一、前言

1、开发Android工具：

Eclipse+ADT插件、Android Studio

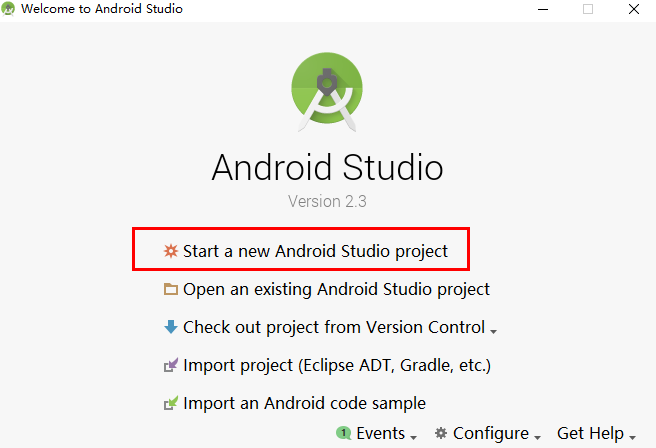
2、Android Studio工作环境：

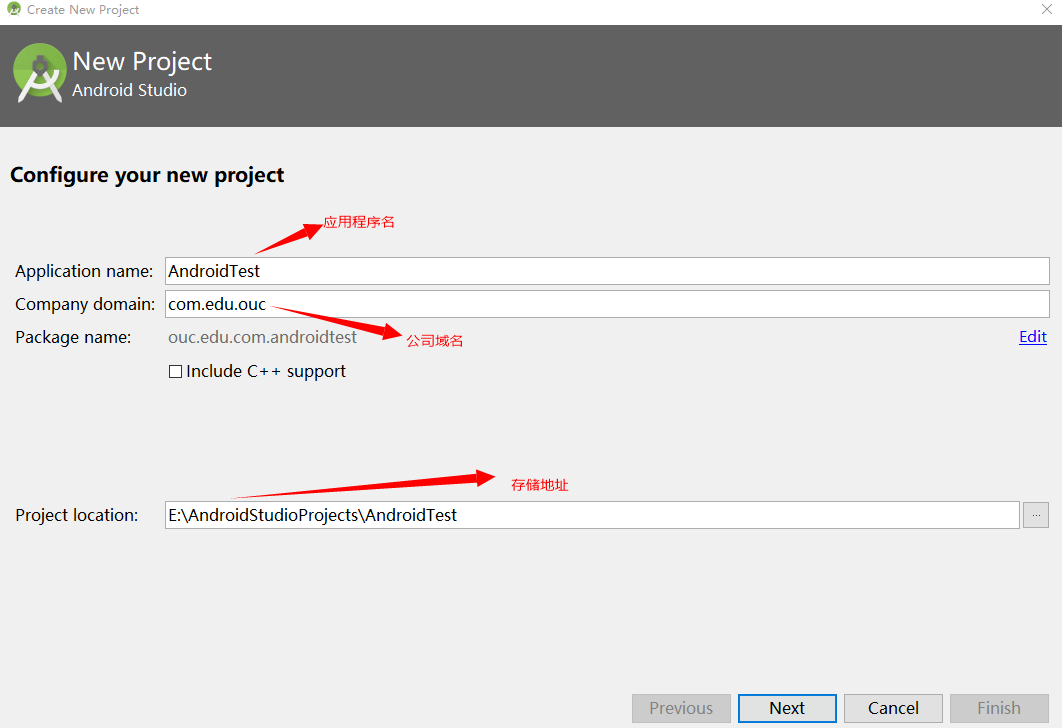
JDK

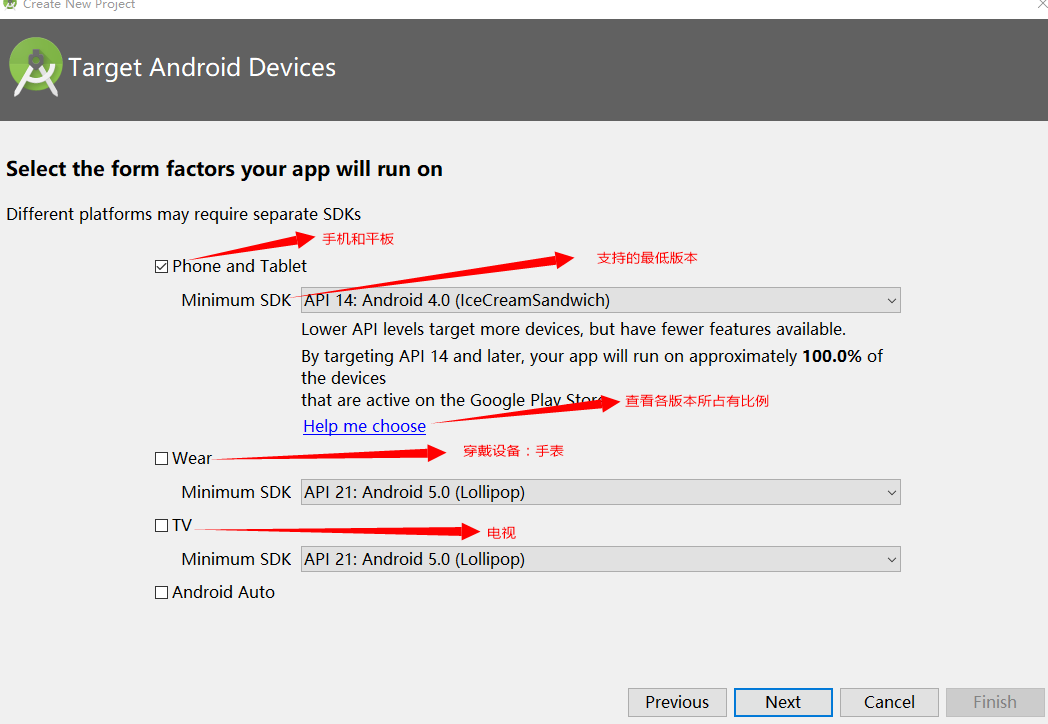
3、Android测试：

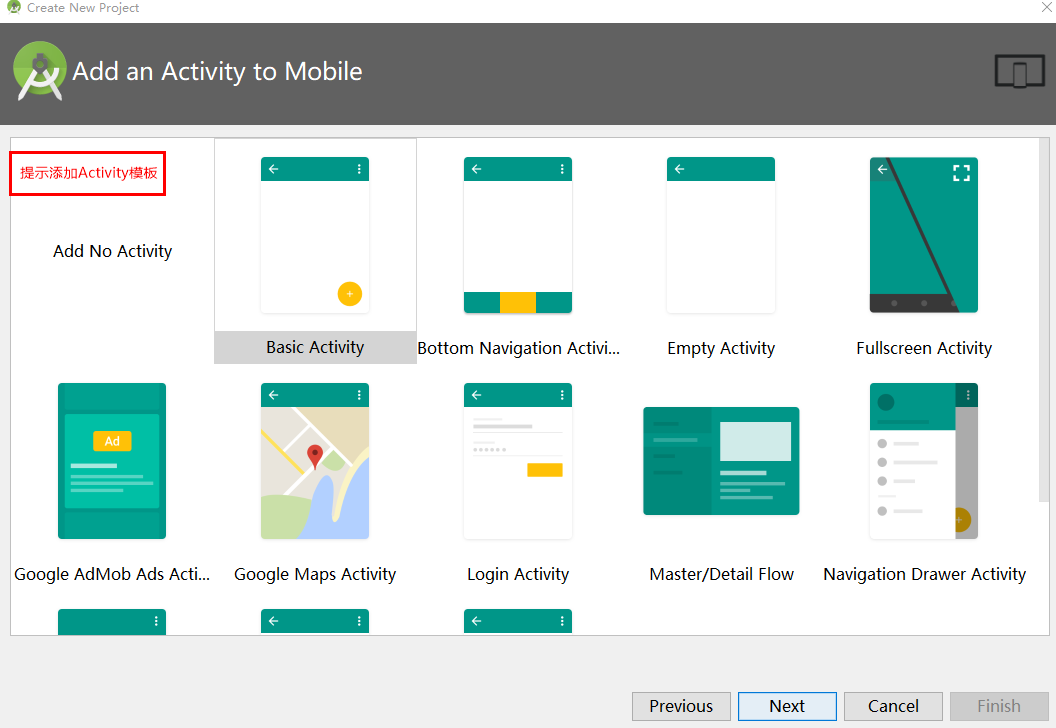
至少有一台安卓设备、Google Nexus模拟器

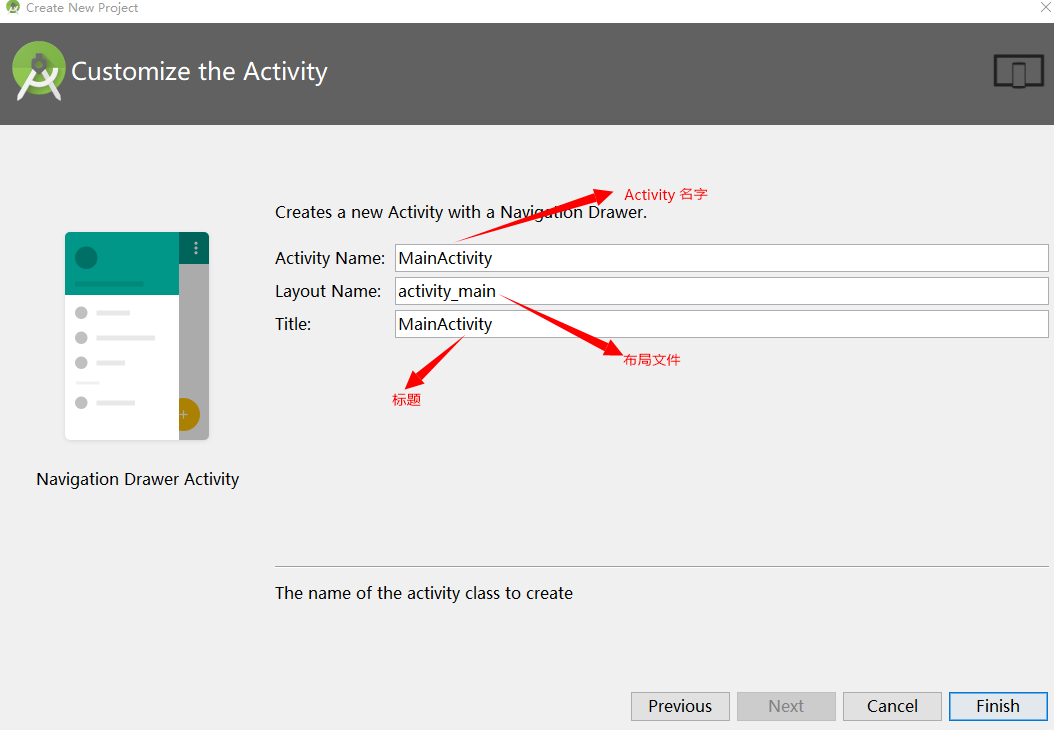
二、创建Android项目

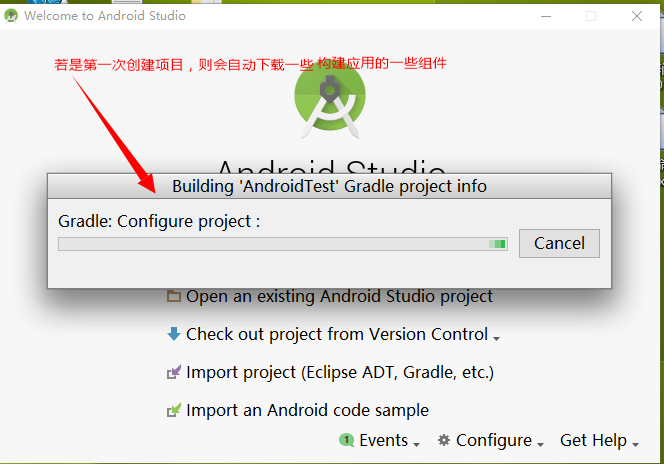


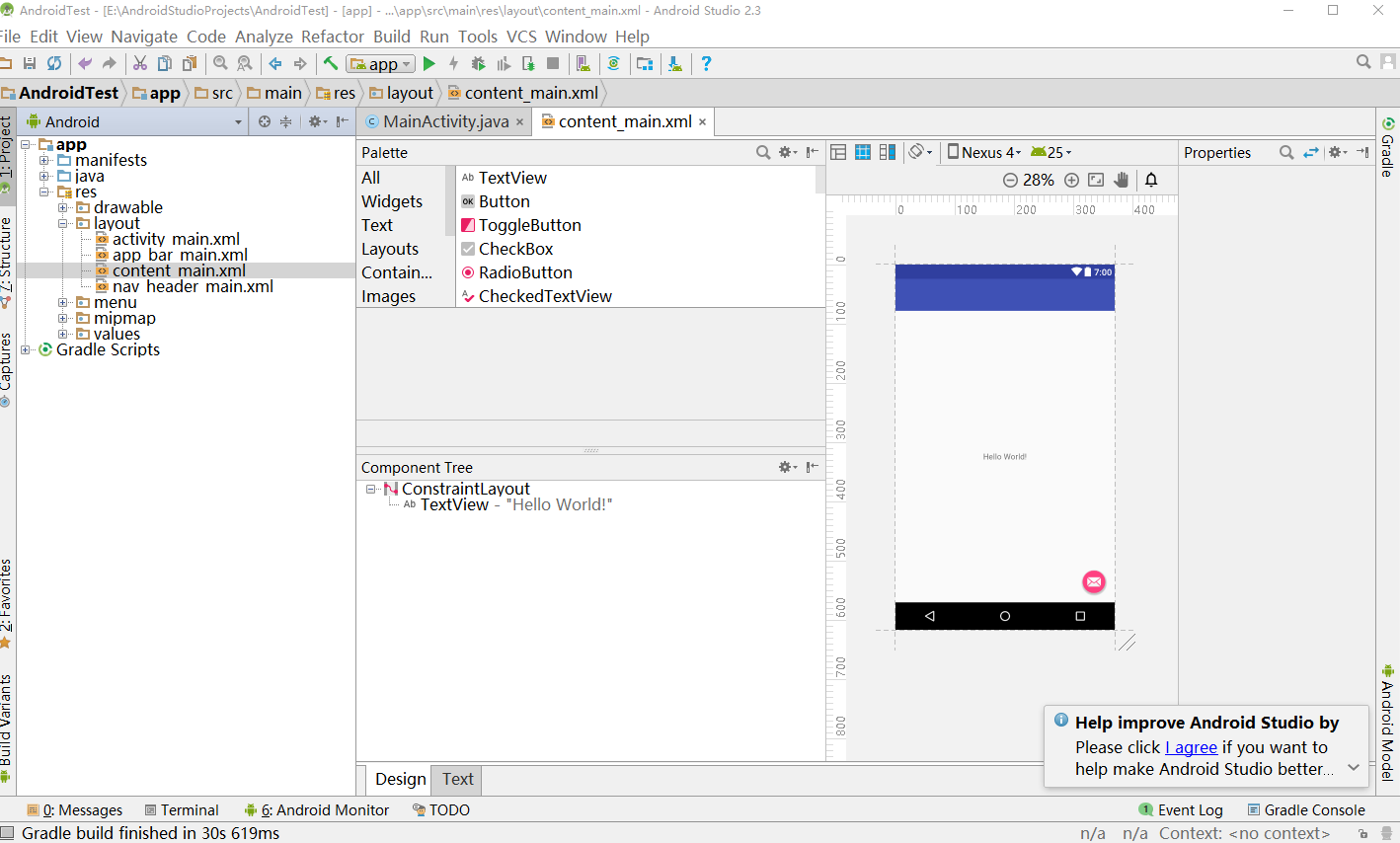




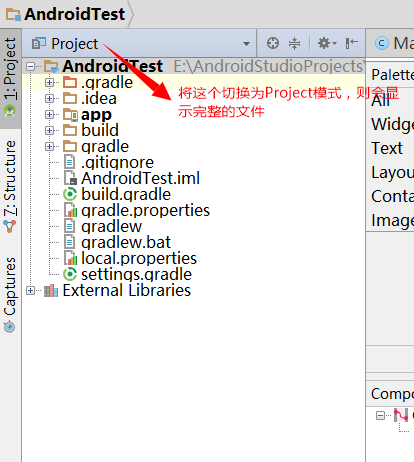


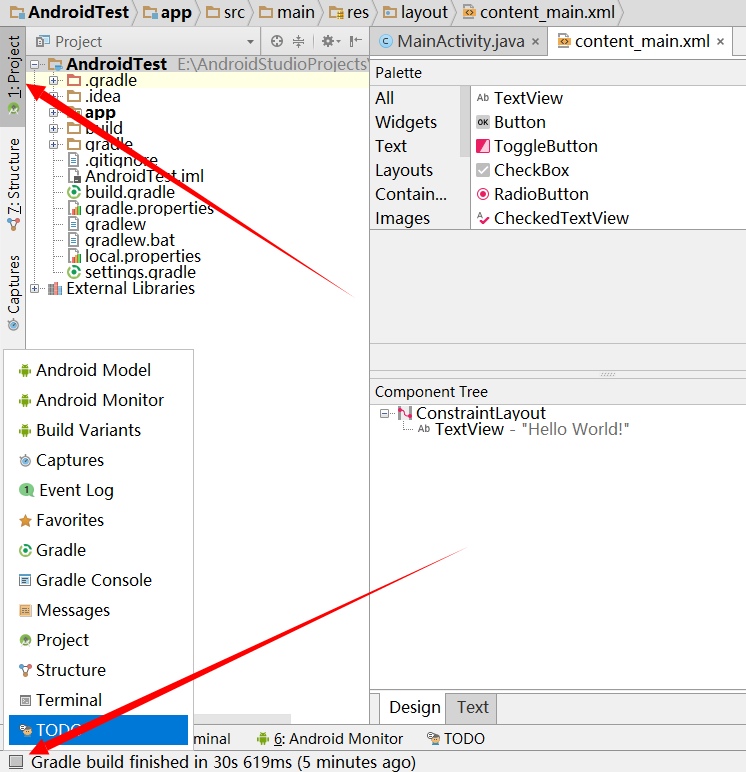






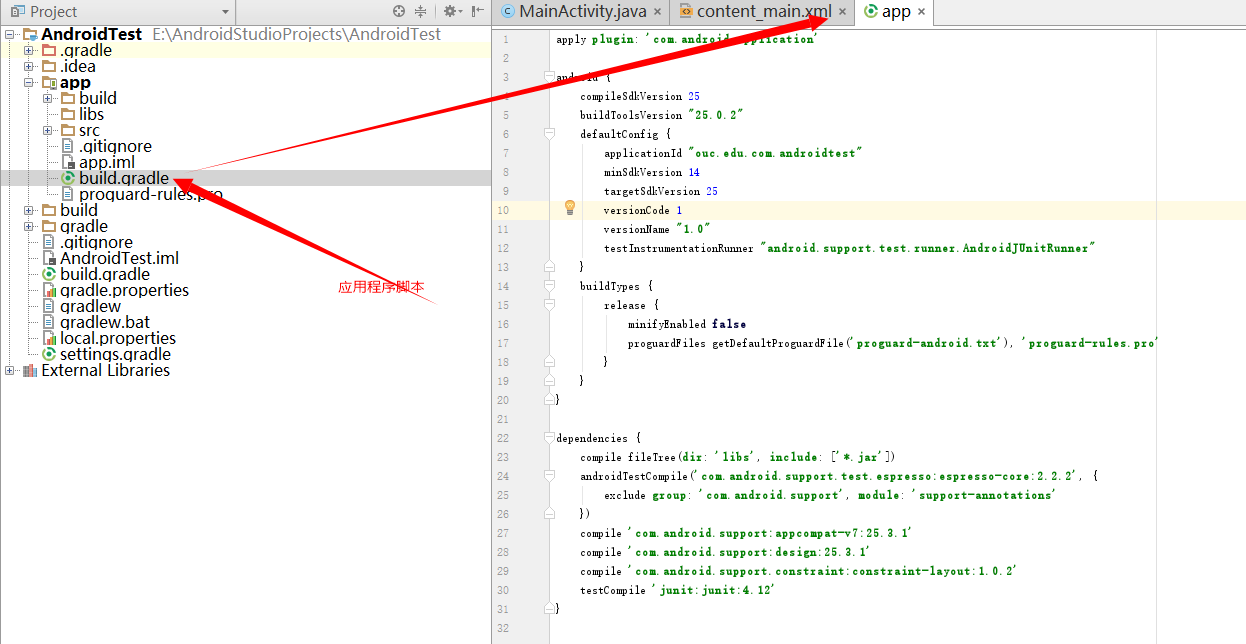
三、熟悉Android Studio运行环境

若在左侧没有Project窗口，可以点击左下角，选择Project



1、App下的Build.gradle:构建应用程序的脚本

通过这个脚本，告诉系统构建那个版本的应用程序



2、Android外观设置

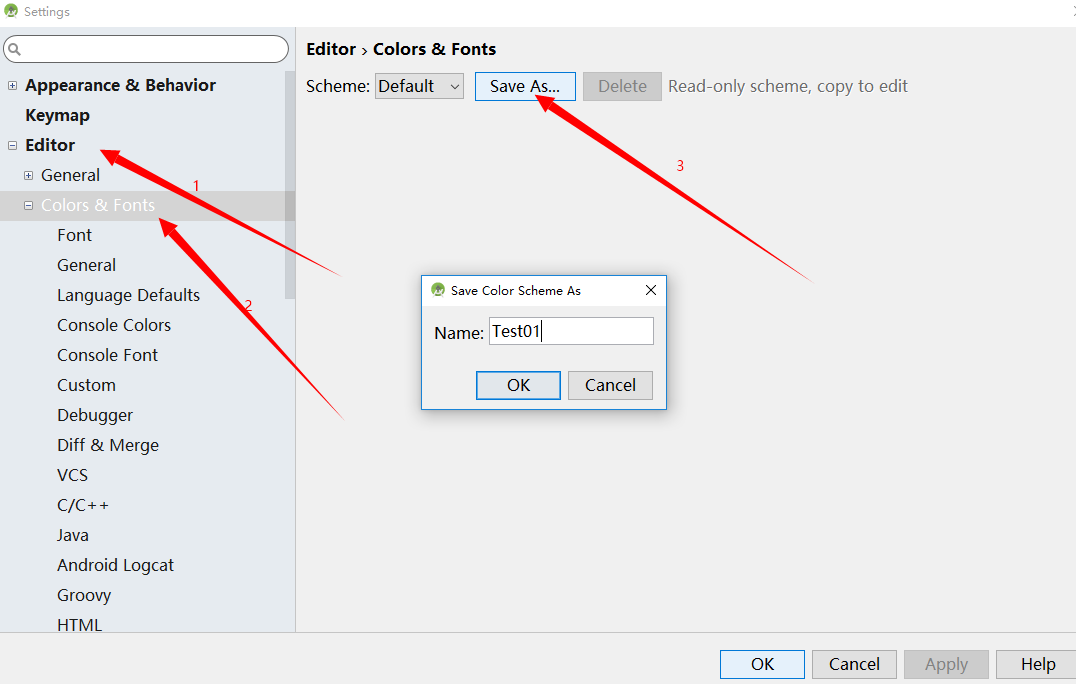
File->Setting

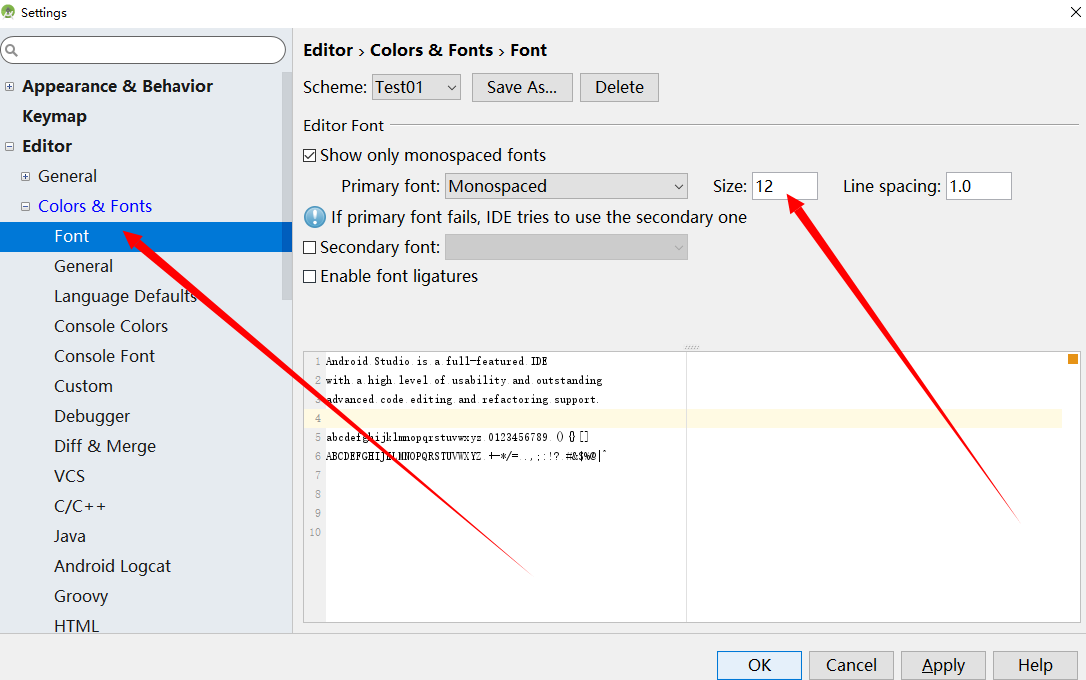
3、显示行号

File->Setting->Editor->General->Appearance->选中Show line numbers

4、自定义字体

File->Setting->Editor->Colors&Fonts->Save As输入标识名称，->ok->Font->输入字体大小即可





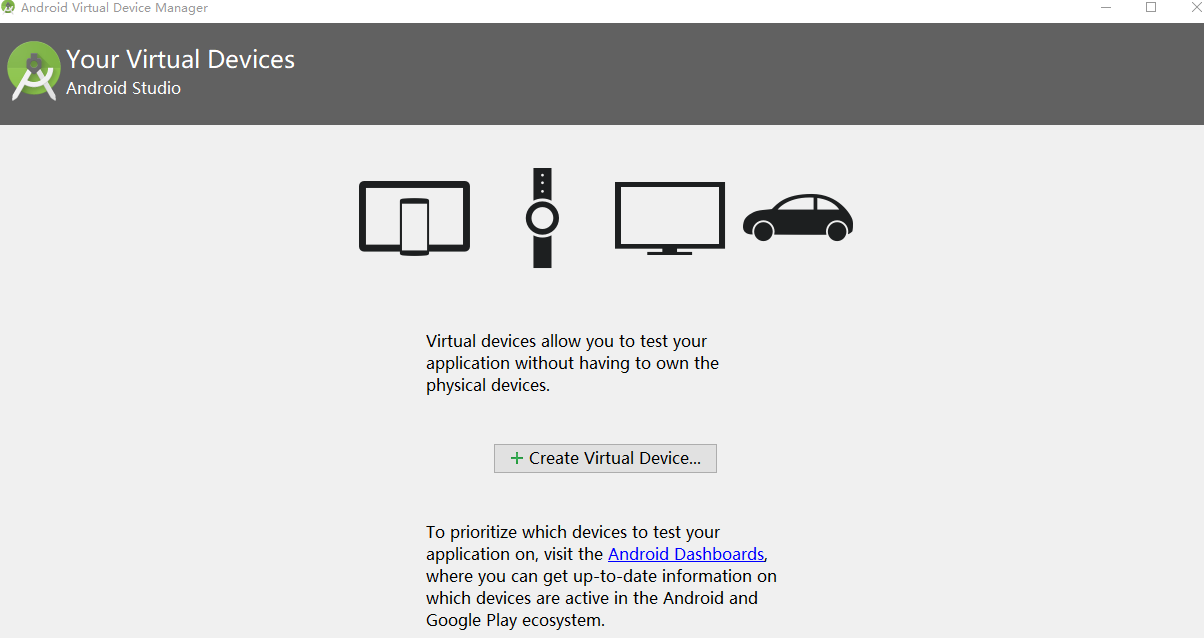
5、查看日志

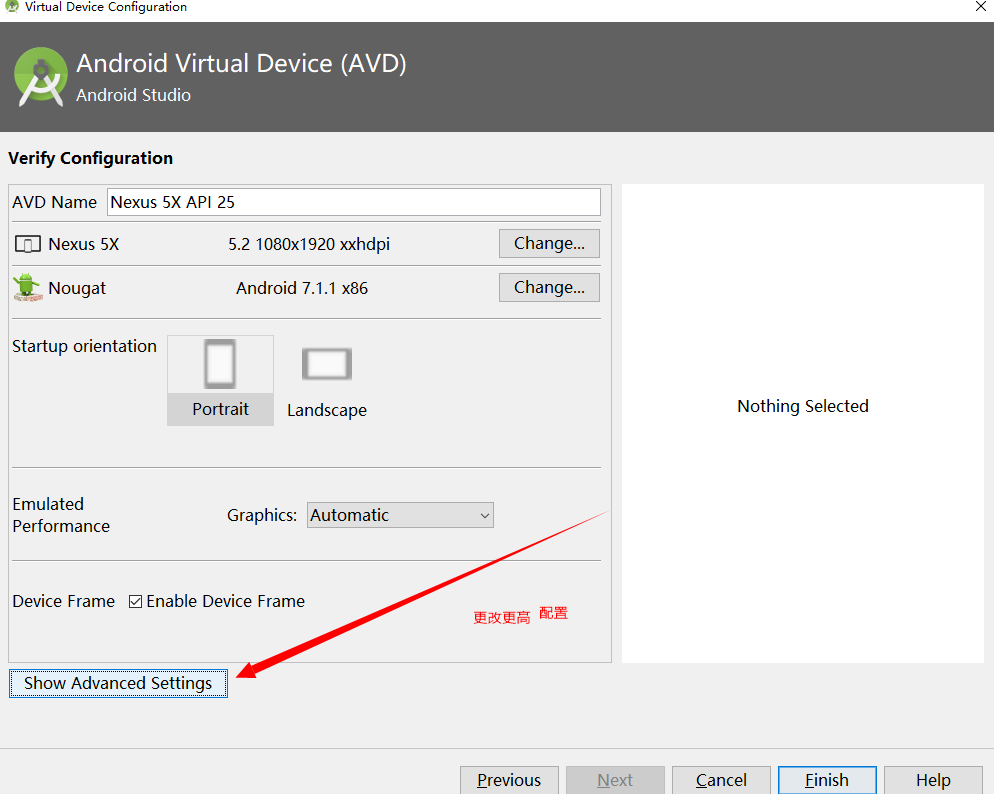
Android monitor->logcat

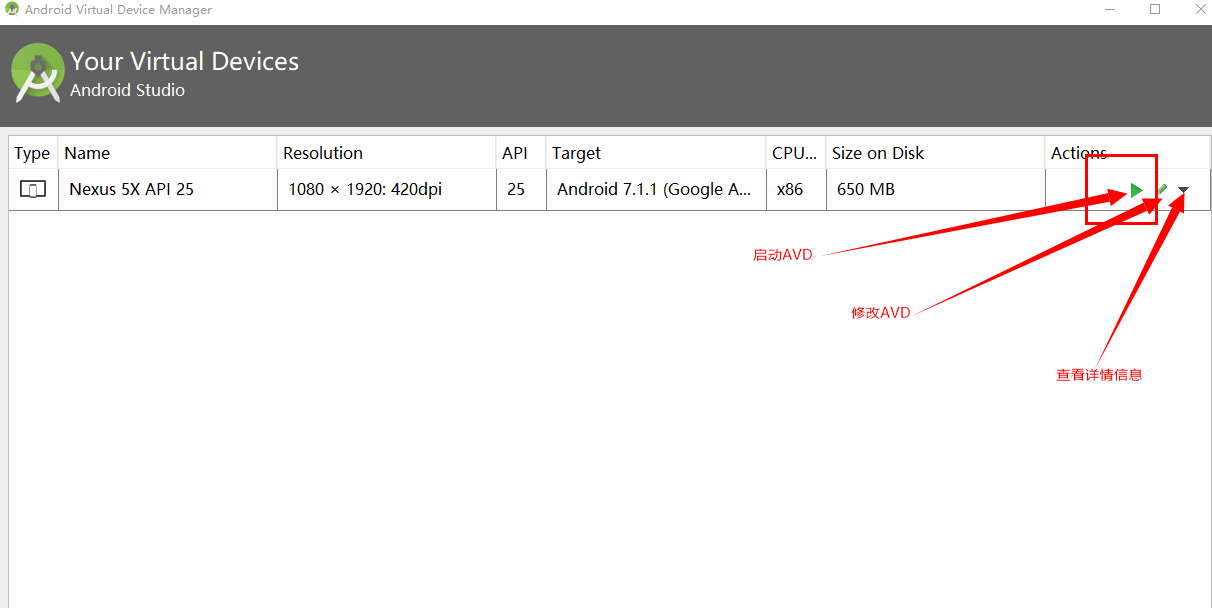
五、测试应用程序

1、android虚拟设备

Tools->Android->AVD Manager

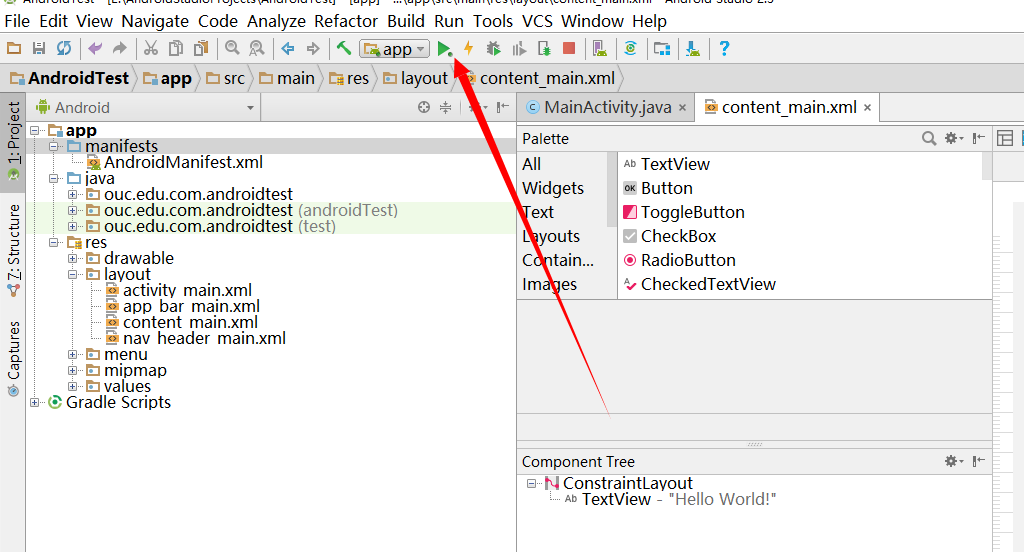




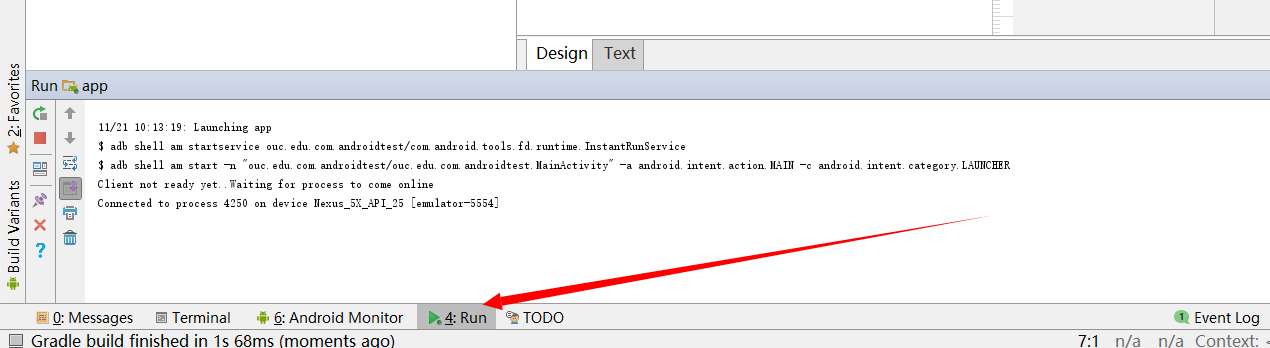


2、在虚拟设备下运行Android项目

（1）直接点击运行按钮



（2）查看日志信息

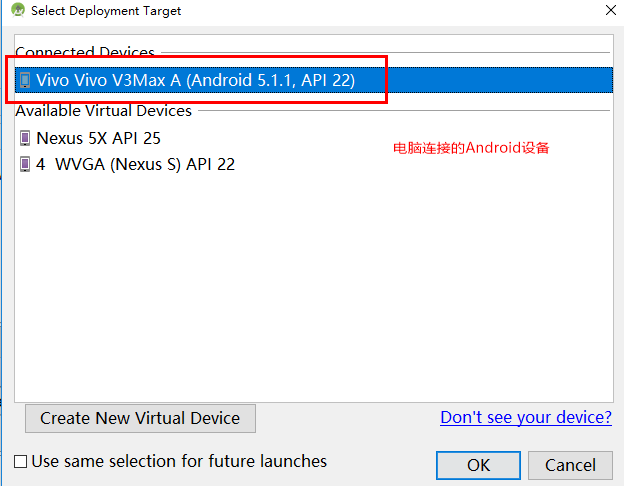


3、在真机上运行Android项目

（1）Tools->Android->SDK Manager,安装Google USB Driver

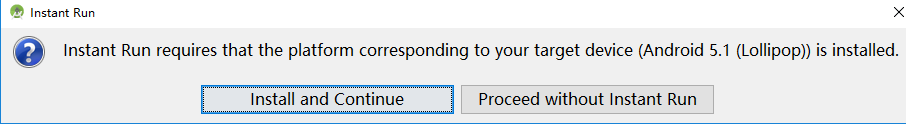
（2）将Android手机通过USB连接电脑，并打开手机的USB调试

（3）直接点击运行项目，我的手机是Vivo V3 MAXA Android5.1.1



出现一个提示框：即时运行需要安装与目标设备（Android 5.1（Lollipop））相对应的平台。

此时选择安装并继续或选择不需要安装也可以，首次安装会在手机弹出是否信任和安装界面，直接安装即可。



六、Android文件目录介绍

将模式切换到Project模式下

1、src->main->java——java源代码存放目录

2、src->main->res——存放资源文件（图片、xml文件等）

3、AndroidManifest.xml——功能清单文件。这个文件列出了应用程序所提供的功能

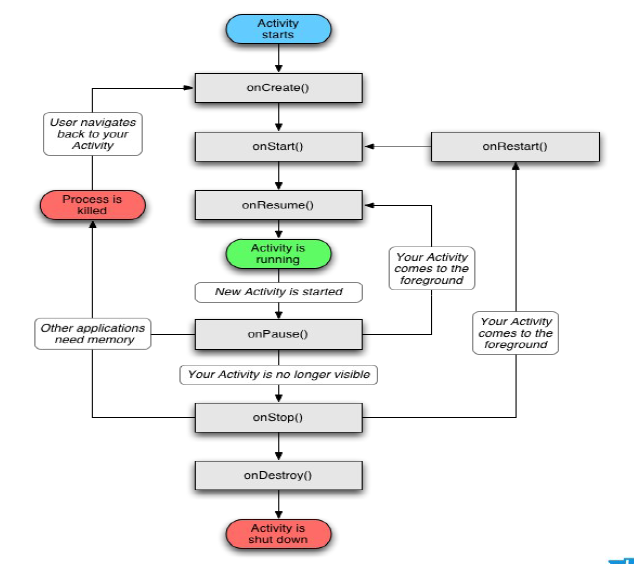
七、Activity

1、简介：

* Activity是Android四大基本组件之一
* 每个Activity对应一个软件界面
* 一个Android程序可以由一个或多个Activity组成
* 对于有界面的程序来说，程序的入口就是一个Activity

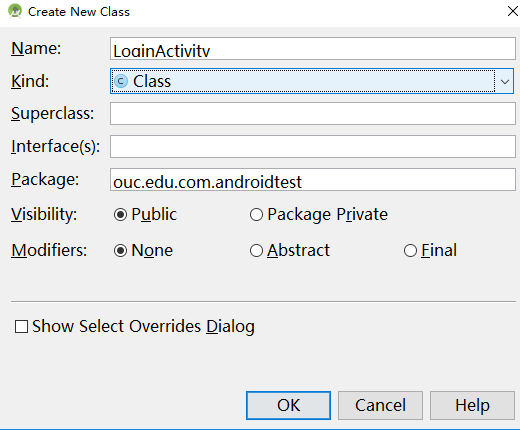
2、Activity生命周期

* 运行——当它在屏幕前台时（位于当前任务堆栈的顶部），它是激活或运行状态。它就是响应用户操作的Activity。方法：Oncreate,Onstart,Onresume
* 暂停——当它失去焦点但仍然对用户可见时，它处于暂停状态。方法：Onresume,Onpuse 若再次重新运行，会调用Onresume方法
* 停止——完全被另一个Activity覆盖时则处于停止状态。它仍然保留所有的状态和成员信息。然而对用户是不可见的，所以它的窗口将被隐藏，如果其它地方需要内存，则系统经常会杀死这个Activity。方法：Onpuse，OnStop,Ondestroy若再次重新运行，会调用Onrestart，Onstart，Onresume方法



3、创建一个Activity

* 在src->main->java文件下新建一个java class文件，并继承Activity，实现其Oncreate方法



* 每个Activity都要加载一个视图，即xml文件

在src->main->res->layout下新建一个布局文件为activity\_login.xml文件，并在LoginActivity中添加此视图setContentView(R.layout.***activity\_login***);

* 在AndroidManifest.xml中注册此Activity（必须在Application标签内注册才有效）

在src->main下的AndroidManifest.xml的Application标签内添加中

<**activity android:name=".LoginActivity"**></**activity**>

注：

<**intent-filter**> *<!—标明应用程序的入口程序，此标签在那个Activity下，应用程序就会先启动哪个Activity-->* <**action android:name="android.intent.action.MAIN"** />  
 <**category android:name="android.intent.category.LAUNCHER"** />  
</**intent-filter**>

八、添加控件方法

1、通过代码添加

在src->main->res->layout下的activity\_main.xml的Test模式下添加

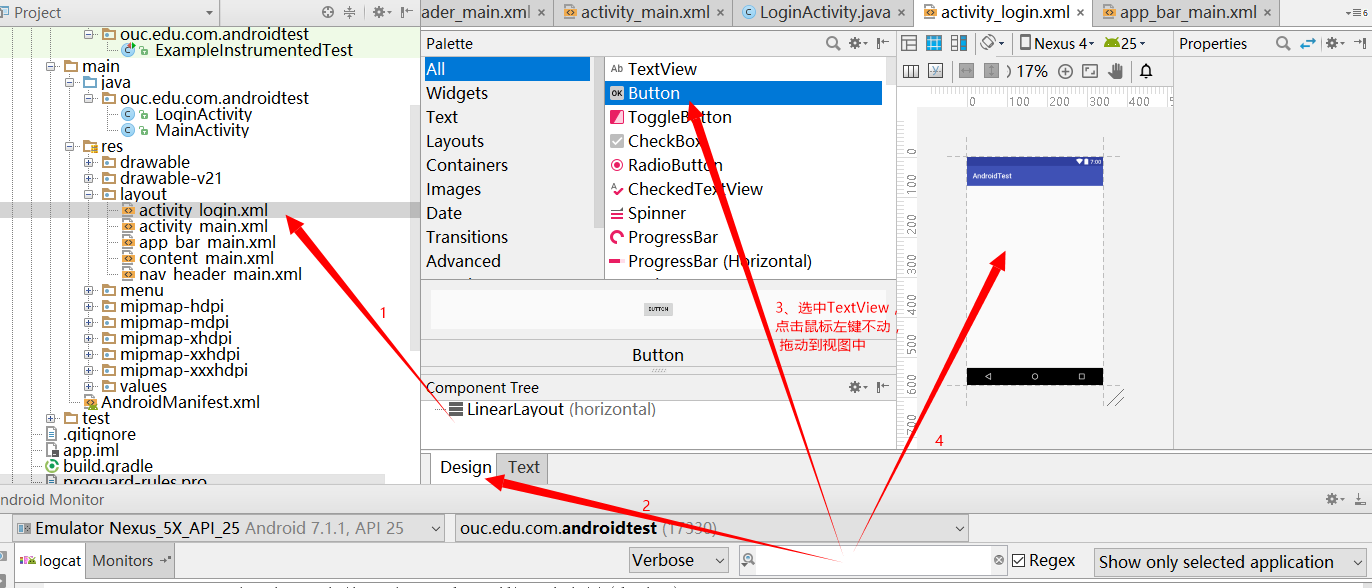
<TextView android:id="@+id/ tv\_test"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="TextView"/>

注：**wrap\_content**：自适应

fill\_content:充满父控件

2、通过拖动控件添加

在src->main->res->layout下的activity\_main.xml的Design模式下，将要添加的控件拖动到视图上



九、添加Button的Click事件

1、为视图添加一个button按钮

2、在Activity中添加代码：

//声明一个Button对象  
private Button button;  
@Override  
protected void onCreate(@Nullable Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 //这里可以使用setContentView(R.layout.xxx) 显示某个视图....  
 setContentView(R.layout.activity\_login);  
 //将通过id找到button实例并强制转换成Button类型  
 button=(Button)findViewById(R.id.btn\_test);  
 //添加button的点击事件  
 button.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 }  
 });  
}

注：若有多个点击事件，可将其进行归类

在Activity中实现OnClickListener接口

Button1.setOnClickListener(this);

Button2.setOnClickListener(this);

Public void onClick(View v){

Switch(v.getId()){

Case R.id.start:

Break;

Default:

Break;

}

}

九、控件的使用

1、RatingBar是很重要的评星控件，在很多软件中都有这评星的功能。在Android中系统也提供了一个专门评星的控件： RatingBar

• 相关属性：

– Android:numStars：指定评星数量

– Android:rating ：指定当前分数

– Android:stepSize：指定分数的增量单位

* 获取星星分数

实现 OnRatingBarChangeListener接口，并重写其星星改变函数

@Override  
public void onRatingChanged(RatingBar ratingBar, float rating, boolean fromUser) {  
 //得到当前星星的数量  
 ratingBar.setRating(rating);  
 //如果当前的星星iD和用户操作的ID是同一个  
 if(ratingBar.getId()==R.id.ratingBar2){  
 tv.setText("分数："+rating);  
 }  
}

2、Gallery一般用于显示图像列表，也可以称为相册控件。这里重点，他需要一个适配器，那么我们需要提供自己的适配器（adapter）。

九、Intent

1、概念：

Intent:意图，负责对应用中一次操作的动作、动作涉及数据、附加数据进行描述 。所以Intent在这里主要是一个媒介的作用。

2、作用

* Intent可以显示Activity，可以发送广播和启动服务。
* 在同一个应用程序中往往会使用Intent对象来指定一个Activity，并通过startActivity或startActivityForResult方法启动一个Activity。
* 通过Intent在不同的Activity中进行携带数据

3、通过Intent实现Activity之间的跳转

* 创建两个Activity（包含两个layout文件）

4、通过Intent传递数据

Intent传递数据方法：putExtra

* 在第一个Activity中的添加传递数据方法

Intent intent=new Intent(LoginActivity.this,MainActivity.class);  
intent.putExtra("name","通过Intent传递过来的数据");  
startActivity(intent);

* 在第二个Activity中添加获取数据方法

String val=getIntent().getStringExtra("name");

十、Service

1、概念

* 是一个应用程序组件，没有图形界面，通常用来处理一些耗时比较长的操作（闹钟）。

2、生命周期

OnCreat（）

OnStartCommand（）

OnDestroy（）

3、创建一个Service

* 创建一个Service类，并实现Serice
* 在一个Activity中启动Service

Intent i=new Intent(LoginActivity.this,MyService.class);  
startService(i);

* 在Manifest.xml中注册Service(application标签内)

<service android:name=”MyService”></service>

十一、BroadcastRecevicer

1、简介

2、创建一个广播接收器

* 发送广播消息

Intent intent =new Intent(LoginActivity.this,MyReceive.class);

intent.putExtra("name","通过Intent传递过来的数据");  
sendBroadcast(intent);

* 接收广播消息

创建一个Broadcast类，并实现BroadcastRecevicer，重写Onreceive方法

Intent.getStrubgExtra(“ name”);

* 在Manifest.xml中注册BroadcastRecevicer (application标签内)

<receiver android:name=”MyReceiver”></ receiver >

十二、Handler

1、简介

2、创建一个Handler

* 通过handler的post方法启动runnable

handler.post(runnable);

* 移除runnable

handler.removeCallbacks(runnable)

* 实例化Handler

Handler handler=new Handler();

Runnable runnable=new Runnable(){

Public void run(){

Systemt.out.println(“runnable”);

//handler是一个消息队列，每执行一次就会传递一个runnable

handler.postDelayed(runnable,3000);//每隔3000毫秒执行一次

}

};

十三、UI组件与布局

1、线性布局：线性布局又分为水平布局和垂直布局，可通过orientation属性进行设置。标签中有一个非常重要的属性：Gravity属性。

* android:layout\_weight**="1"**通过设置控件的layout\_weight属性以控制各个控件在布局中的**相对大小**,线性布局会根据该控件layout\_weight值与其所处布局中所有控件layout\_weight值之和的比值为该控件分配占用的区域。
* android:gravity　属性是对该view中内容的限定．比如一个button 上面的text. 你可以设置该text 相对于view的靠左，靠右等位置．
* 这两个属性可选的值有：top、bottom、left、right、center\_vertical、fill\_vertical、center\_horizontal、fill\_horizontal、center、fill、clip\_vertical。一个属性可以包含多个值，需用“|”分开。其含义如下：

|  |  |
| --- | --- |
| top | 将对象放在其容器的顶部，不改变其大小. |
| bottom | 将对象放在其容器的底部，不改变其大小. |
| left | 将对象放在其容器的左侧，不改变其大小. |
| right | 将对象放在其容器的右侧，不改变其大小. |
| center\_vertical | 将对象纵向居中，不改变其大小.  垂直对齐方式：垂直方向上居中对齐。 |
| fill\_vertical | 必要的时候增加对象的纵向大小，以完全充满其容器.  垂直方向填充 |
| center\_horizontal | 将对象横向居中，不改变其大小.  水平对齐方式：水平方向上居中对齐 |
| fill\_horizontal | 必要的时候增加对象的横向大小，以完全充满其容器.  水平方向填充 |
| center | 将对象横纵居中，不改变其大小. |
| fill | 必要的时候增加对象的横纵向大小，以完全充满其容器. |
| clip\_vertical | 附加选项，用于按照容器的边来剪切对象的顶部和/或底部的内容. 剪切基于其纵向对齐设置：顶部对齐时，剪切底部；底部对齐时剪切顶部；除此之外剪切顶部和底部.  垂直方向裁剪 |
| clip\_horizontal | 附加选项，用于按照容器的边来剪切对象的左侧和/或右侧的内容. 剪切基于其横向对齐设置：左侧对齐时，剪切右侧；右侧对齐时剪切左侧；除此之外剪切左侧和右侧.  水平方向裁剪 |

注：

* center\_vertical和center\_horizontal两个属性值，center\_vertical是指将对象在垂直方向上居中对齐，即在从上到下的方向上选择中间的位置放好；center\_horizontal是指将对象水平方向上居中对齐，即在从左到右的方向上选择中间的位置放好。
* 当我们采用LinearLayout布局时，有以下特殊情况需要我们注意：  
  (1)当 android:orientation="vertical"  时， android:layout\_gravity只有水平方向的设置才起作用，垂直方向的设置不起作用。即：left，right，center\_horizontal 是生效的。

(2)当 android:orientation="horizontal" 时， android:layout\_gravity只有垂直方向的设置才起作用，水平方向的设置不起作用。即：top，bottom，center\_vertical 是生效的。

2、相对布局

* 相对布局可以设置某一个视图相对于其他视图的位置，这些位置包括上、下、左、右。以及可以设置其相对于父级控件的位置。
* 相对布局属性

android:layout\_marginTop="25dip" //顶部距离

android:gravity="left" //空间布局位置

android:layout\_marginLeft="15dip //距离左边距

// 相对于给定ID控件

android:layout\_above 将该控件的底部置于给定ID的控件之上;

android:layout\_below 将该控件的底部置于给定ID的控件之下;

android:layout\_toLeftOf 将该控件的右边缘与给定ID的控件左边缘对齐;

android:layout\_toRightOf 将该控件的左边缘与给定ID的控件右边缘对齐;

android:layout\_alignBaseline 将该控件的baseline与给定ID的baseline对齐;

android:layout\_alignTop 将该控件的顶部边缘与给定ID的顶部边缘对齐;

android:layout\_alignBottom 将该控件的底部边缘与给定ID的底部边缘对齐;

android:layout\_alignLeft 将该控件的左边缘与给定ID的左边缘对齐;

android:layout\_alignRight 将该控件的右边缘与给定ID的右边缘对齐;

// 相对于父组件

android:layout\_alignParentTop 如果为true,将该控件的顶部与其父控件的顶部对齐;

android:layout\_alignParentBottom 如果为true,将该控件的底部与其父控件的底部对齐;

android:layout\_alignParentLeft 如果为true,将该控件的左部与其父控件的左部对齐;

android:layout\_alignParentRight 如果为true,将该控件的右部与其父控件的右部对齐;

// 居中

android:layout\_centerHorizontal 如果为true,将该控件的置于水平居中;

android:layout\_centerVertical 如果为true,将该控件的置于垂直居中;

android:layout\_centerInParent 如果为true,将该控件的置于父控件的中央;

// 指定移动像素

android:layout\_marginTop 上偏移的值;

android:layout\_marginBottom 下偏移的值;

android:layout\_marginLeft 　　左偏移的值;

android:layout\_marginRight 　 右偏移的值;

十四、SQLite数据库存储

1、特点

* 轻量级

SQLite和C\S模式的数据库软件不同，它是进程内的数据库引擎，因此不存在数据库的客户端和服务器。使用SQLite一般只需要带上它的一个动态库，就可以享受它的全部功能。而且那个动态库的尺寸也相当小

* 独立性

SQLite数据库的核心引擎本身不依赖第三方软件，使用它也不需要“安装”，所以在使用的时候能够省去不少麻烦

* 隔离性

SQLite数据库中的所有信息（比如表、视图、触发器）都包含在一个文件内，方便管理和维护

* 跨平台

SQLite数据库支持大部分操作系统，除了我们在电脑上使用的操作系统之外，很多手机操作系统同样可以运行，比如Android、WindowsMobile、Symbian、Palm等

2、简介

* 多语言接口

SQLite数据库支持很多语言编程接口，比如C\C++、Java、Python、dotNet、Ruby、Perl等，得到更多开发者的喜爱。

* 安全性

SQLite数据库通过数据库级上的独占性和共享锁来实现独立事务处理。这意味着多个进程可以在同一时间从同一数据库读取数据，但只有一个可以写入数据。在某个进程或线程向数据库执行写操作之前，必须获得独占锁定。在发出独占锁定后，其他的读或写操作将不会再发生

3、使用SqlLite

十五、SharedPreference储存

1、简介

是Android提供的一种轻量级的数据存储方式，主要用来存储一些简单的配置信息，例如，默认欢迎语，登录用户名和密码等。其以键值对的方式存储,使得我们能很方便进行读取和存入

3、文件储存

4、网络储存

Json转化为实体类

* 安装Gson

在build.gradle文件中的dependencie标签内添加

compile 'com.google.code.gson:gson:2.7'

* 使用

try {  
 JSONObject jsonObject=new JSONObject(json);  
 JSONArray jsonArray=jsonObject.getJSONArray("data");  
 List<UserInfoModel> userInfoModelList=new ArrayList<UserInfoModel>();  
 //这两句代码必须的，为的是初始化出来gson这个对象，才能拿来用  
 Type type1=new TypeToken<List<UserInfoModel>>(){}.getType();  
 userInfoModelList=new Gson().fromJson(jsonArray.toString(),type1);  
 System.out.println("----1212-----"+userInfoModelList.size());  
} catch (JSONException e) {  
 e.printStackTrace();  
}

设置连接超时

HttpClient httpClient=new DefaultHttpClient();  
HttpParams params = null;  
params = httpClient.getParams();  
//若3秒未连接上，则强制断开  
HttpConnectionParams.setConnectionTimeout(params, 3000);  
//若连接上3秒之后没有接收到数据，则强制断开  
HttpConnectionParams.setSoTimeout(params, 3000);

让一个Activity停留几秒后再跳转

private void getHomeActivity(final Context context) {  
 Timer timer=new Timer();  
 TimerTask task=new TimerTask(){  
 public void run(){

Intent intent = new Intent(InitActivity.this, MainActivity.class)；

startActivity(intent);  
 }  
 };  
 timer.schedule(task, 2000);  
}

自动登录

* 用户信息管理类

package ouc.edu.com.function;  
import android.content.Context;  
import android.content.SharedPreferences;  
import android.text.TextUtils;  
import ouc.edu.com.model.UserInfoModel;  
/\*\*  
 \* Created by JHC on 2017/11/23.  
 \* 自动登录  
 \*/  
public class AutoMaticLogin {  
 private static AutoMaticLogin instance;  
 private AutoMaticLogin() {  
 }  
 public static AutoMaticLogin getInstance() {  
 if (instance == null) {  
 instance = new AutoMaticLogin();  
 }  
 return instance;  
 }  
 /\*\*  
 \* 保存自动登录的用户信息  
 \*/  
 public void saveUserInfo(Context context, String username, String password) {  
 SharedPreferences sp = context.getSharedPreferences("userInfo", Context.MODE\_PRIVATE);//Context.MODE\_PRIVATE表示SharePrefences的数据只有自己应用程序能访问。  
 SharedPreferences.Editor editor = sp.edit();  
 editor.putString("USER\_NAME", username);  
 editor.putString("PASSWORD", password);  
 editor.commit();  
 }  
 /\*\*  
 \* 获取用户信息model  
 \*  
 \* @param context  
 \* @param  
 \* @param  
 \*/  
 public UserInfoModel getUserInfo(Context context) {  
 SharedPreferences sp = context.getSharedPreferences("userInfo", Context.MODE\_PRIVATE);  
 UserInfoModel userInfo = new UserInfoModel();  
 userInfo.setLgname(sp.getString("USER\_NAME", ""));  
 userInfo.setLgpwd(sp.getString("PASSWORD", ""));  
 return userInfo;  
 }  
 /\*\*  
 \* userInfo中是否有数据  
 \*/  
 public boolean hasUserInfo(Context context) {  
 UserInfoModel userInfo = getUserInfo(context);  
 if (userInfo != null) {  
 if ((!TextUtils.isEmpty(userInfo.getLgname())) && (!TextUtils.isEmpty(userInfo.getLgpwd()))) {//有数据  
 return true;  
 } else {  
 return false;  
 }  
 }  
 return false;  
 }  
  
}

* 用户信息实体类

package ouc.edu.com.model;  
/\*\*  
 \* Created by JHC on 2017/11/22.  
 \* 用户信息Model  
 \*/  
public class UserInfoModel {  
 /\*\*  
 \*  
 \*/  
 private static final long serialVersionUID = 5790767399049546805L;  
 private int id; // id  
 private String lgname; // 登录名  
 private String lgpwd; // 登录密码  
 private String truename; // 真实姓名  
 private String role; // 角色  
 private String phone; // 电话  
 private String unit; // 单位  
 public int getId() {  
 return id;  
 }  
 public void setId(int id) {  
 this.id = id;  
 }  
 public String getLgname() {  
 return lgname;  
 }  
 public void setLgname(String lgname) {  
 this.lgname = lgname;  
 }  
 public String getLgpwd() {  
 return lgpwd;  
 }  
 public void setLgpwd(String lgpwd) {  
 this.lgpwd = lgpwd;  
 }  
 public String getTruename() {  
 return truename;  
 }  
 public void setTruename(String truename) {  
 this.truename = truename;  
 }  
 public String getRole() {  
 return role;  
 }  
 public void setRole(String role) {  
 this.role = role;  
 }  
 public String getPhone() {  
 return phone;  
 }  
 public void setPhone(String phone) {  
 this.phone = phone;  
 }  
 public String getUnit() {  
 return unit;  
 }  
 public void setUnit(String unit) {  
 this.unit = unit;  
 }  
}

* 登陆成功后写入

String userName = editText\_username.getText().toString();  
String userPwd = editText\_password.getText().toString();  
AutoMaticLogin.getInstance().saveUserInfo(LoginActivity.this, userName, userPwd);

* 自动登录

if (AutoMaticLogin.getInstance().hasUserInfo(context))//自动登录判断，SharePrefences中有数据，则跳转到主页，没数据则跳转到登录页  
{  
   
} else {  
   
}

主页面Tab

新闻列表

用了fragment，这里不再叙述

* 新闻页面

<**RelativeLayout** 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"  
 tools:context="com.example.administrator.wechat.fragment.DiscoveryFragment">  
 <!-- TODO: Update blank fragment layout -->  
 <ListView  
 android:id="@+id/lv\_task\_ok"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent" />  
</RelativeLayout>

* ListView listview\_fragmeng\_task.xml

<?xml version="1.0" encoding="utf-8"?>  
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:layout\_margin="10dp">  
  
 <LinearLayout  
 android:layout\_width="wrap\_content"  
 android:layout\_height="60dp"  
 android:layout\_margin="10dp"  
 android:orientation="horizontal"  
 android:gravity="center">  
 <TextView  
 android:id="@+id/tv\_tis"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_weight="1"  
 android:singleLine="true"  
 android:text="[已完成]"  
 android:textSize="20dp" />  
 </LinearLayout>  
  
 <LinearLayout  
 android:layout\_marginLeft="25dp"  
 android:layout\_width="match\_parent"  
 android:layout\_height="60dp"  
 android:layout\_margin="10dp"  
 android:orientation="vertical"  
 >  
 <TextView  
 android:id="@+id/tv\_title"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:singleLine="true"  
 android:text="11111"  
 android:textSize="18dp"/>  
 <TextView  
 android:id="@+id/tv\_info"  
 android:layout\_marginTop="10dp"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="11111"  
 android:textSize="14dp"/>  
 </LinearLayout>  
 </LinearLayout>

* 自定义适配器
* **package** com.edu.ouc.adapter;  
    
  import android.content.Context;  
  import android.support.v7.widget.RecyclerView;  
  import android.view.LayoutInflater;  
  import android.view.View;  
  import android.view.ViewGroup;  
  import android.widget.BaseAdapter;  
  import android.widget.ImageView;  
  import android.widget.TextView;  
  import android.widget.Toast;  
    
  import com.edu.ouc.model.TaskShowModel;  
  import com.edu.ouc.tasktracking.R;  
    
  import java.util.ArrayList;  
  import java.util.List;  
    
  /\*\*  
   \* Created by JHC on 2017/11/23.  
   \* 这里主要是为ListView写一个填充数据的适配器,通过这个适配器来确定ListView上每个Item显示的数据.  
   \*/  
    
  public class TaskBaseAdapter extends BaseAdapter {  
   private List<TaskShowModel> datas = new ArrayList<TaskShowModel>();//新闻列表集合  
    
   private Context context;  
   public TaskBaseAdapter(Context context, List<TaskShowModel> datas) {  
   this.datas = datas;  
   this.context = context;  
   }  
   @Override  
   public int getCount() {  
   return datas.size(); //返回列表的长度  
   }  
    
   @Override  
   public TaskShowModel getItem(int position) {  
   return datas.get(position); //通过列表的位置 获得集合中的对象  
   }  
    
   @Override  
   public long getItemId(int position) { // 获得集合的Item的位  
   return position;  
   }  
    
   @Override  
   public View getView(final int position, View convertView, ViewGroup parent) {  
   ViewHolder vh;  
   if(convertView==null){  
   vh = new ViewHolder();  
   convertView = LayoutInflater.from(context).inflate(R.layout.listview\_fragment\_task,null);  
   vh.tvTitle = (TextView) convertView.findViewById(R.id.tv\_title);  
   vh.tvDate = (TextView) convertView.findViewById(R.id.tv\_info);  
   convertView.setTag(vh);  
   }else {  
   vh = (ViewHolder) convertView.getTag();  
   }  
   convertView.setOnClickListener(new View.OnClickListener() {  
   @Override  
   public void onClick(View view) {  
   Toast.makeText(context, "点击了"+datas.get(position).getTasktitle(),Toast.LENGTH\_SHORT).show();  
   /\* Intent intent = new Intent(context, Main2Activity.class);  
   context.startActivity(intent);\*/  
   }  
   });  
   /\*vh.img.setOnClickListener(new View.OnClickListener() {  
   @Override  
   public void onClick(View view) {  
   Toast.makeText(context,"点击图片",Toast.LENGTH\_SHORT).show();  
   }  
   });\*/  
   TaskShowModel taskShowModel = datas.get(position);  
   vh.tvTitle.setText(taskShowModel.getTasktitle());  
   vh.tvDate.setText(taskShowModel.getStartDate());  
   return convertView;  
   }  
   protected class ViewHolder {  
   private TextView tvTitle;  
   private TextView tvDate;  
   }  
  }
* framment

package com.edu.ouc.fragment;  
  
  
import android.app.Activity;  
import android.content.Context;  
import android.os.Bundle;  
import android.support.annotation.Nullable;  
import android.support.v4.app.Fragment;  
import android.view.LayoutInflater;  
import android.view.View;  
import android.view.ViewGroup;  
import android.widget.ListView;  
  
import com.edu.ouc.adapter.TaskBaseAdapter;  
import com.edu.ouc.model.TaskShowModel;  
import com.edu.ouc.tasktracking.R;  
  
import java.util.ArrayList;  
import java.util.List;  
  
/\*\*  
 \* A simple {@link Fragment} subclass.  
 \*/  
public class TaskFragment extends Fragment {  
 private Activity activity;  
 private ListView listView;  
 private List<TaskShowModel> datas; //新闻集合对象  
 private TaskBaseAdapter adapter; //自定义适配器对象  
 public TaskFragment() {  
 // Required empty public constructor  
 }  
 @Override  
 public View onCreateView(LayoutInflater inflater, ViewGroup container,  
 Bundle savedInstanceState) {  
 // Inflate the layout for this fragment  
 return inflater.inflate(R.layout.fragment\_task, container, false);  
 }  
  
 @Override  
 public void onActivityCreated(@Nullable Bundle savedInstanceState) {  
 super.onActivityCreated(savedInstanceState);  
 listView=(ListView)getView().findViewById(R.id.lv\_task\_ok);  
 datas=new ArrayList<TaskShowModel>();  
 getDatas();  
 /\*\*  
 \* 实例化Adapter对象(注意:必须要写在在getDatas() 方法后面,不然datas中没有数据)  
 \*/  
 adapter = new TaskBaseAdapter(getActivity(), datas);  
 //将适配器变量的内容加载到List里(也就是把那一堆新闻都放了进去)  
 listView.setAdapter(adapter);  
 }  
 /\*\*  
 \* 通过接口获取新闻列表的方法  
 \* @param url  
 \*/  
 public void getDatas(){  
 for (int i=0;i<20;i++){  
 TaskShowModel taskShowModel=new TaskShowModel();  
 taskShowModel.setTasktitle("文章标题"+i);  
 taskShowModel.setStartDate("时间："+i);  
 datas.add(taskShowModel);  
 }  
 }  
}

开机自启动

* 新建一个接收类，并实现BroadcastRecevicer，重写Onreceive方法

Intent intent =new Intent(context,MyReceive.class);

intent.setFlags(Intent.FLAG\_ACTIVITY\_NEW\_TASK);

context.startActivity(intent);

* 在Manifest.xml注册此类，并添加

<intent-filter> <!—标明应用程序的入口程序，此标签在那个Activity下，应用程序就会先启动哪个Activity-->  
 <action android:name="android.intent.action.BOOT\_COMPLETED" />  
 <category android:name="android.intent.category.HOME" />  
</intent-filter>

视频连接：http://study.163.com/course/courseLearn.htm?courseId=270014#/learn/video?lessonId=377177&courseId=270014

Handler控制PorgressBar

* 启动Handler

handler.post(runnable);

* 创建Handler

Handler handler=new Handler(){

//接收消息

Public void handlerMessage(Message msg){

Pb.setProgess(msg.arg1);//其中pb为ProgresBar的对象

handler.post(runnable);

if(msg.arg1==100){

handler.removeCallbacks(runnable);

}

}

}

Runnable runnable=new Runnable(){

Int i=0;

Public void run(){

I+=10;

//Message是Android系统所提供的的消息对象

Message message=handler.obtainMessage();

//message.arg1是系统提供的，并且会节省系统资源消耗

message.arg1=i;

Try{

Thread.sleep(1000); //由于线程执行太快，故而可使线程睡眠1秒

}catach(InterruptedException e){

e.printStackTrace();

}

//发送消息

handler.sendMessage(message);

}

}

两次返回键退出程序

// 定义一个变量，来标识是否退出

private static boolean isExit = false;

Handler mHandler = new Handler() {

@Override

public void handleMessage(Message msg) {

super.handleMessage(msg);

isExit = false;

}

};

@Override

public boolean onKeyDown(int keyCode, KeyEvent event) {

if (keyCode == KeyEvent.KEYCODE\_BACK) {

exit();

return false;

}

return super.onKeyDown(keyCode, event);

}

private void exit() {

if (!isExit) {

isExit = true;

Toast.makeText(getApplicationContext(), "再按一次退出程序",

Toast.LENGTH\_SHORT).show();

// 利用handler延迟发送更改状态信息

mHandler.sendEmptyMessageDelayed(0, 2000);

} else {

finish();

System.exit(0);

}

}

Thread和Handler

<https://www.cnblogs.com/net168/p/4075126.html>

Android数据传递】Intent传递List和Object和List<Object>(附源码)

**一、传递List**

**传递List<String>的方法**

ArrayList<String> info = new ArrayList<String>();

info.add(name);

info.add(website);

info.add(weibo);

Intent intent = new Intent(MainActivity.this, ResultActivity.class);

intent.putStringArrayListExtra("infoList", info);

startActivity(intent);

**接收List<String>的方法**

ArrayList<String> infoList = new ArrayList<String>();

infoList = getIntent().getStringArrayListExtra("infoList");

**传递List<Integer>的方法**

intent.putIntegerArrayListExtra(key, list);

  接收**List<Integer>的方法**

list =(ArrayList<Integer>) getIntent().getIntegerArrayListExtra(key);

**二、传递Object**

有两种方式来传递Object:Serializable和Parcelable

**2.1 使用Serializable方式**

前提：Object需要实现Serializable接口

用Serializable方式传递Object的语法：bundle.putSerializable(key,object);

用Serializable方式接收Object的语法：object=(Object) getIntent().getSerializableExtra(key);

实现Serializable接口就是把对象序列化，然后再传输，和Java的常用编程没什么明显区别，而且Object不需要明显改变,推荐用这种方式。

**Object实现Serializable**

package com.wirelessqa.testintent;

import java.io.Serializable;

/\*\*

\* OBJECT实现SERIALIZABLE

\* @author bixiaopeng 2013-2-18 上午11:32:19

\*/

public class SerInfo implements Serializable {

private String name;

private String website;

private String weibo;

public SerInfo(){}

public SerInfo(String name, String website, String weibo){

this.name = name;

this.website = website;

this.weibo = weibo;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public String getWebsite() {

return website;

}

public void setWebsite(String website) {

this.website = website;

}

public String getWeibo() {

return weibo;

}

public void setWeibo(String weibo) {

this.weibo = weibo;

}

}

**用Serializable方式传递Object**

SerInfo serInfo = new SerInfo(name, website, weibo);

Intent intent = new Intent();

Bundle bundle = new Bundle();

bundle.putSerializable("serinfo", serInfo);

intent.setClass(MainActivity.this, ResultActivity.class);

intent.putExtras(bundle);

startActivity(intent);

**用Serializable方式接收Object**

//获得Serializable方式传过来的值

SerInfo serInfo = (SerInfo) getIntent().getSerializableExtra("serinfo");

参考连接：<http://blog.csdn.net/wirelessqa/article/details/8589200>

可见（visible)  
XML文件：android:visibility="visible"  
Java代码：view.setVisibility(View.VISIBLE);

不可见（invisible）  
XML文件：android:visibility="invisible"  
Java代码：view.setVisibility(View.INVISIBLE);

隐藏（GONE）  
XML文件：android:visibility="gone"  
Java代码：view.setVisibility(View.GONE);

注：当控件visibility属性为INVISIBLE时，textview依然会占据那部分位置；而控件属性为GONE时，界面则不保留textview控件所占有的空间

弹框-确定，取消

*new AlertDialog.Builder(this).setTitle("提示")//设置对话框标题  
 .setMessage("请确认是否接收任务！")//设置显示的内容  
 .setPositiveButton("确定",new DialogInterface.OnClickListener() {//添加确定按钮  
 @Override  
 public void onClick(DialogInterface dialog, int which) {//确定按钮的响应事件  
 }  
 }).setNegativeButton("返回",new DialogInterface.OnClickListener() {//添加返回按钮  
 @Override  
 public void onClick(DialogInterface dialog, int which) {//响应事件  
 //* ***TODO Auto-generated method stub*** *}  
 }).show();//在按键响应事件中显示此对话框*

弹框—复选框

AlertDialog.Builder builder = **new** AlertDialog.Builder(TaskInfoActivity.**this**);  
builder.setTitle(**"爱好"**);  
**final** String[] hobbies = {**"篮球"**, **"足球"**, **"网球"**, **"斯诺克"**};  
*// 设置一个单项选择下拉框  
/\*\*  
 \* 第一个参数指定我们要显示的一组下拉多选框的数据集合  
 \* 第二个参数代表哪几个选项被选择，如果是null，则表示一个都不选择，如果希望指定哪一个多选选项框被选择，  
 \* 需要传递一个boolean[]数组进去，其长度要和第一个参数的长度相同，例如 {true, false, false, true};  
 \* 第三个参数给每一个多选项绑定一个监听器  
 \*/*builder.setMultiChoiceItems(hobbies, **null**, **new** DialogInterface.OnMultiChoiceClickListener()  
{  
 StringBuffer **sb** = **new** StringBuffer(100);  
 @Override  
 **public void** onClick(DialogInterface dialog, **int** which, **boolean** isChecked)  
 {  
 **if**(isChecked)  
 {  
 **sb**.append(hobbies[which] + **", "**);  
 }  
 Toast.*makeText*(TaskInfoActivity.**this**, **"爱好为："** + **sb**.toString(), Toast.***LENGTH\_SHORT***).show();  
 }  
});  
builder.setPositiveButton(**"确定"**, **new** DialogInterface.OnClickListener()  
{  
 @Override  
 **public void** onClick(DialogInterface dialog, **int** which)  
 {  
 }  
});  
builder.setNegativeButton(**"取消"**, **new** DialogInterface.OnClickListener()  
{  
 @Override  
 **public void** onClick(DialogInterface dialog, **int** which)  
 {  
 }  
});  
builder.show();

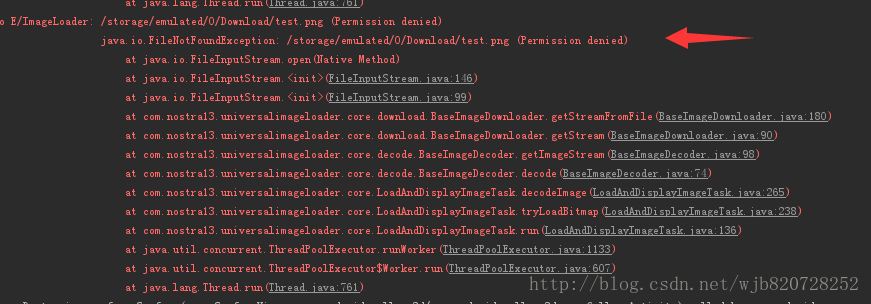
### **解决设置Android 23.0以上版本对SD卡的读写权限无效的问题**

对Android的SD卡进行读取权限设置时：

<uses-permission android:name="android.permission.WRITE\_EXTERNAL\_STORAGE" />

<uses-permission android:name="android.permission.READ\_EXTERNAL\_STORAGE"/>

* 1
* 2

结果报错：   


依然提示我没有权限，于是百度说是版本问题，23.0版本（笔者的版本是25.0）以上的不仅仅要设置上面的权限，还要在对SD卡有读写操作的地方授权，下面是公共类：

public class PermisionUtils {

// Storage Permissions

private static final int REQUEST\_EXTERNAL\_STORAGE = 1;

private static String[] PERMISSIONS\_STORAGE = {

Manifest.permission.READ\_EXTERNAL\_STORAGE,

Manifest.permission.WRITE\_EXTERNAL\_STORAGE};

/\*\*

\* Checks if the app has permission to write to device storage

\* If the app does not has permission then the user will be prompted to

\* grant permissions

\*

\* @param activity

\*/

public static void verifyStoragePermissions(Activity activity) {

// Check if we have write permission

int permission = ActivityCompat.checkSelfPermission(activity,

Manifest.permission.WRITE\_EXTERNAL\_STORAGE);

if (permission != PackageManager.PERMISSION\_GRANTED) {

// We don't have permission so prompt the user

ActivityCompat.requestPermissions(activity, PERMISSIONS\_STORAGE,

REQUEST\_EXTERNAL\_STORAGE);

}

}

}

然后直接在需要授权的地方调用：

//检测读写权限

PermisionUtils.verifyStoragePermissions(this);

* 1
* 2

程序运行的时候，会询问是否授权

Android N7以上版本打开摄像头

Android N 文件系统权限的变化   
为了提高私有文件的安全性，在targetSdk版本为N或者以后版本的app中，其私有目录将会限制访问。这可以防止私有文件元数据的泄露，比如文件大小或者是文件是否存在。但这给开发者带来了很多不利的影响：

文件的owner不能放宽文件权限，如果你使用MODE\_WORLD\_READABLE或者 MODE\_WORLD\_WRITEABLE操作文件，将会触发SecurityException。

**当你跨package域传递file://的URI时**，接收者得到的将是一个无权访问的路径，因此，这将会触发FileUriExposedException。对于这类操作，官方推荐的方式是使用FileProvider，当然你也可以使用ContentProvider。这里只看文字理解起来可能有点小困难，所以我将以调用系统拍照为例说明下：在targetSdk为Android N之前的系统版本时，你可以使用如下方法调用系统相机拍照并存入指定路径中。

Intent intent = new Intent(MediaStore.ACTION\_IMAGE\_CAPTURE);

Uri uri = Uri.fromFile(sdcardTempFile);

intent.putExtra(MediaStore.EXTRA\_OUTPUT, uri);

* 1
* 2
* 3
* 4

当你将targetSdk设置为Android N时，很不幸，在执行到这段代码时app就crash了，crash便是FileUriExposedException。

**方法一：**   
把代码修改下，使用ContentProvider方式传递uri，这样在Android N上便可以正常运行了。

Intent intent = new Intent(MediaStore.ACTION\_IMAGE\_CAPTURE);

ContentValues contentValues = new ContentValues(1);

contentValues.put(MediaStore.Images.Media.DATA, sdcardTempFile.getAbsolutePath());

Uri uri = context.getContentResolver().insert(MediaStore.Images.Media.EXTERNAL\_CONTENT\_URI,contentValues);

* 1
* 2
* 3
* 4

**方法二：**

通过FileProvider解决，实例下载：<https://github.com/honjane/fileProviderDemo>

<http://blog.csdn.net/honjane/article/details/52057132>

android okhttputils

服务器端

@ResponseBody

@RequestMapping(value = "/upload", method = RequestMethod.***POST***)

**public** **void** upload(HttpServletRequest request, HttpServletResponse response) **throws** Exception {

response.setHeader("Access-Control-Allow-Origin", "\*"); // 解决跨域请求

*LOG*.info("-----进入后台----upload接口---开始---");

//获取工厂类实例

DiskFileItemFactory factory = **new** DiskFileItemFactory();

//创建解析器类实例

ServletFileUpload fileUpload = **new** ServletFileUpload(factory);

//fileUpload通过该对象来限制文件的大小

//设置当文件的大小为50KB

//fileUpload.setFileSizeMax(1024\*50);

//设置多个文件的总大小为300mb

fileUpload.setSizeMax(1024\*1024\*300);

**try** {

//解析request

List<FileItem> fileItems = fileUpload.parseRequest(request);

//遍历fileItems，读取表单的信息

**for** (FileItem fileItem : fileItems) {

//判断当前表单项是否是一个普通表单项

**if**(fileItem.isFormField()){

//获取属性名

String fieldName = fileItem.getFieldName();

//获取属性值

String value = fileItem.getString("utf-8");

System.***out***.println(fieldName+" = "+value);

}**else**{

//如果是文件表单项

//获取文件的大小

**long** size = fileItem.getSize();

//判断size是否为0

**if**(size==0){

**continue**;

}

//获取文件的类型

String contentType = fileItem.getContentType();

//获取文件的名字

String name = fileItem.getName();

//判断name中是否包含有路径信息

**if**(name.contains("\\")){

//如果包含则截取字符串

name = name.substring(name.lastIndexOf("\\")+1);

}

//生成一个UUID，作为文件名的前缀

String prefix = UUID.*randomUUID*().toString().replace("-", "");

name = prefix+"\_"+name;

//获取表单项的属性名

String fieldName = fileItem.getFieldName();

System.***out***.println("文件的大小: "+size);

System.***out***.println("文件的类型: "+contentType);

System.***out***.println("文件的名字: "+name);

System.***out***.println("表单项name属性名: "+fieldName);

//获取ServletContext对象

ServletContext context = request.getServletContext();

//获取项目的真实路径

String path = context.getRealPath("/upload");

//判断路径是否存在

File file = **new** File(path);

**if**(!file.exists()){

//如果不存在该路径，则创建一个路径

file.mkdirs();

}

//将文件写入到磁盘中

fileItem.write(**new** File(path+"/"+name));

}

}

PrintWriter out=response.getWriter();

out.println("upload success!!!");//输出的内容要放在body中

//设置3秒钟跳转

//response.setHeader("refresh", "3;../test/index.jsp");

}**catch**(FileSizeLimitExceededException e){

//一但捕获到该异常，则说明单个文件大小超过限制。

//设置一个错误消息

request.setAttribute("msg", "单个文件大小请不要超过50KB");

//转发到index.jsp

request.getRequestDispatcher("test/testUploadFile.jsp").forward(request, response);

}**catch**(SizeLimitExceededException e){

//一但捕获到该异常，则说明单个文件大小超过限制。

//设置一个错误消息

request.setAttribute("msg", "所有文件大小请不要超过300mb");

//转发到index.jsp

request.getRequestDispatcher("test/testUploadFile.jsp").forward(request, response);

}

**catch** (FileUploadException e) {

e.printStackTrace();

} **catch** (Exception e) {

e.printStackTrace();

}

*LOG*.info("--后台----upload成功------");

}

客户端

**public** String upoadFile(File file, String requestURL){  
 String result=**""**;  
 Map<String, String> params = **new** HashMap<String, String>();  
 params.put(**"username"**, **"宋新良"**);*//上传的用户名,没有不用传参数* params.put(**"password"**, **"123456"**);*//上传的密码,没有不用传参数* String url = requestURL ;  
 OkHttpUtils.*post*()*//* .addFile(**"mFile"**, **"server\_afu.png"**, file)*//可以上传多个文件  
 /\* .addFile("mFile", "server\_test.txt", file2)//\*/* .url(url)  
 .params(params)*//* .build()*//* .execute(**new** UpLoadSummaryActivity.MyStringCallback());  
 **return** result;  
}

UpLoadFileToServer upLoadFileToServer=**new** UpLoadFileToServer();  
upLoadFileToServer.upoadFile(**new** File(**imagePaths**.get(0)),**"http://10.0.2.2:8080/TaskTrackingService/upload.do?"**);

*//okhttp-utils的回调类***public static class** MyStringCallback **extends** StringCallback {  
 @Override  
 **public void** onBefore(Request request, **int** id) {  
 *//setTitle("loading...");* }  
 @Override  
 **public void** onAfter(**int** id) {  
 *// setTitle("Sample-okHttp");* }  
  
 @Override  
 **public void** onError(Call call, Exception e, **int** id) {  
 e.printStackTrace();  
 *//tv\_result.setText("onError:" + e.getMessage());* }  
  
 @Override  
 **public void** onResponse(String response, **int** id) {  
 *//Log.e(TAG, "onResponse：complete");  
 //tv\_result.setText("onResponse:" + response);* **switch** (id) {  
 **case** 100:*//http请求的响应码  
 //Toast.makeText(OKHttpActivity.this, "http", Toast.LENGTH\_SHORT).show();* **break**;  
 **case** 101:*//http请求的响应码  
 // Toast.makeText(OKHttpActivity.this, "https", Toast.LENGTH\_SHORT).show();* **break**;  
 }  
 }  
  
 @Override  
 **public void** inProgress(**float** progress, **long** total, **int** id) {  
 *//Log.e(TAG, "inProgress:" + progress);  
 /\*mProgressBar.setProgress((int) (100 \* progress));\*/* }  
}

**dialog**.setCancelable(**false**); *//设置这个对话框不能被用户按[返回键]而取消掉*