1. 控件
2. Menu控件

当用户点击菜单按钮的时候调用

复写onCreateOptionsMenu方法

例子：

Public Boolean onCreateOptionsMenu(Menu menu)

{

menu.add(0,1,1,R.string.exit);//（组、编号【ItemId】、排序、名称）

menu.add(0,2,2,R.string.help);;

}

当用户点击某一个菜单项时调用

复写onOptonsItemSelected方法

例子：

Public void onOptionItemSelected(MenuItem item)

{

If(item.getItemId == 1)

{

Finish();//结束应用程序

}

}

1. RadioGroup和RadioButton
   1. 布局

<RadioGroup

android:id=”@+id/genderGroup”

android:layout\_width=”wrap\_content”

android:layout\_height=”wrap\_content”

android:orientation=”vertical”>

<RadioButton></RadioButton>

<RadioButton></RadioButton>

</RadioGroup>

* 1. 代码

Private RadioGroup genderGroup;

Private RadioButton femaleButton;

Private RadioButton maleButton;

genderGroup.setOnCheckedChangeListener(new RadioGroup.OnCheckedChangeListener(){

@Override

Public void onCheckedChanged(RadioGroup group, int checkId)

{

If(femaleButton.getId == checkId)

{

System.out.Println(“female”);

}else if(maleButton.getId == checkId){}

}

});

1. CheckBox
   1. 布局

<CheckBox

Android:id=”@+id/swim” />

<CheckBox

Android:id=”@+id/read” />

<CheckBox

Android:id=”@+id/run” />

* 1. 代码

CheckBox继承了CompoundButton

swimBox.setOnCheckedChangeListener(new CompoundButton.OnCheckedChangedListener(){

@Override

Public void onCheckedChanged(CompoundButton buttonView,Boolean isChecked){

If(isChecked){};

}

});

1. Toast

提示信息

Toast.makeText(RadioTest.this,”female”,Toast.LENGTH\_SHORT).show();//（当前类，提示信息，长度）

1. ProgressBar
   1. 进度条
   2. <ProgressBar

android:id=”@+id/firstBar”

style=”?android:attr/progressBarStyleHorizontal” //水平方向的风格

android:visibility=”gone” /> //是否可视,gone表示不可视

<ProgressBar

Android:id=”@+id/secondBar”

Style=”?android:attr/progressBarStyle” //默认的风格，圆圈央视

Android:visibility=”gone”

Android:max=”200”> //设置进度条的最大值

* 1. firstBar.setVisibility(View.VISIBLE);//设置进度条处于可见状态

firstBar.setPaogress(i);//设置进度条的当前值

firstBar.setSeconddaryProgress(i+10);//设置第二进度条的当前值

firstBar.setMax(150);//设置进度条最大值

1. ListView
   1. 列表
   2. 继承ListAvtivity

<ListView

android:id=”@id/android:list”

android:drawSelectorOnTop=”false”

android:scrollbars=”vertical” />//滚动条方向

ArrayList<HashMap<String,String>> list = new ArrayList<HashMap<String,String>>();

SimpleAdapter listAdapter = new SimpleAdapter(this,list,R.layout.user,new String(“user\_name”,”user\_ip”),new int[]{R.id.user\_name,R.id.user\_ip});//适配器

setListAdapter(listAdapter);

@Override

Protected void onListItemClick(ListView l, View v, int position, long id)

{

Super.onListItemClick(l,v,postion,id);

}

1. 创建Android Project
   1. 向上兼容
   2. 选择1.6的SDK
   3. Application Name ——应用程序的名称
   4. Create Activity —— 显示的名称
   5. Min SDK Version ——兼容的最低版本
2. Project 目录介绍

gen 目录，插件自动生成，R.java不能编辑

assets 可以防止任意的人间

res 其中的人间会在R.java中生成一个Id

layout main.xml——布局文件

values string.xml——键值对，会在R.java中生成Id

1. AndroidManifest 整个应用程序的配置
2. Activity
   1. Activity的作用
      1. 是一个用户接口，和用户进行界面的交互，空间的容器；
   2. Activity的步骤
      1. 一个Activity就是一个类，并且要继承Activity
      2. 需要复写onCreate()方法
      3. 每个Activity都要在Manifest.xml文件中配置，配置了<intent-activity>就是第一启动画面
      4. 为Activity添加控件
3. Activity和Intent的关系

Activity01 ——Intent intent——Activity02

Intent对象包括：

* 1. Component name ——启动哪一个组件
  2. Action——下一个Activity要做的动作
  3. Data——数据
  4. Category
  5. Extras——额外的一些信息、键值对
  6. Flags

生成一个Intent对象

Intent intent = new Intent();

Intent.setClass(Activity02.this,OtherActivity.class);

Activity02.this.startActivity(intent);

1. Activity的生命周期
2. 生命周期函数

启动一个新的Activity的时候会调用下面三个生命周期函数：

onCreat()——》当Activity第一个被创建的时候就会调用这个方法

onStart()——》当Activity能被户用看到的时候会调用这个方法

onResume()——》当Activity能获得用户焦点的时候调用这个方法

启动第二个Activity的时候：

第一个Activity的onPause()——》当应用程序启动了另外一个Activity的时候调用

onPause()当应该写保存当前数据的代码；

第二个Activity的onCreate() onStart() onResume()方法

第一个Activity的onStop()方法——》当这个Activity处于不可见的状态的时候，调用这个方法（需要全部遮挡），处于停止状态

点击Back之后

第二个Activity的onPause()

第一个Activity的onRestart()

第一个Activity的onStart()

第一个Activity的onResume()

第二个Activity的onStop()

第二个Activity的onDestory()//第二个被销毁，有两种情况会调用，（1）调用了 finish()方法；（2）系统资源不够用

1. Task

Task就是一个栈（先进后出），里面存放了很多Activity，压栈，弹栈

当调用了finish方法之后就会从栈里面消失

3. 窗口风格的Activity的实现

其他没有区别，区别在AndroidManifest.xml文件中

<activity android:name=”.SecondActivity”

android:label=”SecondActivity”

android:theme=”@android:style/Theme.Dialog”/>

指定android:theme=”@android:style/Theme.Dialog”

1. Activity 布局
2. LinearLayout——线性布局

android:orientation=”vertical”//线性布局的方向，vertical为垂直，horizontal为水平

android:layout\_width=”fill\_parent”//宽度填充父控件

android:layout\_height=”wrap\_parent”//高度显示内容即可

android:id //为控件指定相应的Id

android:text //指定控件当中显示的文字，尽量使用string.xml

android:gravity //指定控件里面的内容的基本位置，比如说居中、居右等位置

android:textSize //指定控件当中字体的大小

android:background //指定控件所使用的背景颜色，RGB命名法

android:width //指定控件的宽度

android:height //指定控件的哦高度

android:padding\* //指定控件内容的内边距，也就是说控件当中的内容

android:sigleLine //如果设置为真的话，则将控件的内容在同一行进行显示

android:weight //控件的占的比例

1. TableLayout——表格布局

<TableRow>行

android:stretchColumns=”1” //使用第一列作为拉升的一列以便填充父控件（从0位计数）

3．RelativeLayout——相对布局

（1）两个控件之间的上下左右对齐

Android:layout\_above——将控件的底部至于给定Id的控件之上

android:layout\_above=”@id/xxx”

android:layout\_below——将控件的顶部至于给定Id的控件之下

android:layout\_below=”@id/xxx”

android:layout\_toLeftOf——将控件的右边缘与给定Id的控件的左边缘对齐（即在左边）

android:layout\_toLeftOf=”@id/xxx”

android:layout\_toRightOf——

（2）对齐

android:layout\_alignBaseline

android:layout\_alignBottom

android:layout\_alignRight

android:layout\_alignLeft

android:layout\_alignTop

（2）和父控件进行对齐

Android:alignParentBottom ——如果为True，则将该控件的底部和父控件的底部对齐

android:layout\_alignParentLeft——如果该值为true，则该控件的左边与父控件的左边对齐

android:layout\_alignParentRight

android:layout\_alignParentTop

android:layout\_centerHorizontal ——如果为True，该控件将被至于水平方向的中央

android:layout\_centerInParent ——如果为true,该控件将被至于父控件水平方向和垂直方向

android:layout\_centerVertical——如果为true，该控件将被置于垂直方向的中央

1. Handler
   1. handler.postDelayer(Thread.class,3000);//隔3秒把线程加入到消息队列中
   2. 一个handler就是一个消息队列，消息队列都是先进先出的顺序
   3. handler.post(Thread.class);//马上加入到消息队列中
   4. handler.removeCallbacks(Thread.class);//删除线程
   5. 步骤：创建一个Handler对象；调用Handler的Post方法，将要执行的线程对象添加到队列当中；将要执行的操作写在线程对象的run方法中；在run方法内部，执行postDelayer或者是post方法。
   6. 提供了异步线程处理方案

Handler handler = new Handler();

Runnable updateThreead = new Runnable(){

@Override

Public void run(){

System.out.println(“Update Thread”);

handler.postDelayer(updateThread,3000);

}

}