) {
        Toast.makeText(MainActivity.this, "btn1", Toast.LENGTH_SHORT).show();
    }
});
```

创建事件处理程序

◆ 第三种写法：Activity实现View.OnClickListener接口（最常用）

XML内容：

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

    <Button
        android:id="@+id/btn1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="按钮1" />

    <Button
        android:id="@+id/btn2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="按钮2" />

</LinearLayout>
```

删除
android:onClick="onClick"

创建事件处理程序

实现该接口，那么这个Activity对象就是一个监控器

Activity代码

练一练

```
public class MainActivity extends Activity implements OnClickListener{
    private Button btn1, btn2;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        //初始化View
        initView();
    }
    // 方法：初始化View
    private void initView() {
        btn1 = (Button) findViewById(R.id.btn1);
        btn2 = (Button) findViewById(R.id.btn2);

        //按钮绑定点击事件的监听器
         btn1.setOnClickListener(this);
         btn2.setOnClickListener(this);
    }
    //方法：按钮的单击事件
    @Override
    public void onClick(View v) {
        switch (v.getId()) {
        case R.id.btn1:
            Toast.makeText(MainActivity.this, "btn1", Toast.LENGTH_SHORT).show();
            break;
        case R.id.btn2:
            Toast.makeText(MainActivity.this, "btn2", Toast.LENGTH_SHORT).show();
            break;
        default:
            break;
        }
    }
}
```

```
import android.view.View;
import android.view.View.OnClickListener;
```

将自己（也就是这个Activity本身）注册为监控器
初始化View时候就需要注册！

onClick()和方法1一样

按钮事件处理

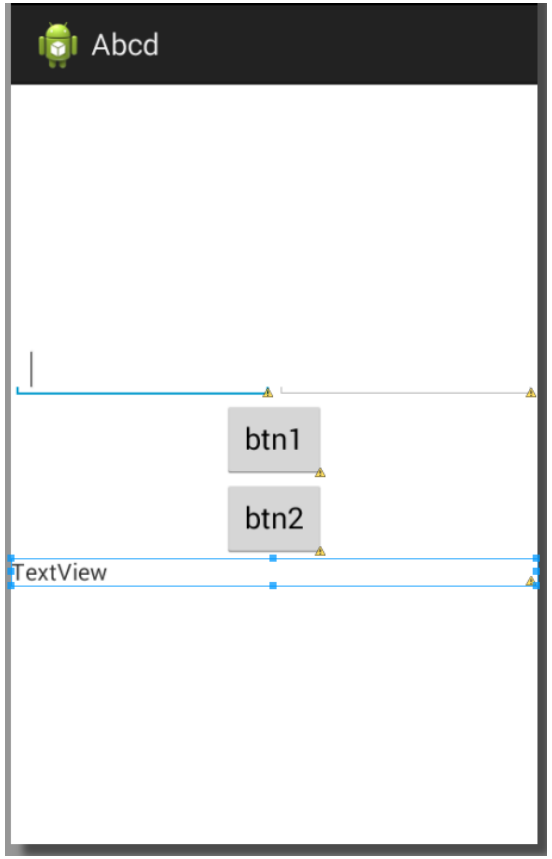
- 长按事件（相当于windows里面的鼠标右键），长按控件超过1s触发。需要实现接口OnLongClickListener，函数为onLongClick(View v)

```
btn1.setOnClickListener(listener);
btn1.setOnLongClickListener(longListener);
View.OnLongClickListener longListener = new View.OnLongClickListener() {
    public boolean onLongClick(View v) {
        label.setText("触发LongClick事件...");
        return false; // 触发LongClick后再触发Click事件
        // return true;  //    只触发LongClick事件
    }
};
// 建立Click事件的监听器对象
View.OnClickListener listener = new View.OnClickListener() {
    public void onClick(View v) {
        label.setText("触发Click事件...");
    }
};
```

课堂练习(1)

1、对按钮事件的响应（setOnClickListener）

2、对按钮长按事件的响应（setOnLongClickListener）




```
        Button  btn1 = (Button)findViewById(R.id.button_event_bt1);

        Button  btn2 = (Button)findViewById(R.id.button_event_bt2);

        btn1.setOnClickListener(this);
        btn2.setOnClickListener(this);

        btn1.setOnLongClickListener(this);
        btn2.setOnLongClickListener(this);


public void onClick(View v) {
    if(v.getId() == R.id.button_event_bt1) {
        Toast.makeText(ButtonEvent.this, "btn1", Toast.LENGTH_SHORT).show();

        EditText  edit_text = (EditText)findViewById(R.id.button_event_edit_text_1);
        TextView  text_view = (TextView)findViewById(R.id.button_event_text_view);
        String str_input = edit_text.getText().toString();
        text_view.setText(str_input);
    }
    if(v.getId() == R.id.button_event_bt2) {
        Toast.makeText(ButtonEvent.this, "btn2", Toast.LENGTH_SHORT).show();

        EditText  edit_text = (EditText)findViewById(R.id.button_event_edit_text_2);
        TextView  text_view = (TextView)findViewById(R.id.button_event_text_view);
        String str_input = edit_text.getText().toString();
        text_view.setText(str_input);
```

更多的事件处理

◆ 键盘事件 KeyDown 和 KeyUp

```
public boolean onKeyDown(int keyCode, KeyEvent event) {
        if (keyCode == KeyEvent.KEYCODE_DEL) {
                Toast.makeText(this, "按下DEL键...", Toast.LENGTH_SHORT).show();
                return true;
        }
        return super.onKeyDown(keyCode, event);
}

public boolean onKeyUp(int keyCode, KeyEvent event) {
        if (keyCode == KeyEvent.KEYCODE_MENU) {
                Toast.makeText(this, "按下MENU键...",  Toast.LENGTH_SHORT).show();
                return true;
        }
        return super.onKeyUp(keyCode, event);
}
```

更多的事件处理

◆ 鼠标事件（触屏） onTouchEvent

```
public boolean onTouchEvent(MotionEvent event) {
    //Log.d(TAG, "aa localapp.PopupWindow.onTouchEvent");
        Toast.makeText(this, "onTouchEvent..." + " " + event,  Toast.LENGTH_SHORT).show();
    //    finish();
    //Log.d(TAG, "aa localapp.PopupWindow.onTouchEvent");
    return super.onTouchEvent(event);
}
```

课堂练习(2)

1、对键盘事件的响应。

- 按回车键，关闭界面。
- 按字母或数字键，在界面上显示。
- 对鼠标事件的响应。

文本编辑事件

找到控件

```
movie_name_filter = (EditText) findViewById(R.id.movie_name_filter);
```

定义事件

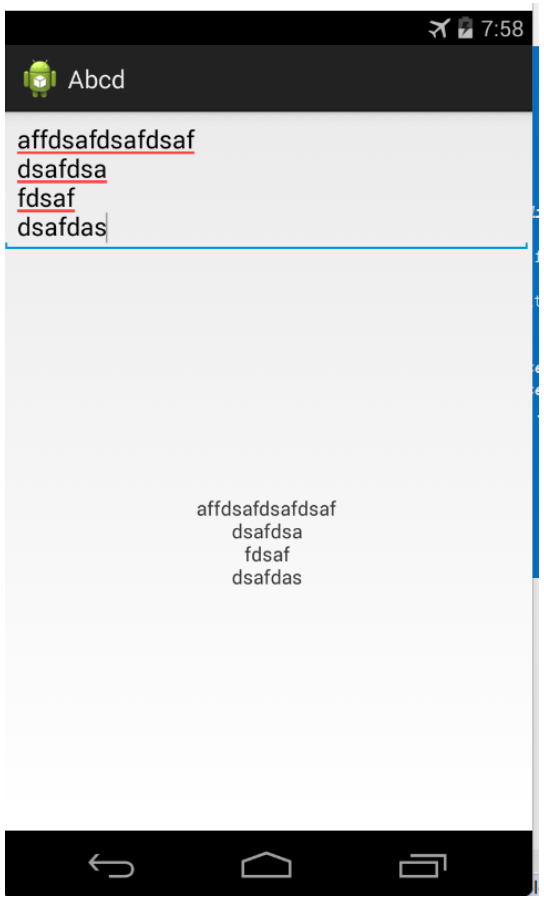
```
private final TextWatcher mTextWatcher = new TextWatcher()      {
    public void beforeTextChanged(CharSequence s, int start, int count, int after)          {
    }

    public void onTextChanged(CharSequence s, int start, int before, int count)          {
    }

    public void afterTextChanged(Editable s) {
        if (s.length() > 0) {
            Toast.makeText(MovieList.this, s.toString(), Toast.LENGTH_SHORT).show();
            Log.d(TAG, s.toString());
      }
};
```

关联事件

```
movie_name_filter.addTextChangedListener(mTextWatcher);
```

课堂练习(3)

3、对输入文本事件的响应




```
public class EditTextTest extends Activity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_edit_text_test);
        initView();
    }

    public void initView() {
        final TextWatcher mTextWatcher = new TextWatcher() {
            public void beforeTextChanged(CharSequence s, int start, int count, int after) {
            }

            public void onTextChanged(CharSequence s, int start, int before, int count) {
            }

            public void afterTextChanged(Editable s) {
                if (s.length() > 0) {
                    TextView view_text = (TextView) findViewById(R.id.view_text_test);
                    view_text.setText(s.toString());
                }
            }
        };
        EditText edit_text = (EditText) findViewById(R.id.edit_text_test);
        edit_text.addTextChangedListener(mTextWatcher);
    }
}
```

listView事件

参考网址

https://www.jianshu.com/p/c79741798fba

填充元素

```
movie_list_view = (ListView) findViewById(R.id.movie_list_view);
adapter =
    new ArrayAdapter(this,android.R.layout.simple_list_item_1, list_movie_name);
movie_list_view.setAdapter(adapter);
```

事件监听

```
movie_list_view.setOnItemClickListener(this);//添加监听
```

事件响应

```
public void onItemClick(AdapterView<?> parent, View v, int position, long id){
    switch (parent.getId()) {
    case R.id.movie_list_view:
        MovieInfo str = adapter.getItem(position);
            Toast.makeText(this, str.getName(), Toast.LENGTH_SHORT).show();
    break;}
    }
```

listView事件

赋值与监听消息

```
ListView list_test_list_view = (ListView) findViewById(R.id.list_test_list_view);

list_name.add("ab");list_name.add("bc");list_name.add("ca");

adapter = new ArrayAdapter<String>(this, android.R.layout.simple_list_item_1);
adapter.addAll(list_name);
list_test_list_view.setAdapter(adapter);
list_test_list_view.setOnItemClickListener(this);// 添加监听
```

接收与处理消息

```
public void onItemClick(AdapterView<?> parent, View v, int position, long id) {
      switch (parent.getId()) {
      case R.id.list_test_list_view:
            TextView list_test_view_text = (TextView) findViewById(R.id.list_test_view_text);
            list_test_view_text.setText(adapter.getItem(position).toString());
            break;
      }
}
```

对列表修改

```
public void filterByText(String s) {
      adapter.clear();
      if (s.isEmpty()) {
            for (String item : list_name) {
                  adapter.add(item);
            }
      } else {
            for (String item : list_name) {
                  if (item.contains(s)) {
                        adapter.add(item);
                  }}}
}
```

课堂练习(4)

4、对listView事件的响应

- 初始化listView。
- 点击listView某一个元素后，更新textView
- 根据editText的内容，过滤更新listView

Spinner 控件

参考地址

http://www.dedeyun.com/it/m/98498.html

填充元素

```
Spinner item_select = (Spinner)findViewById(R.id.item_select);
String [] startArray = {"aa", "bb", "cc"};
arrayAdapter = new ArrayAdapter<String>(this, android.R.layout.simple_dropdown_item_1line,startArray);
item_select.setAdapter(arrayAdapter);
```

事件监听

```
item_select.setOnItemSelectedListener(this);
```

事件响应

```
public void onItemSelected(AdapterView<?> parent, View v, int position, long id) {
    switch(v.getId()) {
    case R.id.item_select:
        Toast.makeText(ComboBoxTest.this, "ComboBoxTest.onItemClick " +
        arrayAdapter.getItem(position).toString(), Toast.LENGTH_SHORT).show();
        break;
    }
}
```

WebView 控件

参考资料

https://blog.csdn.net/huweiliyi/article/details/105892050

加载网址

```
WebView webView = (WebView)findViewById(R.id.web_page_view);
webView.loadUrl("https://www.baidu.com");
```

设置本地加载

```
webView.setWebViewClient(new WebViewClient(){
public boolean shouldOverrideUrlLoading(WebView view, String url) {
            view.loadUrl(url);// 强制在当前 WebView 中加载 url
            return true;
        }});
```

其他配置

```
WebSettings webSettings = webView.getSettings();

 //如果访问的页面中要与Javascript交互，则webview必须设置支持Javascript
 webSettings.setJavaScriptEnabled(true);
//设置自适应屏幕
 webSettings.setUseWideViewPort(true); //将图片调整到适合webview的大小
 webSettings.setLoadWithOverviewMode(false); // 缩放至屏幕的大小
 //缩放操作
webSettings.setSupportZoom(false); //支持缩放，默认为true。是下面那个的前提。
webSettings.setBuiltInZoomControls(true); //设置内置的缩放控件。若为false，则该WebView不可缩放
webSettings.setDisplayZoomControls(false); //隐藏原生的缩放控件
```

WebView 控件


```
@Override
public void onClick(View parent) {
// TODO Auto-generated method stub

switch(parent.getId()) {
case R.id.web_page_select:
EditText editText =
(EditText)findViewById(R.id.edit_web_page_name);
String page_url = editText.getText().toString().trim();
if(!page_url.startsWith("https://") && !
page_url.startsWith("http://")) {
page_url = "https://" + page_url;
}
WebView webView = (WebView)findViewById(R.id.web_page_view);
webView.loadUrl(page_url);
break;
case R.id.web_page_quit:
finish();
break;
}

}
```