

Android 开发

1、Android Studio 安装

下载地址: <https://developer.android.google.cn/studio>

我安装的是 Android Studio 3.4 版本。

确保安装 Android studio 前, 安装好 java JDK, 并配置好环境。
运行下载好的 android studio 的 exe 文件。

1. 点击 next
2. 勾选 Android Virtual Device(下载官方模拟器), next
3. 选择安装目录, 最好不要用中文路径, next
4. 点击 install 进行安装
5. 安装完成后, 点击 finish 运行 Androidstudio
6. 弹出 complete installation 窗口, 选择第三项, 点击 ok
7. 弹出选择 sdk 窗口, 选择 cancel 暂不安装
8. Welcome 窗口, 点击 next
9. Install type 窗口, 选择第一项, 点击 next
10. 选择界面颜色, 点击 next
11. 下载完成后, 点击 finish 运行 Androidstudio
12. 新建一个新的 Androidstudio 工程, 点击第一项
13. 项目名称、位置可以改变, 点击 next
14. 进入 androidstudio 等待项目 build
15. 完成

2、项目结构

Android 项目结构：

manifests 目录：Android 全局描述文件

<application>节点：整个 app 的属性，如 Icon:app 图标

Java 目录：java 源码文件

Res 目录：资源文件

Drawable: 保存位图目录

Layout: 存储 Android 程序的布局文件

Mipmap: 保存启动图标文件，保存路径不同图片分辨率不同。



Value:color.xml: 保存颜色资源；dimens.xml: 保存尺寸资源；string.xml: 字符串资源；styles: 样式资源

(R 文件生成失败，重新清理缓存，bulid->clean project)

3、布局管理器

①线性布局 (LinearLayout)

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    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"
    android:orientation="vertical"
    android:gravity="bottom|center_horizontal"
    tools:context=".MainActivity">
    <Button.../>
    <Button.../>
    <Button.../>
</LinearLayout>
```

②表格布局 (TableLayout)

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity"
    android:orientation="vertical">
    <TableLayout
        android:id="@+id/table1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:shrinkColumns="1"
        android:stretchColumns="2">
    </TableLayout>
    <!--指定第二列按钮被隐藏-->
    <TableLayout
        android:id="@+id/table2"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:collapseColumns="3">
        <Button .../>
        <TableRow>
            <Button .../>
```

```

        <Button.../>
        <Button.../>
    </TableRow>
</TableLayout>
<TableLayout
    android:id="@+id/table3"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:stretchColumns="1,2">
    <!--stretchColumns 设置 1,2 列的按钮拉伸-->
    <Button.../>
    <TableRow>
        <Button .../>
        <Button.../>
        <Button.../>
    </TableRow>
    <TableRow>
        <Button.../>
        <Button.../>
    </TableRow>
</TableLayout>
</LinearLayout>

```

③ 帧布局 (FrameLayout)

```

<?xml version="1.0" encoding="utf-8"?>
<FrameLayout
    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=".MainActivity">
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:width="160dp"
        android:height="160dp" />
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello World!"
        android:layout_gravity="center"
        android:width="140dp"
        android:height="140dp"

```

```

        android:background="#f00"/>
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_gravity="center"
    android:width="120dp"
    android:height="120dp"
    android:background="#000"/>
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_gravity="center"
    android:width="100dp"
    android:height="100dp"
    android:background="#ff0"/>
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_gravity="center"
    android:width="80dp"
    android:height="80dp"
    android:background="#f0f"/>

<Button
    android:layout_width="66dp"
    android:layout_height="wrap_content"
    android:layout_gravity="center"
    android:width="60dp"
    android:height="60dp"
    android:background="#0f0" />
</FrameLayout>

```

④ 网格布局 (GridLayout)

```

<?xml version="1.0" encoding="utf-8"?>
<GridLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:rowCount="11"
    android:columnCount="5"
    android:paddingLeft="40dp"
    tools:context=".MainActivity">

```

```

<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="用户注册"
    android:layout_row="0"
    android:layout_column="2"
    android:layout_gravity="center"
    android:gravity="center"/>
<TextView
    android:id="@+id/t1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_row="1"
    android:layout_column="0"
    android:text="@string/t1" />
<EditText
    android:layout_width="235dp"
    android:layout_height="wrap_content"
    android:layout_row="1"
    android:layout_column="1"
    android:layout_columnSpan="4"
    android:hint="@string/e1" />
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_row="3"
    android:layout_column="0"
    android:text="年龄： " />
<EditText
    android:layout_width="235dp"
    android:layout_height="wrap_content"
    android:layout_row="3"
    android:layout_column="1"
    android:layout_columnSpan="4"
    android:hint="请输入年龄" />
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_row="4"
    android:layout_column="0"
    android:text="性别： " />
<RadioButton
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"

```

```

        android:layout_row="4"
        android:layout_column="1"
        android:hint="男" />
<RadioButton
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_row="4"
    android:layout_column="2"
    android:hint="女"/>
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_row="5"
    android:layout_column="0"
    android:text="兴趣爱好: " />
<CheckBox
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_row="5"
    android:layout_column="1"
    android:hint="下棋" />
<CheckBox
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_row="5"
    android:layout_column="2"
    android:hint="看书" />
<CheckBox
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_row="5"
    android:layout_column="3"
    android:hint="唱歌" />
<CheckBox
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_row="5"
    android:layout_column="4"
    android:hint="跳舞" />
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_row="6"
    android:layout_column="0"

```

```

        android:text="联系方式: " />
<EditText
    android:layout_width="235dp"
    android:layout_height="wrap_content"
    android:layout_row="6"
    android:layout_column="1"
    android:layout_columnSpan="4"
    android:hint="请输入联系方式" />
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_row="7"
    android:layout_column="0"
    android:text="个人简介: " />
<EditText
    android:layout_width="235dp"
    android:layout_height="wrap_content"
    android:layout_row="7"
    android:layout_column="1"
    android:layout_columnSpan="2"
    android:background="#fff"
    android:lines="6"
    android:layout_columnSpan="4"
    android:hint="请输入个人简介" />
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_row="8"
    android:layout_column="2"
    android:text="注册" />
</GridLayout>

```

⑤相对布局 (RelativeLayout)

⑥绝对布局 (AbsoluteLayout)

4、UI 组件

①TextView 及其子类

TextView 子类有 EditText、Button、CheckedTextView、Chronometer、DigitalClock。

②ImageView 及其子类

ImageView 子类有 ImageButton、QuickContactBadge

③AdapterView 及其子类

AdapterView 子类有 AbsListView、AbsSpinner、AdaperViewAnimator

④Adapter 接口及其实现类

Adapter 子类有 ListAdapter、BaseAdapter

⑤时间选择器 (TimePicker)

⑥日期选择器 (DatePicker)

⑦计时器 (Chronometer)

常用方法: `setBase()`: 设置计时器的起始时间; `setFormat()`: 设置显示时间的格式; `start()`: 开始计时; `stop()`: 停止计时; `setOnChronometerTickListener()`: 监听计时器改变的监听器。

5、高级 UI 组件

①ProgressBar(进度条)

分为条形进度条和选择进度条；在约束布局里

style=" ?android:attr/progressBarStyleHorizontal" 。其他布局里

style="@android:style/Widget.ProgressBar.Horizontal"。条形进度条需要通过 线程监控耗时任务，实时给出任务进度，以便页面更新进度条的进度。

旋转进度条：style="?android:attr/progressBarStyle" 。

②SeekBar(拖动条)

属于进度条的子类。通过拖动拖动条调节图片的大小，明亮，透明度等。总之调节一些属性，图片、音量、视频进度等等。

③RatingBar(星级评分条)

android:rating="5": 设置默认星级为五

④ImageView(图像视图)

android:src=" @drawable/xxx" : 图像资源一般放置在 drawable 资源中，不同于背景放置在 mipmap 资源中。

⑤ImageSwitcher(图像切换器)

可以设置图像切换时的动画，如淡入淡出。需要创建视图工厂，为图像切换器指定图像，并通过监听方法监听单击图片的事件，从而进行切换到下一张图片。

⑥网格视图 (GridView)

⑦下拉表框 (Spinner)

下拉表框能够通过定位资源或者定义适配器来填充下拉框的数据。定位资源：使用 <Spinner> 中的 entries 属性，在 layout 下 value 目录中创建数组资源文件 arrays.xml，通过 string-array 属性设置填充的数据 (item)。适配器方式：定义数组放入下拉框数据，定义适配器放入数据并设置下拉框样式，在使用下拉框对象添加适配器。

⑧ (ListView) 列表视图

列表视图与下列表框一样，同样能够通过定位资源或定义适配器来使用。在项目开发中一班使用适配器方法，除非想要写死用 xml。

⑨ScrollView(滚动视图)

在 <scrollview> 中直接添加控件，如果内容超过当前屏幕，就会出现垂直滚动条。水平滚动条为：<HorizontalScrollView>。

滚动视图同样可以在 xml 文件或者 java 文件中添加。

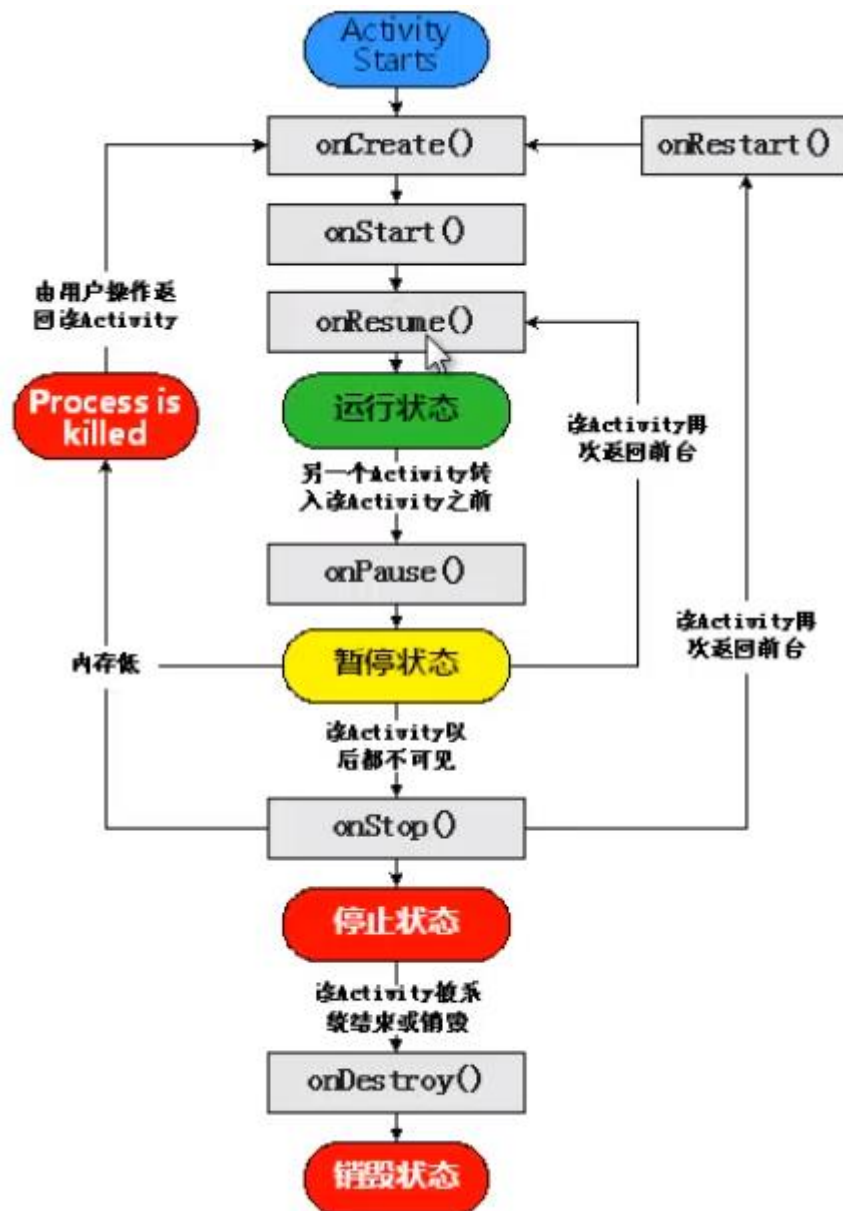
⑩选项卡

第一步：在布局文件中添加 TabHost、TabWidget 和 TabContent 组件；第二步编写个标签页的 xml 布局文件；第三步获取并初始化 TabHost 组件；第四步为 TabHost 对象添加标签页。

5、Activity

①基本概念：

Android 的四大组件之一，主要是提供一个屏幕，用于用户的交互。Activity 拥有四种基本转态：运行状态、暂停状态、停止状态、销毁状态。活动声明周期图：



②创建、配置、启动、关闭、刷新 activity

一个程序只有一个入口 Activity，在 `AndroidManifests.xml` 中设置设置程序入口：

```
<category android:name="android.intent.category.LAUNCHER" />
```

配置创建 Activity，第一种方法：在 `MainActivity.java` 同级目录下创建一个新的 activity 名为 `DatilActivity.java`，继承 Android 的 Activity，重写方法 `onCreate()`，指定布局文件 `setContentView(R.layout.xxx)`；最后在 `manifests` 目录下的 `AndroidManifests.xml` 文件中配置 `DatilActivity`。

第二种方法：右击包名新建 Activity，选择 empty Activity 创建一个新的 Activity，

Android 导向会自动配置 Activity,在创建的时候还可以指定布局文件。

启动 Activity, 创建 Intent 对象 (Intent 相当于 activity 的意图):

Intent intent(MainActivity.this,MyActivity.class);第二个参数就是你想要启动的 Activity。然后使用 startActivity(intent);启动新的 Activity。

关闭 activity: 在监控事件中调用 finish() 方法。关闭当前 activity 后, 页面会返回上一个调用的 activity 中。如果该方法在入口 activity 中程序会退出。

刷新 Activity: 在监控事件中调用 onCreate(null)。

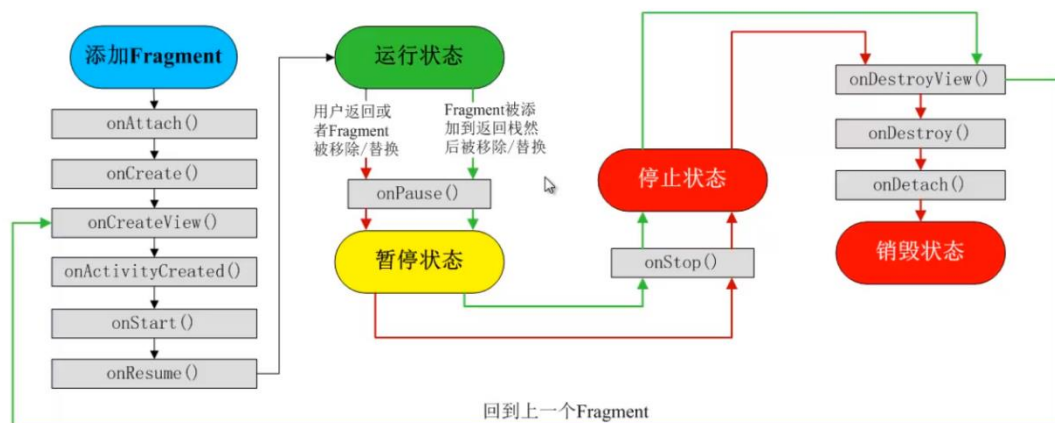
③多个 Activity 使用

Activity 之间传递数据,Intent 对象不能存储数据,需要将数据先存在 Bundle 对象中,保存为键值对形式, 在通过 Intent 对象的 putExtras() 方法将 Bundle 对象加载到 Intent 对象中,在通过 startActivity(intent)启动要传数据的 Activity 就可以了。新的 Activity 通过 Bundle 数据的键值 getString(“键值”)取出。

④调用另一个 Activity 并返回结果

启动 Activity 时使用 startActivityForResult(“要跳转的 activity”,返回码); 新的 Activity 用 setResult()方法返回结果。再在 activity 中继承 onActivityResult()方法接收数据。

⑤Fragment 概念和生命周期



fragment 是 Android3.0 后引入的一个新的 API, 初衷是为了适应大屏幕的平板电脑。Fragment 可以理解为小型的 Activity, 即 Activity 片段。它有自己的生命周期, 且必须依赖于 Activity。通过 Fragment 可以将屏幕划分为几块, 进行分组, 实现模块化化管理。

⑥Fragment 的使用

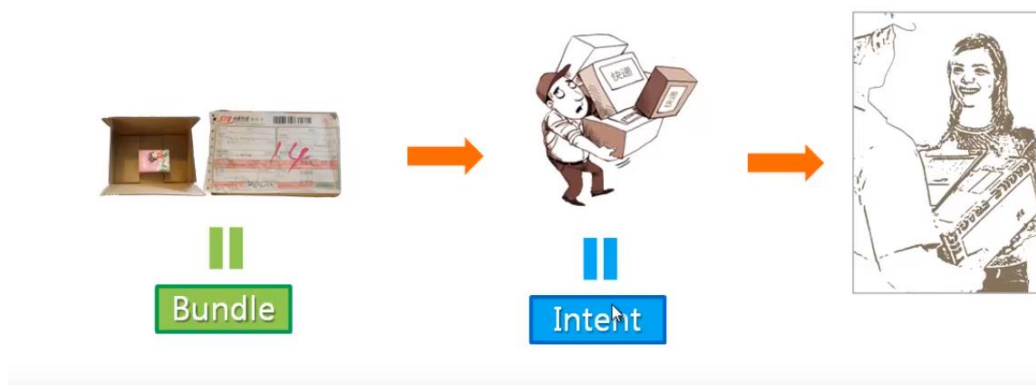
将 Fragment 添加到 Activity 中通常有两种方式: 一是直接在 Activity 的布局文件中生命 fragment; 二是通过 java 代码将 fragment 添加到已存的 ViewGroup 中。开发当中一般使用第二种方式。

过程: 创建 java 类, 继承 Fragment, 创建布局文件, 重写 onCreateView() 方法, 为 fragment 指定布局文件。第一种添加方式, 直接在 Activity 的布局文件中使用 Fragment 标记添加, 通过 Fragment 标记的 name 属性指定要显示的 fragment java 类就可以了。第二种方式, 在 Activity 中实例化 Fragment 对象, 然后获取 FragmentTransaction 实例, 即 FragmentTransaction ft = getSupportFragmentManager().beginTransaction(); 再使用 FragmentTransaction 的 add() 方法, add() 两个参数, 第一个参数指定 Fragment 要放入哪个容器当中, 第二个指定 fragment。最后使用 commit() 方法提交。

6、Android 应用核心 Intent

①Intent 概述

Intent 主要作用是为了实现组件间的通讯。如图所示：



Intent 相当于快递员，快递员传送快递(携带信息的 Bundle 对象)传递给(用户)组件。具体事例如下图所示。



②Intent 应用

第一种应用：开启一个 Activity，比如登录界面。

第二种应用：开启一个 server，比如下载一个文件或应用。

第三种应用：发送广播，发送广播到接收广播，需要 intent 传递。

③Intent 对象属性

Component name 属性：设置 Intent 对象组件名称的，通过设置 component name 可以启动其他的 Activity。

```
//定义componentName 对象
ComponentName componentName = new ComponentName(
    包名 → pkg: "hncj.edu.cn.intent", |
    cls: "hncj.edu.cn.intent.DetailActivity");
intent.setComponent(componentName);
startActivity(intent);          activity完整路径
```

Action 和 data 属性: action 是动作, data 是事件、目的或者说数据。比如我想要喝水。如下图是常见的 Activity Action Intent 常量。

常量名称	常量值	意义
ACTION_MAIN	android.intent.action.MAIN	应用程序入口
ACTION_VIEW	android.intent.action.VIEW	显示数据给用户
ACTION_ATTACH_DATA	android.intent.action.ATTACH_DATA	指明附加信息给其他地方的一些数据
ACTION_EDIT	android.intent.action.EDIT	显示可编辑的数据
ACTION_PICK	android.intent.action.PICK	选择数据
ACTION_CHOOSER	android.intent.action.CHOOSER	显示一个Activity选择器
ACTION_GET_CONTENT	android.intent.action.GET_CONTENT	获得内容
ACTION_DIAL	android.intent.action.DIAL	显示打电话面板
ACTION_CALL	android.intent.action.CALL	直接打电话
ACTION_SEND	android.intent.action.SEND	直接发短信
ACTION_SENDTO	android.intent.action.SENDTO	选择发短信
ACTION_ANSWER	android.intent.action.ANSWER	应答电话
ACTION_INSERT	android.intent.action.INSERT	插入数据
ACTION_DELETE	android.intent.action.DELETE	删除数据
ACTION_RUN	android.intent.action.RUN	运行数据
ACTION_SYNC	android.intent.action.SYNC	同步数据
ACTION_PICK_ACTIVITY	android.intent.action.PICK_ACTIVITY	选择Activity
ACTION_SEARCH	android.intent.action.SEARCH	搜索
ACTION_WEB_SEARCH	android.intent.action.WEB_SEARCH	Web搜索
ACTION_FACTORY_TEST	android.intent.action.FACTORY_TEST	工厂测试入口点

具体实例: 拨打电话、发送邮箱(注意要在 AndroidManifest.xml 开启相应权限)。

```

case R.id.phone:
    intent.setAction(intent.ACTION_DIAL);
    intent.setData(Uri.parse("tel:03752089092"));
    startActivity(intent);
case R.id.mail:
    intent.setAction(intent.ACTION_SENDTO);
    intent.setData(Uri.parse("mailto:zs@hncj.edu.cn"));
    startActivity(intent);

```

Category 属性: 对执行动作的类别进行描述的。同样的也可以通过 category 提供的常量进行设置。常量表如下图所示。

Category常量	对应字符串	描述
CATEGORY_DEFAULT	android.intent.category.DEFAULT	默认的Category
CATEGORY_BROWSABLE	android.intent.category.BROWSABLE	指定该Activity能被浏览器安全调用
CATEGORY_TAB	android.intent.category.TAB	指定Activity作为TabActivity的Tab页
CATEGORY_LAUNCHER	android.intent.category.LAUNCHER	Activity显示顶级序列表中
CATEGORY_INFO	android.intent.category.INFO	用于提供包信息
CATEGORY_HOME	android.intent.category.HOME	设置该Activity随系统启动而运行
CATEGORY_PREFERENCE	android.intent.category.PREFERENCE	该Activity是参数面板
CATEGORY_TEST	android.intent.category.TEST	该Activity是一个测试
CATEGORY_CAR_DOCK	android.intent.category.CAR_DOCK	指定手机被插入汽车底座(硬件)时运行该Activity
CATEGORY_DESK_DOCK	android.intent.category.DESK_DOCK	指定手机被插入桌面底座(硬件)时运行该Activity
CATEGORY_CAR_MODE	android.intent.category.CAR_MODE	设置该Activity可在车载环境下使用

具体实例：点击返回按钮，应用返回桌面。

```
Intent intent = new Intent();  
intent.setAction(intent.ACTION_MAIN);  
intent.addCategory(intent.CATEGORY_HOME);  
startActivity(intent);
```

Extras 属性：主要通过 intent 向组件添加附加信息的，通常情况下这些信息是以键值对的形式保存的。通常在多个 activity 交换数据时使用，使用方法：放入信息 putExtras(); 取出信息 getExtras()。具体实例在多个 Activity 中已经实现。

Flags 属性：表示不同来源的标记，多数用于只是 Android 如何启动 Activity 以及启动以后如何对待。标记都以常量形式定义在 Intent 类中。

FLAG_ACTIVITY_NEW_TASK: 默认的跳转类型, 会重新创建一个新的 Activity (第一种)
FLAG_ACTIVITY_SINGLE_TOP: 相当于 Activity 加载模式中的 singleTop (第二种)
FLAG_ACTIVITY_CLEAR_TOP: 相当于加载模式中的 singleTask (第三种)
FLAG_ACTIVITY_REORDER_TO_FRONT: 如果 activity 在 task 存在, 拿到最顶端, 不会启动新的 Activity
FLAG_ACTIVITY_NO_HISTORY: 被启动的 Activity 一旦退出, 他就不会存在于栈中

示例改变应用返回桌面，再点击应用显示的 Activity。

```
Intent intent = new Intent(MainActivity.this, DetailActivity.class);  
intent.setFlags(intent.FLAG_ACTIVITY_NO_HISTORY);  
startActivity(intent);
```

④显式 Intent 和隐式 Intent

显式 Intent 就是为 Intent 指定具体组件，如跳转到具体的 Activity 中。隐式 Intent 就是不指定具体组件，而是通过 Action、data、category 属性，设置常量通过 Android 系统自动匹配目标组件。区别就是显式 Intent 多用于在应用程序内部传递信息；隐式 Intent 多用于不同应用程序之间传递信息。

⑤Intent 过滤器

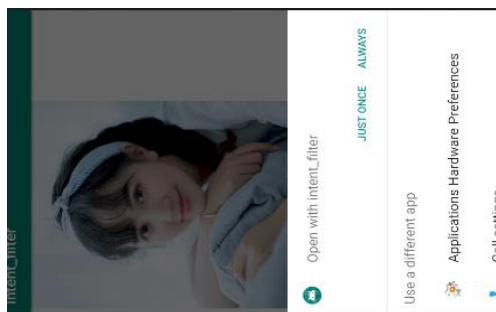
主要应用于用隐式 Intent 启动 Activity 时。通过在 AndroidManifest.xml 文件的 <intent-filter> 标记配置的。主要由 <action> 指定组件所能响应的动作、<category>：是以哪种方式执行 intent 请求动作、<data>：向 action 提供要操作的数据。组成配置。

实例：通过过滤器选择图片用哪个应用查看。在 MainActivity 中并不指定跳转的 activity，而是在 AndroidManifest.xml 文件进行过滤器配置。

```
Intent intent = new Intent();  
intent.setAction(intent.ACTION_VIEW);  
startActivity(intent);
```

```
<activity android:name=".showActivity">  
  <intent-filter>  
    <action android:name="android.intent.action.VIEW"/>  
    <category android:name="android.intent.category.DEFAULT"/>  
  </intent-filter>  
</activity>
```

效果图：



7、调试程序

①DDMS

D:\apps\AppData\Local\Android\Sdk\tools\lib\monitor-x86_64\monitor.exe

②日志信息输出

Log 类位于 android.util 包中，主要有五个方法。

1. Log.v() v，即 Verbose，中文：详细的，输出最最普通的信息。

两重载：

```
public static int v(String tag, String msg)
```

```
public static int v(String tag, String msg, Throwable tr)
```

tag 为标签，一般为调用类的名称，msg 为输出信息，其他输出方法类似。



2. Log.d() d，即 Debug，输出调试信息。

3. Log.i() i，即 Information，输出一般信息。


4. Log.w() w，即 Warning，警告信息。

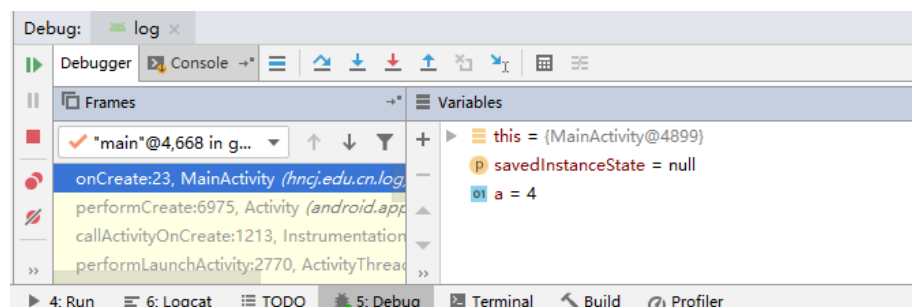
5. Log.e() e，即 Error，输出错误信息。


③Android Studio 编辑器调试


有红色波浪线的代码通过 alt+enter 弹出提示修改信息。编辑器右上角有  或  图标，鼠标点击有错误或警告的信息。在编辑器右侧通过短红线和短黄线，点击可定位到出现错误或警告的代码行。


④Android Studio 调试器调试


在编辑器左侧打断点，点击调试工具 ，进入调试界面。





 单步跳过，运行一行代码，不进入调用方法的内部，然后跳到下一个断点 (F8)。

 单步跳入，跳入调用方法或对象的内部单步执行程序 (F7)。

 强制单步跳入，跳入所有被调用的方法 (ALT+SHIFT+F7)。

 单步跳出，跳出当前进入的方法 (shift+F8)。

 跳入到下一个断点，如果没断点或异常，会直接运行到程序结束。

 查找某一变量当前的值，也可以让几个变量进行简单的计算。

8、Android 事件处理和手势

①事件处理

②基于监听的事件处理。

监听事件处理前面多次应用就不说了。

③基于回调的事件处理。

回调事件处理即：当某一动作发生时调用的方法。区别：基于回调的事件处理方式一般应用于通用性事件；基于监听的事件处理方式一般应用于某些特定的事件处理事件。

④物理按键事件处理

Android 为物理按钮（音量键，返回键等）都提供了三种方法：onKeyDown() 按下；onKeyUp() 抬起；onKeyLongPress() 长按。判断按下的是哪个键，android 也为常用的物理键提供了对应的常量。

电源键	KEYCODE_POWER
后退键	KEYCODE_BACK
菜单键	KEYCODE_MENU
HOME 键	KEYCODE_HOME
相机键	KEYCODE_CAMERA
音量键	KEYCODE_VOLUME_UP / KEYCODE_VOLUME_DOWN
搜索键	KEYCODE_SEARCH
方向键	KEYCODE_DPAD_CENTER KEYCODE_DPAD_UP KEYCODE_DPAD_DOWN KEYCODE_DPAD_LEFT KEYCODE_DPAD_RIGHT
键盘键	KEYCODE_0...KEYCODE_9 KEYCODE_A...KEYCODE_Z

⑤单击事件

⑥长按事件和触摸事件

长按事件：要求长按时间在 2 秒以上，方法：setOnLongClickListener()

触摸事件：方法：setOnTouchListener。如果同时发生单击事件和触摸事件，触摸事件先执行。单击事件触发一个动作，触摸事件触发两个动作。

⑦手势检测

方法：onDown() 触摸事件按下时触发；onFling() 手指在屏幕拖过时触发；onLongPress() 手指长按时触发；onScroll(); onShowPress(); onSingleTapUp();

⑧自定义手势的创建和导出

在 AndroidStudio 模拟器中有一个  应用可以创建自定义手势，在通过 DDMS 到文件管理器中找到自定义的手指资源文件（gestues 文件），既可以导出文件。

⑨手势的识别

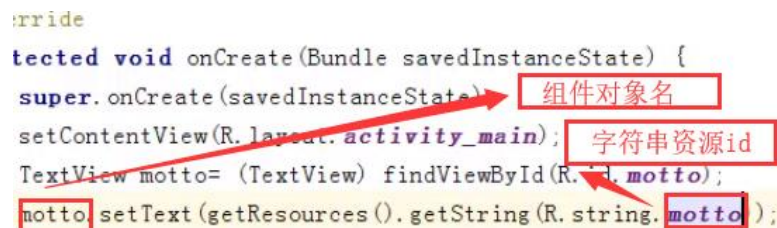
9、Android 应用的资源

资源文件都可以自定义，根据标签不同，设置资源类型。

①字符串资源

字符串资源可以在 xml 布局文件中或 java 文件中使用。Android 默认的字符串资源文件是 strings.xml，主要是通过<string>标记表示的。在 xml 中使用：@string/字符串资源 id；在 Java 中使用：通过组件的 setText() 方法设置，通过 getResources().getString(R.string.字符串资源 id) 获取资源。

```
override  
protected void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.activity_main);  
    TextView motto= (TextView) findViewById(R.id.motto);  
    motto.setText(getResources().getString(R.string.motto));  
}
```



②颜色资源

颜色资源 Android 中用：# 透明度 RGB 表示。Android 默认的字符串资源文件是 colors.xml，主要是通过<color>标记表示的。使用时与字符串资源相同。如设置字体颜色：调用方法 setTextColor(), getResources().getColor(R.color.)

③尺寸资源

dp: 设备独立像素。根据手机屏幕大小不同，显示的尺寸不同，可以根据屏幕大小自动调整。主要用于设置边距和组件大小。sp: 可伸缩像素，用于设置字体大小，可以根据手机字体大小而改变。通过<dimen>标记定义尺寸。使用时与字符串资源相同。如：设置字体大小：调用方法 set textSize(), getResources().getDimension(R.dimen.)

④布局资源

Layout 目录下的就是布局资源。可以在 java 代码中使用也可以在 xml 其他布局文件中使用。Java 中使用：setContentView(R.layout.布局文件名)；xml 其他布局文件中使用：<include layout="@layout/布局文件名" />

⑤数组资源

数组资源主要有三种标签定义：<array>: 普通类型数组，可以放置颜色、大小、字符串等资源；<integer-array>: 整型数组；<string-array> 字符串数组。数组里面的元素用<item>标签。使用时同样和字符串资源相同。


⑥drawable 资源

可以放入图片资源。。9.png 图片制作 (draw9patch.bat)。StateListDrawable 资源 (状态列表资源)。可以根据组件处于不同的状态改变组件的形状，颜色等等。如根据输入框是否获得焦点改变输入框中字体的颜色；根据按钮的点击改变按钮的大小等等。

⑦mipmap 资源

通常保存的都是图标，使用时和字符串资源相同。Mipmap 和 drawable 资源区别：mipmap 主要存储应用的启动图标；drawable 主要存储作为背景或修饰的图片文件、9patch 图片、Shape 图形资源、StateListDrawable 资源等。

⑧主题资源

在局部文件设计视图，通过  AppTheme 可以选择更多系统提供的视图主题。我们也可以使用自定义的主题资源，在 res 目录下 values 目录中有 styles.xml 就是主题资源文

件。可以修改相关内容，对应用的外观主题进行设置。如修改屏幕背景图片：

```
<style name="bgTheme" parent="@style/AppTheme">
    <item name="android:windowNoTitle">false</item>
    <item name="android:background">@drawable/bc</item>
</style>
```

使用的时候可以在 java 中或者 AndroidManifest.xml 中使用。可以为整个 app 设置主题也可以为一个 activity 设置主题。AndroidManifest.xml 中使用：theme="@style/bgTheme"。在 java 中使用：需要在加载布局资源文件之前使用 setTheme(R.style.主题资源 id)。

⑨样式资源

主题是设置整个 app 或窗口样式的，样式是设置组件的样式的。同样在 style 中定义设置样式资源。样式资源一般在布局文件中使用。比如定义字体颜色：

```
<style name="title">
    <item name="android:textSize">30dp</item>
    <item name="android:textColor">#9C27B0</item>
</style>
<style name="context">
    <item name="android:textSize">20dp</item>
    <item name="android:textColor">#009688</item>
```

在布局文件中使用：style="@style/title",style="@style/context".

⑩菜单资源

创建菜单资源文件：res->new->Directory 输入文件名 menu,menu->Menu resource file 输入菜单资源文件名 menu.xml。使用<item>标记定义菜单项。

```
<item android:id="@+id/about" android:title="关于"></item>
<item android:id="@+id/setting" android:title="设置"></item>
```

创建选项菜单如下图所示：

```
//第一步重写onCreateOptionsMenu()方法，添加一个选项菜单
@Override
public boolean onCreateOptionsMenu(android.view.Menu menu) {
    MenuInflater menuInflater = new MenuInflater(context: this)
    menuInflater.inflate(R.menu.menu, menu);
    return super.onCreateOptionsMenu(menu);
}
```

```
//第二步对各个菜单项处理
@Override
public boolean onOptionsItemSelected(@NonNull MenuItem item) {
    switch(item.getItemId()){
        case R.id.about:
            Intent intent = new Intent(packageContext: Menu.this, MenuAbout.class);
            startActivity(intent);
            break;
        case R.id.setting:
            Intent intent1 = new Intent(packageContext: Menu.this, MenuSetting.class);
            startActivity(intent1);
            break;
    }
    return super.onOptionsItemSelected(item);
}
```

创建上下文菜单：在某一组件长按时弹出的菜单。上下文菜单的实现与选项菜单基本相同。唯一一点不同就是为上下文菜单的控件注册上下文菜单。如下图为一个文本框注册上下文菜单。

```
textView = (TextView)findViewById(R.id.textView8);  
registerForContextMenu(textView); //为文本框注册上下文菜单
```

弹出菜单：默认情况下，弹出式菜单（PopupMenu）会在指定组件的上方或下方弹出。使用弹出菜单与前两种菜单不同：1. 调用 new PopupMenu 创建下拉菜单，anchor 代表要激发弹出菜单的组件。2. 调用 MenuInflater 的 inflate() 方法将菜单资源填充到 PopupMenu 中。3. 调用 PopupMenu 的 show() 方法显示弹出式菜单。如下图所。

```
public void popupMenuClick(View view){  
    popupMenu = new PopupMenu(context, this, view);  
    getMenuInflater().inflate(R.menu.popupmenu, popupMenu.getMenu());  
    popupMenu.setOnMenuItemClickListener(  
        new PopupMenu.OnMenuItemClickListener() {  
            @Override  
            public boolean onMenuItemClick(MenuItem menuItem) {  
                switch (menuItem.getItemId()) {  
                    case R.id.hide:  
                        popupMenu.dismiss();  
                    default:  
                        Toast.makeText(context, Menu.this, text: "if  
                }  
            }  
            return true;  
        }  
    );  
    popupMenu.show();  
}
```

```
android:onClick="popupMenuClick"
```

10、Action bar 的使用(顶部标题栏)

①显示和隐藏 Action bar

隐藏 Action bar 在 AndroidManifest.xml 中 activity 添加主题后缀带有.NoActionBar 的主题就可以隐藏。也可以在 java 文件中控制 Action bar 的显示和隐藏。第一步获取 ActionBar 对象;第二步通过 ActionBar 对象的 show() 方法和 hide() 方法控制 ActionBar 的显示或隐藏

```
final ActionBar actionBar =getSupportActionBar();
```

```
actionBar.show(); actionBar.hide();
```

②添加 Action Item

`<!--showAsAction`

`always`:菜单项一直显示在ActionBar上

`ifRoom`:如果ActionBar有空间就显示在ActionBar上

`never`:不显示ActionBar上, 显示在溢出菜单中-->

```
<menu xmlns:android="http://schemas.android.com/apk/res/android"
      xmlns:app="http://schemas.android.com/apk/res-auto">
    <item android:id="@+id/search" android:icon="@drawable/search"
          android:title="search"
          app:showAsAction="ifRoom"></item>
    <item android:id="@+id/share" android:icon="@drawable/share"
          android:title="share"
          app:showAsAction="ifRoom"></item>
    <item android:id="@+id/setting"
          android:title="设置"
          app:showAsAction="never"></item>
    <item android:id="@+id/about"
          android:title="关于"
          app:showAsAction="never"></item>
```

在 java 程序中添加菜单。

```
//第一步重写onCreateOptionsMenu()方法, 添加一个选项菜单
@Override
public boolean onCreateOptionsMenu(android.view.Menu menu) {
    MenuInflater inflater = new MenuInflater(context, this)
    inflater.inflate(R.menu.menu, menu);
    return super.onCreateOptionsMenu(menu);
}
```

③添加 Action View

在 Action Bar 上添加组件。



④ActionBar 实现 tab 导航

第一步：设置使用 Tab 导航方式；第二步：添加多个 Tab 标签页，并为每个 Tab 标签页添加事件监听。（未实现例子）

⑤ActionBar 实现层级式导航

步骤：



具体实现：

```
//判断父Activity是否为空，不为空设置导航图标显示
if(NavUtils.getParentActivityName( sourceActivity: friendsActivity. this) !=null) {
    getSupportActionBar().setDisplayHomeAsUpEnabled(true); //显示向左的箭头图标导航
}
```

```
<activity android:name=".friendsActivity"
    android:label="朋友圈">
    <meta-data
        android:name="android.support.PARENT_ACTIVITY"
        android:value=".MainActivity"/>
</activity>
```


11、消息、通知和广播

①Toast 消息提示框

```
Toast.makeText(context: MainActivity.this, text: "消息内容", Toast.LENGTH_SHORT).show();
```

②使用 AlertDialog 实现对话框



③使用 Notification 显示通知

使用 Notification 显示通知的步骤如下图。



(实例未成功)

④发送、接收广播

(实例未成功)

⑤使用 AlarmManager 设置闹钟

(实例未成功)



12、图像图形处理技术

①使用画布画图

第一步创建一个继承 View 类的自定义 view；第二步在布局管理器中添加自定义 View。

②绘制几何图形、文本

1.Canvas相当于画布，所有的图形都在其上面绘制并显示出来。Paint相当于画笔，可以设置不同颜色等，画出不同图形。

以下均在onDraw(Canvas canvas)执行，定义了Paint paint = new Paint();

2.背景设置颜色 canvas.drawColor()，例如：canvas.drawColor(Color.WHITE);

3.去锯齿paint.setAntiAlias(true);

4.设置paint的颜色paint.setColor(Color.RED);

5.设置paint的style 空心： paint.setStyle(Paint.Style.STROKE);

实心： paint.setStyle(Paint.Style.FILL);

6.设置paint的外框宽度 paint.setStrokeWidth(3);

7.画圆： canvas.drawCircle(cx, cy, radius, paint);

8.画正方形： canvas.drawRect(left, top, right, bottom, paint);

9.画长方形： canvas.drawRect(left, top, right, bottom, paint);

10.画椭圆： RectF re = new RectF(left, top, right, bottom);

11.画三角形： Path path = new Path();
path.moveTo(10, 330);//第一个点
path.lineTo(70, 330);//第二个点
path.lineTo(40, 270);//第三个点
path.close();
canvas.drawPath(path, paint);

12.画梯形： Path path1 = new Path();
path1.moveTo(10, 410);
path1.lineTo(70, 410);
path1.lineTo(55, 350);
path1.lineTo(25, 350);
path1.close();
canvas.drawPath(path1, paint);

13.画扇形： canvas.drawArc(oval, startAngle, sweepAngle, useCenter, paint)

例如： canvas.drawArc(new RectF(330, 0, 480, 150), 0, 270, true, paint);

14.设置渐变色： Shader mShader = new LinearGradient(0, 0, 100, 100, new int[] {
Color.RED, Color.GREEN, Color.BLUE, Color.YELLOW }, null,
Shader.TileMode.REPEAT);

paint.setShader(mShader);

15.写字： paint.setTextSize(24);

canvas.drawText(text, x, y, paint);

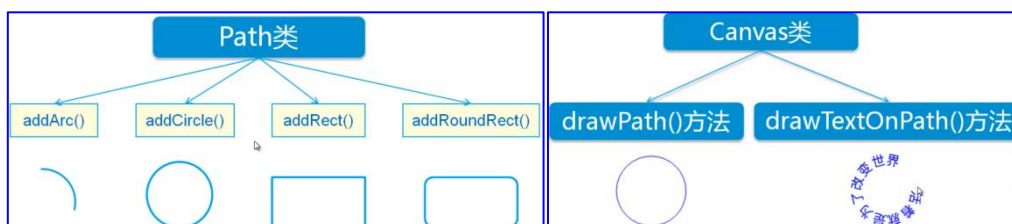
生成自定义view后,使用主Activity调用这个view。假设我们的View名是MyView, 则应 setContentView(new MyView(this))。也可以直接加载到布局文件中如
frameLayout.addView(new MyView(this))。(Moduel:paint)

③绘制图形（位图对象）

通过 BitmapFactory 类的 decodeFile() 方法创建位图对象，主要是通过图片的路

径来创建的；在通过 canvas 的 drawBitmap 方法将图片绘制到屏幕上；再用 Bitmap 的 createBitmap() 方法可以在原图上挖取一小块区域，再次通过 canvas 的 drawBitmap 方法绘制到屏幕上。（Moduel:paint）

④绘制路径



⑤逐帧动画

首先创建一个动画资源文件，drawable 下新建一个 xml 文件，用 animaction-list 标记加入 item 设置图片资源、和播放间隔。在主 activity 中 imageview 中设置背景，引用这个资源。在主 activity 类中设置 imageview 的监听事件，通过实例化 AnimationDrawalbe 对象，用 iamgeview 实例化对象引用 getBackground()//获取动画资源，在通过一个变量控制动画播放状态。关键代码如下。（Moduel:paint）

```

private boolean flag = true ;//记录播放状态
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_donghua);

    ImageView imageView = (ImageView)findViewById(R.id.image);
    final AnimationDrawable anim = (AnimationDrawable) imageView.getBackground();//获取动画资源
    imageView.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            if(flag==true){
                anim.start();//播放动画
                flag = false;
            }else{
                anim.stop();//停止动画
                flag = false;
            }
        }
    });
}
  
```

⑥补间动画

Android 主要提供了四种补间动画：分别是透明度渐变动画、旋转动画、缩放动画、平移动画。首先在 res 目录下创建 anim 目录，然后在 anim 目录下创建动画资源文件，通过以下标记添加相应动画，通过各种标签下不同属性控制动画运动属性。

（Moduel:paint）



13、多媒体应用开发

①播放音频

使用 MediaPlayer 播放音频，创建 MediaPlayer 有两种方式如下图。



使用 SoundPool 播放音频，与 MediaPlayer 相比，它延迟短、占用资源少、且支持同时播放多个音频。不过 SoundPool 只能播放短小的音频。主要用来播放按键音或提示音，也可以在游戏中播放密集且短暂的声音等。基本步骤创建 SoundPool 对象，load() 加载音频，play() 播放音频。（media）

②播放视频

播放视频也有两种方式：VideoView 和 MediaPlayer(声音)+SurfaceView(画面)。使用 MediaPlayer 播放视频的步骤：先定义 SurfaceView 组件、创建 MediaPlayer 对象、加载视频、把视频输出到 SurfaceView 中，通过 setDisplay() 方法显示到屏幕上，最后通过 MediaPlayer 提供的 pause() \start() \stop() 的方法来控制视频的播放。

使用 VideoView 播放视频步骤：第一步通过 view->Tools Windows->Device File Explorer 打开模拟器 SD 卡文件管理面板，在 storage 目录下的 self->primary 目录，右击 primary 选择 upload 传入要播放的视频文件。第二步在布局文件中加入 VideoView 组件。第三步在 AndroidManifest.xml 中为应用设置访问内存的权限。具体如下图。

```
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
<uses-permission android:name="android.permission.MOUNT_UNMOUNT_FILESYSTEMS"
    tools:ignore="ProtectedPermissions"/>
```

第四步在 activity 中加载要播放的视频文件，如下图所示。

```
/*加载要播放的视频*/
File file = new File( pathname: Environment.getExternalStorageDirectory()+"/video.mp4");
if(file.exists()){
    video.setVideoPath(file.getAbsolutePath()); //指定要播放的视频
}else{
    Toast.makeText( context: videoView.this, text: "要播放的视频文件不存在", Toast.LENGTH_SHORT)
}
```

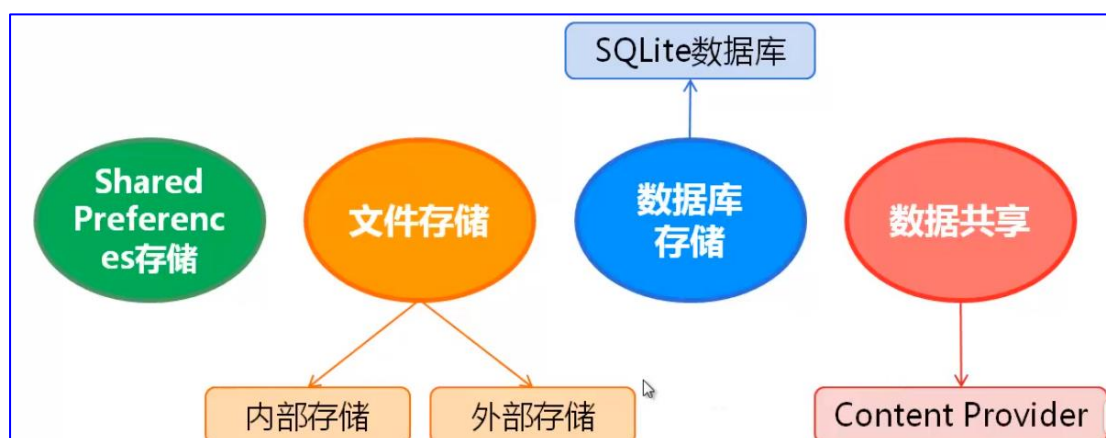
第五步通过 MediaController 对象控制视频的播放，具体如下图所示。

```
/*控制视频的播放*/
//创建mediacontroller对象
android.widget.MediaController mc = new android.widget.MediaController( context: videoView.this);
video.setMediaController(mc); //让video 和Mediacontroller关联
video.requestFocus(); //让videoView获得焦点
video.start(); //播放视频
video.setOnCompletionListener(new MediaPlayer.OnCompletionListener() {
    @Override
    public void onCompletion(MediaPlayer mediaPlayer) {
        Toast.makeText( context: videoView.this, text: "视频播放完毕", Toast.LENGTH_SHORT);
    }
});
```

- ③控制摄像头拍照（真机调试未实现）
- ④控制摄像头录像

14、数据存储技术

① 总体概述

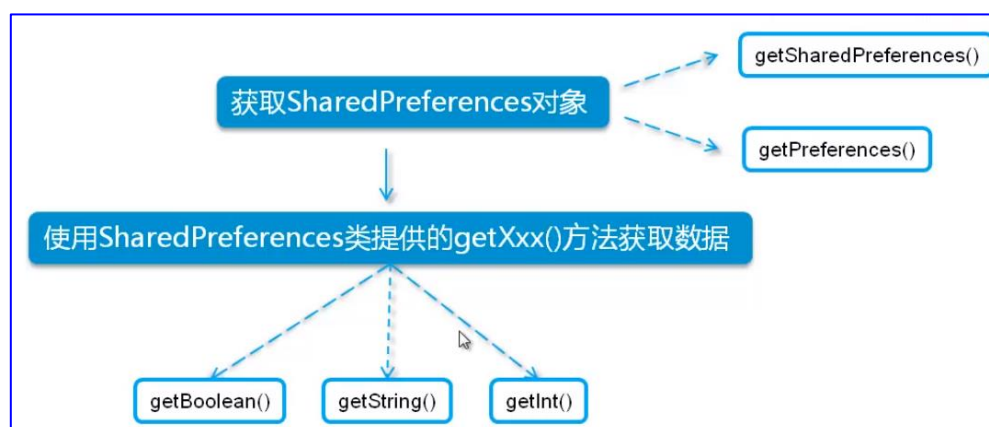


② 使用 SharedPreferences 存储、读取数据

使用 SharedPreferences 存储数据的步骤: 获取 SharedPreferences 对象主要有两种方式: 通过 `getSharedPreferences()` 方法或 `getPreferences()` 方法。主要步骤如下图所示。



使用 SharedPreferences 读取数据的步骤。



通过以上方法步骤实现 QQ 自动登录和手动登录, 通过 SharedPreferences 对象的 `getSharedPreferences()` 方法指定保存文件的文件名, 再通过这个方法的 `getString()` 方法获取数据。在保存数据的时候, 通过 SharedPreferences.Editor 对象的 `putString()`

方法保存字符串类型的数据，最后用 Editor 对象的 commit() 方法提交数据。具体实现如下图。

```
//获取Shared Preferences对象
final SharedPreferences sp = getSharedPreferences( name: "mrsoft", MODE_PRIVATE); //只能本应用访问
//*****实现自动功能*****
String username = sp.getString( s: "username", s1: null); //获取账号信息
final String password = sp.getString( s: "password", s1: null); //获取账号信息
if(username!=null&&password!=null){
    if(username.equals("mr")&&password.equals("mrsoft")){
        //通过Intent跳转到好友列表页面
        Intent intent = new Intent( packageContext: MainActivity. this, MessageActivity. class);
        startActivity(intent);
    }
} else{
    //*****实现手动登录并储存密码和账号*****
    login.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            String in_username = usernameET.getText().toString(); //获取输入的账号
            String in_password = passwordET.getText().toString(); //获得输入的密码
            SharedPreferences.Editor editor = sp.edit(); //获取Editor对象
            if(in_password.equals("mrsoft")&&in_username.equals("mr")){
                editor.putString( s: "username", in_username); //保存账号
                editor.putString( s: "password", in_password); //保存密码
                editor.commit(); //提交数据
                Intent intent = new Intent( packageContext: MainActivity. this, MessageActivity. class);
                startActivity(intent);
                Toast.makeText( context: MainActivity. this, text: "已保存账号和密码", Toast. LENGTH_LONG). show();
            } else{
                Toast.makeText( context: MainActivity. this, text: "账号或错误", Toast. LENGTH_LONG). show();
            }
        }
    });
}
```

密码账号信息保存路径：/data/data/<包名>/shared_prefs/mrsoft.xml

③内部存储

文件存储：通过 Java 的 IO 流读取磁盘上的文件。内部储存的文件路径：data->data-><包名>->files。内部存储的特点：默认只能被创建它的应用访问到；当应用卸载后，内部存储中的文件也会被删除；一旦手机内部存储空间耗尽手机将无法使用。具体实现步骤如下图。

写入文件	读取文件
① 获取FileOutputStream对象	① 获取FileInputStream对象
② 调用write()方法	② 调用read()方法
③ 调用flush()方法	③ 调用close()方法
④ 调用close()方法	

④外部储存

外部储存并不是指可移动的 SD 卡，内部储存和外部储存简单区分：将手机连接到电脑上能够被电脑识别的手机文件就是外部储存。

读、写外部存储空间上文件的步骤如下图所示。



⑤数据库存储-sqlite3 的使用

SQLite 数据库特点: 占用资源少、运行效率高、可移植性强、安全可靠。操作 SQLite 数据库的方式: sqlite3 工具; java 代码。

通过 sqlite3 工具操作 sqlite 数据库: 通过 cmd

1. 进入\apps\AppData\Local\Android\Sdk\platform-tools 目录下;
2. adb shell
3. mkdir /data/data/hncj.edu.datastorage(包名)/database 创建数据库根目录
如果 API 23 即 Android6.0 以上的版本, 没有 root 权限, 会拒绝修改。新建一个 API23 的 AVD 就行了。
4. cd /data/data/hncj.edu.datastorage(包名)/database 打开数据库
5. Sqlite3 启动 sqlite 工具
6. 进入 sqlite 工具, 创建表, 插入数据, 查找数据等操作数据库。

⑥数据库存储-使用代码操作 sqlite



⑦Content Provider

Content provider 使用基于数据模型的简单表格来提供其中的数据。用于不同应用间的数据交互。比如通过其它应用获取联系人信息、短信信息等。下图就是创建和使用 content Provider 的步骤。



15、Handel 消息处理

①handel 消息传递机制

线程和进程：在 Android 中一个运行的 app 既是一个进程，而一个进程可以包含多个线程。多个线程主要分为两大类主线程和子线程，在 Android 中子线程不允许操作主线程中的组件。而如果我们必须在子线程中更新主线程的 UI 组件，就需要用到 Handel。我们可以用子线程将要实现的操作通知给 handel，再由 handel 更改主线程中的组件。

Handel 的作用：第一在任意线程中发送消息；第二在主线程中获取并处理消息。

②Message 语法

Message 是封装了需要传递的数据交由 Handler 处理的对象。主要有两个作用：一、用于存放传递的数据；二、是主线程和子线程传递数据的载体。在 Android 中 handel 并不是单独工作的，与 handler 一起工作的还有 Looper\MessageQueue 的关系。Looper：循环管理器，主要管理 MessageQueue 消息队列。在 Android 当中一个线程包含一个 looper，一个 looper 包含一个 MessageQueue，一个消息队列包含多个 Message。工作流程：创建或获取一个 Message，通过 Handler 发送给 looper 管理的 messageQueue，looper 会按照队列先进先出的顺序取出 messagequeue 中的待处理的 message 返回给 handler 进行处理。如果没有 message，looper 会处于待命状态。创建使用 message，如下图。

- ① 使用 `Message.obtain()` 或 `Handler.obtainMessage()` 方法创建 Message
- ② 携带 `int` 型信息，优先使用 Message 的 `arg1` 和 `arg2` 属性
- ③ 使用 `Message.what` 来标识信息

③Looper 对象

在 message 中大致了解 Looper 其实是个循环，不断的取出消息，发送给 Handler 处理。每个 Handler 必须持有一个 Looper。在主线程中系统会自动创建 Looper 对象，而在子线程中需要手动创建 Looper 对象。在子线程中创建 Looper 对象的步骤如下图。



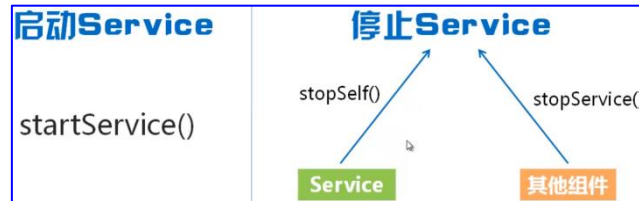
即在子线程中实例化 Handler 对象之前，调用 `Looper.prepare()` 初始化 Looper 对象；在 Handler 发送消息后再调用 `Looper.loop()` 方法启动 Looper。

16、Service 应用

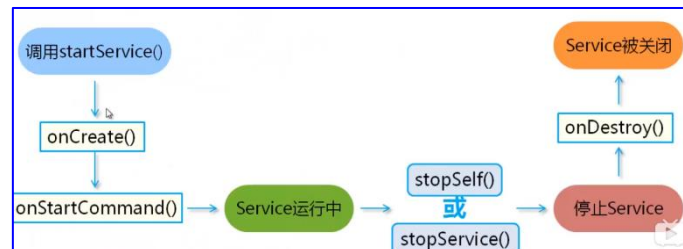
Service：能够在后台长时间运行，并且没有用户界面的应用程序组件。按照启动方式不同 service 可以分为：started Service 和 Bound Service。

①Service 基本用法 startedService

创建配置 Service：右击包名，new->Service->Service 新建 service 类。重写方法 onCreate() onStartCommand() onDestroy() 方法分别创建、启动、销毁 service。接着通过主进程中通过 Intent 对象使用 startService() 方法启动 service。如下图。



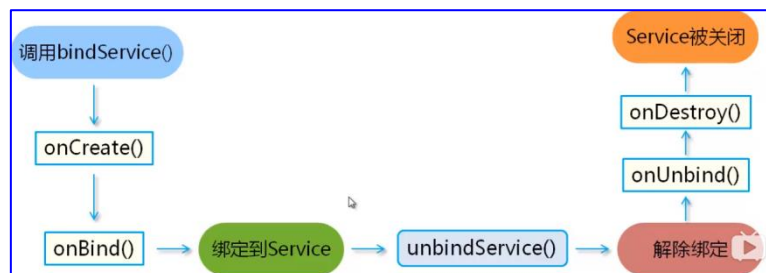
Service 的生命周期如下图所示。（Module: service）



②Service 基本用法 Bound Service

实现 Bound Service 基本步骤：创建一个 Service，重写 onBind() 方法；在 service 中创建一个 MyBinder 的内部类继承 Binder 类；修改 onBind() 方法，返回 IBinder 对象（用于与它绑定的组件进行通信的）；在 Activity 中创建 ServiceConnection 对象，用于获取 onBind() 方法返回到 IBinder 对象；在 Activity 中的 onStart() 方法中调用 bindService() 方法绑定 Service；在 onStop() 方法中调用 unbindService() 方法解除绑定的 Service。（Moudel:service）

Bound Service 的生命周期如下图。



③IntentService

IntentService 是继承自 Service 并处理异步请求的一个类，在 IntentService 内有一个个线程来处理耗时操作。优点：IntentService 会自动开启或结束 service。启动 IntentService 最好使用 startService 方式启动。应用：在不影响用户操作的情况下，在后台默默做一些任务。如下载或上传文件。

17、传感器

- ①磁场传感器（真机测试未实现）
- ②加速度传感器
- ③方向传感器
- ④感光传感器

18、位置服务与地图应用

①获取 LocationProvider

LocationProvider 用于提供定位信息。常用的 LocationProvider:



获取 LocationProvider: 1. 通过 LocationManager 对象的 getAllProviders() 方法获取所有可用的 LocationProvider (位置源)。 2. 通过名称获取 LocationProvider: LocationManager 对象的 getProvider() 方法。 3. 通过 Criteria 类获得 LocationProvider。具体如下图。

```
LocationManager locationManager = (LocationManager) getSystemService(LOCATION_SERVICE);
List<String> providerNames = locationManager.getAllProviders(); //获取所有的LocationProvider名称
StringBuilder stringBuilder = new StringBuilder(); //字符串构建起
for(Iterator<String> iterator = providerNames.iterator(); iterator.hasNext();){ //迭代器
    stringBuilder.append(iterator.next()+"\n");
}
textView.setText(stringBuilder.toString()); //显示获取的Locationprovider名称

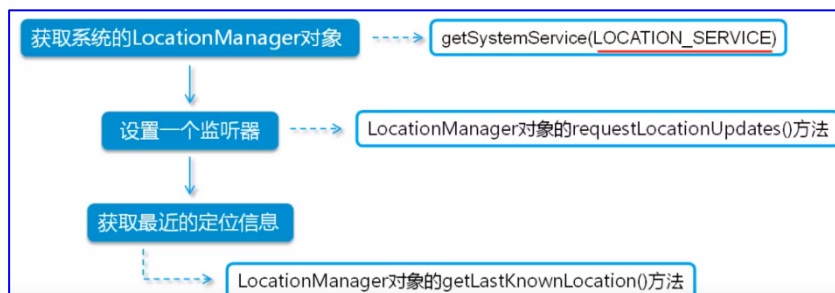
LocationManager locationManager = (LocationManager) getSystemService(LOCATION_SERVICE);
LocationProvider locationProvider = locationManager.getProvider(LocationManager.GPS_PROVIDER);
textView.setText(locationProvider.getName()); //显示获取的Locationprovider名称

LocationManager locationManager = (LocationManager) getSystemService(LOCATION_SERVICE);
//获取最佳的locationProvider
Criteria criteria = new Criteria(); //创建一个过滤条件对象
criteria.setCostAllowed(false); //不收费的
criteria.setAccuracy(Criteria.ACCURACY_FINE); //使用精度最准确的
criteria.setPowerRequirement(Criteria.POWER_LOW); //使用耗电量最低的
String provider = locationManager.getBestProvider(criteria, enabledOnly: true); //获取最佳的LocationProvider
textView.setText(provider); //显示获取的最佳的locationProvider
```

②获取定位信息

获取定位信息的步骤:

首先获取 LocationManager 对象, 通过 getSystemService(LOCATION_SERVICE) 获取的; 设置监听器; 通过自定义的 LocationUpdates() 方法将位置坐标显示在文本上, 主要是通过字符串构建器 StringBuilder 来实现的。在获取定位信息时需要调用 LocationManager 对象的 getLastKnownLocation() 方法来实时获取最新的位置信息。并且把获取的信息传到自定义的方法中。如下图所示。



③应用百度地图 API 开发地图 APP 实例

真机运行如果手动卸载会出现以下错误：

```
Error while executing: am start -n "pagekageName/pagekageName.ac.WelcomeActivity" -a android.intent.a
Starting: Intent { act=android.intent.action.MAIN cat=[android.intent.category.LAUNCHER] cmp=pagekageName
Error type 3
Error: Activity class {pagekageName/pagekageName.ac.WelcomeActivity} does not exist.

Error while Launching activity
```

app 没有卸载干净。解决办法：

adb uninstall packageName命令运行：

<https://www.bilibili.com/video/av22836860/?p=185>

19、Android 与 Javascript 交互

根据布局代码生成对用的 java 代码工具：

<https://www.buzzingandroid.com/tools/android-layout-finder/>

①加载网页

使用 WebView 控件加载网页，加载网页通常有以下几种方式：

```
//方式一：加载一个网页
webView.loadUrl("http://www.baidu.com");

//方式二：加载应用资源文件内的网页
webView.loadUrl("file:///android_asset/test.html");

//方式三：加载一段代码
webView.loadData(String data,String mimeType, String encoding);
```

具体实现如下图：

```
private void initWebView() { //加载网页
    webView = new WebView( context: this);
    WebSettings webSettings = webView.getSettings();
    webSettings.setJavaScriptEnabled(true); //设置支持js
    //不调用浏览器
    webView.setWebViewClient(new WebViewClient());
    /*加载网络网页和本地网页*/
    //网络网页（需要联网权限）
    // webView.loadUrl("https://www.baidu.com/index.php?tn=monline_3_dg");
    //本地网页
    webView.loadUrl("file:///android_asset/test.html");
    setContentView(webView);
}
```

②Java 与 Js 互调

Java 调用 Js 格式：

```
webView.loadUrl("javascript:javaCall("+number+"")");
setContentView(webView);
```

js方法名 参数值

Js 调用 Java：配置 Javascript 接口类：

```
//添加javascript接口
//js通过Android这个字段调用AndroidAndJsInterface中的任何方法
webView.addJavascriptInterface(new AndroidAndJsInterface(), name: "Android");
```

实现 Javascript 接口类

```
class AndroidAndJsInterface{
    @JavascriptInterface 注意一定要添加注解或者修改api版本为16以下
    public void showToast() {
        Toast.makeText( context: JavaAndJsCallActivuty. this, text: "调用成功", Toast.LENGTH_SHORT).show();
    }
}
```

附

①剪切板

实例：长按文本框弹出上下文菜单点击复制按钮复制文本。

```
//获取剪贴板管理器：
ClipboardManager cm = (ClipboardManager) getSystemService(Context.CLIPBOARD_SERVICE);
// 创建普通字符型ClipData
ClipData mClipData = ClipData.newPlainText(label: "Label", text: "这里是要复制的文字");
// 将ClipData内容放到系统剪贴板里。
cm.setPrimaryClip(mClipData);
```

②ImageButton

背景图片透明：android:background="#00000000"。

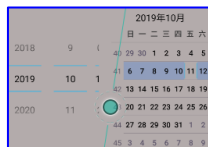
③DatePicker

android:theme="@android:style/Theme.DeviceDefault.Dialog"

android:theme="@android:style/Theme.Material"



android:theme="@android:style/Theme.Holo.Light.DarkActionBar"



android:theme="@android:style/Theme.Black"

android:theme="@android:style/Theme.Dialog"

android:theme="@android:style/Theme.Light.NoTitleBar"

android:theme="@android:style/Theme.NoTitleBar"

android:theme="@android:style/Theme.Translucent.NoTitleBar"

android:theme="@android:style/Theme.InputMethod"

android:theme="@android:style/Theme.WithActionBar"



style="@android:style/Widget.DatePicker"



④Message、MessageQueue、Looper、Handler 详解

<https://www.cnblogs.com/spec-dog/p/3795594.html>

