

Android基础入门

—— 余婷 副教授 yut@hznu.edu.cn ——

学习目标/Target



了解1G~5G的通信技术，能够说出1G~5G技术的发展内容

掌握Android Studio开发环境的搭建步骤，能够独立搭建Android Studio开发环境

掌握编写简单Android程序的步骤，能够编写一个Hello World程序

掌握资源的管理与使用方式，能够灵活使用程序中的资源

掌握单元测试与Logcat的使用，能够完成对程序的调试

章节概述/ Summary



Android是Google公司基于Linux平台开发的手机及平板电脑的操作系统，它自问世以来，受到了前所未有的关注，并迅速**成为移动平台最受欢迎的操作系统之一**。Android手机随处可见，如果能加入Android开发者行列，编写自己的应用程序供别人使用，想必是件诱人的事情。那么从今天开始，我们将开启Android开发之旅，并逐渐成为一名出色的Android开发者。





01

Android简介

02

Android开发环境搭建

03

开发第一个Android程序

04

Android程序结构



05

资源的管理与使用

06

程序调试

1.1



Android简介

>>> 1.1 Android简介



- 了解1G~5G的通信技术，能够说出1G~5G技术的发展内容
- 了解Android的发展历史，能够说出Android各版本对应的系统名称和图标
- 了解Android的体系结构，能够说出Android系统的4种分层结构
- 了解Dalvik虚拟机，能够说出Dalvik虚拟机编译文件的过程



1.1.1 通信技术



- **第一代通信技术 (1G)**：是指最初的模拟、仅限语音的蜂窝电话标准。



- **第二代通信技术 (2G)**：是指第2代移动通信技术，代表为GSM，以数字语音传输技术为核心。传输速度9.6k/s。



- **第三代通信技术 (3G)**：是指将无线通信与国际互联网等多媒体通信结合的新一代移动通信系统。3G通信网在室内、室外和行车的环境中能够分别支持至少2M/s、384K/s以及144K/s的传输速度。



- **第四代通信技术 (4G)**：又称IMT-Advanced技术，它包括了TD-LTE 和 FDD-LTE。4G通信网最高甚至可以达到100M/s的传输速度。



- **第五代通信技术 (5G)**：传输速度可达20Gbps。

>>> 1.1.2 Android发展历史

Android操作系统最初是由安迪·鲁宾（Andy Rubin）开发出的，后来被Google收购，并于2007年11月5日正式向外界展示了这款系统。随后Google以Apache开源许可证的授权方式，发布了Android操作系统的源代码。



>>> 1.1.2 Android发展历史

2008年9月发布Android**第1个**版本Android1.1。

2009年4月30日，Android1.5 **Cupcake**（纸杯蛋糕）正式发布。

2009年9月5日，Android1.6 **Donut**（甜甜圈）版本发布。

.....

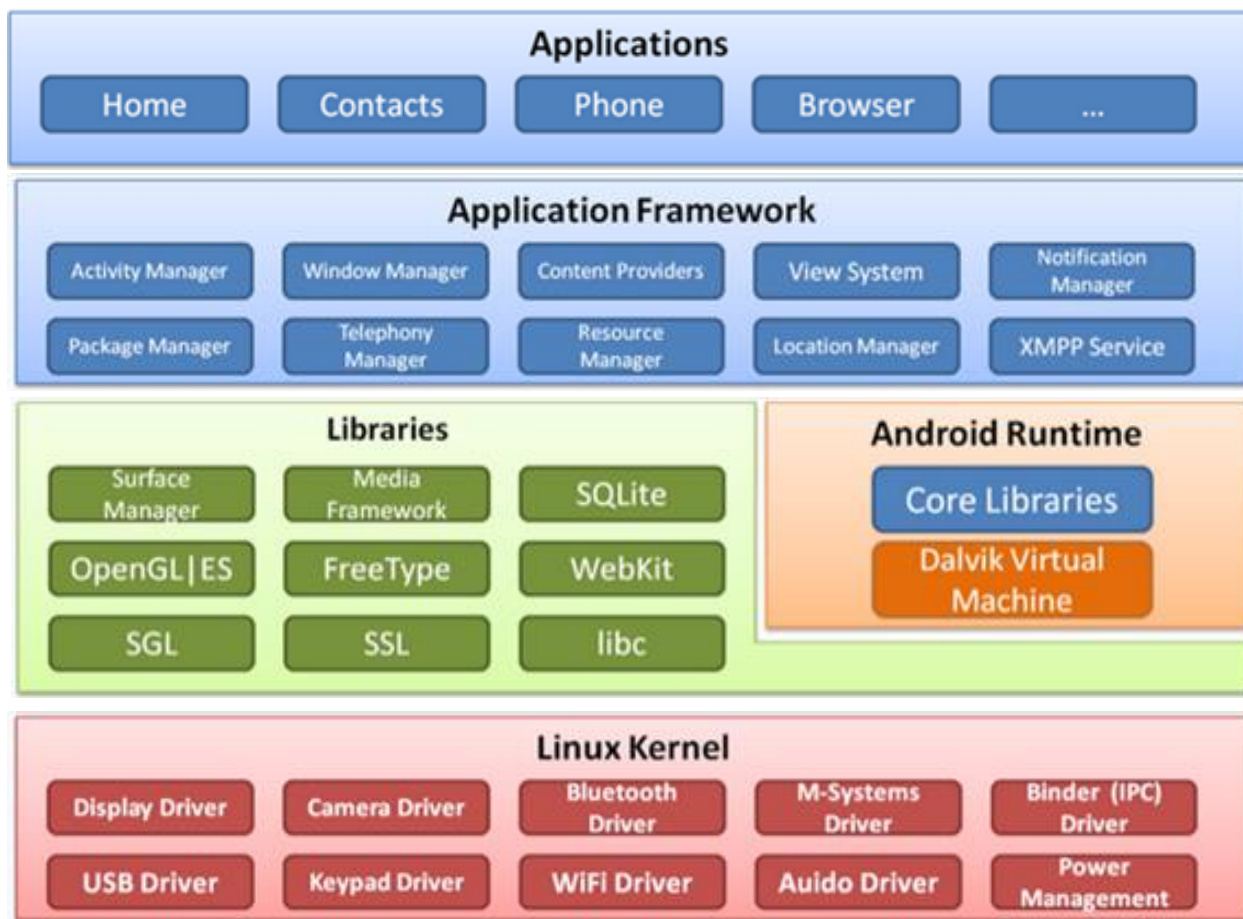


Android1.5 Cupcake
(纸杯蛋糕)



Android9.0 Android pie
(派)

>>> 1.1.3 Android体系结构



应用程序层



应用程序框架层



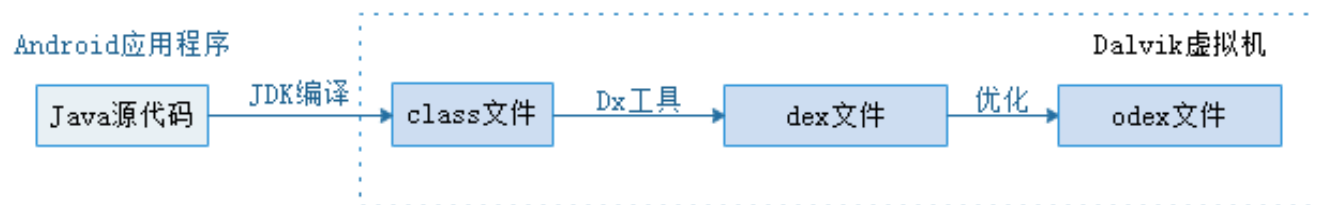
核心类库层



Linux内核层

>>> 1.1.4 Dalvik虚拟机

Dalvik是Google公司设计的，用于在Android平台上运行的虚拟机，其指令集基于寄存器架构，执行其特有的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。每一个Android应用在底层都会对应一个独立的Dalvik虚拟机实例，其代码在虚拟机的解释下得以执行，Dalvik虚拟机编译文件的过程如下图所示。



Dalvik虚拟机编译文件过程



1.2

Android开发环境搭建

>>> 1.2 Android开发环境搭建



- 掌握Android Studio开发环境的搭建步骤，能够独立搭建Android Studio开发环境
- 掌握模拟器创建的步骤，能够独立创建模拟器
- 掌握在Android Studio中下载SDK的步骤，能够独立下载SDK

>>> 1.2.1 Android Studio 安装

俗话说，“工欲善其事，必先利其器”。在开发Android程序之前，先要搭建开发环境。最开始Android是使用Eclipse作为开发工具的，但是在2015年底，Google公司声明不再对Eclipse提供支持服务，Android Studio将全面取代Eclipse。接下来，本部分将针对Android Studio开发工具的环境搭建进行讲解。



>>> 1.2.1 Android Studio 安装步骤

下载Android Studio

步骤1

Android Studio安装包可以[从中文社区进行下载](#)。这里我们以Windows 64系统为例，[下载ANDROID STUDIO 3.2.0版本](#)。Android Studio下载页面如下图所示。

步骤2

步骤3



ANDROID STUDIO 3.2.0 DOWNLOAD FROM DL.GOOGLE.COM

立即开始使用 Android Studio

Android Studio 包含用于构建 Android 应用所需的所有工具。

下载 ANDROID STUDIO
3.2 FOR WINDOWS 64-BIT (923 MB)

* 版本 : 3.2.0
* 发布日期 : SEPTEMBER 2018

选择其他平台 **1**

Platform	Android Studio package	Size	SHA-256 checksum
Windows (64-bit)	android-studio-ide-181.5014246-windows.exe	923 MB	978673a7babf51a9dca67729213c178995c1039a496dc1dfccb4c095b842c753
	Recommended		
	android-studio-ide-181.5014246-windows.zip	1000 MB	0a301ae52a41f0a55d10e78f8390b931abd6eb31af932a520738438d0d6bcfe9
Windows (32-bit)	No .exe installer		
	android-studio-ide-181.5014246-windows32.zip	999 MB	e28a862663ae90cb9e329ce69cf431306b7933af10ed06e428a8c31900ed2ef0
Mac	android-studio-ide-181.5014246-mac.dmg	988 MB	a3499a64970bf97d95a3bb27ebe571a56cee77510fa8a6d4745d6fbc24d252e1
Linux	android-studio-ide-181.5014246-linux.zip	1006 MB	e671d48cad66589860c510871167309b88c3f1f5e22a691cba053764c11a2a6c

请参阅 [Android Studio 发行说明](#)。

>>> 1.2.1 Android Studio 安装步骤

安装Android Studio

步骤1

成功下载Android Studio安装包后，双击后缀名为.exe的文件，[进入Welcome to Android Studio Setup](#)页面，如下图所示。

步骤2

步骤3



>>> 1.2.1 Android Studio 安装步骤

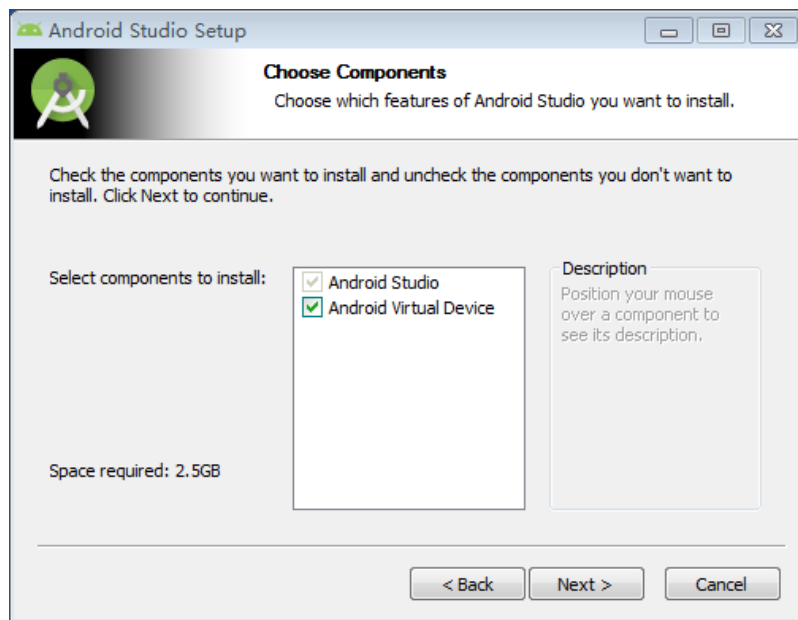
安装Android Studio

步骤1

步骤2

步骤3

单击上一页图中的“Next”按钮，进入Choose Components页面，如下图所示。



>>> 1.2.1 Android Studio 安装步骤

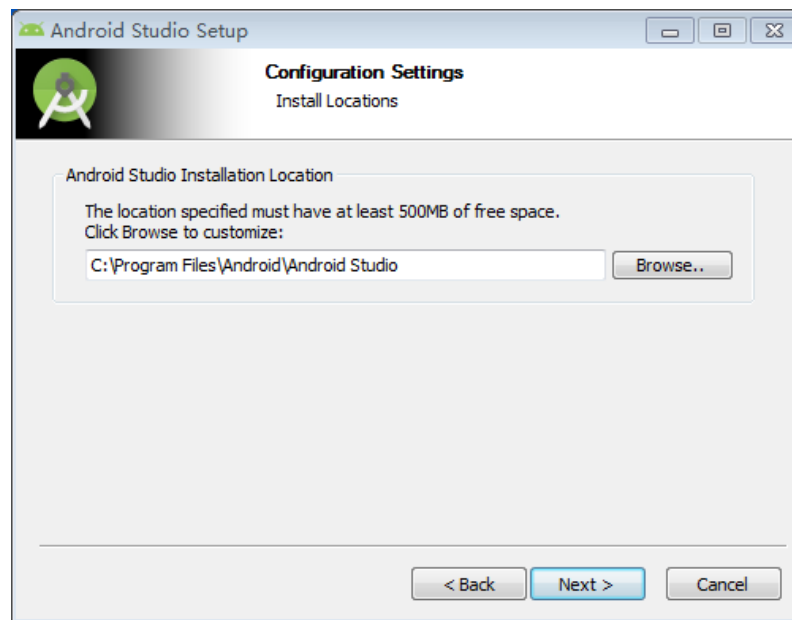
安装Android Studio

步骤1

步骤2

步骤3

单击上一页图中的“Next”按钮，进入Configuration Settings页面，如下图所示。



>>> 1.2.1 Android Studio 安装步骤

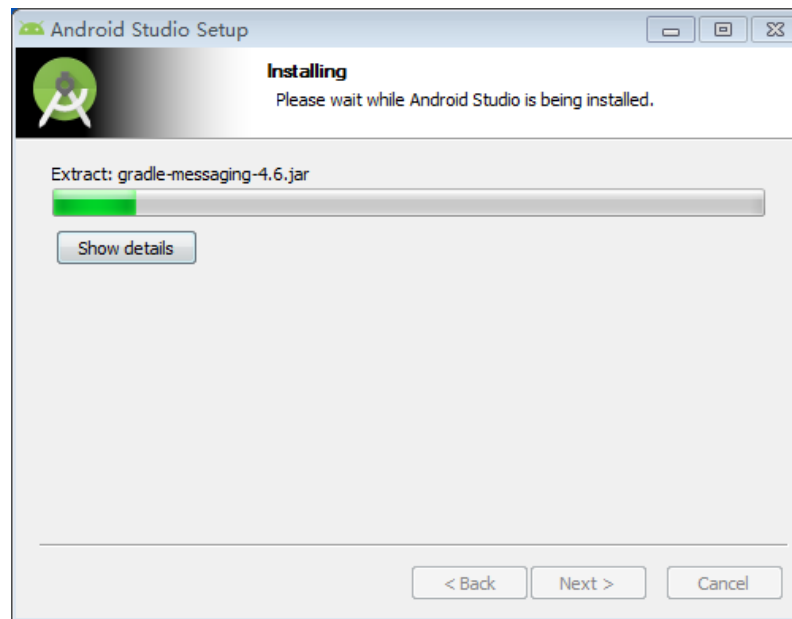
安装Android Studio

步骤1

步骤2

步骤3

单击上一页图中的 “Install” 按钮进入Installing页面开始安装，如下图所示。



>>> 1.2.1 Android Studio 安装步骤

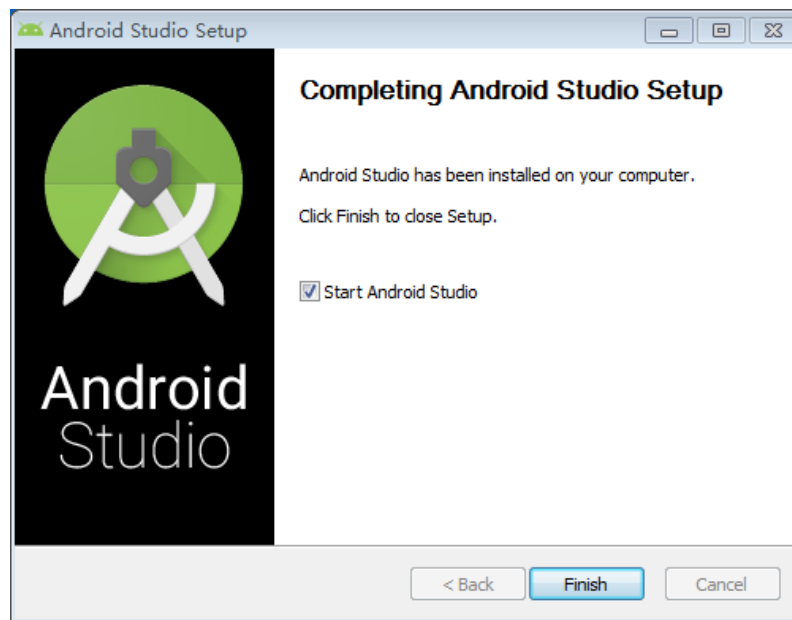
安装Android Studio

步骤1

安装完成后，单击上一页图中的“Next”按钮进入Completing Android Studio Setup 页面，如下图所示。

步骤2

步骤3



单击图中的“Finish”按钮，至此，Android Studio的安装全部完成。

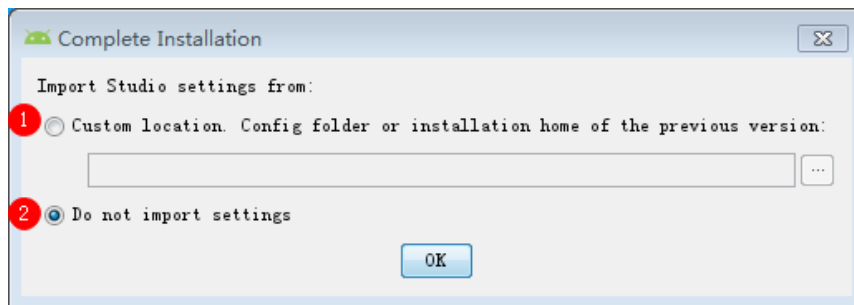
>>> 1.2.1 Android Studio 安装步骤

配置Android Studio

步骤1

如果我们在上一页图的页面中勾选了Start Android Studio选项，安装完成之后Android Studio会自动启动，会弹出一个Complete Installation对话框（选择导入Android Studio配置文件位置的窗口），如下图所示。

步骤2



步骤3

图中包含2个选项，其中选项①表示自定义Android Studio配置文件的位置，选项②表示不导入配置文件的位置。如果之前安装过Android Studio，想要导入之前的配置文件，则可以选择选项①，否则，选择选项②，此处可以根据实际情况进行选择。

>>> 1.2.1 Android Studio 安装步骤

配置Android Studio

我们选择选项上一页图中的②之后会进入Android Studio的开启窗口，如下图所示。

步骤1

步骤2

步骤3



androidstudio

Powered by the IntelliJ® Platform

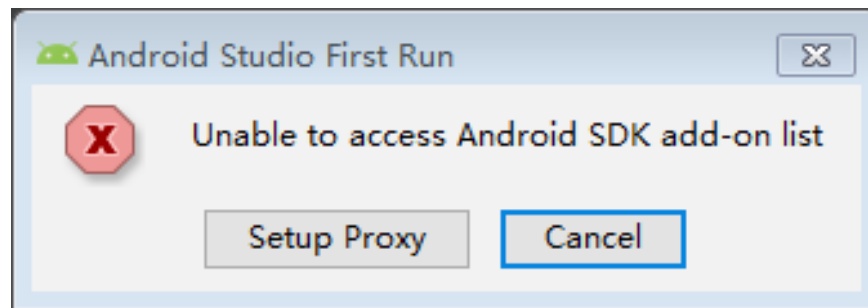
>>> 1.2.1 Android Studio 安装步骤

配置Android Studio

步骤1

上一页图中的进度完成之后，会弹出Android Studio First Run对话框，如下图所示。

步骤2



步骤3

>>> 1.2.1 Android Studio 安装步骤

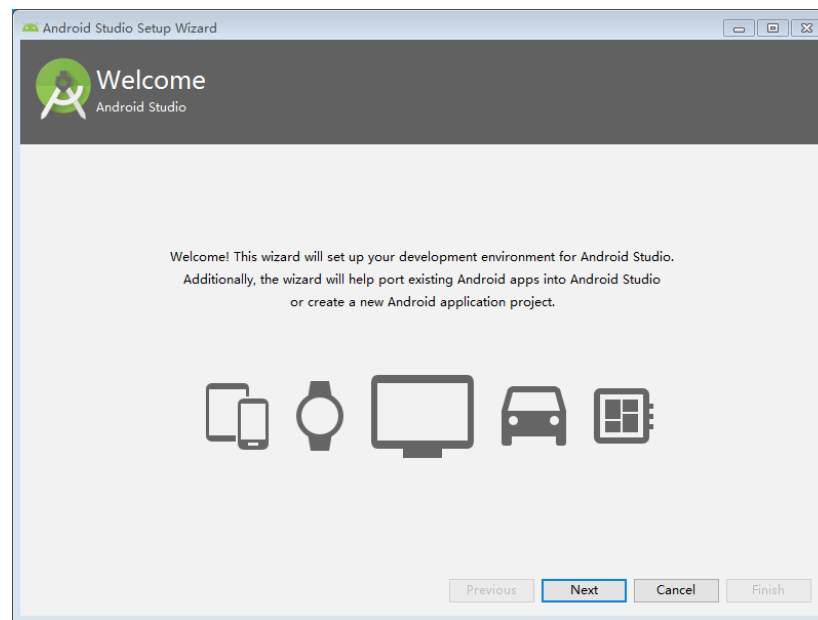
配置Android Studio

步骤1

步骤2

步骤3

单击上一页图中的“Cancel”按钮之后进入Welcome Android Studio页面，如下图所示。



>>> 1.2.1 Android Studio 安装步骤

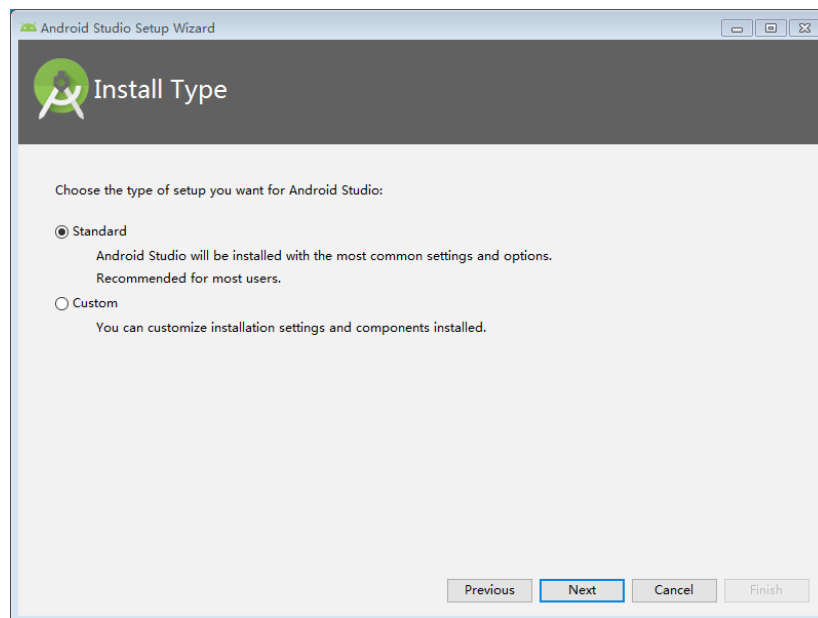
配置Android Studio

步骤1

步骤2

步骤3

单击上一页图中的“Next”按钮进入Install Type页面，如下图所示。



>>> 1.2.1 Android Studio 安装步骤

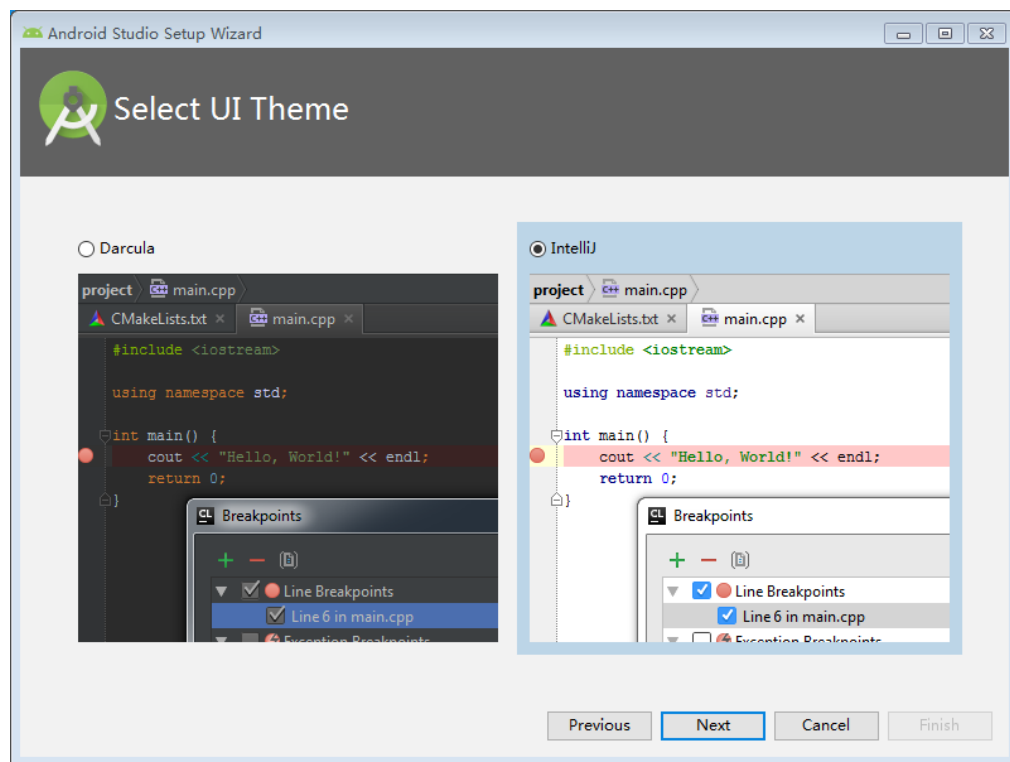
配置Android Studio

步骤1

单击上一页图中的“Next”按钮进入 [Select UI Theme \(选择UI主题\)](#) 页面，如下图所示。

步骤2

步骤3



>>> 1.2.1 Android Studio 安装步骤

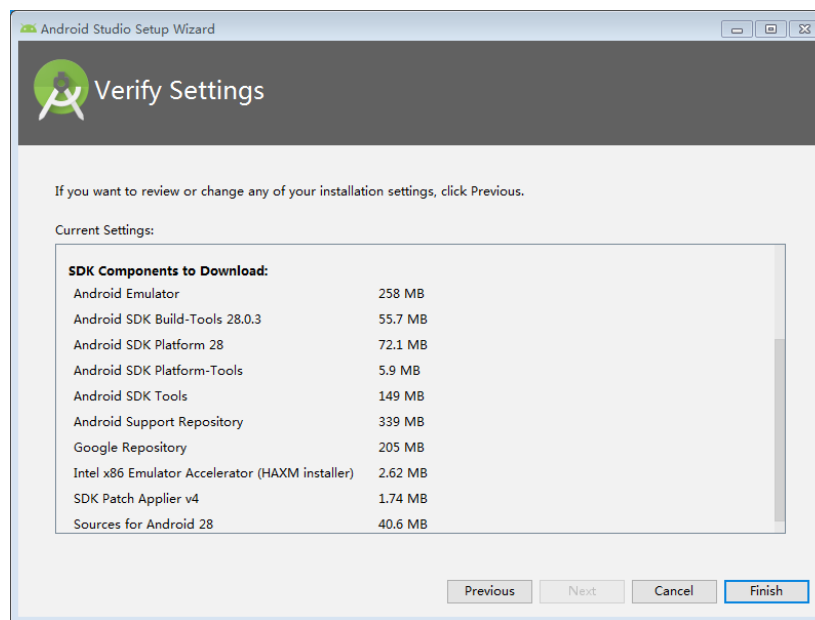
配置Android Studio

单击上一页图中的“Next”按钮进入Verify Settings页面，如下图所示。

步骤1

步骤2

步骤3



>>> 1.2.1 Android Studio 安装步骤

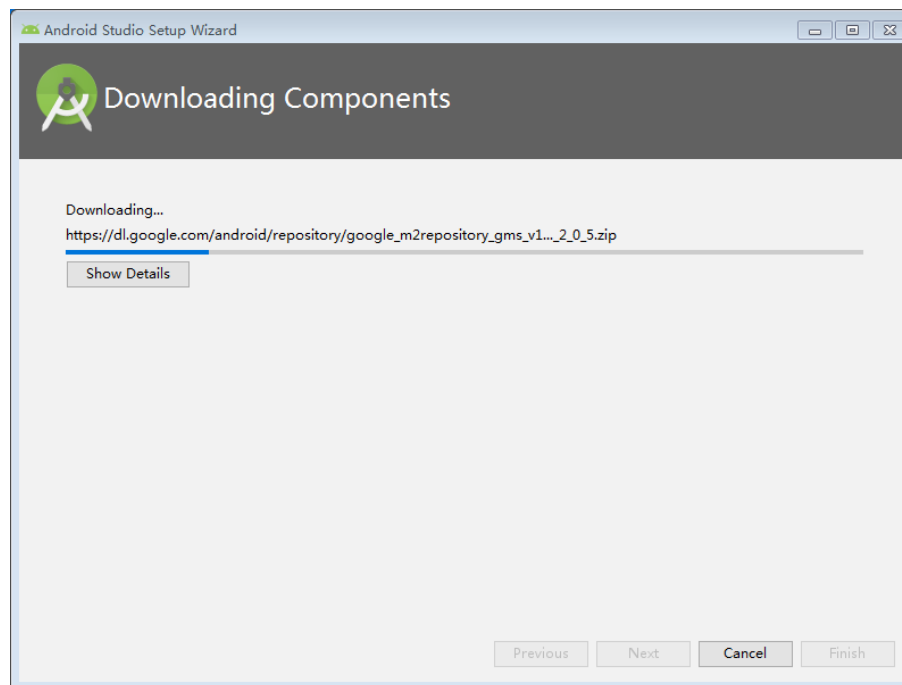
配置Android Studio

步骤1

步骤2

步骤3

单击上一页图中的“Finish”按钮进入[Downloading Components](#)页面，如下图所示。



>>> 1.2.1 Android Studio 安装步骤

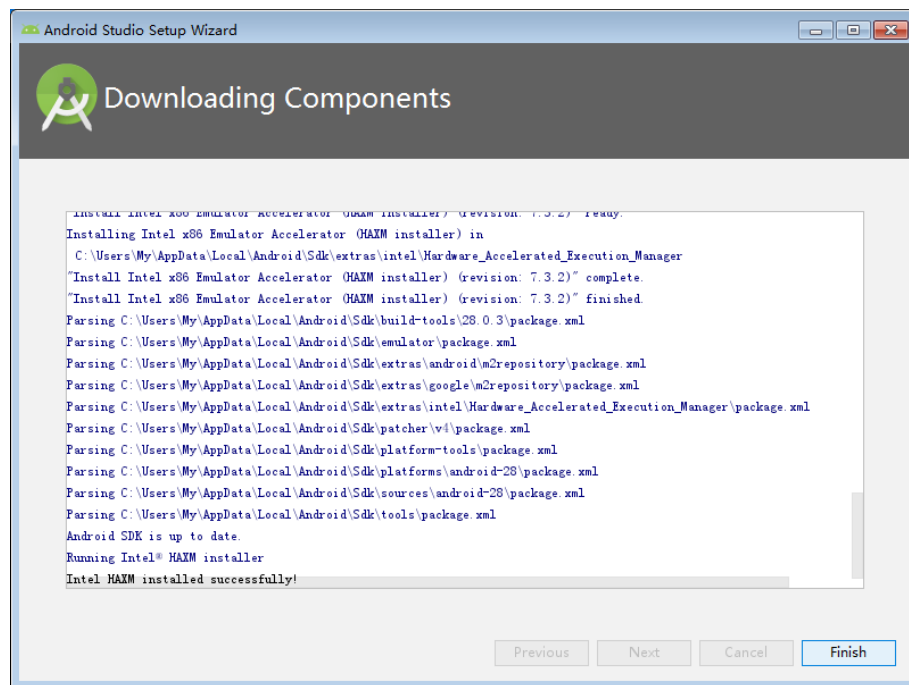
配置Android Studio

步骤1

步骤2

步骤3

下载完成后，会显示Downloading Components（下载完成）页面，如下图所示。



>>> 1.2.1 Android Studio 安装步骤

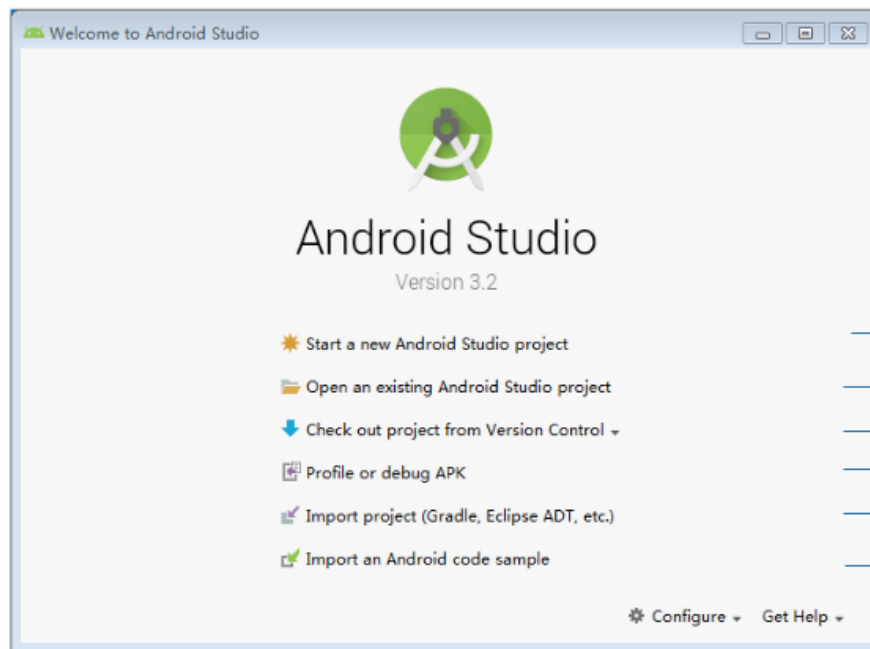
配置Android Studio

单击上一页图中的“Finish”按钮，进入Welcome to Android Studio窗口，如下图所示。

步骤1

步骤2

步骤3



创建AndroidStudio项目

打开已经存在的AndroidStudio项目

从版本控制系统中导入项目

配置和调试APK

导入非Android Studio项目

导入官方样例

至此，Android Studio工具的配置已经完成。

>>> 1.2.2 模拟器创建

Android程序可以运行到手机和平板等物理设备上，当运行Android程序时，没有手机或平板等物理设备，可以使用Android系统提供的模拟器。模拟器是一个可以运行在电脑上的虚拟设备。在模拟器上可预览和测试Android应用程序。



>>> 1.2.2 模拟器创建

单击ADV Manager标签

步骤1

当创建完第一个Android程序时，在Android Studio中，单击导航栏中的 图标会进入Your Virtual Devices页面，如下图所示。

步骤2

步骤3



>>> 1.2.2 模拟器创建

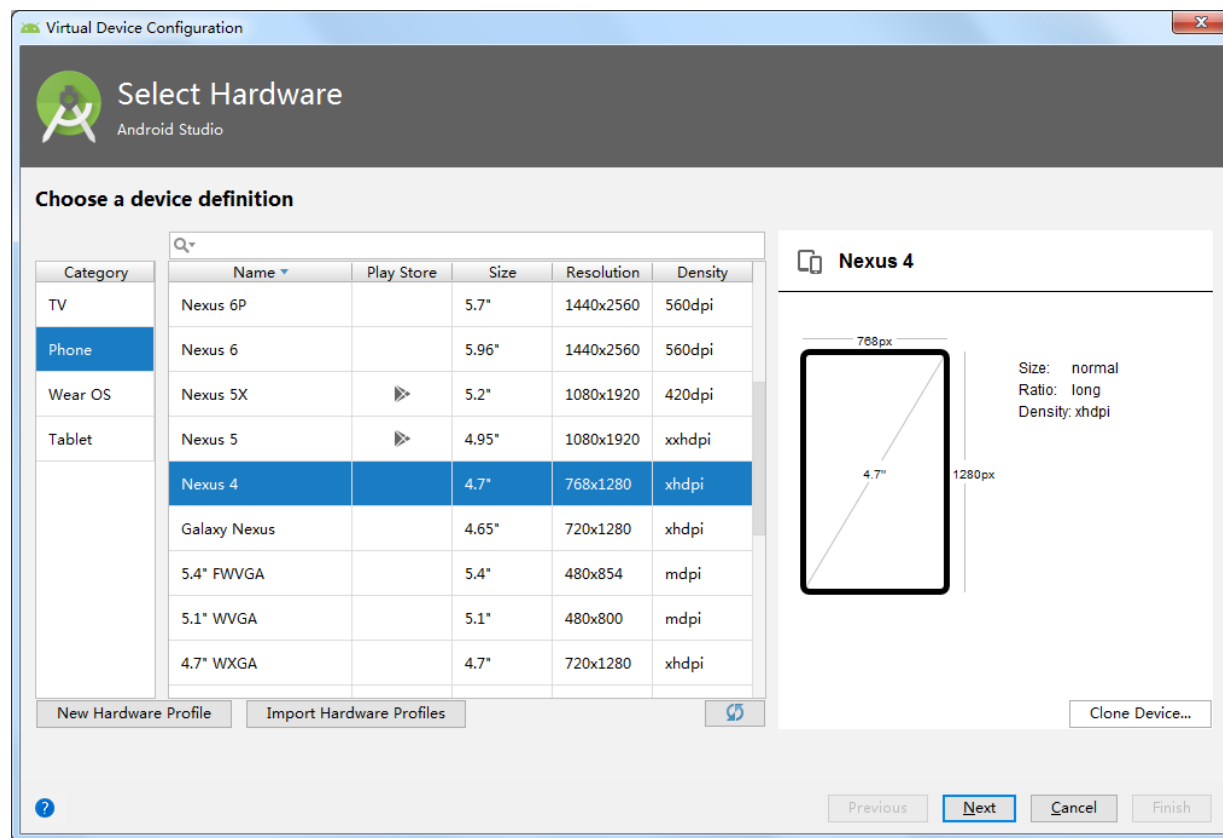
选择模拟设备

步骤1

单击上一页图中的 “+ Create Virtual Device...” 按钮，此时会进入选择模拟设备的Select Hardware页面，如下图所示。

步骤2

步骤3



>>> 1.2.2 模拟器创建

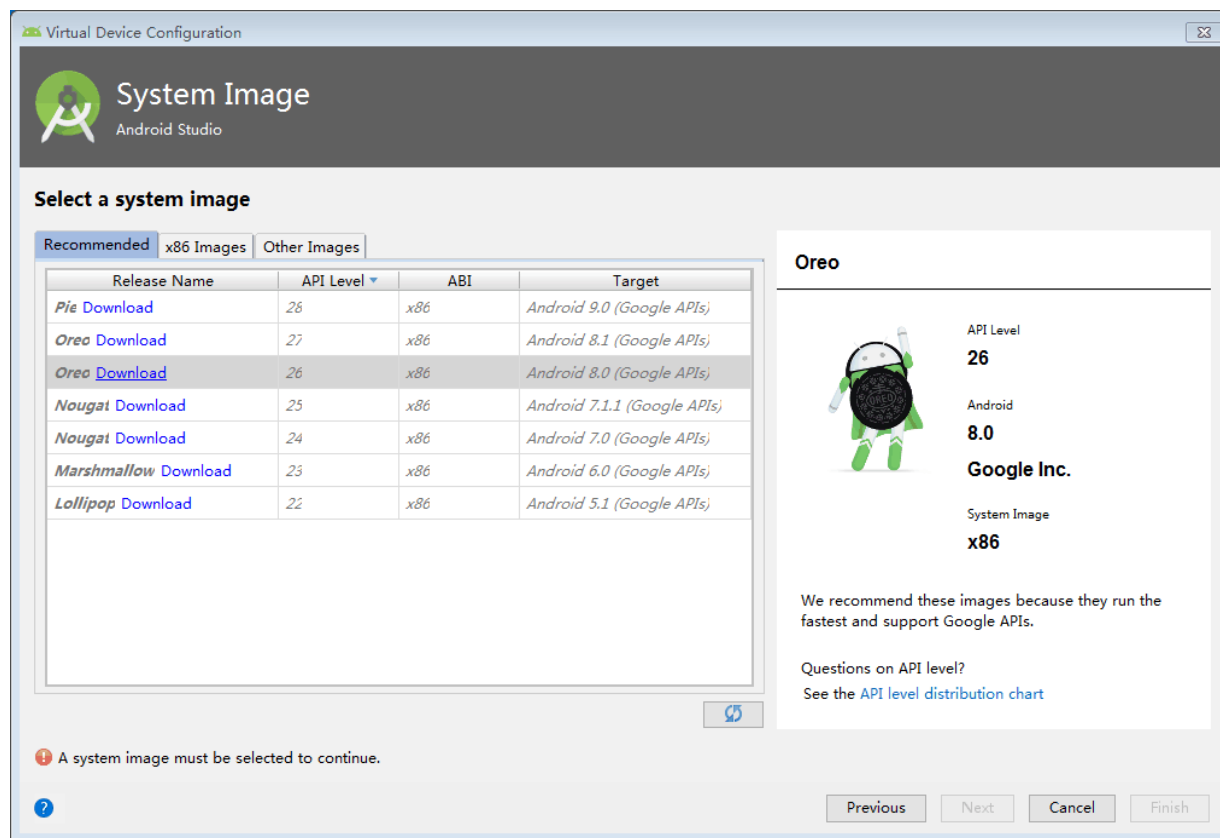
下载SDK System Image

步骤1

我们选择上一页图中的【Phone】→【Nexus 4】选项（此选项可根据自己需求选择不同屏幕分辨率的模拟器），单击“Next”按钮进入 [System Image](#) 页面，如下图所示。

步骤2

步骤3



>>> 1.2.2 模拟器创建

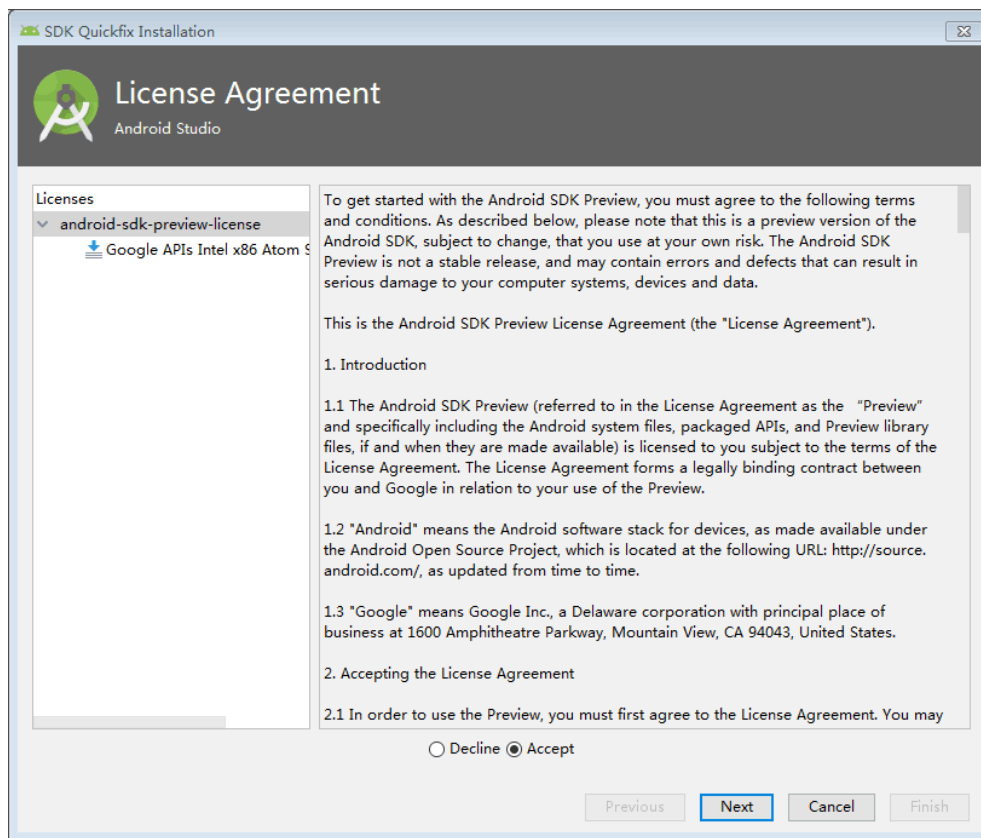
下载SDK System Image

步骤1

选中上一页图中的Oreo系统版本，单击“Download”进入License Agreement 页面，如下图所示。

步骤2

步骤3



>>> 1.2.2 模拟器创建

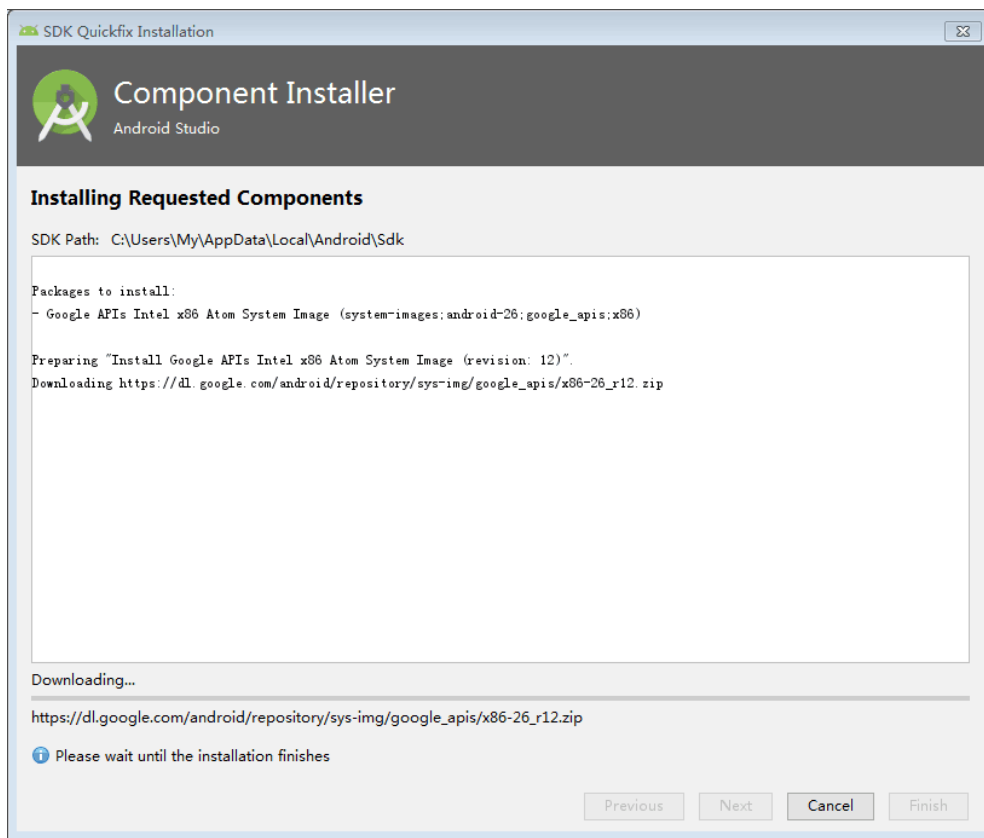
下载SDK System Image

步骤1

选中上一页图中的“Accept”按钮接受页面中显示的信息，并单击“Next”按钮进入Component Installer页面，如下图所示。

步骤2

步骤3



>>> 1.2.2 模拟器创建

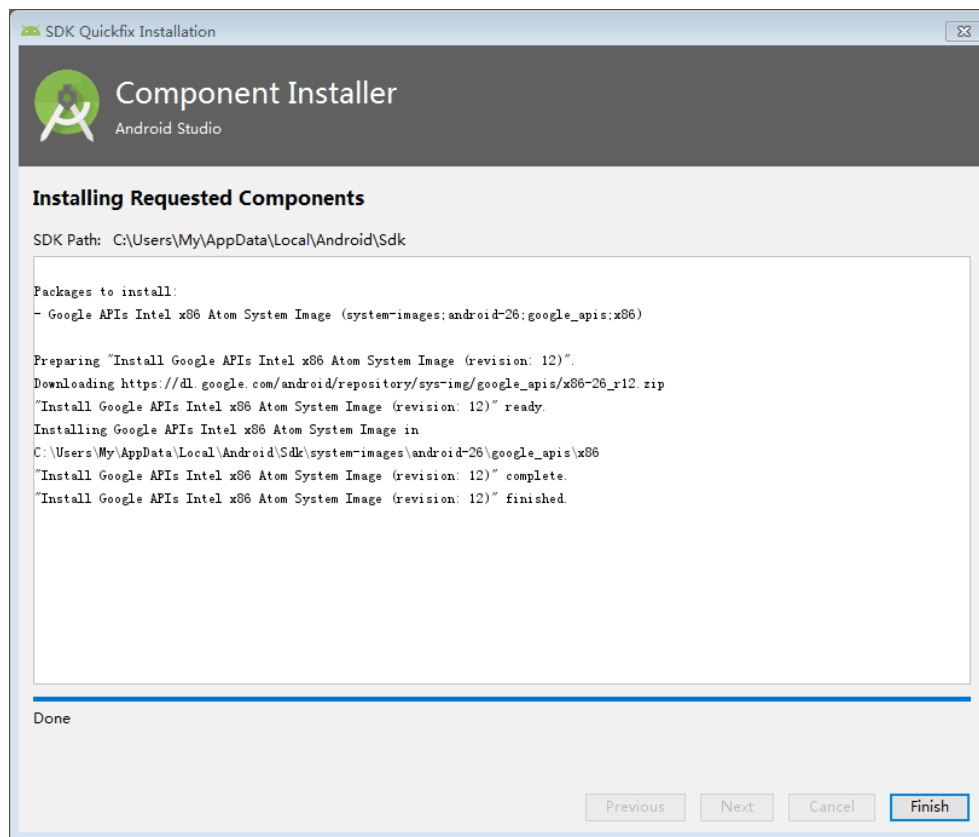
下载SDK System Image

下载完成后的Component Installer的页面，如下图所示。

步骤1

步骤2

步骤3



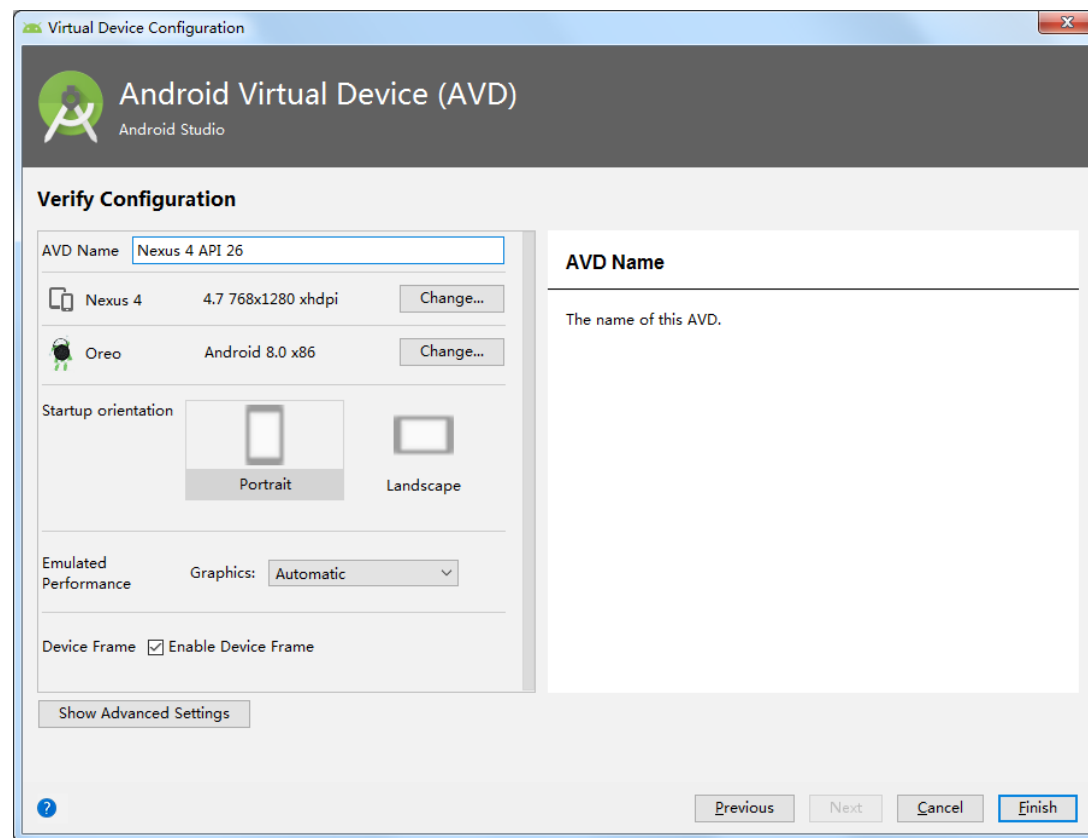
>>> 1.2.2 模拟器创建

创建模拟设备

此时选中System Image页面中系统版本名称为Oreo的条目，单击“Next”按钮进入[Android Virtual Device \(AVD\) 页面](#)，如下图所示。

步骤4

步骤5



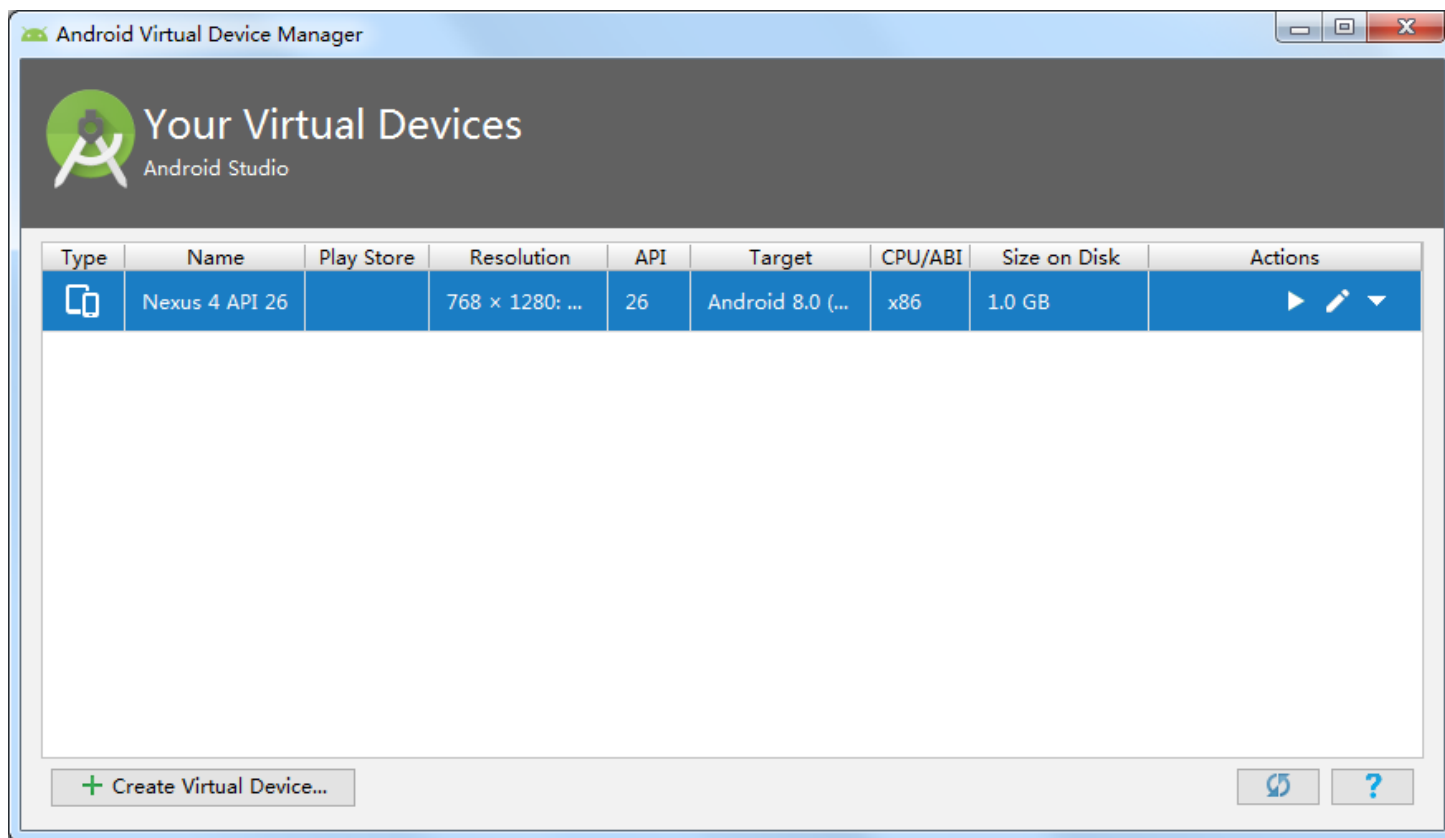
>>> 1.2.2 模拟器创建

创建模拟设备

单击上一页图中的“Finish”按钮，完成模拟器的创建。此时在Your Virtual Devices页面中会显示创建完成的模拟器，如下图所示。

步骤4

步骤5



>>> 1.2.2 模拟器创建

打开模拟设备

单击上一页图中的“启动”按钮 启动模拟器，启动完成后的Android模拟器界面，如下图所示。

步骤4

步骤5



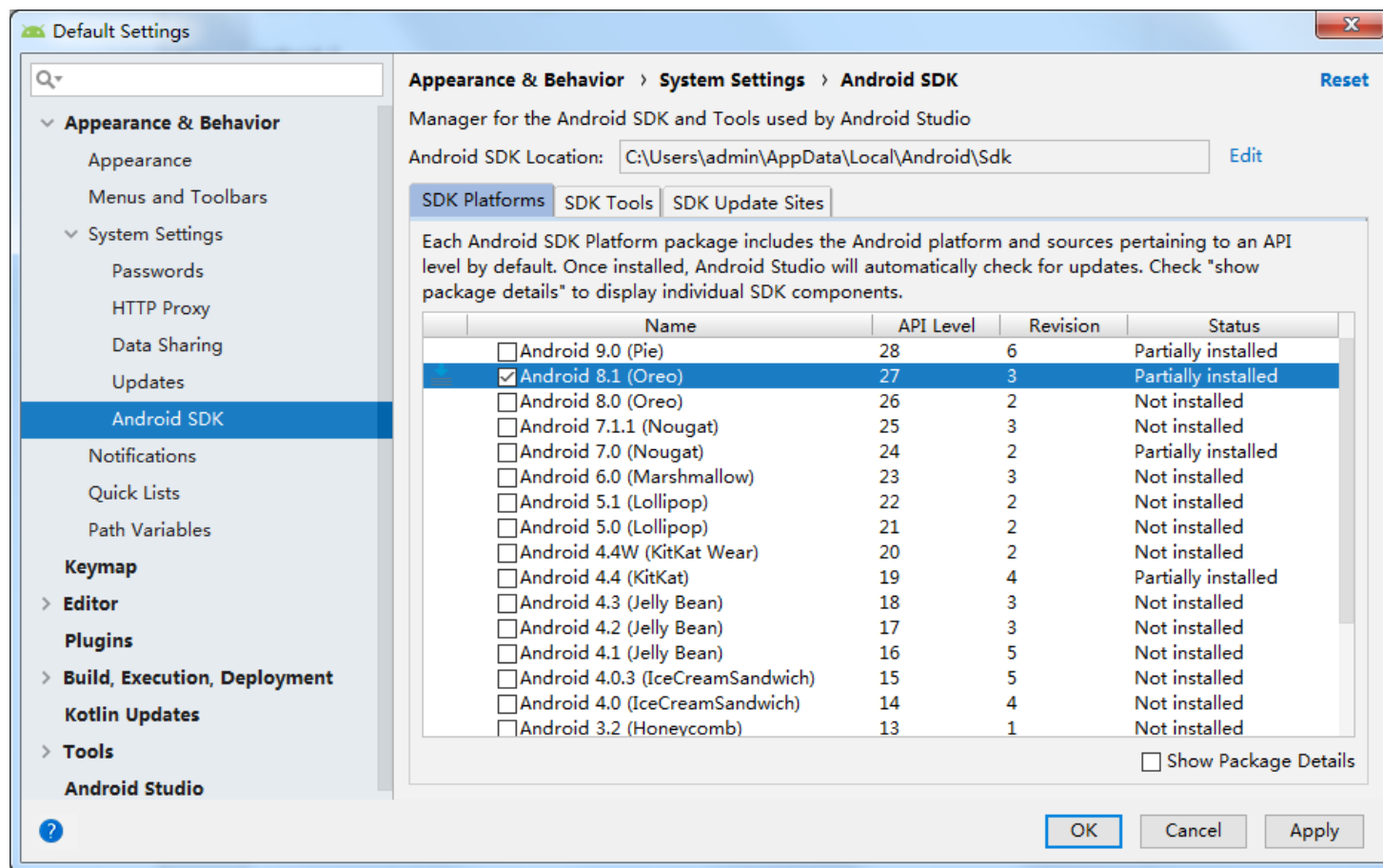
1.2.3 在Android Studio中下载SDK

下载SDK版本

打开Android Studio, 单击导航栏中的  图标, 进入Default Settings窗口, 如下图所示。

步骤1

步骤2



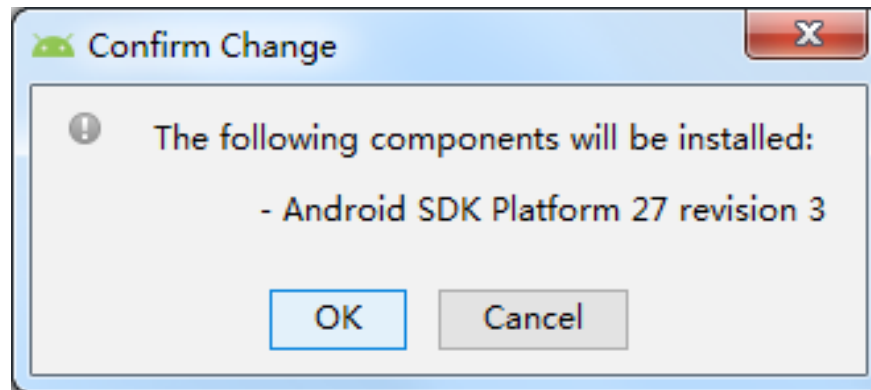
>>> 1.2.3 在Android Studio中下载SDK

下载SDK版本

在SDK Platforms选项卡下选择Android 8.1 (Oreo)条目，单击图1-31中的“OK”按钮会弹出确认安装SDK组件的Confirm Change窗口，如下图所示。

步骤1

步骤2



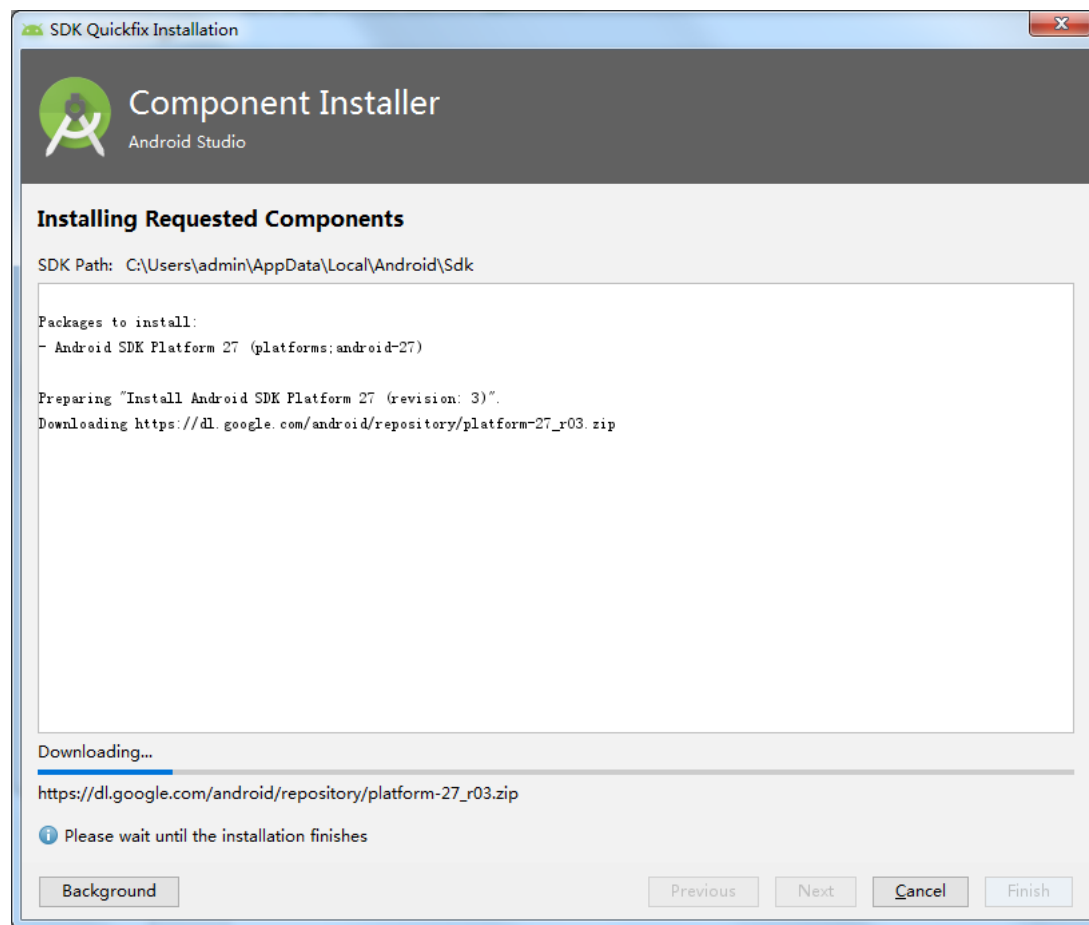
>>> 1.2.3 在Android Studio中下载SDK

下载SDK版本

单击上一页图中的“OK”按钮，进入Component Installer下载页面，如下图所示。

步骤1

步骤2



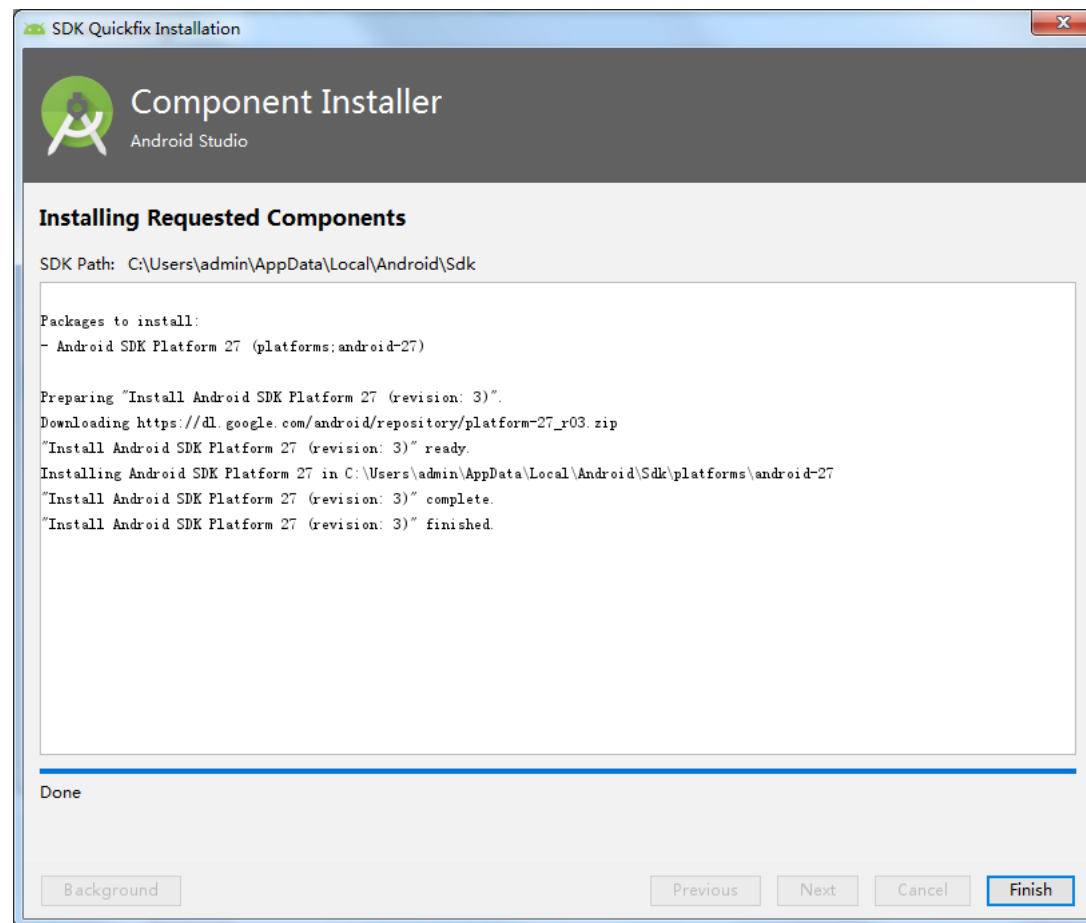
>>> 1.2.3 在Android Studio中下载SDK

下载SDK版本

下载完成后的Component Installer页面，如下图所示。

步骤1

步骤2



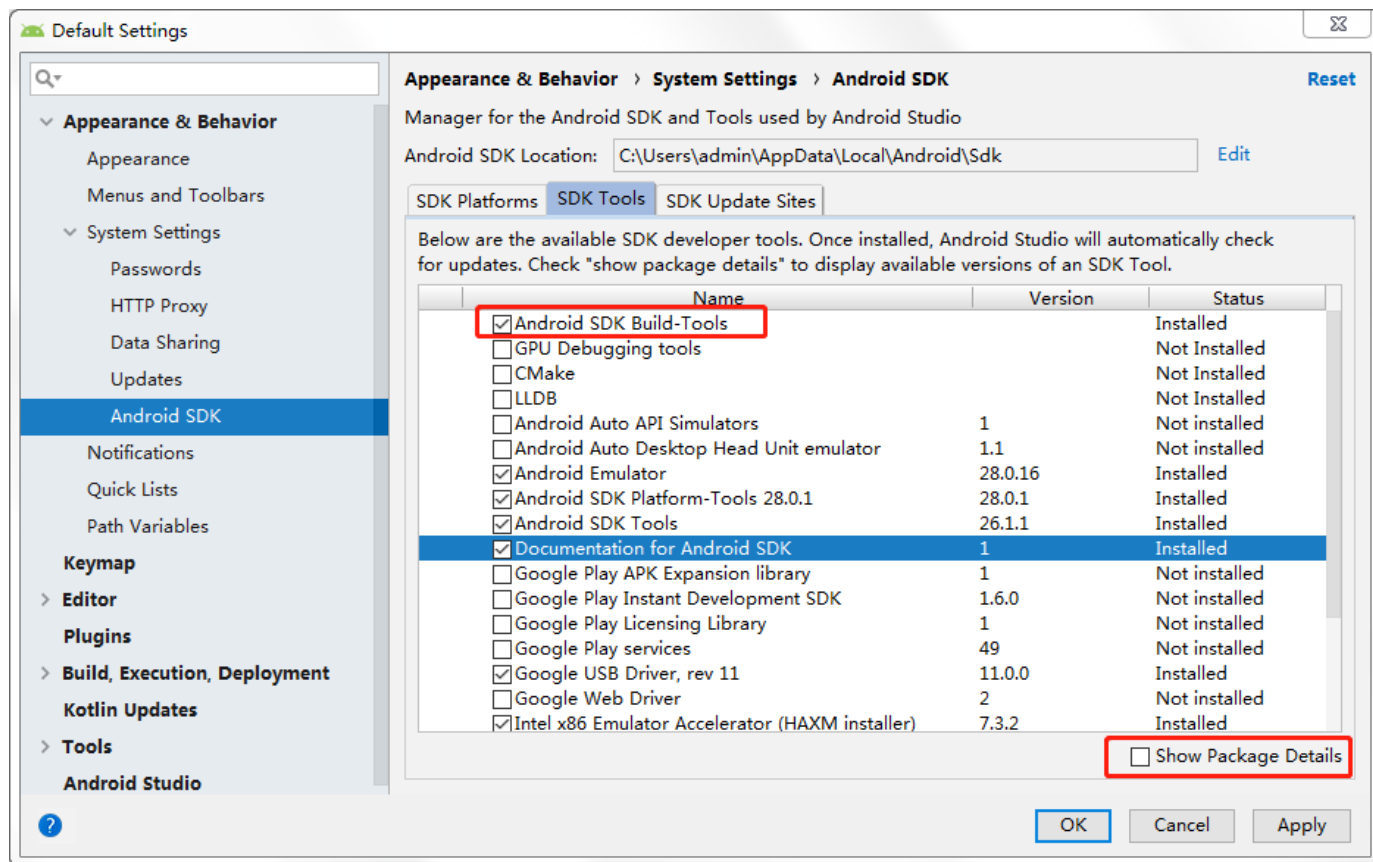
1.2.3 在Android Studio中下载SDK

下载Tools工具

在Default Settings窗口中的SDK Tools选项卡下，勾选Android SDK Build-Tools选项，如下图所示。

步骤1

步骤2



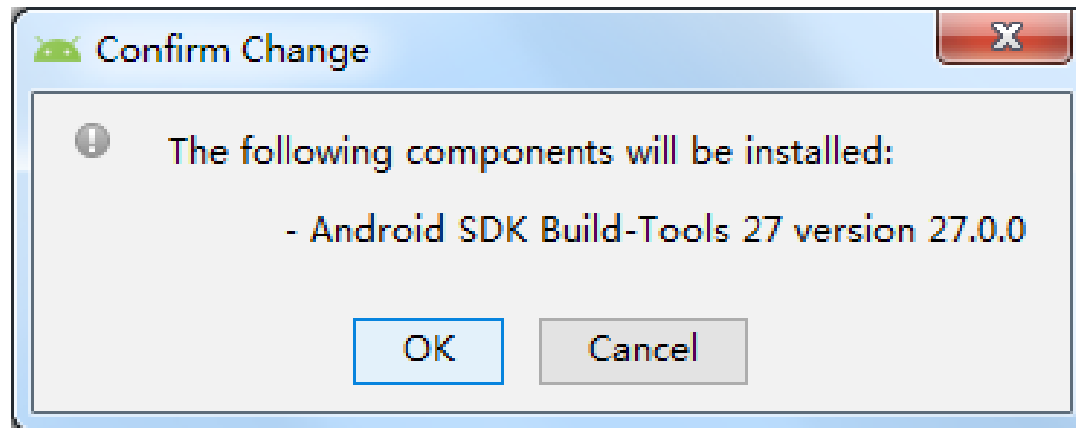
>>> 1.2.3 在Android Studio中下载SDK

下载Tools工具

步骤1

接着勾选Default Settings窗口右下角的Show Package Details选项，会打开Android SDK Build-Tools中的SDK版本列表信息，在列表中勾选27.0.0条目，单击“OK”按钮会弹出Confirm Change窗口，如下图所示。

步骤2



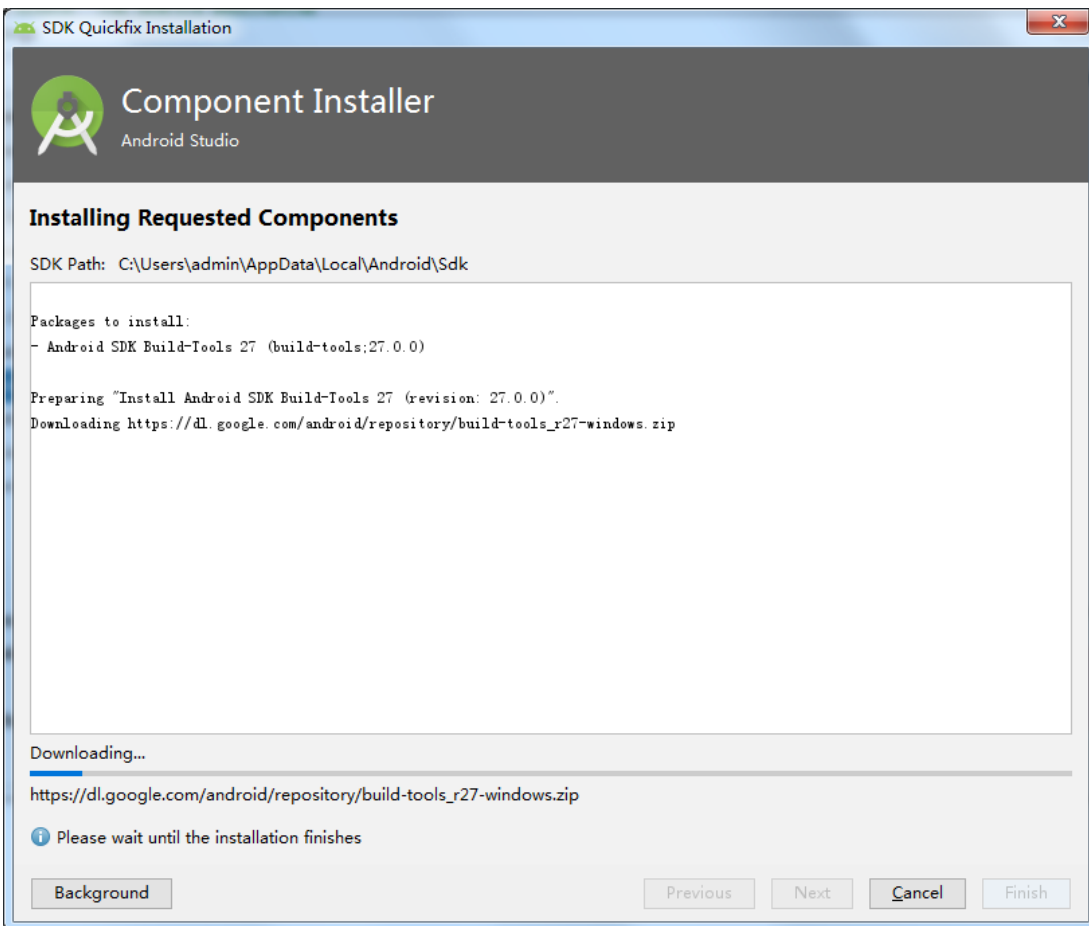
>>> 1.2.3 在Android Studio中下载SDK

下载Tools工具

单击上一页图中的“OK”按钮进入Component Installer下载页面，如下图所示。

步骤1

步骤2



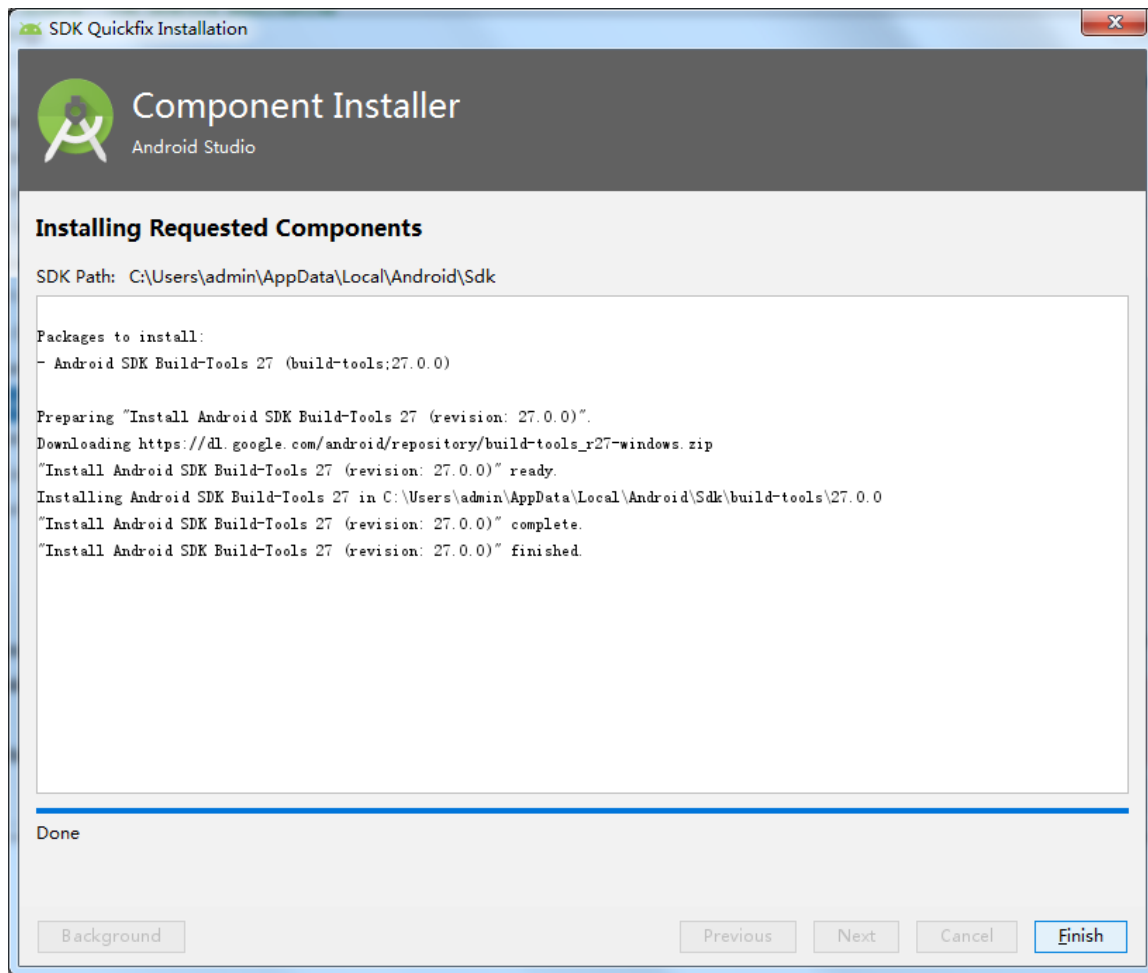
>>> 1.2.3 在Android Studio中下载SDK

下载Tools工具

一段时间之后，SDK下载完成，Component Installer下载完成页面的显示如下图所示。

步骤1

步骤2





1.3

开发第一个Android程序

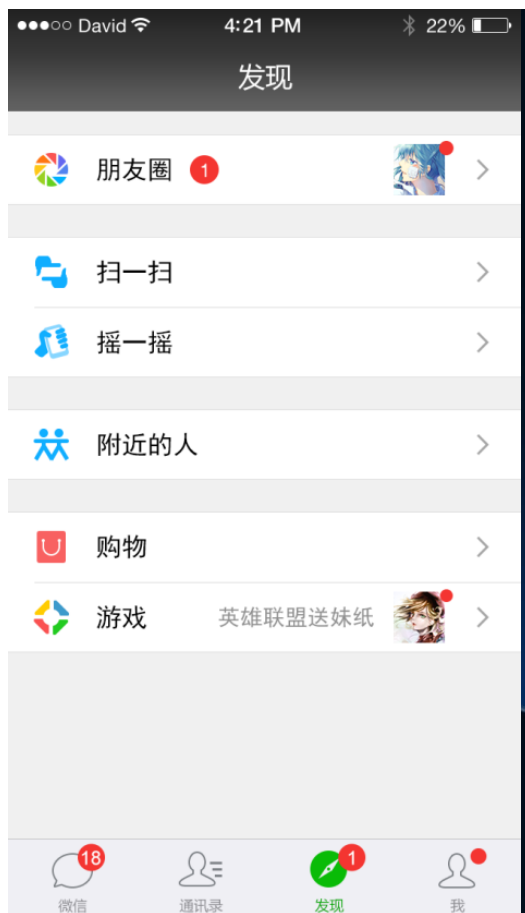
>>> 1.3 开发第一个Android程序



- 掌握编写简单Android程序的步骤，能够编写一个Hello World程序

1.3 开发第一个Android程序

学习Android可以开发出精美的APP，比如我们常见的QQ、微信和淘宝APP等。



>>> 1.3 开发第一个Android程序

前面小节中已经完成了Android开发环境的搭建，接下来使用Android Studio工具开发第一个Android程序，具体步骤如下：

1

创建程序：

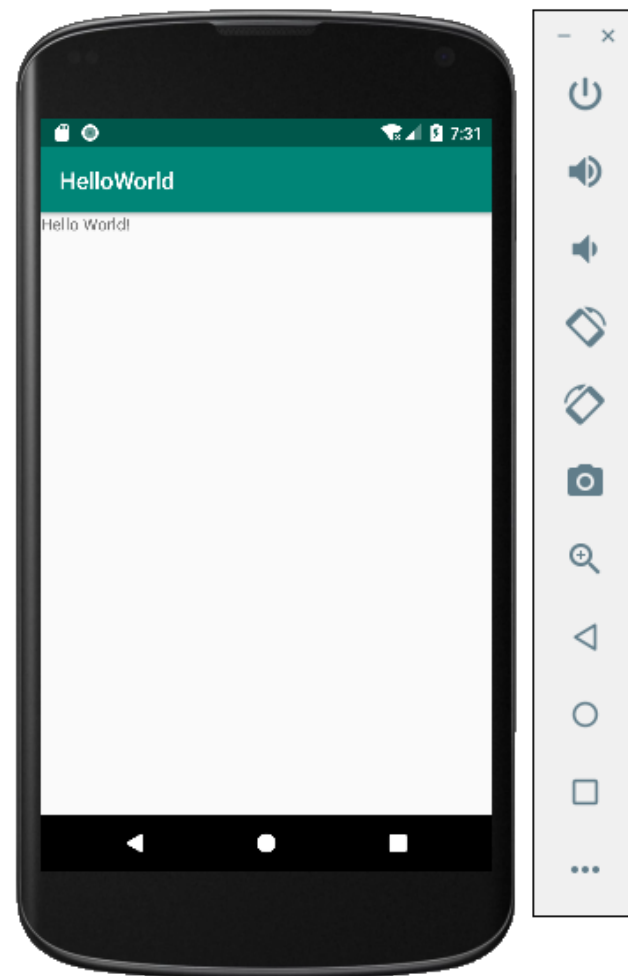
- ① 创建名为Hello World的程序
- ② 指定包名为cn.hznu.helloworld

2

启动模拟器： 点击工具栏中【AVD Manager】标签启动模拟器

3

运行程序： 点击工具栏中的运行按钮运行程序





1.4

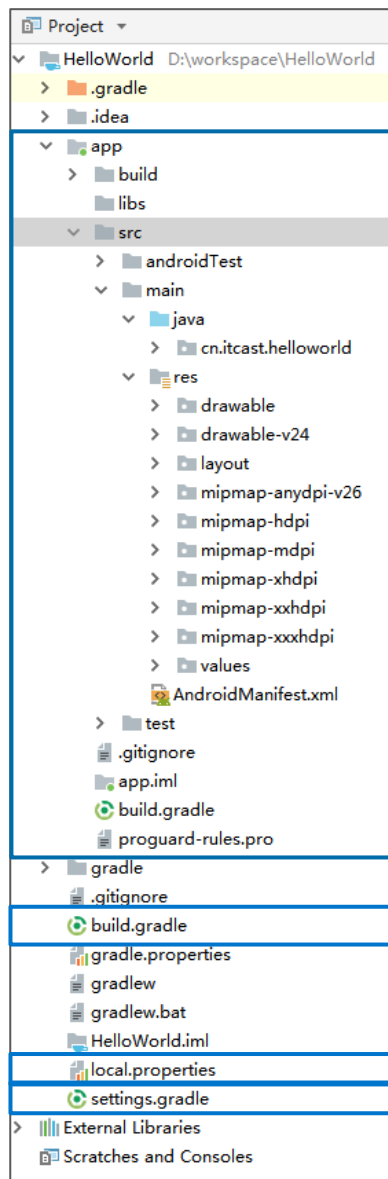
Android程序结构

>>> 1.4 Android程序结构



- 熟悉Android程序结构，能够归纳Android程序中常用的文件和文件夹的作用

>>> 1.4 Android程序结构



存放程序的代码和资源等文件

程序的gradle构建脚本

指定项目中所使用的SDK路径

配置在Android中使用的子项目(Moudle)



1.5

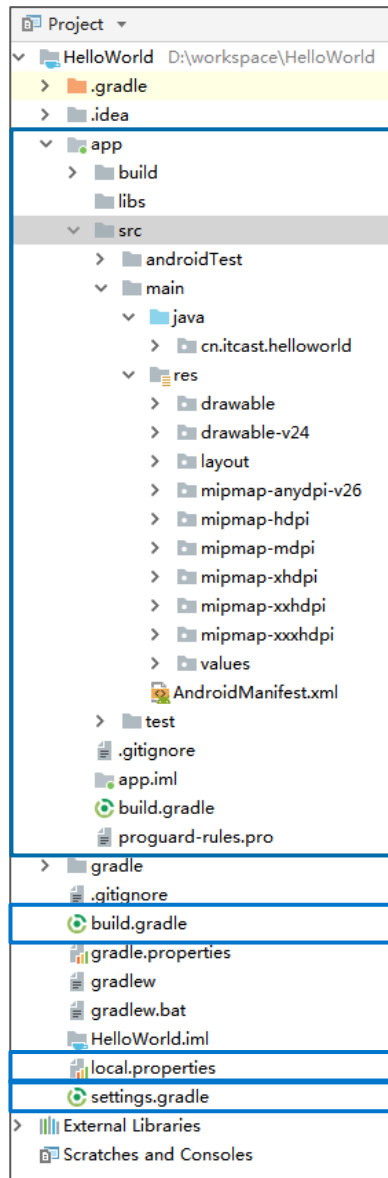
资源的管理与使用

>>> 1.5 资源的管理与使用



- 掌握资源的管理与使用方式，能够灵活使用程序中的资源

>>> 1.5 资源的管理与使用



存放程序的代码和资源等文件

程序的gradle构建脚本

指定项目中所使用的SDK路径

配置在Android中使用的子项目(Moudle)

1.5.1 图片资源

图片资源：扩展名为.png、.jpg、.gif、.9.png等的文件。

图片资源分类

- 应用图标资源：存放在mipmap文件夹中
- 界面中使用的图片资源：存放在drawable文件夹中

屏幕密度匹配规则

密度范围值	mipmap文件夹	drawable文件夹
120~160dpi	mipmap_mdpi	mipmap_mdpi
160~240dpi	mipmap_hdpi	drawable_hdpi
240~320dpi	mipmap_xdpi	drawable_xdpi
320~480dpi	mipmap_xxdpi	drawable_xxdpi
480~640dpi	mipmap_xxxdpi	drawable_xxxdpi

1.5.1 图片资源

调用图片资源的方式有两种，具体如下：

(1) 通过Java代码调用图片资源

```
//调用mipmap文件夹中资源文件  
getResources().getDrawable(R.mipmap.ic_launcher);  
  
//调用以drawable开头的文件夹中的资源文件  
getResources().getDrawable(R.drawable.icon);
```

(2) 在XML布局文件中调用图片资源

```
@mipmap/ic_launcher //调用mipmap文件夹中的资源文件  
@drawable/icon       //调用以drawable开头的文件夹中的资源文件
```

1.5.2 主题和样式资源

主题：包含一种或多种格式化属性的集合，在程序中调用主题资源可改变窗体的样式。

主题资源定义位置：在res/values目录下的styles.xml文件中

定义主题资源的标签：

`<style> </style>`：定义主题的标签

`<item> </item>`：设置主题样式的标签

```
<style name="AppTheme" parent="Theme.AppCompat.Light.DarkActionBar">
```

```
<item name="colorPrimary"> @color/colorPrimary </item>
```

用于指定主题名称

用于指定继承的父主题

```
<item name="colorPrimaryDark"> @color/colorPrimaryDark </item>
```

```
<item name="colorAccent"> @color/colorAccent </item>
```

定义状态栏的颜色

```
</style>
```

各个控件被选中的颜色

1.5.2 主题和样式资源

想要调用styles.xml文件中定义的主题，可以在AndroidManifest.xml文件中设置，也可以在代码中设置。

(1) 在AndroidManifest.xml文件中设置主题

```
android:theme="@style/AppTheme"
```

(2) 在Java代码中设置主题

```
setTheme(R.style.AppTheme);
```

1.5.2 主题和样式资源

样式：设置View的宽度、高度和背景颜色等信息。

样式资源定义位置：res/values目录下的styles.xml文件中

样式的标签：

`<style> </style>`：定义样式的标签

`<item> </item>`：设置控件样式的标签

在XML布局文件中引用样式

```
style="@style/textViewStyle"
```


1.5.3 布局资源

布局资源：通常用于搭建程序中的各个界面。

布局资源存放位置：res/layout文件夹中

调用布局资源的方式有2种：

(1) 通过Java代码调用布局资源文件

```
//在Activity的onCreate()方法中调用activity_main.xml布局文件  
setContentView(R.layout.activity_main);
```

(2) 在XML布局文件中调用布局资源文件

```
//在XML布局文件中调用activity_main.xml布局文件  
<include layout="@layout/activity_main"/>
```

1.5.4 字符串资源

字符串：用于显示界面上的文本信息。

字符串资源定义位置：res/values目录下的strings.xml文件中

字符串标签：

```
<string> </string>：定义字符串的标签
```

```
<resources>
```

```
    <string name="app_name">字符串</string>
```

```
</resources>
```

1.5.4 字符串资源

调用字符串资源的方式有2种：

(1) 通过Java代码调用字符串资源

```
getResources().getString(R.string.app_name);
```

(2) 在XML布局文件中调用字符串资源

```
@string/app_name
```

>>> 1.5.5 颜色资源

颜色：用于显示控件的不同色彩效果。

颜色资源定义位置：res/values/colors.xml文件中

颜色标签：

```
<color> </color>：定义颜色的标签
```

```
<resources>
```

```
    <color name="colorPrimary">#3F51B5</color>
```

```
</resources>
```

1.5.5 颜色资源

调用颜色资源的方式有2种：

(1) 通过Java代码调用颜色资源

```
getResources().getColor(R.color.colorPrimary);
```

(2) 在XML布局文件中调用颜色资源

```
@color/colorPrimary
```



定义颜色值



在Android中，颜色值是由RGB（红、绿、蓝）三原色和一个透明度（Alpha）表示，颜色值必须以“#”开头，“#”后面显示Alpha-Red-Green-Blue形式的内容。其中，Alpha值可以省略，如果省略，表示颜色默认是完全不透明的。一般情况下，使用以下4种形式定义颜色

#RGB

#ARGB

#RRGGBB

#AARRGGBB

1.5.6 尺寸资源

尺寸：用于设置View的宽高和View之间的间距值。

尺寸资源定义位置：res/values/dimens.xml文件中，如果程序中没有dimens.xml文件，可自行创建。

尺寸的标签：

```
<dimen> </dimen>： 定义尺寸的标签
```

```
<resources>  
    <dimen name="activity_horizontal_margin">16dp</dimen>  
</resources>
```

1.5.6 尺寸资源

调用尺寸资源的方式有2种：

(1) 通过Java代码调用尺寸资源

```
getResources().getDimension(R.dimen.activity_horizontal_margin);
```

(2) 在XML布局文件中调用尺寸资源

```
@dimen/activity_horizontal_margin
```




Android支持的尺寸单位



尺寸单位:

- ✓ **px** (pixels, 像素): 每个px对应屏幕上的一个点。
- ✓ **dp** (Density-independent Pixels, 设备独立像素): 是一种与屏幕密度无关的尺寸单位。
- ✓ **sp** (scaled pixels, 比例像素): 主要处理字体的大小, 可以根据用户字体大小首选项进行缩放。
- ✓ **in** (inches, 英寸): 标准长度单位。
- ✓ **pt** (points, 磅): 屏幕物理长度单位, 1磅为1/72英寸。
- ✓ **mm** (Millimeters, 毫米): 屏幕物理长度单位。

1.6

程序调试



>>> 1.6 程序调试



- 掌握单元测试与Logcat的使用，能够完成对程序的调试

1.6.1 单元测试

单元测试是指在Android程序开发过程中对最小的功能模块进行测试，单元测试包括Android单元测试和JUnit单元测试。

Android单元测试

1. 该测试方式执行测试的时候需要连接Android设备。
2. 速度比较慢。
3. 适合需要调用Android API的单元测试。

JUnit单元测试

1. 该测试方式不需要依赖Android设备，在本地即可运行。
2. 速度快。
3. 适合只对Java代码功能进行的单元测试。

1.6.1 单元测试

Android Studio 3.2版本在创建项目时，会默认在app/src/androidTest和app/src/test文件夹中创建Android单元测试类ExampleInstrumentedTest和Junit单元测试类ExampleUnitTest。

(1) Android单元测试类ExampleInstrumentedTest

- 使用@RunWith(AndroidJUnit4.class)注解ExampleInstrumentedTest类
- @Test注解类中的方法

(2) Junit单元测试类ExampleUnitTest

- @Test注解类中的方法

1.6.1 单元测试

Android单元测试

ExampleInstrumentedTest.java类中的代码


```
package cn.itcast.helloworld;  
@RunWith(AndroidJUnit4.class)  
public class ExampleInstrumentedTest {  
    @Test  
    public void useAppContext() {  
        // Context of the app under test.  
        Context appContext = InstrumentationRegistry.getTargetContext();  
        assertEquals("cn.itcast.helloworld", appContext.getPackageName());  
    }  
}
```

注解类

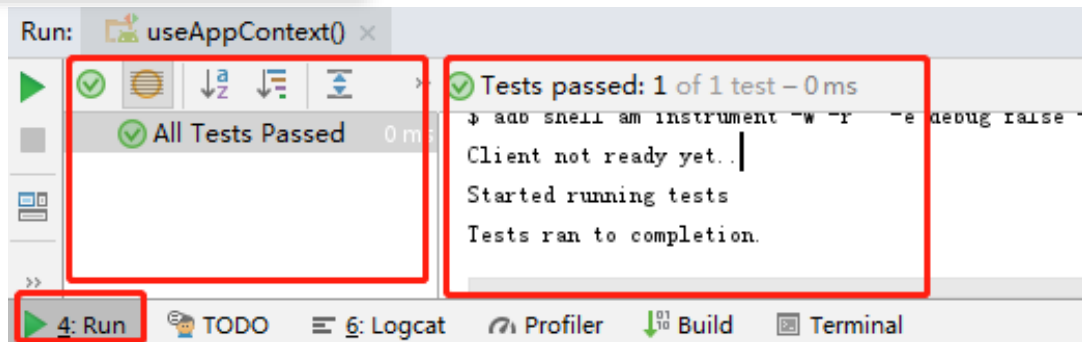
注解方法

断言，期望两个参数值相等

1.6.1 单元测试

在上一页代码中的方法useAppContext()上右击，在弹出框中选择【Run useAppContext()】。将程序运行到模拟器后，在Android Studio底部导航栏中单击“ 4: Run”图标查看运行成功的结果。

测试结果正常



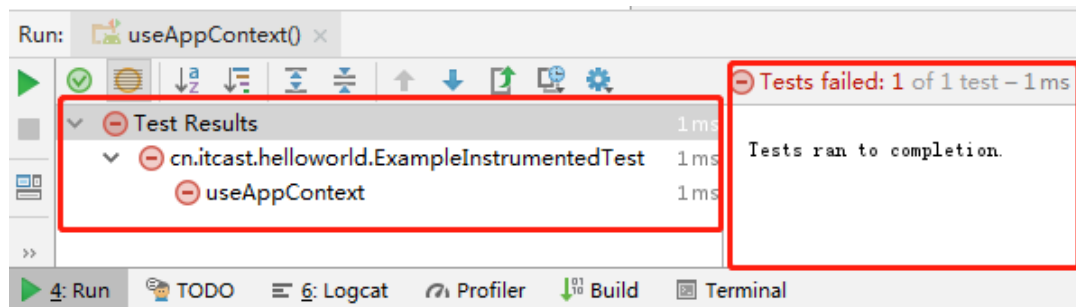
1.6.1 单元测试

接下来修改文件ExampleInstrumentedTest.java中assertEquals()方法的参数，使测试方法useAppContext()时，显示错误信息，修改的具体代码如下：

```
assertEquals("helloworld", appContext.getPackageName());
```

运行程序，运行失败的结果如下图所示。

测试结果错误



1.6.1 单元测试

Junit单元测试

ExampleUnitTest.java类中的代码

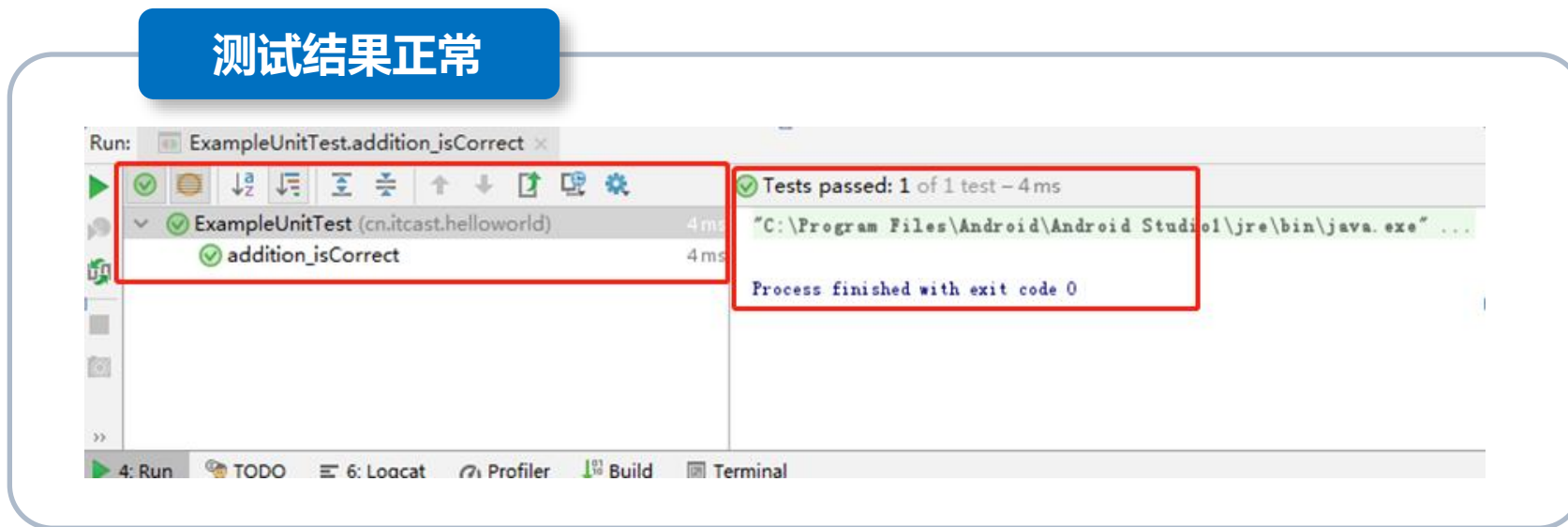
```
package cn.itcast.helloworld;  
public class ExampleUnitTest {  
    @Test  
    public void addition_isCorrect() {  
        assertEquals(4, 2 + 2);  
    }  
}
```

注解方法

断言，期望两个参数值相等

1.6.1 单元测试

在上一页代码中的方法`addition_isCorrect()`上右击，在弹出框中选择“Run `addition_isCorrect()`”选项。程序运行结束后，在Android Studio底部导航栏中单击“ ”图标查看运行成功的结果，如下图所示。



1.6.1 单元测试

接下来修改文件ExampleUnitTest.java中的assertEquals()方法中的参数，使测试addition_isCorrect()方法时，显示错误信息，修改的具体代码如下：

```
assertEquals(4, 1 + 2);
```

运行程序，运行失败的结果如下图所示。

测试结果错误



>>> 1.6.1 单元测试



注意

Android Studio 3.2版本在创建项目时，会自动在build.gradle文件中添加单元测试的支持库，如果在进行单元测试时，程序中的build.gradle文件中没有添加单元测试的支持库，则需要手动进行添加。

```
dependencies {  
    .....  
    testImplementation 'junit:junit:4.12'  
    androidTestImplementation 'com.android.support.test:runner:1.0.2'  
    androidTestImplementation  
        'com.android.support.test.espresso:espresso-core:3.0.2'  
}
```

1.6.2 Logcat的使用

LogCat是Android中的命令行工具，用于获取程序从启动到关闭的日志信息。

Log类所输出的日志内容分为六个级别。

级别	Log类中的静态方法
Verbose	Log.v()
Debug	Log.d()
Info	Log.i()
Warning	Log.w()
Error	Log.e()
Assert	Log.wtf()

>>> 1.6.2 Logcat的使用

打印信息的标签

Log.v("MainActivity", "Verbose");

需要打印的信息

Log.d("MainActivity", "Debug");

Log.i("MainActivity", "Info");

Log.w("MainActivity", "Warning");

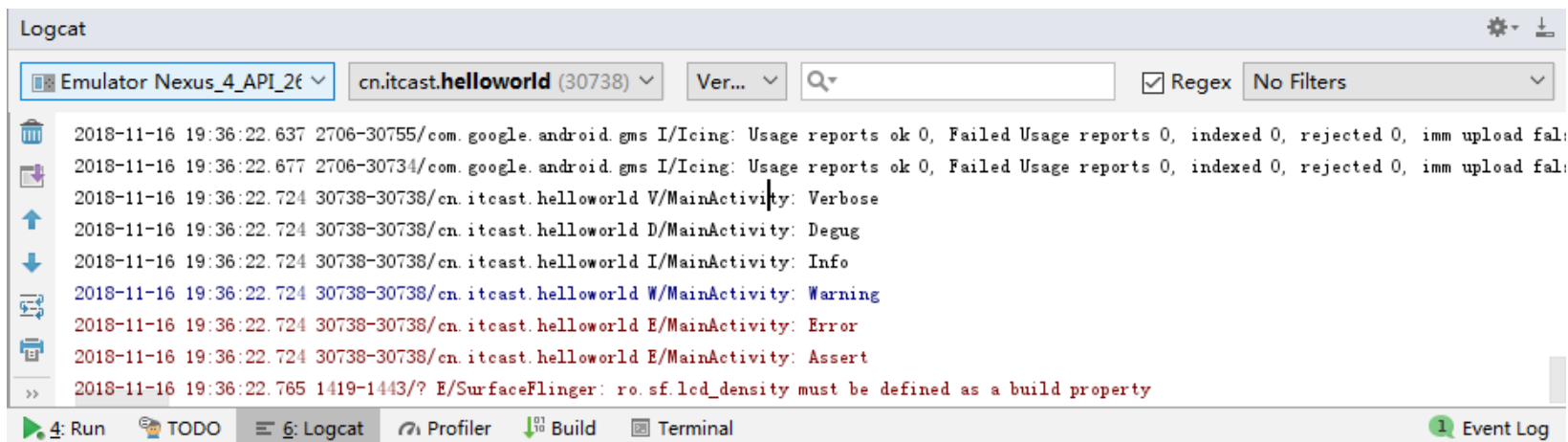
Log.e("MainActivity", "Error");

Log.wtf("MainActivity", "Assert");

>>> 1.6.2 Logcat的使用

运行上一页中的程序，此时Logcat窗口中打印的Log信息，如下图所示。

测试结果

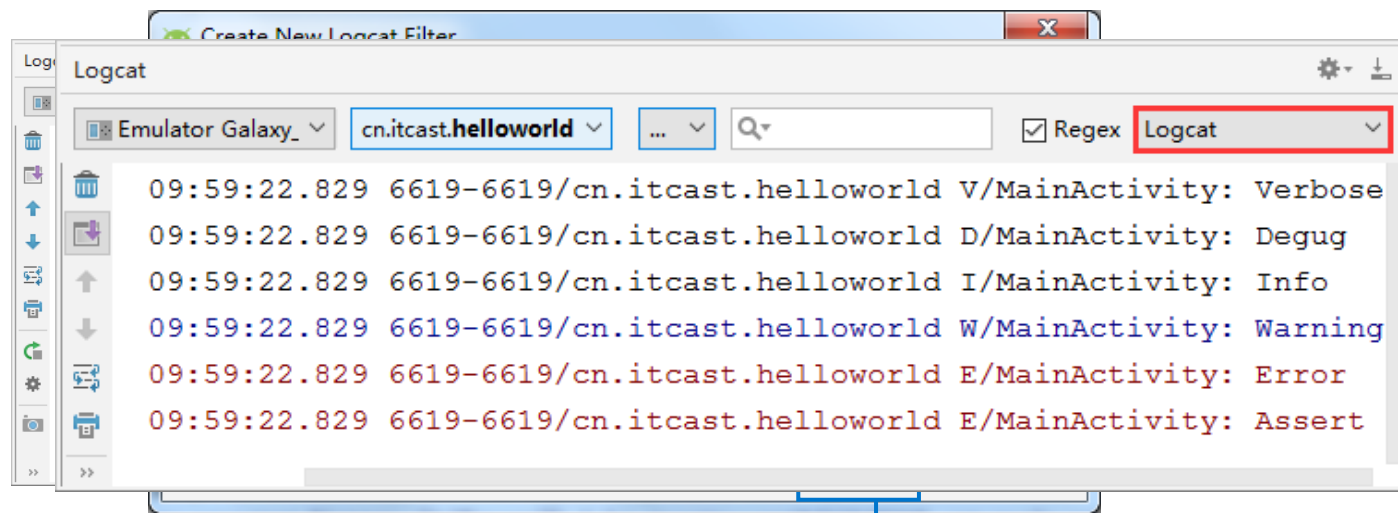


```
Logcat
Emulator Nexus_4_API_26  cn.itcast.helloworld (30738)  Ver...  Q  [x] Regex  No Filters

2018-11-16 19:36:22.637 2706-30755/com.google.android.gms I/Icing: Usage reports ok 0, Failed Usage reports 0, indexed 0, rejected 0, imm upload fal
2018-11-16 19:36:22.677 2706-30734/com.google.android.gms I/Icing: Usage reports ok 0, Failed Usage reports 0, indexed 0, rejected 0, imm upload fal
2018-11-16 19:36:22.724 30738-30738/cn.itcast.helloworld V/MainActivity: Verbose
2018-11-16 19:36:22.724 30738-30738/cn.itcast.helloworld D/MainActivity: Debug
2018-11-16 19:36:22.724 30738-30738/cn.itcast.helloworld I/MainActivity: Info
2018-11-16 19:36:22.724 30738-30738/cn.itcast.helloworld W/MainActivity: Warning
2018-11-16 19:36:22.724 30738-30738/cn.itcast.helloworld E/MainActivity: Error
2018-11-16 19:36:22.724 30738-30738/cn.itcast.helloworld E/MainActivity: Assert
2018-11-16 19:36:22.765 1419-1443/? E/SurfaceFlinger: ro.sf.lcd_density must be defined as a build property

Run  TODO  Logcat  Profiler  Build  Terminal  Event Log
```

>>> 1.6.2 Logcat的使用

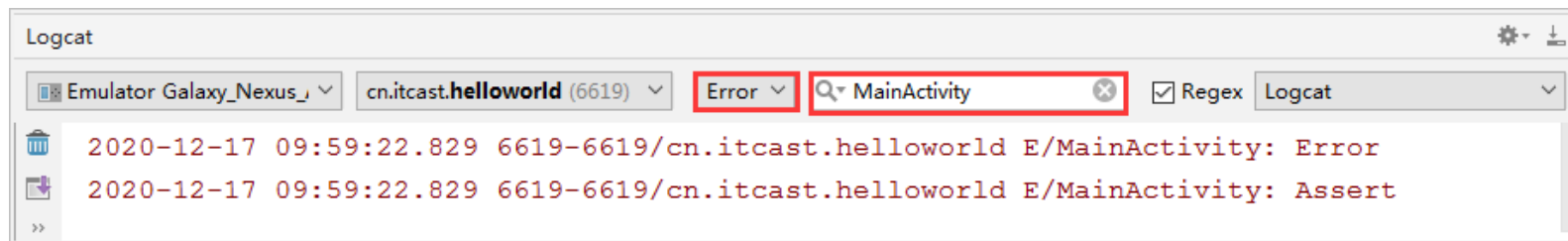


TAG过滤信息

点击创建完成

>>> 1.6.2 Logcat的使用

除了设置过滤器过滤所需的信息外，还可以输入TAG信息、根据Log级别等方式过滤信息。



LogCat区域中日志信息根据级别不同显示不同的颜色

级别	显示信息	日志信息颜色
verbose(V)	全部信息	黑色
debug(D)	调试信息	蓝色
info(I)	一般信息	绿色
warning(W)	警告信息	橙色
error(E)	错误信息	红色
assert	断言失败后的错误消息	红色

本 章 小 结

本章主要讲解了Android的基础知识，首先介绍了Android的发展历史以及体系结构，然后讲解Android开发环境的搭建，接着开发了一个HelloWorld程序，帮助大家了解Android项目的创建、程序的结构，以及资源文件的使用。最后介绍了程序调试，包括单元测试和Logcat的使用。通过本章的学习，希望读者能对Android有一个大致的了解，并会独立搭建Android开发环境，为后续学习Android知识做好铺垫。

