西南交通大學
Southwest Jiaotong University

移动商务应用开发

第2章 Android控件与页面交互

学习内容

2.1 界面结构概述

2.2 常见UI控件

2.3 消息框和对话框

2.4 基于监听的事件处理

学习目标

了解

- Android界面结构

掌握

- 常见UI控件的使用
- 消息框和对话框的使用
- 基于监听的事件处理

学习内容

2.1 界面结构概述

2.2 常见UI控件

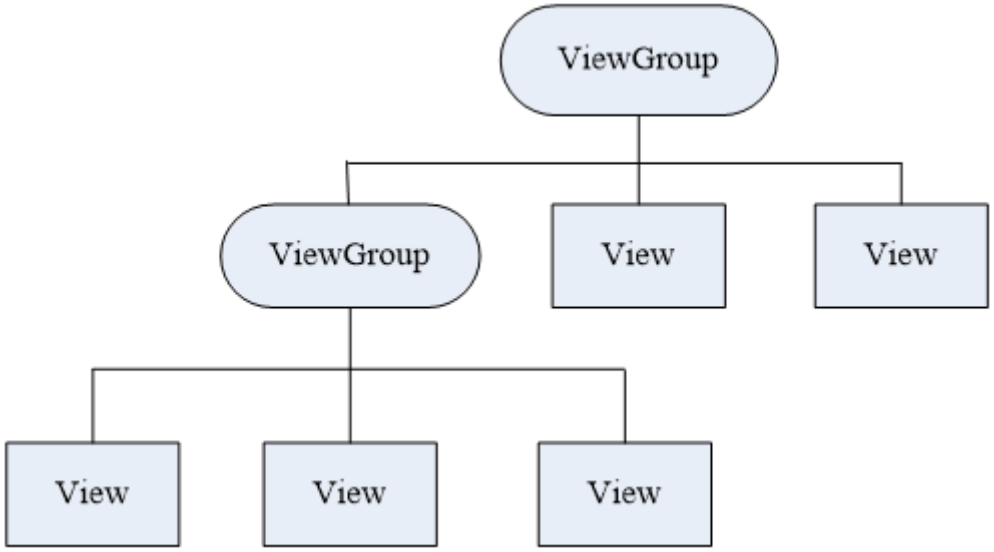
2.3 消息框和对话框

2.4 基于监听的事件处理

主要内容

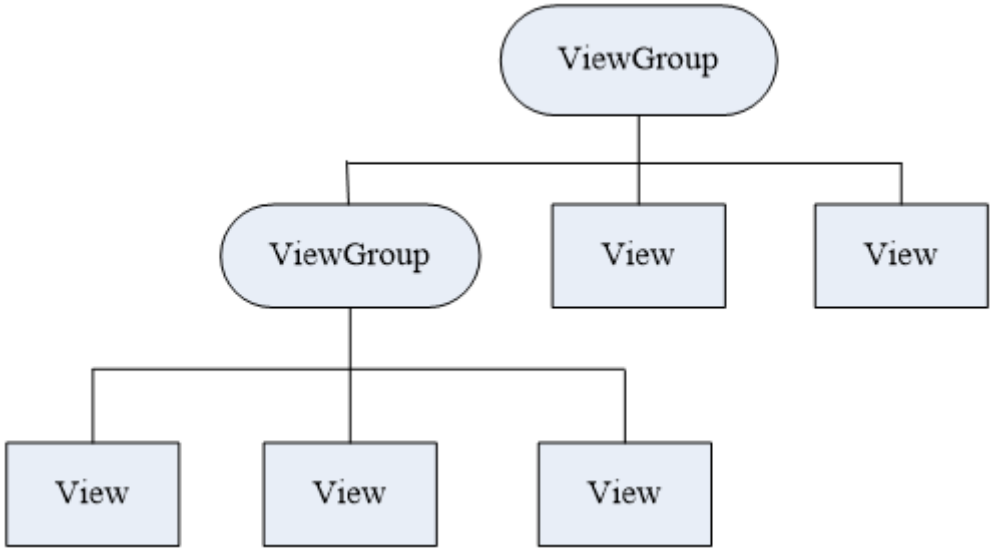
界面结构概述

- Android页面中的所有元素均使用View和ViewGroup对象的层次结构进行构建。
  - view对象通常称为"UI 控件"，用户可查看并进行交互。
  - viewGroup对象通常称为“容器”或“布局”，用户不可见



界面结构概述

- 一个界面布局文件有且仅有一个viewGroup对象作为根节点。
- 一个viewGroup对象中可以存放多个view对象和viewGroup对象。
- Android应用的绝大多数U控件都是放在android.widget、android.view包及其子包中。



界面结构概述

- Android开发讲究逻辑层和视图层相分离的原则，所以推荐使用xml布局文件的方法来控制页面布局。
- 布局文件放在res/layout目录下。
- 可以通过`setContentView(R.layout.布局名)`的方式指定应用中显示的布局。
- 可以通过`findViewById(R.id.控件ID)`的方式来获取布局文件中的控件。

学习内容

2.1 界面结构概述

2.2 常见UI控件

2.3 消息框和对话框

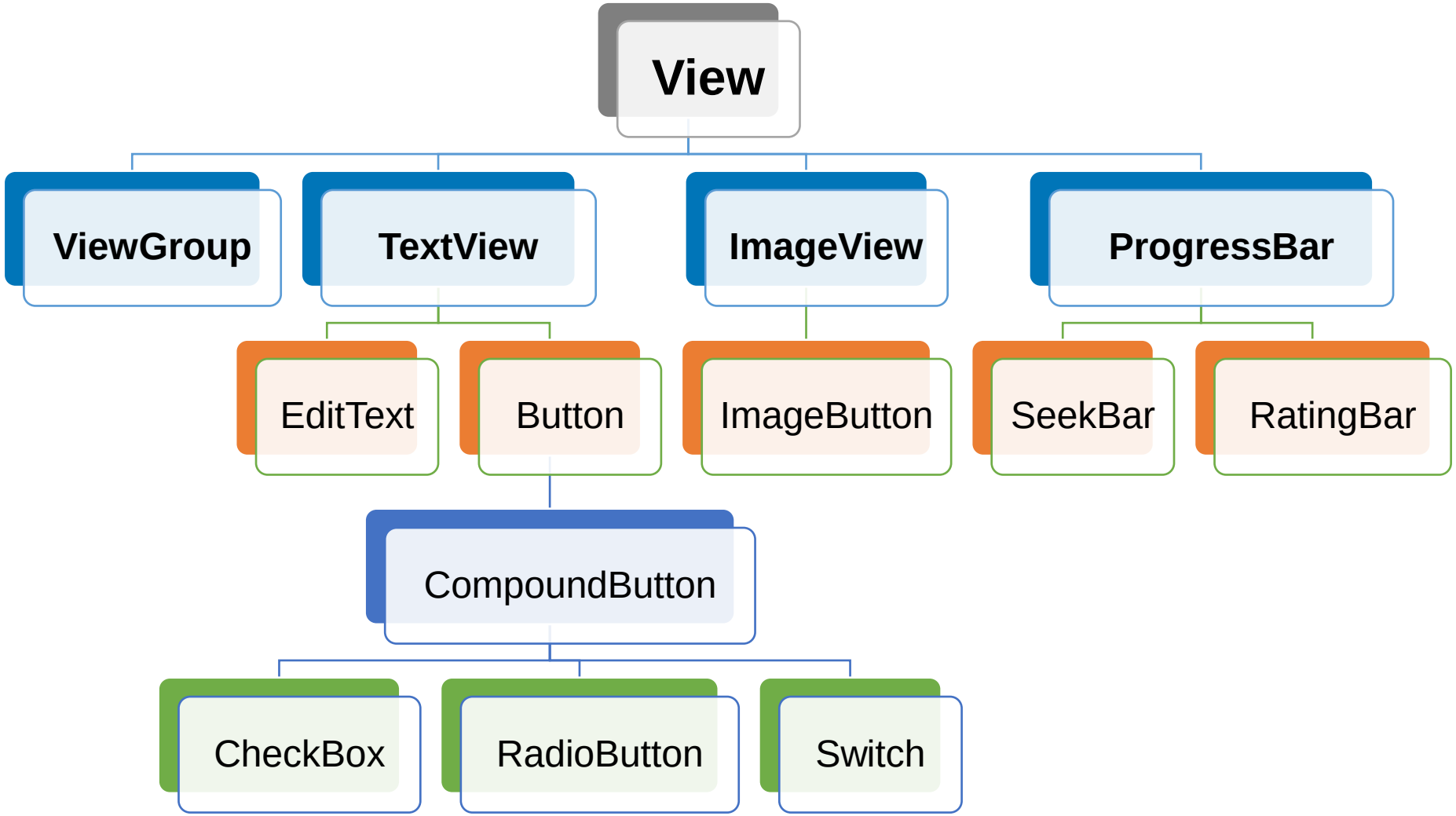
2.4 基于监听的事件处理

主要内容

常见UI控件

常见UI控件	功能描述
TextView	显示文本
EditView	输入文本
Button	按钮
Switch	开关
RadioButton	单选按钮
CheckBox	多选框
ImageView	显示图片
ImageButton	图片按钮
ProgressBar	进度条
SeekBar	拖动条
RatingBar	评分

常见UI控件


View
ViewGroup
TextView
ImageView
ProgressBar
EditText
Button
ImageButton
SeekBar
RatingBar
CompoundButton
CheckBox
RadioButton
Switch

TextView

- 绝对尺寸
- match_parent (或者0dp)
- wrap_content

➢功能：显示文本。

属性	功能描述
android:id	设置控件的唯一id
android:layout_width	设置控件宽度
android:layout_height	设置控件高度
android:text	设置文本内容
android:textColor	设置文本颜色
android:textSize	设置文本大小
android:textStyle	设置文本样式
android:textAlignment	设置文本对齐方式
android:background	设置背景
android:backgroundTint	设置背景颜色渲染
android:backgroundTintMode	设置背景颜色渲染的模式
android:layout_margin	设置控件外边距
android:padding	设置控件内边距

TextView

```
<TextView
      android:id="@+id/textView6"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:background="#D7D7D1"
      android:text="欢迎界面"
      android:textAlignment="center"
      android:textColor="@color/design_default_color_primary"
      android:textSize="24sp" />
```

备注：本节课PPT中示例代码不是完整代码，省略了layout_constraint部分。

EditView

- 功能：输入文本
- EditText是TextView的子类，所以TextView的属性也适用于EditText。
- 其他属性：

- text：普通文本
- textPassword：密码
- textEmailAddress：邮箱
- number：数字
- 其他

属性	功能描述
android:hint	设置控件输入为空时显示的提示文本
android:textColorHint	设置控件输入为空时显示的提示文本的颜色
android:ems	设置显示的最大长度
android:inputType	设置输入文本的类型
android:singleLine	设置为一行

EditView

```
<EditText
    android:id="@+id/editTextText2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:ems="10"
    android:hint="请输入用户名"
    android:inputType="text"
    android:singleLine="true"
    android:textAlignment="center"
    android:textColorHint="#009688"
    android:textSize="20sp"/>
```

Button

- 功能：按钮
- Button是TextView的子类，所以TextView的属性也适用于Button。

```
<Button
      android:id="@+id/button"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:backgroundTint="#018786"
      android:text="按钮"
      android:textSize="20sp"
      android:textStyle="bold" />
```

Switch

- 功能：开关按钮
- Switch是Button的子类，所以也具有Button的属性。
- 其他属性：

属性	功能描述
android:checked	设置是否打开

```
<Switch
      android:id="@+id/switch1"
      android:layout_width="wrap_content"
      android:layout_height="0dp"
      android:checked="true"
      android:text="开关"
      android:textSize="20sp" />
```

CheckBox

- 功能：多选框
- CheckBox是Button的子类，也具有Button的属性
- 其他属性：

属性	功能描述
android:checked	设置是否选中

CheckBox

```
<CheckBox
    android:id="@+id/checkBox"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:checked="true"
    android:text="语文"/>

<CheckBox
    android:id="@+id/checkBox5"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="数学"/>

<CheckBox
    android:id="@+id/checkBox6"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:checked="true"
    android:text="英语" />
```

RadioButton

- 功能：单选按钮
  - 多个RadioButton要放在RadioGroup中才能实现单选效果。
  - RadioGroup继承ViewGroup，是不可见容器。
- RadioButton是Button的子类，也具有Button的属性
- 其他属性：

属性	功能描述
android:checked	设置是否选中

RadioButton

```
<RadioGroup
     android:layout_width="wrap_content"
     android:layout_height="wrap_content"
     android:orientation="horizontal">

     <RadioButton
          android:id="@+id/radioButton"
          android:layout_width="wrap_content"
          android:layout_height="wrap_content"
          android:layout_marginEnd="20dp"
          android:checked="true"
          android:text="男" />

     <RadioButton
          android:id="@+id/radioButton3"
          android:layout_width="wrap_content"
          android:layout_height="wrap_content"
          android:layout_marginStart="20dp"
          android:text="女" />
 </RadioGroup>
```

ImageView

- 功能：显示图片

属性	功能描述
android:id	设置控件的唯一id
android:layout_width	设置控件宽度
android:layout_height	设置控件高度
android:srcCompat	设置显示的图片
android:scaleType	设置所显示的图片的缩放类型
android:alpha	设置透明度
android:background	设置背景
android:backgroundTint	设置背景颜色渲染
android:backgroundTintMode	设置背景颜色渲染的模式
android:layout_margin	设置控件外边距
android:padding	设置控件内边距

ImageView

```
<ImageView
      android:id="@+id/imageView2"
      android:layout_width="0dp"
      android:layout_height="0dp"
      android:scaleType="fitCenter"
      app:layout_constraintDimensionRatio="1:1"
      app:srcCompat="@drawable/boy" />
```

ImageButton

- 功能：图片按钮
- ImageButton是ImageView的子类，也具有ImageView的属性

```
<ImageButton
      android:id="@+id/imageButton"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      app:srcCompat="@android:drawable/ic_delete" />
```

ProgressBar

- 功能：进度条

属性	功能描述
android:id	设置控件的唯一id
android:layout_width	设置控件宽度
android:layout_height	设置控件高度
style	设置进度条类型
android:max	指定进度条的最大进度值
android:progress	指定进度条的当前进度值
android:indeterminate	设置进度条的进度值是否为不确定状态
android:scaleX	设置宽度缩放比例
android:scaleY	设置高度缩放比例
android:layout_margin	设置控件外边距
android:padding	设置控件内边距

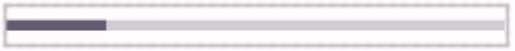
ProgressBar

水平进度条

```
<ProgressBar
    android:id="@+id/progressBar2"
    style="?android:attr/progressBarStyleHorizontal"
    android:layout_width="200dp"
    android:layout_height="wrap_content"
    android:layout_marginStart="24dp"
    android:layout_marginEnd="24dp"
    android:max="100"
    android:progress="20"
    app:layout_constraintHorizontal_bias="0.5" />
```

圆形进度条

```
<ProgressBar
    android:id="@+id/progressBar3"
    style="?android:attr/progressBarStyle"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:indeterminate="true" />
```



SeekBar

- 功能：拖动条，用户可以拖动。
- SeekBar是ProgressBar的子类，所以也具有ProgressBar的属性。

```
<SeekBar
    android:id="@+id/seekBar"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_marginStart="24dp"
    android:layout_marginEnd="24dp"
    android:max="100"
    android:progress="20" />
```

RatingBar

- 功能：评分，用户可以打分。
- RatingBar是ProgressBar的子类，所以也具有ProgressBar的属性。
- 其他属性：

属性	功能描述
android:rating	设置当前分值

```
<RatingBar
      android:id="@+id/ratingBar2"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:rating="3.5"
      android:scaleX="0.8"
      android:scaleY="0.8" />
```

学习内容

2.1 界面结构概述

2.2 常见UI控件

2.3 消息框和对话框

2.4 基于监听的事件处理

主要内容

消息框 Toast

- Toast是Android系统提供的小型弹出式窗口，用于向用户提示即时消息。它显示在应用程序界面的最上层，显示一段时间后会自动消失，也不获得焦点。

static Toast	makeText(Context context, int resId, int duration) Make a standard toast that just contains a text view with the text from a resource.
static Toast	makeText(Context context, CharSequence text, int duration) Make a standard toast that just contains a text view.

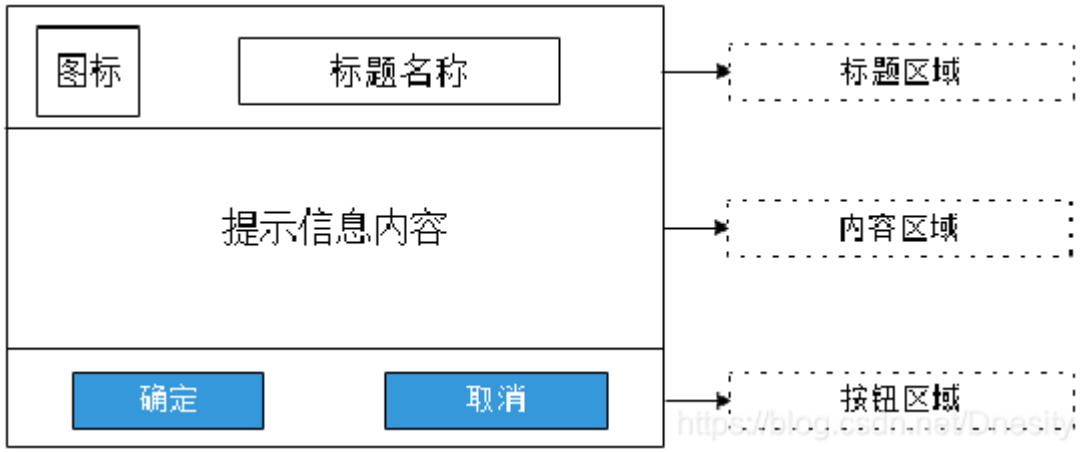
- 第一个参数：上下文环境，表示显示Toast的窗口。
- 第二个参数：Toast弹出窗口中显示的文本内容。
- 第三个参数：Toast弹出窗口显示的时间，`Toast.LENGTH_LONG` 或 `Toast.LENGTH_SHORT`

示例：

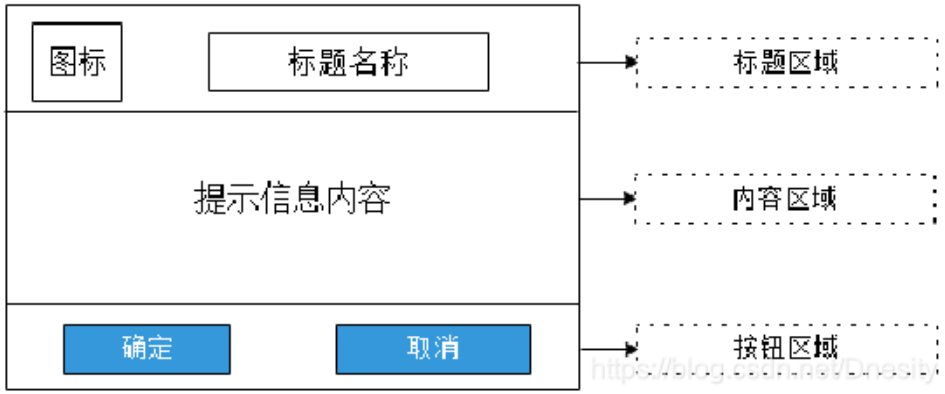
```
1 Toast.makeText(MainActivity.this, "提交成功", Toast.LENGTH_SHORT).show;
```

对话框 AlertDialog

- AlertDialog是Android系统提供的弹出式对话框。当对话框显示的时候，下面的Activity会失去焦点，用户的注意力将全部集中在对话框上进行操作；当对话框消失后，Activity将重新获得焦点。



对话框 AlertDialog



标题区域

- `setIcon()` 设置图标
- `setTitle()` 设置标题名称

按钮区域

- `setPositiveButton()` 设置确定按钮
- `setNegativeButton()` 设置取消按钮

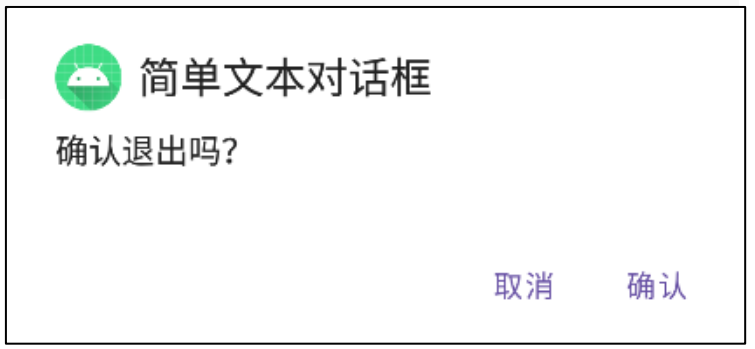
内容区域

- `setMessage()` 设置对话框内容为简单文本
- `setSingleChoiceItems()` 设置对话框内容为单选列表
- `setMultiChoiceItems()` 设置对话框内容为多选列表

简单文本对话框

```
AlertDialog.Builder builder = new AlertDialog.Builder(MainActivity.this);
builder.setIcon(AppCompatResources.getDrawable(MainActivity.this, R.mipmap.ic_launcher_round))
    .setTitle("简单文本对话框")
    .setMessage("确认退出吗? ")
    .setPositiveButton(getString(R.string.sure), new DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialog, int which) {
            dialog.dismiss();
            finish(); // 结束Activity
        }
    })
    .setNegativeButton(getString(R.string.cancel), new DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialog, int which) {
            dialog.dismiss();
        }
    })
    .create()
    .show();
```

创建Builder

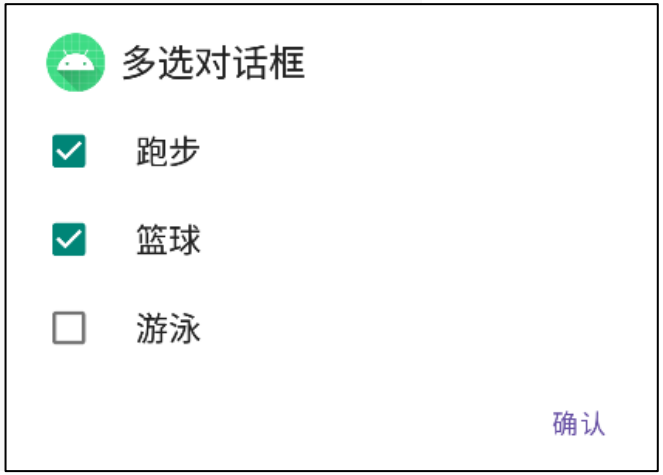


单选对话框

```
final String[] grades = new String[]{"大一","大二","大三","大四"};
buttonGrade.setOnClickListener(new View.OnClickListener() {
    int checkedItem = 0; // 默认选择第一个
    @Override
    public void onClick(View v) {
        // 创建对话框
        AlertDialog.Builder builder = new AlertDialog.Builder(MainActivity.this);
        builder.setIcon(R.mipmap.ic_launcher_round)
            .setTitle("单选对话框")
            .setSingleChoiceItems(grades, checkedItem, new DialogInterface.OnClickListener() {
                @Override
                public void onClick(DialogInterface dialog, int which) {
                    checkedItem = which;
                    String str = "你选择了" + grades[which];
                    Toast.makeText(MainActivity.this, str, Toast.LENGTH_SHORT).show();
                }
            })
            .setPositiveButton(getString(R.string.sure), new DialogInterface.OnClickListener() {
                @Override
                public void onClick(DialogInterface dialog, int which) {
                    buttonGrade.setText(grades[checkedItem]);
                    dialog.dismiss();
                }
            })
            .create()
            .show();
    }
});
```

多选对话框

```
String[] hobbies = new String[]{"跑步","篮球","游泳"};
buttonGrade.setOnClickListener(new View.OnClickListener() {
    boolean[] checkedItems = new boolean[]{false, false, false};
    @Override
    public void onClick(View v) {
        AlertDialog.Builder builder = new AlertDialog.Builder(MainActivity.this);
        builder.setIcon(getDrawable(R.mipmap.ic_launcher_round))
            .setTitle("多选对话框")
            .setMultiChoiceItems(hobbies, checkedItems, new DialogInterface.OnMultiChoiceClickListener() {
                @Override
                public void onClick(DialogInterface dialog, int which, boolean isChecked) {
                    checkedItems[which] = isChecked;
                }
            })
            .setPositiveButton(getString(R.string.sure), new DialogInterface.OnClickListener() {
                @Override
                public void onClick(DialogInterface dialog, int which) {
                    StringBuilder sb = new StringBuilder();
                    for (int i = 0; i < checkedItems.length; i++) {
                        if (checkedItems[i]) {
                            sb.append(hobbies[i] + " ");
                        }
                    }
                    buttonGrade.setText(sb.toString());
                }
            })
            .create()
            .show();
    }
});
```



练习：UI设计

功能描述：

- 设计界面

技能要点：

- 学会使用各种UI控件

练习：UI设计

辅助线

横向辅助线：5% 13% 30% 35% 43% 52% 60% 68% 76% 84% 92%

纵向辅助线：10% 22% 90%

字体

标题: 28sp

性别、爱好、年级、评分、提交、退出: 18sp, 左对齐

男、女、跑步、篮球、游泳、选择: 14sp

ratingBar: scale=0.6

宽度和高度

ImageView: 0dp, 0dp, 1:1

EditTextUsername: 0dp, 0dp

EditTextPassword: 0dp, 0dp

其余均可为wrap_content, wrap_content

学习内容

2.1 界面结构概述

2.2 常见UI控件

2.3 消息框和对话框

2.4 基于监听的事件处理

主要内容

基于监听的事件处理

- 当用户在应用界面执行一系列操作时，程序需要响应用户的各种事件，如单击、长按、触碰、键盘输入等，对此，Android提供了基于监听的事件处理机制。
- 基于监听的事件处理是基于面向对象的事件处理机制，由3类对象组成：
  - 事件源（Event Source）：产生事件的来源，**通常是各个组件**，如按钮、窗口、菜单等。
  - 事件（Event）：事件封装了界面组件上发生的特定事件，**一般指用户的各种操作**，如单击、长按、触摸等。
  - 事件监听器（Event Listener）：负责监听事件源发生的事件，并对不同的事件做相应的处理。

基于监听的事件处理

- 创建事件监听器的方式：

创建事件监听器的方式	适用情况
匿名内部类作为事件监听器	不会重用
内部类作为事件监听器	被Activity内多个控件重用
外部类作为事件监听器	被不同Activity重用
Activity本身作为事件监听器	事件监听器少且简单
直接绑定在控件标签	只适用于android:onClick

匿名内部类作为事件监听器

匿名内部类

```
buttonOk.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        // 代码块
    }
});
// new View.OnClickListener(){}是接口
```

内部类作为事件监听器

```
// 以下为内部类，定义在MainActicity类，作为MainActicity类的成员
class MyListener implements View.OnClickListener{
    @Override
    public void onClick(View v) {
        // 代码块
    }
}

// 以下在在MainActicity类中使用
buttonOk.setOnCheckedChangeListener(new MyListener());
```

外部类作为事件监听器

```
// 以下类定义在MainActicity类之外
class MyListener implements View.OnClickListener{
    @Override
    public void onClick(View v) {
        // 代码块
    }
}

// 以下在MainActicity类中使用
MyListener listener = new MyListener();
buttonOk.setOnClickListener(listener);
```

Activity本身作为事件监听器

```
public class MainActivity extends AppCompatActivity
implements View.OnClickListener{
    @Override
    public void onClick(View v) {
        // 代码块
    }
}
```

直接绑定到控件标签

- 先在MainActivity中定义方法

```
1 public void onSubmit(View view) {
2     // 代码块
3 }
```

- 然后在xml文件中将上述方法直接绑定到某个控件上

  （只适用于`android:onClick`）

```
1 <Button
2     android:id="@+id/button1"
3     android:layout_width="wrap_content"
4     android:layout_height="wrap_content"
5     android:text="@string/submit"
6     android:onClick="onSubmit" />
```

练习：事件监听

功能描述：

- 实现页面交互

技能要点：

- 创建事件监听器的不同方法
- 实现不同事件的监听

练习：事件监听

1. 切换性别时，更换头像图片
2. 选择爱好时，弹出Toast并显示爱好
3. 点击年龄旁边按钮，弹出单选对话框，选择某项时弹出Toast，选择确认后更改按钮显示内容
4. 或者点击年龄旁边按钮，弹出多选对话框，选择某项时弹出Toast，选择确认后更改按钮显示内容
5. 评分时，弹出Toast并显示评分
6. 点击退出时，弹出对话框提示是否确认退出
7. 点击提交时判断用户名和密码是否为空，为空则弹出Toast并提示，不为空则则弹出Toast并显示提交的内容

本章小结

2.1 界面结构概述

2.2 常见UI控件

2.3 消息框和对话框

2.4 基于监听的事件处理

作业

本章作业

- 按照课堂要求实现UI设计
  - 截图：运行后的界面
- 按照课堂要求实现事件监听
  - 截图：提交成功后弹出Toast的页面
  - 视频：运行后全过程
    - **用户名和密码分别用：姓名 + 学号**

作业提交方式

- 截图+视频，文件名：学号+姓名+第2课作业
- 邮件给助教，主题：学号+姓名+第2课作业
- 截止时间：下周二

作业示例

运行过程中，输入的用户名和密码分别用：姓名 + 学号