第一章 安卓基础入门

通信技术

1G：语音蜂窝电话标准

2G：数字语音传输技术为核心，GSM，增加接收数据的功能

3G：图像、音乐、视频

4G：高质量图像、音乐、视频，最高**100Mbit/s**

5G：**10Gbit/s**

安卓体系结构

Android系统采用分层架构

应用程序层：核心应用程序的集合

应用框架层：提供构建应用程序用到的API

核心类库：包含了系统库和Android运行环境

Linux内核：为安卓设备提供了底层的驱动

Dalvik虚拟机

Dalvik是Google公司设计用于Android平台的虚拟机，其指令集基于寄存器架构，执行特有的dex文件来完成对象生命周期管理，堆栈管理，线程管理，安全异常管理，垃圾回收等重要功能

安卓程序结构

app：

用于存放程序的代码和资源等内容

目录：

libs：用于存放第三方jar包

src/androidTest: 存放测试代码

src/main/java： 存放程序代码

src/AndroidManifest.xml!:整个程序的配置文件，在该文件中配置程序所需要的权限和注册程序中用到的四大组件
app/build/gradle:该文件是app的gradle的注册脚本

build.gradle:该文件是程序的gradle构建脚本

local.properties:该文件用于指定项目中所使用的的SDK路径

setting.gradle:该文件用于配置在安卓程序中使用到的子项目:(moudle)

资源的管理与使用

1.图片资源：

安卓中图片资源包括png jpg gif等文件

存放在mipmap开头的文件中

根据设备屏幕密度不同，自动匹配不同文件夹图片资源

120-160 mdpi

160-240 hdpi

240-320 xdpi

320-480 xxdpi

480-640 xxxdpi

通过java代码调用图片资源

在Activity的方法中可以**getResourse().getDrawable()**方法调用图片资源

示例代码

```
getResourse().getDrawable(R.mipmap.ic_launcher);//调用mipmap文件夹中资源文件
```

2.主题和样式

安卓中的样式与主题，都是用于为界面元素定义显示风格，它们的定义方式比较类似，具体介绍如下

1.主题:

主题是包含一种或多种的格式化集合属性集合，在程序中调用主题资源可以改变窗体的样式，对整个应用或摸个Activity存在全局性影响。
主题资源定义在res/values目录下的styles.xml文件中

2.样式

通过改变主题可以改变整个窗体样式，当主题不能何止View控件的具体样式，因此我们需要创建一个样式来美化View控件，样式存放在res/values目录下的styles.xml文件中
在布局文件中的View控件通过Style属性调用textViewStyle样式的示例代码如下:

```
<TextView
....
style ="@style/textViewstyle"/>
```

3.布局资源

res目录下有一个layout文件夹，该文件夹中存放的是程序中的所有布局资源文件，这些布局资源通常用于搭建程序中的各个界面

如果想要在程序中调用布局资源文件，调用方法有两种方式，一种是通过java代码调用，一种是在xml文件中调用，具体如下:

1.java代码调用

Activity中，找到onCreate() 方法，在该方法中通过**setContentView()**方法，来加载Activity对应的布局资源文件

```
setContentView(R.layout.activity_main);
```

2.xml

```
<include layout="@layout/activity_main"/>
```

4.字符串资源

程序调试

单元测试

Junit 单元测试

1. android单元测试：依赖android设备，慢，调用ExampleInstrumentedTest.java
2. Junit单元测试：不依赖,本地即可，快，适合java ExampleUnitTest.java

LogCat

六个级别

verbose 全部信息 白色

debug　调试信息 蓝色

info　　一般信息 绿色

warning 警告信息 黄色

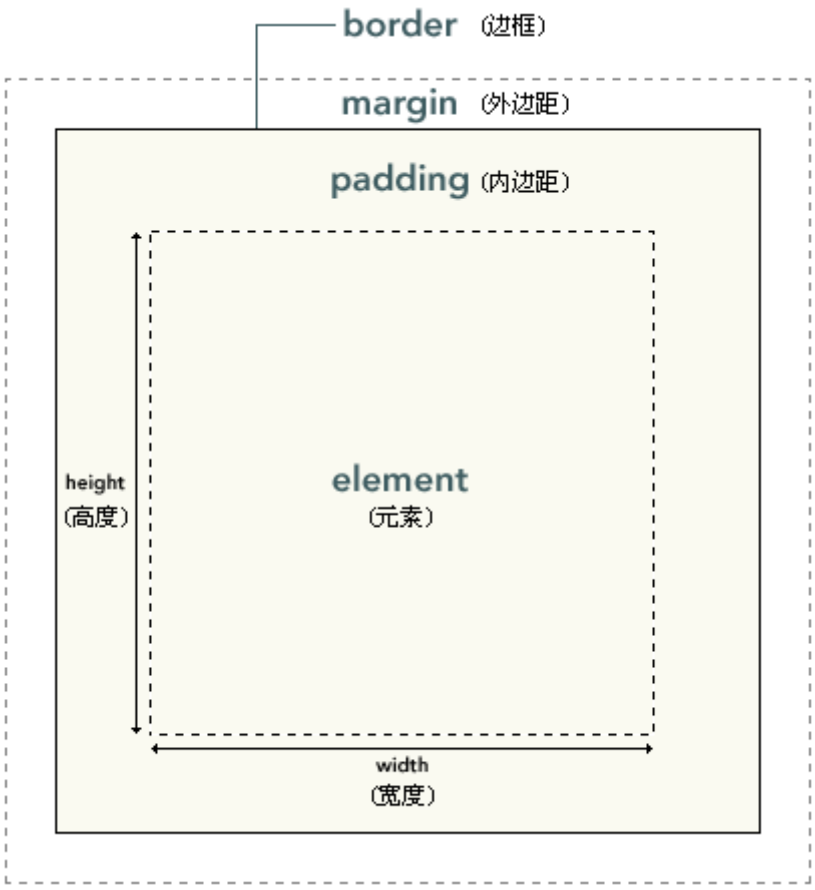
error　　错误信息 红色

assert 断言失败错误信息 红色

基本控件

TextView

主要就是用来显示文件的



设置控件id

```
android:id = "@+id/text_view"
android:id = "@+id/textview"
```

设置宽度

```
android:layout_width = "match_parent" //填充完父布局的宽度
android:layout_width = "wrap_content" //自适应布局
android:layout_width = "10dp"
```

设置高度

```
android:layout_height = "match_parent"
android:layout_height = "wrap_content"
android:layout_height = "10dp"
```

设置颜色

```
android:background = "#CCCCCC"
```

设置内边距

```
android:padding = "12dp"
android:paddingTop
android:paddingBottom
android:paddingRight
android:paddingLeft
```

设置外边距

```
android:margin = "12dp"
android:marginTop
android:marginBottom
```

设置文字内容、字体大小、样式

```
android:text = "hello world"
android:textSize = "12sp"
android:textColor = "#FFFFFF"
android:textStyle = "bold"//粗
```

总体代码演示

```
<TextView
    android:id = "@+id/demo1"
    android:layout_width = "wrap_content"
    android:layout_height = "wrap_content"
    android:text = "hello world"
/>
```

java

```
public class TextViewActivity extends AppCompatActivity {
    @Override
    protected void onCreate(@Nullable Bundle savedInstanceState) {
        super.onCreate(saveInstanceState);
        setContentView(R.layout.demo1);
    }
}
```

```
TextView demo2 = findViewByid(R.id.demo1)
demo2.setText("hello")
```

```
<activity android:name = ".TextViewActivity" android:exported="true"/>
```

字符串一般在res/values/strings/xml

```
<string name = "hello">你好</string>
```

```
android:text = "@string/hello"
----------------------------
demo1.setText(R.string.hello);
```

设置文本颜色

```
透明度-R-G-B
#RGB
#ARGB
#RRGGBB
#AARRGGBB
```

```
<TextView
    android:textColor="#F606060"
    android:textColor="@color/black"
/>
```

```
<resources
    <color name = "black">#FF000000</color>
/>
```

常见尺寸单位

```
px: 1px表示一个物理像素
dp: 这个最常用，与像素密度相关
sp: 与缩放无关的抽象像素
```

文字尺寸sp，非文字dp

表格线、分隔线、阴影线 px

第二章基本控件

TextView

显示文本信息

gravity 设置文本内容 如 center

lines 设置行数

EditText

用于获取用户的输入，继承TextView类

```
andorid:hint              设置EditText内容为空时，显示的文本
andorid:textColorHint     设置hint颜色
andorid:editable          设置是否可以编辑
andorid:password          设置是否为密码框，显示内容自动为星号
```

输入限制

```
android：inputType属性

none              普通字符
text              普通字符
textPassword      密码格式
number            数字格式
date              日期键盘
datetime          时间日期
```

```
<EditText
          android:id = "@+id/edit_text"
          android:layout_width = "wrap_content"
          android:layout_height = "wrap_conetent"
          android:hint = "请输入..."
          android:maxLength = "40"
/>
<EditText
          android:id = "@+id/edit_text"
          android:layout_width = "wrap_content"
          android:layout_height = "wrap_conetent"
          android:password = "true"
/>
<EditText
          android:id = "@+id/edit_text"
          android:layout_width = "wrap_content"
          android:layout_height = "wrap_conetent"
          android:inputType = "data"
/LinearLayout>
```

Button

Button控件表示按钮，通过用户点击来执行操作，当Button控件被点击时，会触发点击效果

```
<Button
        android:id = "@+id/bt3"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:onClick = "Click"
        android:text = "点击按钮"
/>
```

RadioButton

单选按钮，把其放在RadioGroup中实现单选功能，button的子类

RadioGroup继承LinearLayout，用 adnroid:orientation 控制方向

```
<RadioGroup
            android:layout_width = "wrap_content"
            android:layout_heigt = "wrap_content"
            android:centerVertical = "true"       垂直排列
            android:centerHorizental = "true">    水平排列
            <RadioButton
                        android:id = "@+id/kk1"
                        android:layout_width = "match_parent"
                        android:layout_height = "match_parent"
                        android:text = "man"
                        android:checked = "true"/>    默认被选
            <RadioButton
```

```
                        android:id = "@+id/kk2"
                        android:layout_width = "match_parent"
                        android:layout_height = "match_parent"
                        android:text = "woman"
                        android:checked = "false"/>
</RadioGroup>
```

CheckBox

复选按钮可以选择若干个选项，CheckBox是Button子类，支持使用Button的所有类型

```
shuttlecock.setOnCheckedChangeListener(this);
```

```
<TextView
          android:id = "@+id/tv_hobby"
          android:layout_width = "wrap_content"
          android:layout_height = "wrap_content"
          android:text = "my hobby"
          android:textSize = "20sp"/>
<CheckBox
          android:id = "@+id/ch_book"
          android:layout_width =
          android:layout_height =
          android:textSize = "20sp"
          android:text = "read"
          />
<CheckBox
          android:id = "@+id/ch_sport"
          android:layout_width =
          android:layout_height =
          android:textSize = "20sp"
          android:text = "sport"
          />
```

ListView

可以为我们提供一个列表外加纵向滚动的功能，比如使用微信的联系人

使用

```
divider 分割线颜色
dividerHeight 分割线高度
```

数据适配器，数据和视图之间的桥梁

```
string.xml
    <string-array name = "item">
        <item>张三</item>
        <item>李四<item>
    </string-array>
    <ListView
              android:layout_width = "wrap_content"
              android:layout_height = "wrap_content"
              android:entries="@array/itemx"></ListView>
```

ImageView

图像视图，直接继承View类，主要功能是用于显示图片

src：设置图片资源

scaleType：设置图片缩放类型

maxWidth：最大宽度

maxHeight：最大高度

```
<ImageView
           android:id = "@+id/image"
           android:scaleType = "center"
           android:src = "@drawable/logo"/>
```

AlterDialog对话框的使用

标题、内容、按钮

show方法显示

dismiss取消

自定义View

onMeasure 测量

onDraw 图像

onlayout

第三章 常见布局

View视图

所有的ui元素都是View 和 ViewGroup构建的

只能有一个viewgroup容器

LinearLayout

线性布局，要么是横向的，要么是竖向的

android:orientation 规定布局内子控件方向

值	描述
horizontal	水平布局
vertical	垂直布局

均等分布

android:layout_weight 权重

```
想让每一个子视图使用相同的大小空间，使weight = 1
```

使用代码

```
<TextView
          android:layout_width = "0dp"
          android:layout_height = "20dp"
          android:layout_weight = "1"
          android:text = "aaa"/>
<TextView
          android:layout_width = "0dp"
          android:layout_height = "20dp"
          android:layout_weight = "1"
          android:text = "bbb"/>
```

RelativeLayout

相对布局，根据父容器和兄弟控件作为参考来确定控件位置的布局方式

```
android:layout_alignParentLeft = "true"     父容器左边
android:layout_alignParentRight = "true"    父容器右边
android:layout_alignParentTop = "true"      父容器顶部
android:layout_alignParentBottom = "true"   父容器底部
android:layout_centerHorizental = "true"    水平方向居中
android:layout_centerVertical = "true"      垂直方向居中
android:layout_centerInParent = "true"      正中心
```

设置在子控件上面

```

```

FrameLayout

帧布局

左上角开始，后面的覆盖前面的

```
<Button
        android:layout_gravity = "right">
</Button>

top 把对象放在其容器的顶部，不改变其大小
bottom 把对象放在容器底部，不改变其大小
left   把对象放在容器左侧，不改变其大小
right
```

TableLayout

表格布局使以行数和列数来确定位置进行排列，可用于制作表格，其中TableRow为其子控件

```
android:collapseColumns 隐藏某一列，下标从0开始
android:strechColumns   拉伸某列
android:shrinkColumns   收缩某列
```

ConstraintLayout

约束布局

AbsoluteLayout

绝对布局

```
<Button
        android:layout_width = "10dp"
        android:layout_height = "10dp"
        android:layout_x = "100dp"
        android:layout_y = "120dp"
        android:text = "账号"
        android:textSize = "30sp"
/>
```

GridLayout

网格布局

```
 <android:columnCount = "4"
         rowCount = "5"
         orientation = "horizontal"
```

```
<android:layout_row = "0" 指定第几行显示
<android:layout_column = "3"
<android:layout_rowSpan = "2"占两格
<android:layout_columnSpan = "2"占两格
```

第四章 事件

点击事件的三种方式

第一种方式

设置**onClick**

首先要有控件才能实现点击事件

```
<Button
        android:onClick="buttonClick"
        android:id="@+id/btn1"
        android:text="Button"/>
```

```
public class MainActivity extents AppCompatAcitivity {
    @Override
    protected void oncreate(Bundle savedInstancestate){
        super.oncreate(savedInstancestate);
        setcontentview(R.layout.activity_main);
    public void buttonclick(view view){
        Toast.makeText(this,"按钮1被点击啦",ToaSt.LENGTH_LONG).show();
    }
}
```

Toast使用

这个是常用组件

轻量级信息提醒机制 一段时间后自动消失，不会打断当前操作，也不会获得焦点

```
Toast.makeText(MainActivity.this,"aaa",Toast.LENGTH_SHORT).show();
```

```
//第一个参数:当前的上下文环境。可用getApplicationcontext()或Activity的context
// 第二个参数:要显示的字符串。也可是R.string中字符串ID
//第三个参数:显示的时间长短。Toast默认的有两个LENGTH_LONG(长)和LENGTH_SHORT(短)，也可以使用毫秒如2000ms
Toast toast=Toast.makeText(mcontext,"你好 张无忌~"，Toast.LENGTH_SHORT);
//显示toast信息toast.show();
Toast.makeText(this,"hello",Toast.LENGTH_SHORT);
toast.show()
```

第二种方式

匿名内部类

在onCreat()里面声明并绑定控件，然后注册监听器

重写onClick方法

```
<Button
        android:onClick="buttonClick"
        android:id="@+id/btn1"
        android:text="Button"/>
```

```
public class MainActivity extents AppCompatAcitivity {
    @Override
    protected void onCreate(Bundle savedInstancestate){
        super.oncreate(savedInstancestate);
        setContentView(R.layout.activity_main);

    Button btn2 = findViewByID(R.id.btn2);
    btn2.setOnClickListener(new View.OnClickListener() {
        @Override
        public void buttonclick(view view){
            Toast.makeText(this,"按钮1被点击啦",ToaSt.LENGTH_LONG).show();
        }
    });
}
```

```
Toast.makeText(this,"hello",LENGTH_SHORT).show()
```

第三种方式

实现OnclickListener接口

```
<Button
        android:onClick="buttonClick"
        android:id="@+id/btn1"
        android:text="Button"/>
```

```
public class MainActivity extents AppCompatAcitivity {
    @Override
    protected void onCreate(Bundle savedInstancestate){
        super.oncreate(savedInstancestate);
        setContentView(R.layout.activity_main);
        Button btn2 = findViewByID(R.id.btn2);
        btn2.setOnClickListener(this);
    }

    @Override
    public void onClick(View view) {
        if(view.getID == R.id.btn2) {
            Toast.makeText(MainAcitivity.this,"按钮2被点击
了",Toast.LENGTH_LONG).show();
        }
    }
}
```

RadioButton的监听事件

遍历RadioGruop中获取RadioButton

isChecked：用来判断是否选中

```
for (int i = 0; i < rgSex.getChildCount(); i++) {
    if(rgSex.isCheck()) {
        Toast.makeText(getApplicationContext(),"dfdf",Toast.LENGTH_SHROT).show();
        break;
    }
}
```

CheckBox的监听事件

isChecked：用来判断是否选中

```
if(isChecked()) {
    String text = checkBox.getNext().toString()l
    Toast.makeText(MainAcitivity.this,"111",1).show();
}
```

第五章 Activity组件

Activity基础（了解

概述

是一个应用组件，用户可以与其提供的屏幕进行交互

以执行打电话、拍照、查看地图等操作

组成

java文件，layout文件，并且要在清单中注册

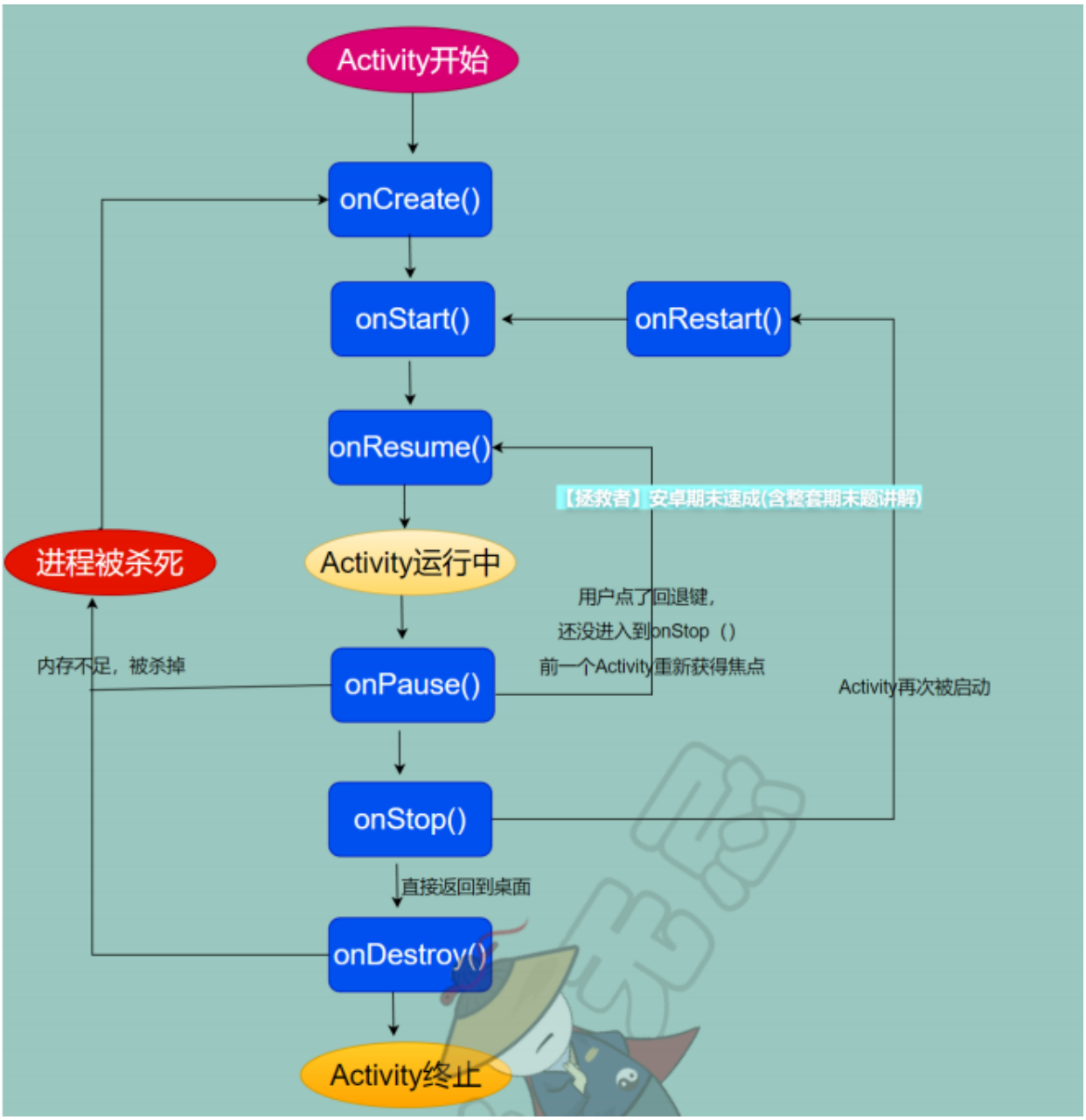
AnroidMainfest.xml - 清单文件

　他告诉系统我们的app有哪些activity，用到了什么权限等

MainActivity是应用启动的第一个activity

Activity的生命周期（了解

```
onCreate()  此处创建视图 将 数据绑定到列表
onStart() 互动前最后的准备工作，即将可见
onResume() 获取焦点
onPause() 暂停，覆盖锁屏使用
onStop()  对用户不可见的时候调用
onRestart() 重启
onDestroy() 销毁时调用，最后一个，确保释放该Activity
```



启动

create start resume

退出

pause stop　destroy

任务 返回栈

启动另一个activity时，新的会被送到栈顶，获得焦点

上一个会停止，

Intent的使用

有两种跳转方式 “意图”

显示Intent

通过指定具体的activity实现

开启自己应用内的activity

```
Intent intent = new Intent(MainActivity.this,TestActivity.class)//当前act，要启动的act
startActivity(intent);
```

直接使用组件的名称定义

隐示Intent

通过指定一组数据或者动作实现

开启其他应用的activity

IntentFilter主要属性

action

指定activity可以完成哪些动作，系统中定义了很多常用的动作

```
action_call 启动电话
action edit 显示数据编辑页面
action main 启动项目的初始界面
action sync 同步服务器和移动设备的数据
```

```
action battery low 警告电池电量低
       headset plug 耳机插入 拔掉设备
       screen on 屏幕已打开
       timezone changed 时区设置改变
```

data

表示要传递的数据

```
打电话 tel
访问网络 http
内容提供者提供的数据content
指向手机通讯录联系人的uri content contacts
```

category

可以0-n个

要全部匹配

action添加的额外信息

指定当前动作被执行的环境，activity被激活的环境

```
default 默认
browsable 被浏览器安全调用
home 随系统启动而运行
launcher
preference 参数面板
```

Activity的数据传递

返回数据给上一个

intent使用Bundle对象存放待传递的数据信息

发数据

```
Bundle bundle = new Bundle();
bundle.putString("name","zwj");
bundle.putInt("age",99);
intent.putExtras(bundle);
startActivity(intent);
```

读数据

```
Bundle bundle = this.getIntent().getExtras();
String name = bundle.getString("name");
int age = bundle.getInt("age");
                getFloat
                   String
                   Double
                   Boolean
                   StringArray
                   ...
```

```
<EidtText
          android:id = "@+id/et1"
          android:layout_width = "match_parent"
          android:layout_height = "wrap_content"
          android:hint = "请输入名字"
          android:textSize = "10px">
</EidtText>
<EidtText
          android:id = "@+id/et2"
          android:layout_width = "match_parent"
          android:layout_height = "wrap_content"
          android:hint = "请输入年龄"
          android:textSize = "10px">
```

```
</EidtText>
<Button
        android:id
        android:layout_width
        android:layout_height
        android:text
        android:background>
</Button>

<activity android:name = "aa" android:exported = "true"></activity>
<activity android:name = "bb" android:exported = "true"></activity>
```

Demo7

```
  //拿到用户输入的数据
        EditText et1= findViewById(R.id.et1);
        EditText et2= findViewById(R.id.et2);

        Button btn1= findViewById(R.id.btBound);

        btn1.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                //构建Bound对象
                Bundle bundle = new Bundle();
                bundle.putString("name",et1.getText().toString());
                bundle.putInt("age",Integer.parseInt(et2.getText().toString()));
                //构建Intent
                Intent intent = new
Intent(Demo7Activity.this,Demo8Activity.class);
                intent.putExtras(bundle);
                startActivity(intent);
            }
        });
```

Demo8

```
      //读取intent的数据给bundle对象
        Bundle bundle = this.getIntent().getExtras();
        String name = bundle.getString("name");
        int age = bundle.getInt("age");
        //给两个组件渲染值
        TextView tv1 = findViewById(R.id.tv1);
        TextView tv2 = findViewById(R.id.tv2);
        tv1.setText("姓名为"+name);
        tv2.setText("年龄为"+age);
```

返回数据给上一个

第六章 Android数据存储

1.文件存储数据

适合存储大量数据

```
FileOutputStream fos = openFileOutput(String name,int mode)
FileInputStream  fis = openFileinput (String name)
```

默认位置/data/user/<包>/files/文件名

应用被卸载，该文件也会被删除

内部存储

```
mode 取值
mode_private         只能被当前程序读写
     append          内容可以追加
     world readable 可以被其他读取
     world writeable
```

默认私有

写入步骤

```
fos.write(content.getBytes());
```

读取步骤

```
fis = openFileputs("data.txt");
byte[] buffer = new byte[fis.available()];
fis.read(buffer);
```

外部存储

以文件的形式存储在外设

路径 mnt/sdcard/目录下

2.sharedpreferences存储数据

轻量级的存储类，适合用于保存软件配置参数

背后使用xml文件存放数据，文件存放在/data/user/shared_prefs目录下

优点：简单、方便、轻量级

缺点：适合存储少量数据，格式只能是基本的数据类型

无法条件查询

写入

只能获取数据

```
editor.putString("username","hfb");//存入string内容
editor.putInt("age",11);
edit.commit();
```

读取

```
SharedPreferences sp = getSharePreferences("data",mode_private);
String data = sp.getString("username","")
```

删除

```
editor.remove("name");
editor.clear();
edit.commit();
```

3.SQLite存储数据

运算速度快、占用资源少， 支持sql语法

并发读写性能不好

写入

查询

删除

修改

4.contentProvider存储数据

应用之间数据交换

5.网络存储

通过网络提供的存储空间来存储/获取数据信息

第七章 其他控件

对话框

ToolBar

导航栏

ProgressBar

进度条

下拉框

RatingBar

评分

第八章 进阶

##AarryAdapter

概念

ListView用于展示大量数据，需要借助适配器adapter来完成

构造方法

```
ArrayAdapter<String>(Context context,int resource,int textViewResourceId,String[]
objects)
ArrayAdapter(Context context,int resource, List<T> objects)
```

context 当前上下文环境

int resource 实例化使用的布局资源id

string【】 字符串数组

list 视图显示的集合

使用

```
Activity xml 中 加入 ListView控件
另一个xml 中写出ListView中item组成，比如设置一个imageview 和 textview
使用findViewById() 加载ListView后，实例化Adapter，然后为ListView加载Adapter
```

item_view.xml

SimpleAdapter

Fragment

让程序更加合理充分利用大屏幕的空间

```
创建类继承Fragment
实现onCreateView方法
静态使用或动态调用
```

不能独立存在，要嵌入activity中

Service

Service是Android中的四大组件之一，

长生命周期，没有可视化界面

运行于后台的一种程序，用户切换到另外的应用场景

Service将持续在后台运行

```
starService() 启动服务
stop 关闭
bind 绑定
unbind 解绑
```

```
onCreate() 创建服务
onStartCommand 开始服务
onDestroy        销毁服务
onBind           绑定服务
onUnBind         解绑服务
```

IntentService

1.service执行耗时任务，创建子线程来做，容易错误

2.于UI线程交互使用handler机制

提供intentService结合这两个

```
定义IntentService的子类：重写无参构造、复写onhandleintent方法
在manifest.xml中注册服务
activity中开启service服务
```

静态注册广播

广播是Android四大组件之一，全称Broadcast Receiver

分类

广播可分为 静态注册广播 动态注册广播

静态是在manifest中

动态是在代码中注册

有序广播

逻辑和无序广播一样，priority设置优先级，优先级高的先收到

数值越大，优先级越高

动态注册广播

要用onDestroy取消注册

习题 查漏

四大组件

activity

service

content provider

不同程序之间进行数据共享的标准api

最重要的就是数据模型和URI

broadcast receiver

实现不同组件之间的通信

使用了观察者模式

简答题

介绍五种布局

1.线性布局LinearLayout

2.相对布局RelativeLayout

3.表格布局

4.绝对布局

5.框架布局

adapter有什么用，常见的有什么

连接后端数据和前端显示的适配器接口

arrayadapter

simple

list

base

activity和intent

activity呈现了一个用户可操作的可视化用户界面

intent解决android的应用各组件之间的联系

android平台分层体系架构

应用程序层：核心应用程序的集合

应用框架层：提供构建应用程序用到的API

核心类库：包含了系统库和Android运行环境

Linux内核：为安卓设备提供了底层的驱动

通信技术

1G：语音蜂窝电话标准

2G：数字语音传输技术为核心，GSM，增加接收数据的功能

3G：图像、音乐、视频

4G：高质量图像、音乐、视频，最高**100Mbit/s**

5G：**10Gbit/s**

简述Android中的存储方式及特点。

Android平台提供的五种数据存储方式，分别为文件存储、SharedPreferences、SQLite数据库、ContentProvider和网络存储，这些存储方式的特点如下。

(1)文件存储：Android提供了openFileInput()和openFileOutput()方法来读取设备上的文件，其读取方式与Java中I/O程序是完全一样的。

(2)SharedPreferences：这是Android提供的用来存储一些简单的配置信息的一种机制，他采用了XML格式将数据存储到设备中。通常情况下，我们使用SharedPreferences存储一些应用程序的各种配置信息，如用户名、密码等。

(3)SQLite数据库：SQLite是Android自带的一个轻量级的数据库，他运算速度快，占用资源少，还支持基本SQL语法，一般使用他作为复杂数据的存储引擎，可以存储用户信息等。

(4)ContentProvider：Android四大组件之一，主要用于应用程序之间的数据交换，他可以将自己的数据共享给其他应用程序使用。

(5)网络存储：需要与Android网络数据包打交道，将数据存储到服务器上，通过网络提供的存储空间来存储/获取数据信息。

简述显式Intent和隐式Intent的区别。

显式意图和隐式意图的区别如下所示：

1、显式意图：需要明确指定激活组件的名称。

2、隐式意图：不明确指定组件名，Android系统会根据隐式意图使用IntentFilter匹配相应的组件，匹配的属性主要有action、category、data。

简述生命周期的方法及调用时机。

Activity一共有7个方法，这些方法和调用的时机具体如下：

1、onCreate()：Activity创建时调用，通常做一些初始化设置。

2、onStart()：Activity即将可见时调用。

3、onResume()：Activity获取焦点时调用。

4、onPause()：当前Activity被其他Activity覆盖或屏幕锁屏时调用。

5、onStop()：Activity对用户不可见时调用。

6、onDestroy()：Activity销毁时调用。

7、onRestart()：Activity从停止状态到再次启动时调用。

简述广播机制的实现过程。

Android中的广播使用了观察者模式，即基于消息的发布/订阅事件的模式。广播发送者和接收者分别处于观察者模式中的消息发布和订阅两端。广播机制的实现过程具体如下：

1. 广播接收者通过Binder机制在AMS(Activity Manager Service)中进行注册。
2. 广播发送者通过Binder机制向AMS发送广播。
3. AMS查找符合相应条件（IntentFilter/Permission）的广播接收者，将广播发送到相应的消息循环队列中。
4. 执行消息循环时获取到发送的广播，然后回调广播接收者中的onReceive()方法并在该方法中进行相关处理。

编写一个程序实现在界面中间显示“I Love Android”。

在activity_main.xml文件中编写布局的代码如下。

```
<?xml version="1.0" encoding="utf-8"?>

<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"

android:layout_width="match_parent"

android:layout_height="match_parent">

  <TextView
```

```
android:layout_width="wrap_content"

android:layout_height="wrap_content"

android:text="I Love Android"

android:layout_centerInParent="true"/>

</RelativeLayout>
```