概述

- 疯狂Android讲义（92页-168页）
    1. TextView的botton及其剩余部分
    2. ImageView组件
    3. AdapterView组件
    4. ProgressBar组件
    5. ViewAnimator组件
    6. 各种杂项组件
    7. 对话框组件

疯狂Android第二章

TextView的Button部分

9patch图片做背景

- 背景：当按钮的内容太多时，Android会自动缩放整张图片，以保证背景图片能覆盖整个按钮。但这种缩放整张图片的效果可能并不好。
- 作用：可能存在的情况是我们只想缩放图片中某个部分，这样才能保证按钮的视觉效果。比如聊天框

单选钮(RadioButton) 和复选框(CheckBox)

- 单选钮(RadioButton)、复选框(CheckBox)是用户界面中最普通的UI组件，它们都**继承了Button类**，因此都可直接使用Button支持的各种属
  性和方法。
- RadioButton、CheckBox与普通按钮的不同：它们多了一个可选中的功能，因此RadioButton、CheckBox都可额外指定一个android:checked属性，该属性用于指定RadioButton、CheckBox 初始时是否被选中。
- RadioButton与CheckBox 的不同：一组RadioButton 只能选中其中一个，因此RadioButton通常要与RadioGroup一起使用，用于定义一组单选钮。
- 委托式事件处理机制的原理：当事件源上发生事件时，该事件将会激发该事件源上的监听器的特定方法。
- 注意：如果在XML布局文件中默认选中了某个单选钮，则必须为该组单选钮的每个按钮都指定android:id 属性值;否则，这组单选钮不能正常工作。

状态开关按钮(ToggleButton) 和开关(Switch)

- 状态开关按钮(ToggleButton) 和开关(Switch)也是由Button派生出来的，因此它们的本质也是按钮，Button 支持的各种属性、方法也适用于ToggleButton 和Switch。
-

	CheckBox	**ToggleButton、Switch**
相同	都可以提供两种状态	都可以提供两种状态
不同	只是自身是否被选中	切换程序的状态

时钟(AnalogClock 和TextClock)

- TextClock 本身就继承了TextView也就是说， 它本身就是文本框，只是它里面显示的内容总是当前时间。
    - 与TextView不同的是，为TextClock设置android:text属性没什么作用。
    - TextClock 能以24小时制或12小时制来显示时间，而且可以由程序员来指定时间格式。
- AnalogClock则继承了View 组件，它重写了View的OnDraw(方法，它会在View上绘制模拟时钟。
- TextClock和AnalogClock的不同
    - TextClock 显示数字时钟，可以显示当前的秒数;
    - AnalogClock 显示模拟时钟，不会显示当前的秒数。

计时器(Chronometer)

- Chronometer组件与TextClock都继承自TextView,因此它们都会显示一段文本。但Chronometer并不显示当前时间，它显示的是从某个起始时间开始，一共过去了多长时间。

ImageView 及其子类

- ImageView继承自View组件，不仅能显示图片，任何Drawable 对象都可使用ImageView来显示。
- ImageView 派生了ImageButton、ZoomButton、 FloatingActionButton 等组件
- 能够设置图片的最大高度，最大长度，缩放，src路径
- **ImageButton**:图片按钮。
    - Button与ImageButton的区别：Button 生成的按钮上显示文字，而ImageButton上则显示图片。
    - 为ImageButton 按钮指定android:text 属性没用(ImageButton 的本质是ImageView),即使指定了该属性，图片按钮上也不会显示任何文字。
    - 使用ImageButton,图片按钮可以指定android:src属性，该属性既可使用静止的图片，也可使用自定义的Drawable对象，这样即可开发出随用户动作改变图片的按钮。
- **QuickContactBadge**:显示关联到特定联系人的图片。
- **ZoomButton**：ZoomButton 可以代表“放大”和“缩小”两个按钮。
    - ZoomButton的行为基本类似于ImageButton,只是Android默认提供了btn minus、 btn plus两个Drawable资源，只要为ZoomButton的android:src 属性分别指定btn minus、 btn plus,即可实现“缩小”和“放大”按钮。
    - Android还提供了一个ZoomControls组件，该组件相当于同时组合了“放大”和“缩小”两个按钮，并允许分别为两个按钮绑定不同的事件监听器。
- **FloatingActionButton**：它代表一个 悬浮按钮。该按钮默认是 一个带默认填充色的圆形按钮，当用户单击该按钮时，该按钮可以显示-一个波纹效果。
    - FloatingActionButton 位于support库中，因此必须先添加然后才能使用。可以直接修改对应模块的build. gradle文件来添加FloatingActionButton

AdapterView及其子类

- AdapterView 本身是一个抽象基类，它派生的子类在用法上十分相似，只是显示界面有一定的区别
- 特征
    - AdapterView 继承了ViewGroup, 它的本质是**容器。**
    - AdapterView可以包括多个“列表项”，并将多个“列表项”以合适的形式显示出来。
    - AdapterView显示的多个“列表项”由Adapter提供。调用AdapterView的setAdapter(Adapter)方法设置Adapter即可。

Adapter接口及其实现类

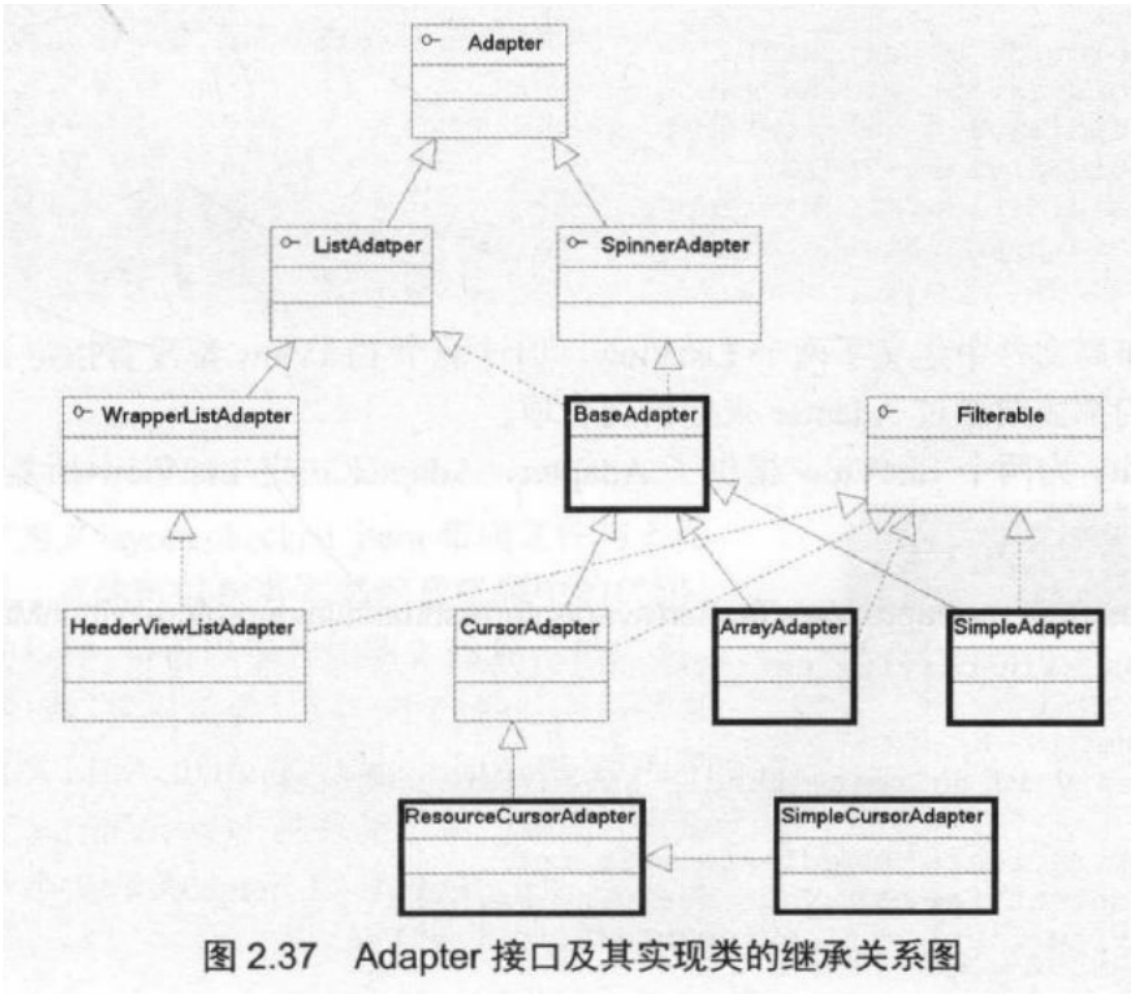
- Adapter本身只是一个接口，它派生了ListAdapter 和SpinnerAdapter 两个子接口，其中ListAdapter为AbsListView**提供列表项**，而SpinnerAdapter为AbsSpinner提供列表项。几乎所有的Adapter都继承了BaseAdapter, 而BaseAdapter同时实现了ListAdapter、SpinnerAdapter两个接口，因此BaseAdapter及其子类可以同时为AbsListView、AbsSpinner 提供列表项。
- 

图 2.37　Adapter 接口及其实现类的继承关系图
  - 常用的实现类
    - **ArrayAdapter**: 通常用于将数组或List集合的多个值包装成多个列表项。
      - new ArrayAdapter（ ）时必须指定如下三个参数。
        1. Context: 这个参数无须多说，它代表了访问整个Android应用的接口。几乎创建所有组件
           都需要传入Context对象。
        2. textViewResourceId:一个资源 ID,该资源ID代表一个TextView,该TextView组件将作
           为ArrayAdapter的列表项组件。
        3. 数组或List:该数组或List将负责为多个列表项提供数据。
      - 如果程序的窗口仅仅需要显示一个列表，则可以直接让Activity继承ListActivity 来实现，ListActivity 的子类无须调用setContentView()方法来显示某个界面，而是可以直接传入一个内容Adapter, ListActivity的子类就呈现出一个列表。
    - **SimpleAdapter**: 可用于将List集合的多个对象包装成多个列表项。
      - new SimpleAdapter（ ）时必须指定如下五个参数
        1. Context
        2. 该参数应该是一个`List<? extends Map<String, ?>>`类型的集合对象，该集合中每个`Map<String, ?>`对象生成一一个列表项。

3. 该参数指定一个界面布局的ID。
4. 该参数应该是一个String[]类型的参数，该参数决定提取`Map<String, ?>`对象中哪些key对应的value来生成列表项。
5. 该参数应该是一个int[]类型的参数，该参数决定填充哪些组件。

- 问题：提供的列表项的value怎么和组件相对应？
  - 猜测按两个数组对应的下标进行匹配（待验证）

- **SimpleCursorAdapter**: 与SimpleAdapter基本相似，只是用于包装Cursor提供的数据。
- **BaseAdapter**: 通常用于被扩展。扩展BaseAdapter可以对各列表项进行最大限度的定制。

自动完成文本框(AutoComplete TextView)

- 自动完成文本框(AutoCompleteTextView)**从EditText派生**而出，实际上它也是一个文本编辑框，但它比普通编辑框多了一个功能:当用户输入一定字符之后，自动完成文本框会显示一个下拉菜单，供用户从中选择，当用户选择某个菜单项之后，AutoCompleteTextView按用户选择自动填写该文本框。
  - 使用：setAdapter（ ）只要为它设置-一个Adapter即可，该Adapter 封装了AutoCompleteTextView预设的提示文本。
- AutoCompleteTextView还派生了一个子类: MultiAutoCompleteTextView, MultiAutoCompleteTextView允许输入多个提示项，多个提示项以分隔符分隔。MultiAutoCompleteTextView 提供了setTokenizer( ) 方法来设置分隔符。

可展开的列表组件(ExpandableListView)

- ExpandableListView是ListView 的子类，它在普通ListView 的基础上进行了扩展，它把应用中的列表项分为几组，每组里又可包含多个列表项。
  - ExpandableListView的用法与普通ListView的用法非常相似，只是ExpandableListView所显示的列表项应该由ExpandableListAdapter提供。ExpandableListAdapter也是一一个接口
  - 实现ExpandableListAdapter
    - 扩展BaseExpandableListAdapter实现ExpandableListAdapter.
    - 使用SimpleExpandableListAdapter将两个List集合包装成ExpandableListAdapter.
    - 使用SimpleCursorTeeAdapter将Cursor中的数据包装成SimpleCursorTreeAdapter.

Spinner

- 弹出(下拉)一个菜单供用户选择。
- Spinner与Gallery都继承了AbsSpinner, AbsSpinner 继承了AdapterView, 因此它也表现出AdapterView的特征:只要为AdapterView提供Adapter即可。Gallery和Spinner都是一个列表选择框。
  - Spinner与Gallery区别
    1. Spinner 显示的是一个垂直的列表选择框;而Gallery显示的是一个水平的列表选择框。
    2. Spinner 的作用是供用户选择;而Gallery则允许用户通过拖动来查看上一个、下一个列表项。
- Spinner与ListView的区别：Spinner可以以弹出或下拉方式显示列表选择框。

AdapterViewFlipper

- AdapterViewFilpper继承了AdapterViewAnimator, 它也会显示Adapter提供的多个View组件，但它**每次只能显示一个View组件**，程序可通过showPrevious()和showNext()方法控制该组件显示上一个、下一个组件。
- AdapterViewFilpper可以在多个View 切换过程中使用渐隐渐显的动画效果。除此之外，还可以调用该组件的startFlipping()控制它"自动播放"下一个View组件。

StackView

- StackView也是AdapterViewAnimator的子类，它也用于显示Adapter 提供的一系列View。StackView将会以"堆叠(Stack)"的方式来显示多个列表项。
- 控制方式
    - 拖走StackView中处于顶端的View, 下一个View 将会显示出来。将上一个View 拖进StackView,将使之显示出来。
    - 通过调用StackView的showNext()、showPrevious()控制显示下一个、 上一个组件。

RecyclerView

- RecyclerView的用法与ListView相似，同样使用Adapter来生成列表项
- RecyclerView需要使用改进的RecyclerView.Adapter,改进后RecyclerView.Adapter 只要实现三个方法。
    - onCreateViewHolder(ViewGroup viewGroup, int i):该方法用于创建列表项组件。使用该方法所创建的组件会被自动缓存。
    - onBindViewHolder(ViewHolder viewHolder, int i):该方法负责为列表项组件绑定数据，每次组件重新显示出来时都会重新执行该方法。
    - gettemCount():该方法的返回值决定包含多少个列表项。

ProgressBar及其子类

- ProgressBar 本身代表了进度条组件，它还派生了两个常用的组件: SeekBar 和RatingBar。ProgressBar 及其子类在用法上十分相似，只是显示界面有一定的区别

进度条(ProgressBar)

- 通常用于向用户显示某个耗时操作完成的百分比。进度条可以动态地显示进度，因此避免长时间地执行某个耗时操作时，让用户感觉程序失去了响应，从而更好地提高用户界面的友好性。
- 通过style为ProgressBar指定风格，可以使用水平和环形进度条

拖动条(SeekBar)

- 拖动条和进度条的区别：进度条采用颜色填充来表明进度完成的程度，而拖动条则通过滑块的位置来标识数值一而且拖动条允许用户拖动滑块来改变值
- 使用场景：对系统的某种数值进行调节，比如调节音量等。
- 改变滑块的属性
    - android:thumb:指定一个Drawable对象，该对象将作为自定义滑块。
    - android:tickMark:指定一个Drawable对象，该对象将作为自定义刻度图标。

星级评分条(RatingBar)

- 星级评分条与拖动条的用法、功能都十分接近:它们都允许用户通过拖动来改变进度。
- RatingBar 与SeekBar的区别: RatingBar 通过星星来表示进度。

ViewAnimator及其子类

- ViewAnimator 是一个基类，它继承了FrameLayout,因此它表现出FrameLayout的特征，可以将多个View组件“叠”在一起。ViewAnimator可以在View切换时表现出动画效果。
    - **FrameLayout**
        - **ViewAnimator**
            - **ViewSwitcher**：当程序控制从一个View切换到另一一个View时，ViewSwitcher支持指定动画效果。
                - 使用：为了给ViewSwitcher添加多个组件，通过调用ViewSwitcher 的setFactory(ViewSwitcher.ViewFactory)方法为之设置ViewFactory,并由该ViewFactory为之创建View即可。
                - **ImageSwitcher**
                    - ImageSwitcher继承了ViewSwitcher, 并重写了ViewSwitcher 的showNext()、showPreviousQ方法，因此ImageSwitcher使用起来更加简单。
                    - 使用：
                        1. 为ImageSwitcher提供一个ViewFactory,该ViewFactory生成的View组件必须是**ImageView**。
                        2. 需要切换图片时，只要调用ImageSwitcher 的setImageDrawable(Drawable drawable)、setImageResource(int resid)和setImageURI(Uri uri)方法更换图片即可。
                - **TextSwitcher**
                    - 与ImageSwitcher相似，使用TextSwitcher也需要设置一个ViewFactory.但是TextSwitcher 所需的ViewFactory 的makeView0方法必须返回一个**TextView**组件。
            - **ViewFlipper**
                - ViewFlipper与前面介绍的AdapterViewFlipper 有较大的相似性，它们可以控制组件切换的动画效果。
                - ViewFlipper和AdapterViewFlipper 的区别：ViewFlipper 需要开发者通过addView(View v)添 加多个View, 而AdapterViewFlipper则只要传入一个Adapter, Adapter 将会负责提供多个View。
                - ViewFlipper可以指定与AdapterViewFlipper相同的XML属性。

各种杂项组件

消息（Toast）组件

- Toast是一种非常方便的提示消息框，它会在程序界面上显示一个简单的提示信息。
    - 特点
        - Toast提示信息不会获得焦点。
        - Toast提示信息过一段时间会自动消失。
    - 流程
        1. 调用Toast的构造器或makeText()静态方法创建一个Toast对象。

2. 调用Toast的方法来设置该消息提示的对齐方式、页边距等。
3. 调用Toast的show0方法将它显示出来。

日历视图(CalendarView) 组件

- 日历视图(CalendarView)可用于显示和选择日期，用户既可选择一个日期，也可通过触摸来滚动日历。
- 监听：如果希望监控该组件的日期改变，则可调用CalendarView的setOnDateChangeListener()方法为此组件的点击事件添加事件监听器。

日期、时间选择器(DatePicker 和TimePicker)

- DatePicker和TimePicker都从FrameLayout 派生而来，其中DatePicker供用户选择日期;而TimePicker则供用户选择时间。
- DatePicker和TimePicker在FrameLayout的基础上提供了一些方法来获取当前用户所选择的日期、时间
- 监听：如果程序需要获取用户选择的日期、时间，则可通过为DatePicker 添加OnDateChangedListener进行监听、为TimePicker添加OnTimerChangedListener进行监听

数值选择器(NumberPicker)

- 数值选择器用于让用户输入数值，用户既可以通过键盘输入数值，也可以通过拖动来选择数值。
-

方法	说明
setMinValue(int minVal)	设置该组件支持的最小值
setMaxValue(int maxVal)	设置该组件支持的最大值
setValue(int value)	设置该组件的当前值

搜索框(SearchView)

- SearchView是搜索框组件，它可以让用户在文本框内输入文字，并允许通过监听器监控用户输入，当用户输入完成后提交搜索时，也可通过监听器执行实际的搜索。
-

方法	说明
setIconifiedByDefault (boolean iconified)	设置该搜索框默认是否自动缩小为图标。
setSubmitButtonEnabled(boolean enabled)	设置是否显示搜索按钮。
setQueryHint(CharSequence hint)	设置搜索框内默认显示的提示文本。
setOnQueryTextListener (SearchView .OnQueryTextListener listener)	为该搜索框设置事件监听器。

- 提供结果：如果为SearchView增加一个配套的ListView,则可以为SearchView提供搜索的结果

滚动视图(ScrollView)

- SrollView**由FrameLayout派生**而出，用于为普通组件添加滚动条的组件。ScrollView里**最多只能包含一个组件**，而ScrollView的作用就是**为该组件添加垂直滚动条**。
- **HorizontalScrollView**：**为组件添加水平滚动条**
- 水平滚动条和垂直滚动条可以同时使用

通知Channel

- Notification是显示在手机状态栏的通知。Notification 所代表的是一种具有全局效果的通知，程序一般通过NotificationManager服务来发送Notification。
- Notification.Builder 类，可以轻松地创建Notification对象。常用方法：

方法	说明
setDefaults()	设置通知LED灯、音乐、振动等。
setAutoCancel()	设置点击通知后，状态栏自动删除通知。
setContentTitle()	设置通知标题。
setContentText()	设置通知内容。
setSmallIcon()	为通知设置图标。
setLargeIcon( )	为通知设置大图标。
setTick()	设置通知在状态栏的提示文本。
setContentIntent()	设置点击通知后将要启动的程序组件对应的PendingIntent。

- Android 8加入了通知Channel帮助用户来统一管理通知，开发者可以为不同类型的通知创建同一个通知Channel,而用户则可通过该Channel统一管 理这些通知的行为一所有使用同一个Channel的通知都具有相同的行为。
  - 重要性、闪光灯、声音、震动、锁屏、免打扰
- Android 9增强了通知参与者的支持和消息支持更丰富的数据
- 使用流程
  1. 调用getSystemService(NOTIFICATION_ SERVICE)方 法获取系统的NotificationManager 服务。
  2. 创建NotificationChannel对象，并在NotificationManager 上创建该Channel对象。
  3. 通过构造器创建一个Notification. Builder对象。
  4. 为Notification.Builder设置通知的各种属性。
  5. 创建MessagingStyle和Message,通过Message设置消息内容，为Notification. Builder设置MessagingStyle后创建Notification。
  6. 通过NotificationManager发送Notification。

对话框Dialog

- 分类
  - AlertDialog:功能最丰富、实际应用最广的对话框。
    - ProgressDialog:进度对话框，这种对话框只是对进度条的包装。
    - DatePickerDialog:日期选择对话框，这种对话框只是对DatePicker的包装。
    - TimePickerDialog:时间选择对话框，这种对话框只是对TimePicker的包装。

AlertDialog

- 分区
  - 标题
    - 图标区
    - 文字区
  - 内容区
  - 按钮区
    - 取消
    - 确认
- 使用流程
  1. 创建AlertDialog. Builder对象。
  2. 调用AlertDialog. Builder的setTitle()或 setCustomTitle()方法设置标题。
  3. 调用AlertDialog Builder的setIcon()方法设置图标。
  4. 调用AlertDialog Builder的相关设置方法设置对话框内容。

方法	说明
setMessage()	设置对话内容为简单文本。
setItems()	设置对话框内容为简单列表项。
setSingleChoiceItems()	设置对话框内容为单选列表项。
setMultiChoiceItems()	设置对话框内容为多选列表项。
setAdapter()	设置对话框内容为自定义列表项。
setView()	设置对话框内容为自定义View。
  5. 调用AlertDialog.Builder 的setPositiveButton()、setNegativeButton()或 setNeutralButton()方法添加多个按钮。
  6. 调用AlertDialog. Builder的create()方法创建AlertDialog 对象，再调用AlertDialog 对象的show()方法将该对话框显示出来。