

实验一

基本 UI 界面设计

【实验目的】

- 1.熟悉 Android Studio 开发工具操作
- 2.熟悉 Android 基本 UI 开发,并进行 UI 基本设计

【实验内容】

实现一个 Android 应用，界面呈现如下效果：



要求：

- (1) 该界面为应用启动后看到的第一个界面
- (2) 各控件的要求如下：

要求只用一个 ConstraintLayout 实现整个布局；

标题字体大小 20sp，与顶部距离 20dp，居中；

图片与标题的间距为 20dp，居中；输入框整体距屏幕右边间距 20dp，上下两栏间距 20dp，内容（包括提示内容）如图所示，内容字体大小 18sp；

学号对应的 EditText 只能输入数字，密码对应的 EditText 输入方式为密码；

两个单选按钮整体居中，字体大小 18sp，间距 10dp，默认选中的按钮为第一个；

两个按钮整体居中，与上方控件间距 20dp，按钮间的间距 10dp，文字大小 18sp。按钮背景框左右边框与文字间距 10dp，上下边框与文字间距 5dp，圆角半径 10dp，背景色为 #3F51B5

(3) 使用的布局和控件：ConstraintLayout、TextView、EditText、Button、ImageView、RadioGroup、RadioButton

【基础知识】

Android 的组件分为布局和控件。布局，就是让控件在里面按一定的次序排列好的一种组件，本身并不提供内容。控件，就是显示内容的组件，比如显示一张图片，显示文字等等。现在的主流是用 ConstraintLayout。最常用的控件有以下几种：用于显示文字的 TextView、用于显示图片的 ImageView、用于接受用户输入的输入框 EditText、按钮 Button、单选按钮 RadioButton，等等。以下简要介绍本次实验使用到的布局和组件。

ConstraintLayout

约束布局，根据布局中的其他元素或视图，确定 View 在屏幕中的位置，受到三类约束，即其他视图，父容器(parent)，基准线(Guideline)。

layout_constraint[本源位置]_[目标位置]="[目标 ID]"

例如：app:layout_constraintBottom_toBottomOf="parent"

约束当前 View 的底部至目标 View 的底部，目标 View 是 constraintLayout。即，把当前 View 的底部对齐到 constraintLayout 的底部。

全部边界与 constraintLayout(父容器)边界对齐，则为居中。

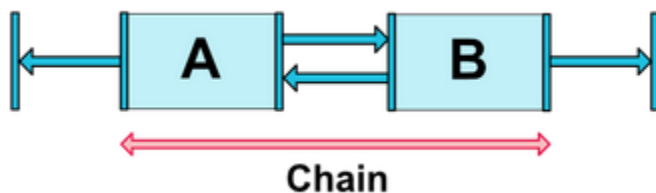
```
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Hello World!"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintTop_toTopOf="parent" />
```

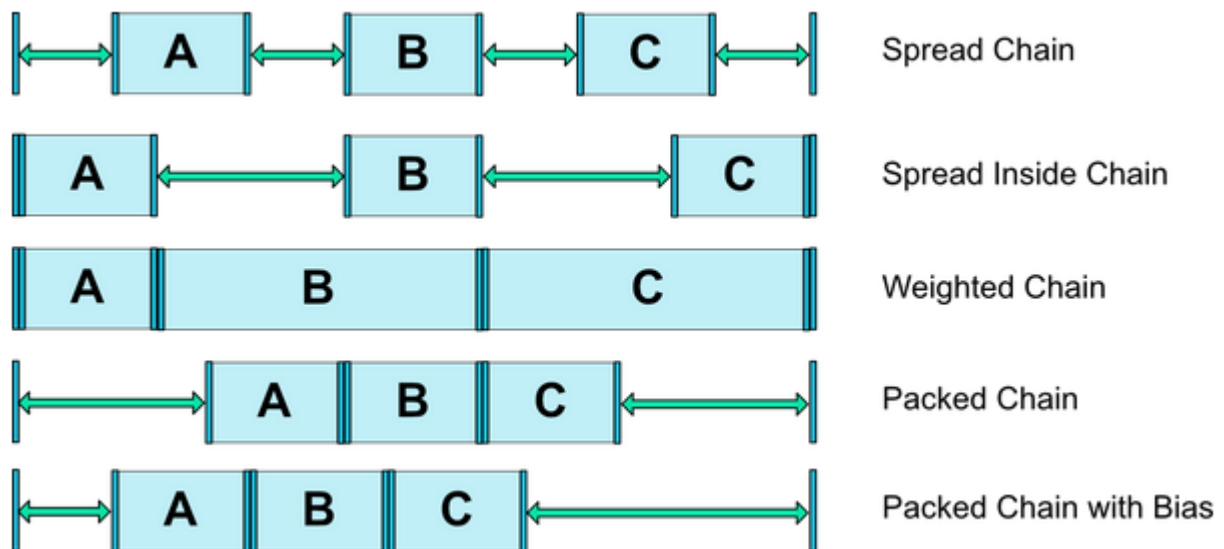
ConstraintLayout 除了与视图约束以外, 还支持与引导线(Guideline)约束. 如, 设置垂直引导线(Guideline)距离左侧 22dp,TextView 左侧与引导线约束,

```
<android.support.constraint.Guideline
    android:id="@+id/guideline"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:orientation="vertical"
    app:layout_constraintGuide_begin="22dp" />
```

表示的 Guideline 是垂直的, 与左边屏幕相距 22dp。

支持链样式, 如果一组小部件通过双向链接链接在一起, 则视为链条。





通过 `app:layout_constraintHorizontal/Vertical_chainStyle` 设置链接的样式。

更多属性请参阅：

<https://developer.android.google.cn/reference/android/support/constraint/ConstraintLayout.html>

TextView

用于显示文字内容的控件，通过设置 `text` 的值来显示要显示的内容，常用的属性有 `textColor`，用于设置文字的颜色，`textSize`，用于设置文字大小。示例：

```
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_margin="10dp"
    android:textColor="@color/colorAccent"
    android:textSize="25sp"
    android:text="第一次实验"/>
```

效果图：

第一次实验

关于 `@color/colorAccent` 这种形式的资源引用后面会讲

EditText

用于接受用户输入的输入框，常用属性除了和 TextView 相同的 textColor 和 textSize 之外，还有 inputType，用于设置输入框中文本的类型，如果设置为 textPassword，会使输入的文字变为小点 (·)，hint，用于设置当输入框为空时的提示内容。以一个密码输入框做示例：

```
<EditText
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:textColor="@color/primary_black"
    android:textSize="@dimen/normal_text"
    android:inputType="textPassword"
    android:hint="请输入密码"/>
```

效果：

未输入前：

输入之后：

ImageView

显示图片的控件，通过设置 src 来设置要显示的图片

```
<ImageView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:src="@mipmap/sysu"/>
```



关于 ImageView 的 src 和 background 属性的区别，自行查阅资料。

RadioButton 和 RadioGroup

RadioButton 是单选按钮，一组单选按钮需要包括在一个 RadioGroup 中，并且需要对 RadioGroup 和其包括的每一个 RadioButton 都设置 id，RadioButton 才能正常工作。示例：

```
<RadioGroup
    android:id="@+id/id0"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:orientation="horizontal">
    <RadioButton
        android:id="@+id/id1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="男"/>
    <RadioButton
        android:id="@+id/id2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="女"/>
</RadioGroup>
```

可通过设置 checked 属性来使某个 RadioButton 被选中

组件的介绍就到这里，下面简单介绍以下几个重要的通用属性

layout_width 和 layout_height

这两个属性分别指定所属组件的宽度和高度，属性值可以是具体的距离，如：20dp，更常见的是指定为 match_parent 或者 wrap_content，match_parent 指的是组件的宽度或高度与父组件的宽度或高度一致，如果组件没有父组件，那么组件的宽度或高度与屏幕的宽度或高度一致。wrap_content 指组件的宽度或高度刚好可以包裹住组件内的子组件即可。

layout_gravity 和 gravity

这两个属性的基本属性值有五种：top、bottom、center、left、right，可以使用组合属性，如 left|bottom 表示左下角。区别在于 layout_gravity 用于指定设置该属性的组件相对于父组件的位置，gravity 用于指定指定设置该属性的组件下的子组件的位置。

layout_margin 和 padding

layout_margin 指定外边距，padding 指定内边距，这两个属性配合上四个方向还各有四个属性，如 layout_marginTop，paddingTop 等。

关于自定义背景边框

当需要将一个 button 设置为圆角矩形时，光设置 button 的属性是达不到效果的，需要定义一个背景边框来达到这个效果



这种自定义的背景边框定义在 drawable 文件夹下，所以为了不把它和图片混杂在一起，习惯上把图片放在 mipmap 文件夹下。

定义的方法如下：

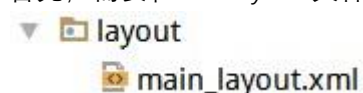
在 drawable 文件夹下新建一个 Drawable resource file，填写 file name，然后把自动生成的 selector 标签改为 shape，shape 下有多个属性，padding，radius，solid 等等，具体怎么使用参见这篇教程：

<http://blog.csdn.net/sysukehan/article/details/52022307>

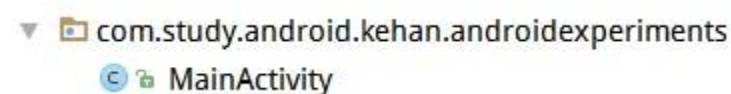
在 Button 中设置 background 为 @drawable/加上刚刚填写的文件名即可引用这个自定义的背景。

如何在应用中显示布局

首先，需要在 res/layout 文件夹下写好布局文件



然后创建一个 java 文件



在该文件中将布局引入


```
public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(@Nullable Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main_layout);
    }
}
```

然后在注册文件中注册，将该 Activity 设置为应用启动时第一个加载的 Activity

```
<activity android:name=".MainActivity">
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</activity>
```

然后就可以运行了

【拓展知识】

关于资源的引用

Android 项目中不建议使用硬编码来使用资源，建议将各类资源定义在 res 文件夹中的 values 文件夹下，字符串资源定义在 strings.xml 中，颜色资源定义在 colors.xml 中，距离，字体大小资源定义在 dims.xml 中。图片资源例外，是存放在 res 文件夹中的 mipmap 文件夹下或者 drawable 文件夹

下。给个示例看一下怎么定义： colors.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="colorPrimary">#3F51B5</color>
    <color name="colorPrimaryDark">#303F9F</color>
    <color name="colorAccent">#FF4081</color>

    <color name="primary_black">#D5000000</color>
    <color name="primary_white">#FFFFFF</color>
</resources>
```

通过键值对来定义，使用的时候用@color/colorAccent 即可引用到对应资源，注意是@color 不是 @colors，这里和文件名是不相同的。其它资源的引用也一样。

关于自定义 style

style 定义在 res/values/styles.xml 文件中，也是一种资源。例如当多个 TextView 具有相同的 layout_height, layout_width, textSize, textColor 设定时，如果每次都要把这些属性设定一次会很烦，而且如果全部需要修改的话（改变字体大小）也不方便，因此可以把这些属性统一定义为一个 style，这样使用的时候只要直接引用这个 style 即可。

定义 style 的方法：在 styles.xml 文

件中定义自定义 style

```
<style name="my_edittext_style">
    <item name="android:layout_height">wrap_content</item>
    <item name="android:layout_width">wrap_content</item>
    <item name="android:textColor">@color/primary_black</item>
    <item name="android:textSize">@dimen/normal_text</item>
</style>
```

一个自定义 style 下包含多个键值对，引用的时候设置 style="@style/my_edittext_style" 即可，注意不要写成 android:style="@style/my_edittext_style"。在引用 style 之后，还可以继续定义属性，如果有重复，以继续定义的属性为准。

【检查内容】

- 1、布局显示是否正常
- 2、是否熟悉相应布局和控件的使用

【提交说明】

- 1、deadline：下一次实验课前一天晚上 12 点
- 2、提交作业地址：ftp://edin.sysu.edu.cn
- 3、文件命名及格式要求：学号_姓名_labX.zip（姓名中文拼音均可）

4、目录结构：

```
14331111_huashen_lab1 --  
    |  
    -- lab1实验报告.pdf  
    |  
    -- lab1_code（包含项目代码文件）
```

其中项目代码文件为项目文件夹，*提交之前先 clean

