2018

计算机软件技术基础

Android 项目描述

程源

广东机电职业技术学院

2018/9/1

目录

1 快速入门

1.1 阅读指南

本讲首先介绍如何搭建 Android 的开发环境，接着将通过一个简单的应用——WebWorld 带领读者体验互联网世界，同时介绍编写 Android 应用需要掌握的一些新的概念、UI 组件和 Activity 的生命周期。学完本章，如果没能理解全部内容，也不必担心。后续章节我们将再次温习并理解这些概念。如果不了解 java 语言，也不熟悉面向对象编程，请找本 java 入门书籍看看。

两个学习网站：

http://www.android-studio.org/，AS 中文社区官网，可以在这里下载安装最新版的 AS，并且这里面有很多教程，可以参看。

http://www.androiddevtools.cn/index.html，AndroidDevTools（安卓开发工具）这里收集整理了许多实用的工具、开发教程。

1.2 搭建开发环境

准备开发前，你需要安装 Android Studio。Android Studio 是基于流行的 IntelliJ IDEA 创建的用于 Android 开发的一套集成开发工具。Google 发布 Android Studio 已经有一段时间了，最近推出了 3.1.2 版本。

Android Studio 的安装包括：

- Android SDK　最新版本的 Android SDK。
- Android SDK 工具和平台工具　用来测试与调试应用的一套工具。

- Android 模拟器系统镜像　用来在不同虚拟设备上开发测试应用。

Google 一直在积极开发和更新 Android Studio 版本。因此，请注意了解你当前在用版本和本讲义所用版本之间的差异。

1.2.1 配置 java 开发环境

首次安装的话，你还需要从 http://www.oracle.com 下载并安装 Java 开发者套件。

下载之后双击进入安装界面，只需根据提示点击“下一步”，并记下安装路径，以备后面将要使用。继续点击“下一步”，等待 jdk 安装。之后进入 jre 安装页面，继续“下一步”，直至安装完成，点击“关闭”即可。

注意 不要将 jre 的目录设置成与 jdk 相同的目录，否则会覆盖后者，导致 Android Studio 不能启动。

这样还不够，还需添加一个系统变量。在“我的电脑”右键“属性”→高级系统设置→环境变量→系统变量→新建，进入如下图所示的“新建系统变量”界面，于“变量名”填 **JAVA_HOME**，“变量值”填入前面记下的 jdk 安装路径（不是 jre 的路径），如图 1-1 所示。

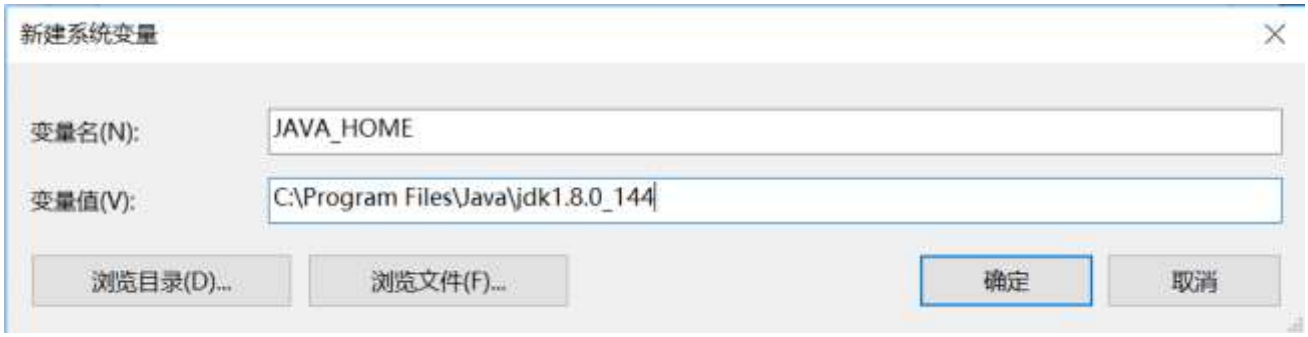

图 1-1 JAVA HOME 设置

以上就是 java 环境的安装和配置的全过程。

1.2.2 Android Studio 的下载与安装

从 http://www.android-studio.org/找到最新版本的 android studio（九月份是 3.2 版）之后，选择你所需要的版本，如不清楚，点击最上边一个下载即可。下载完成后解压，选择那个比较大的应用程序安装，之后一直点击 **Next**。看到是否同意界面时，点击 **I Agree**，

进入安装路径界面。你可以选择默认安装路径，也可以专门建立文件夹作为安装路径。之后直接点击 **install** 即进入安装界面，等待安装完成。进度条满之后，点击 **next**，然后点击 **finish**，然后便进入了最后一步，现在 AndroidStudio 已经在启动中了。

1.2.3 添加 SDK 包

Android Studio 自带最新版本的 SDK 和系统模拟器镜像。但若想在 Android 早期版本上测试应用，还需额外下载相关工具组件。可通过 Android SDK 管理器来配置安装这些组件。在 Android Studio 中，选择 **Tools→Android→SDK Manager** 菜单项，如图 1-2 所示。

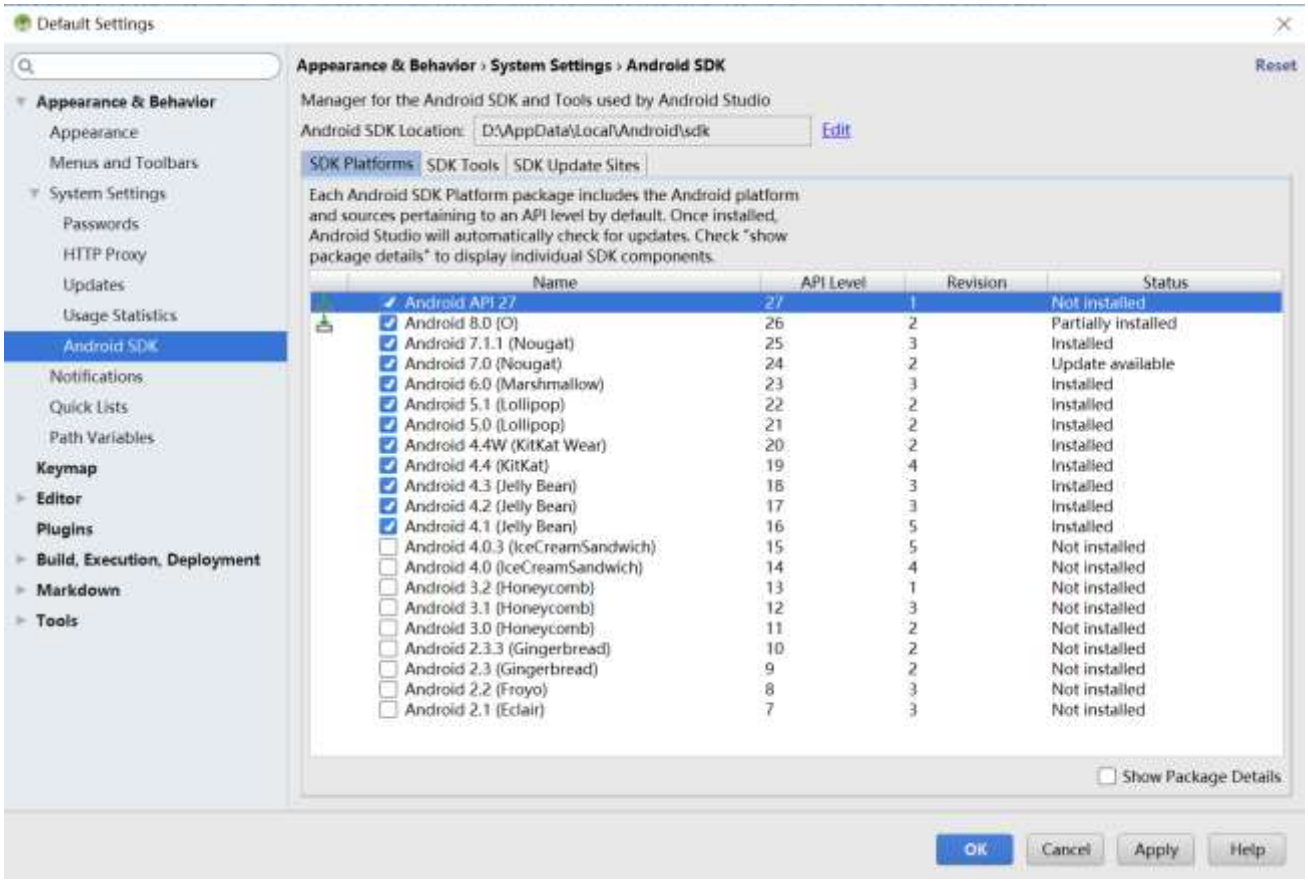

图 1-2 Andriod SDK 管理器

选择并安装需要的 Android 版本和工具。下载这些组件需要一定时间，请耐心等待。通过 Android SDK 管理器，还可以及时获取 Android 最新发布内容，比如新系统平台或新版本工具等。

1.3 体验互联网

对于现代人来说，智能手机之所以成为二十一世纪最重要的发明之一，就是因为它能随时随地上网。Android 系统更是为互联网而生，几行代码就可以带你进入互联网世界。如果你有过桌面系统那庞大、复杂，需要消耗大量内存的 Web 浏览器的开发经历，体验完本节后，你可能就不愿意再回去了。

我们即将要开发的这个名为 WebWorld 的应用将带你进入互联网世界。尽管 Android Studio 已经为你准备好了一切，你还是需要至少在三个文件中填上 8 行代码，包括提供的一个入口，如百度（http://www.baidu.com/）。用户通过 WebWorld 应用可以显示网站首页，还可以通过点击链接浏览更多的网页。

实际开发中这种需求是存在的。事实上，现在很多 App 里都内置了 Web 网页，比如说电商平台，如淘宝、京东等。对于大多数需要帮助文档的应用，普遍做法是以网页的形式提供帮助文档，这样会方便后期的更新与维护。打开浏览器查看帮助文档，既不专业，又妨碍应用行为的定制，同时也无法将网页整合进自己的用户界面。

图 1-3 显示了 WebWorld 应用的界面。

图 1-3 WebWorld 应用显示百度网站首页

WebWorld 应用由一个 **activity**（活动）和一个 **layout**（布局）组成……。可能你已经迫不及待了，在进一步介绍概念之前，我们先来创建 WebWorld 应用。

1.3.1 创建 Android 项目

首先我们创建一个 Android 项目。Android 项目包含组成一个应用的全部文件。

启动 Android Studio。

如果是首次运行 Android Studio，会看到如图 1-4 所示的 Welcome 对话框。

图 1-4 Android Studio 的 Welcome 对话框

从该对话框选择 **Start a New Android Studio project**。

如果没有出现过上述对话框，说明你之前已经创建过项目。在这种情况下选择 **File→New Project**……。

可以看到新建项目向导。在该向导的 **Applicable name**（应用名称）处输入 WebWorld 作为项目名称（见图 1-5）。在 **Company Domain**（公司域名）处输入 jointhope.studio.com(请根据你们小组的名字或你的喜好起名)。这样做后可以看到 **Package name**（包名）自动变成 **com.studio.jointhope.webworld**。对于 **Project Location**，可以使用任意位置。

注意，包名遵循了“DNS”反转约定，亦即将企业、组织或公司的域名反转后，在尾部附加上项目名称。遵循此约定可以保证包名的唯一性。这样，同一设备和 Google Play 商店的各类应用就可以区分开来。

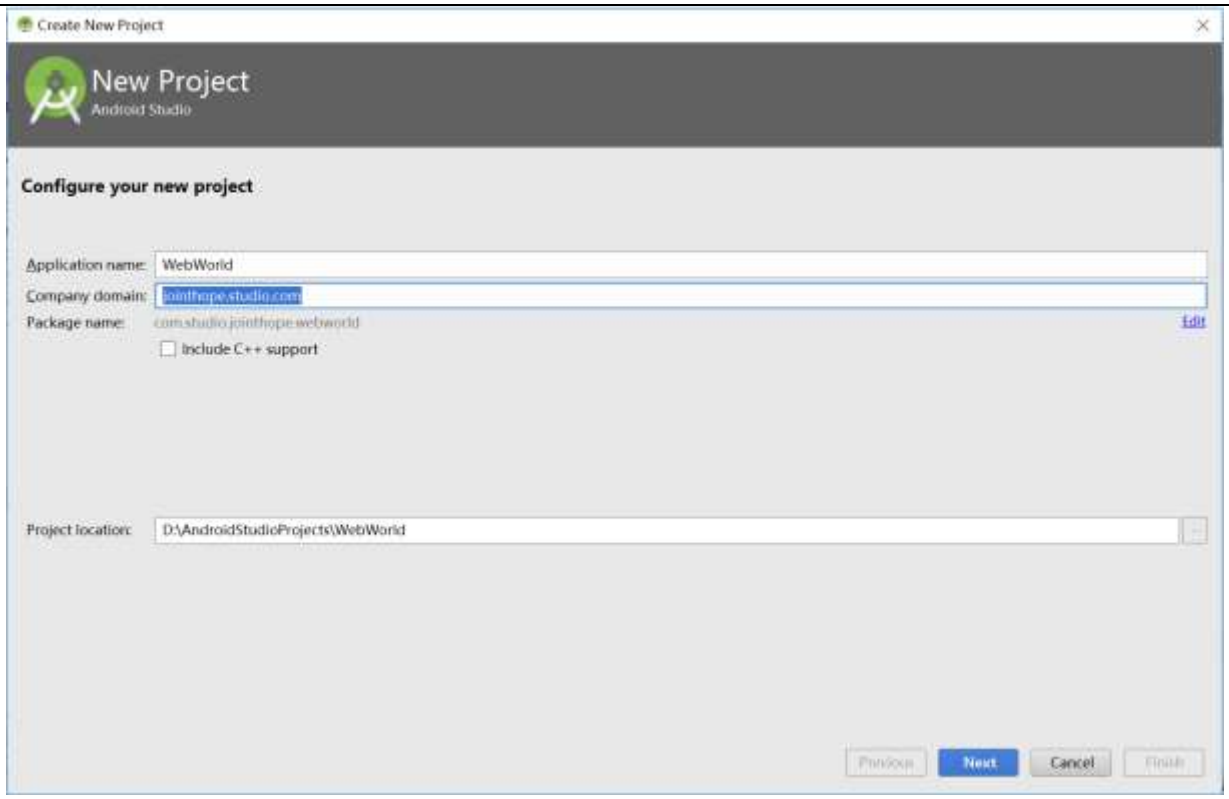

图 1-5 创建新 Android 项目

点击 **Next**，下一个屏幕用来配置应用如何与不同版本的 Android 设备适配。WebWorld 应用支持智能手机和平板，所以只需选择 **Phone and Tablet**。选择最低支持 SDK（Minimum SDK）版本，API 19:Android 4.4(KitKat)，见图 1-6。

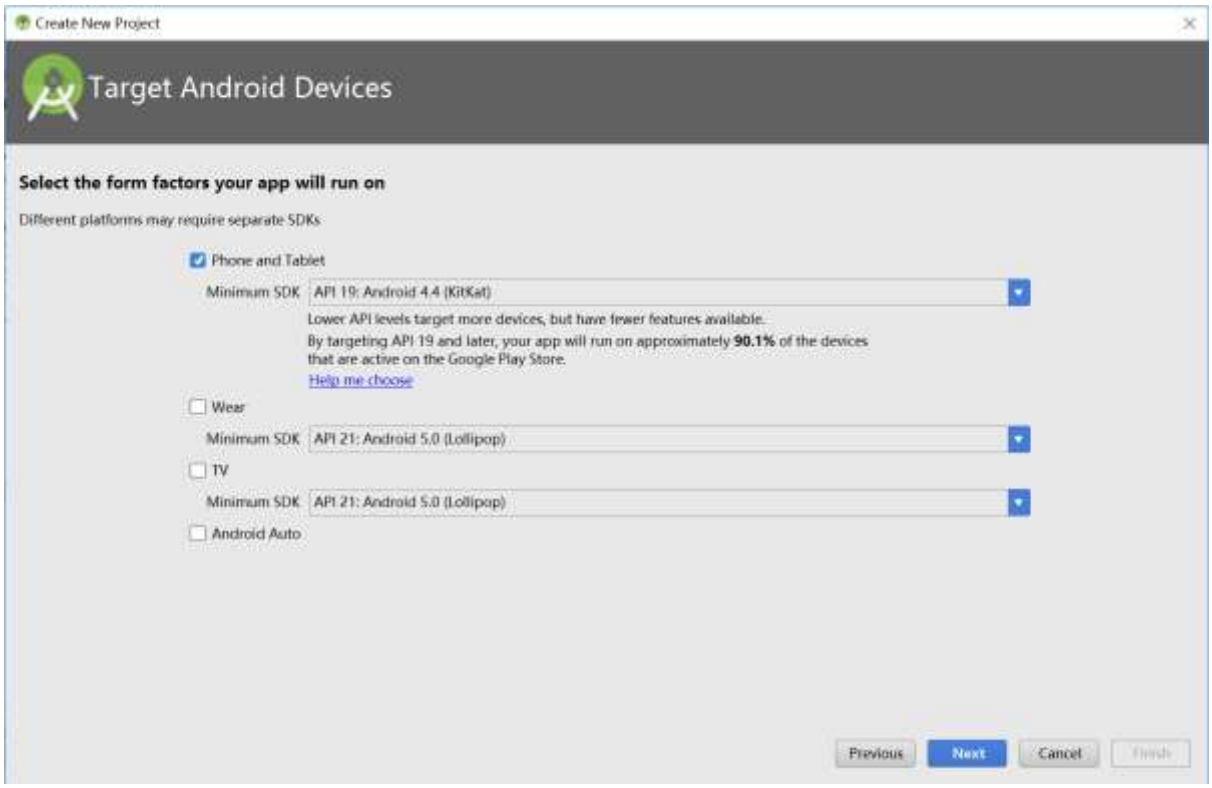

图 1-6 确定支持的装置

点击 **Next**，系统会提示为 WebWorld 应用选择一个模板（图 1-7）。这里我们选择 **Empty Activity**。

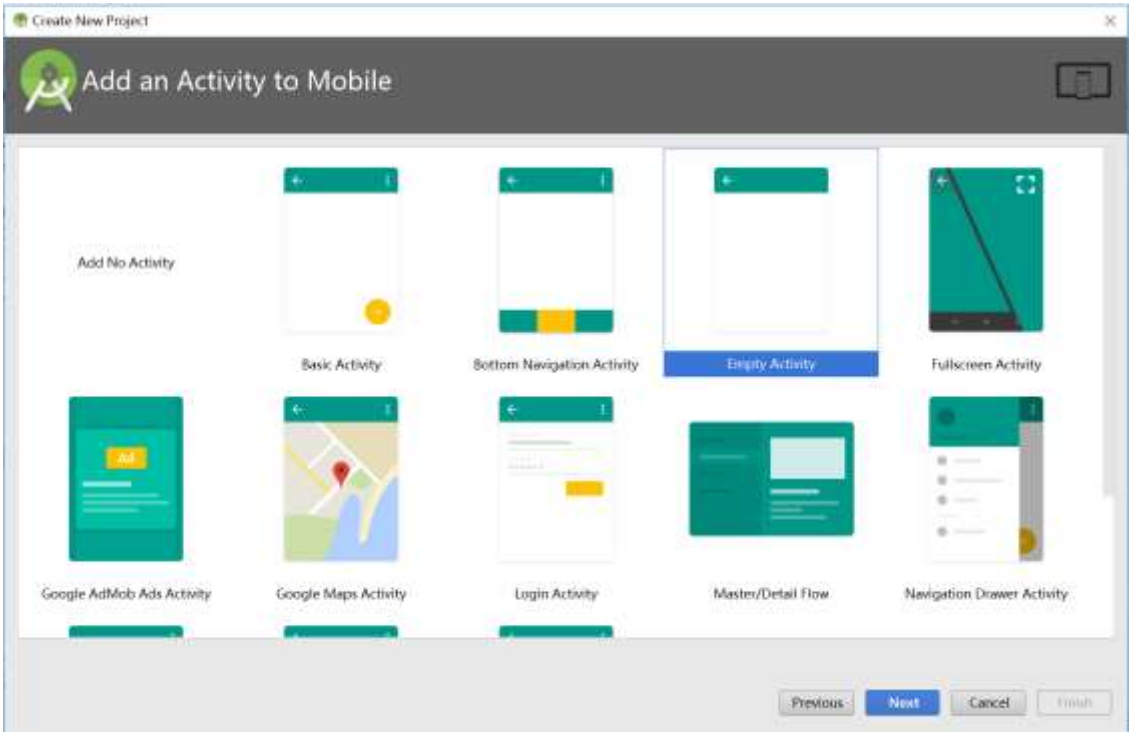

图 1-7 选择 Activity 类型

点击 **Next**，在应用向导的最后一个窗口，将 **Activity Name** 命名为 WebWorldActivity，如图 1-8 所示。注意名子的 Activity 后缀。尽管不是必须的，但建议遵循这一命名约定。

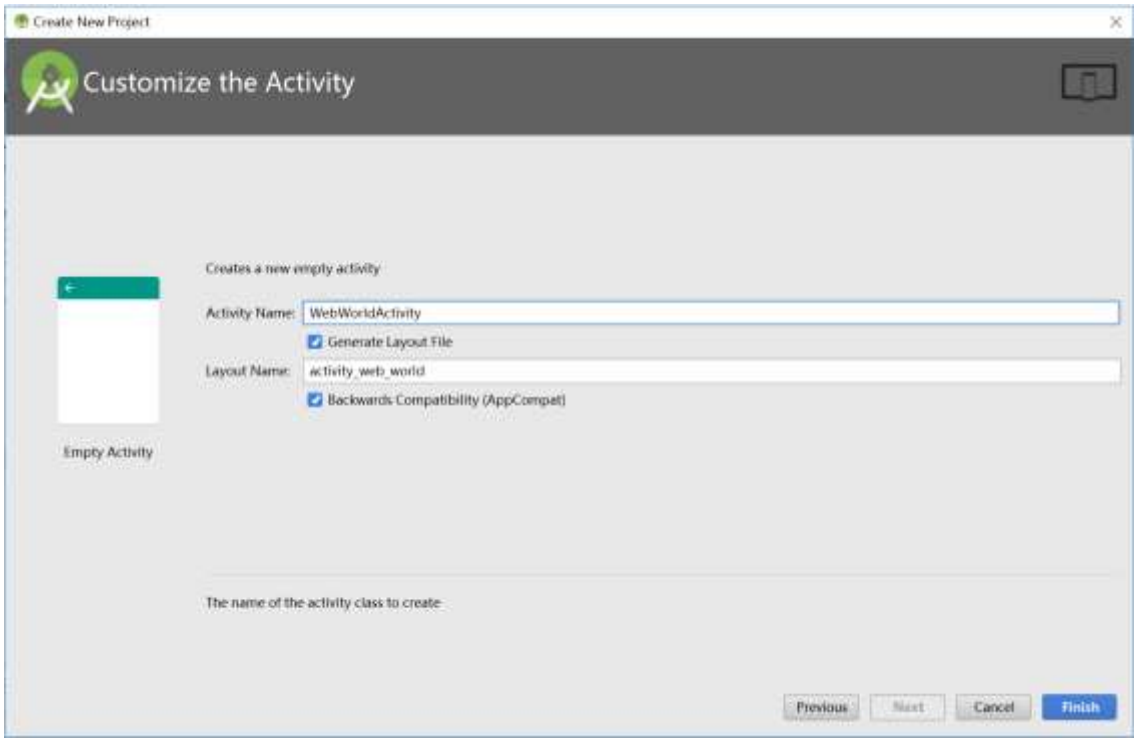

图 1-8 配置新的 Activity

可以看到 **Layout Name**（布局名称）自动更新为 activity_web_world，体现布局与 **activity** 的对应关系。布局的命名规则是：将 **activity** 名称的单词顺序倒过来并全部转换为小写字母，然后在单词间添加下划线。

点击 **Finish**，Android Studio 将创建并在一个新窗口打开你的项目，如图 1-9 所示。

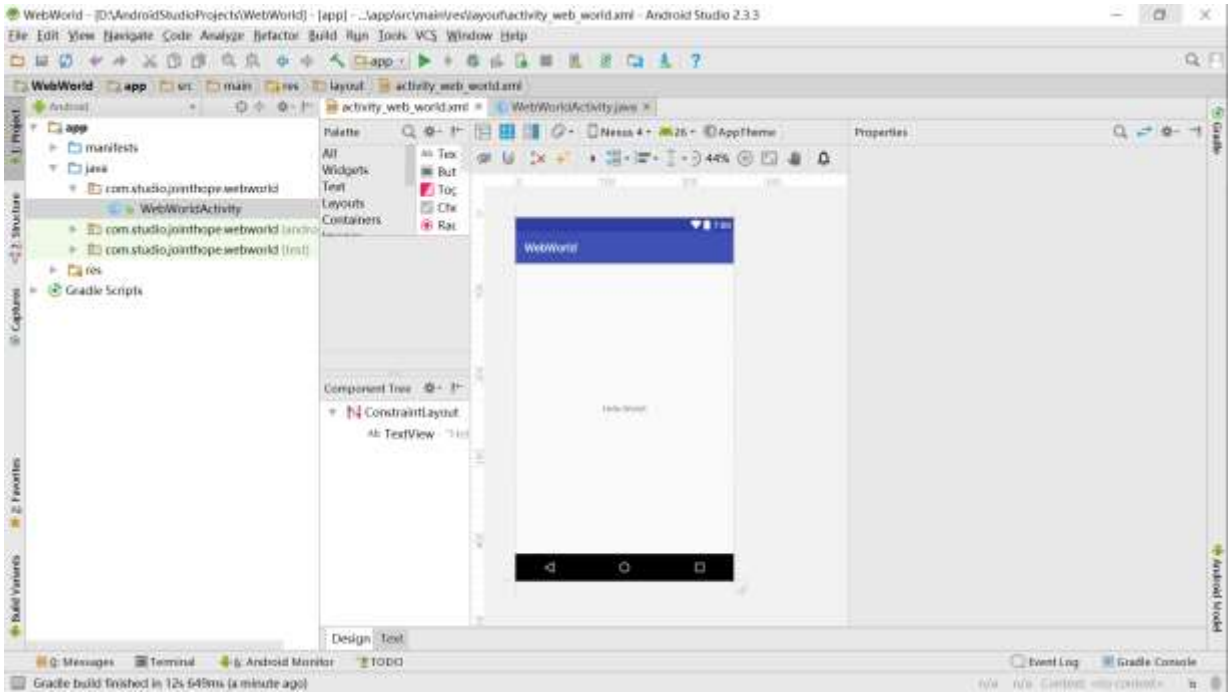

图 1-9 新项目窗口

在编辑器的上方，Android Studio 已经为我们准备了布局文件“**activity_web_world.xml**”和 java 文件“**WebWorldActivity.java**”两个选项卡。这里编辑器已经打开了 activity_web_world.xml 文件。如果想看到 WebWorldActivity 的 java 代码，可单击编辑器顶部的“**WebWorldActivity.java**”选项卡。

1.3.2 android 基本概念

1.3.2.1 布局的概念

在编辑器的上方，单击“activity_web_world.xml”选项卡，就可以看到如图 1-9 所示的布局文件。如果想在编辑器中看到布局文件的代码，可单击底部的“**Text**”选项卡。

Layout（布局）定义了一系列用户界面对象以及它们显示在屏幕上的位置。组成 layout 的定义保存在 XML 文件中。每个定义用来创建屏幕上的一个对象，如按钮或文本信息。

WebWorld 应用包含一个名为 activity_web_world.xml 的布局文件，该布局文件中的 XML 标签定义了图 1-9 所示的用户界面。

目前，activity_web_world.xml 定义了默认的布局，如代码清单 1-1 所示。

代码清单 **1-1** 默认的 **activity** 布局（**activity_web_world.xml**）

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
    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="com.studio.jointhope.webworld.WebWorldActivity">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello World!"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent"/>

</android.support.constraint.ConstraintLayout>
```

应用 activity 的布局默认定义了两个组件（**widget**）：**ConstraintLayout** 和 **TextView**。

组件（**widget**）是组成用户界面的构造模块。组件可以显示文字或图像、与用户交互，甚至是布置屏幕上的其他组件。按钮、文本输入控件和选择框等都是组件。

Android SDK 内置了多种组件，通过配置各种组件可以获得所需的用户界面及行为。每一个组件是 **View** 类或其子类（如 **TextView** 或 **Button**）的一个具体实例。

图 1-9 展示了代码清单 1-1 中定义的 ConstraintLayout 和 TextView 是如何在屏幕上显示的。

其中 ConstraintLayout 是该布局的根元素。ConstraintLayout 在 2016 年由 Google I/O 推出。从支持力度看，将成为主流布局样式，完全代替其他布局，减少布局的层级，优化渲染性能。在新版 Android Studio 中，ConstraintLayout 已替代 RelativeLayout，成为 HelloWorld 项目的默认布局。Constraint 翻译为约束，ConstraintLayout 就是在子 View 上添加各种约束条件来控制每个子 View 所在的位置以及显示的尺寸。而且这一切都是通过可视化操作

来实现的。当然，可视化操作的背后仍然还是使用的 XML 代码来实现的，只不过这些一切都是 Android Studio 自动生成的。

组件与 XML 元素一一对应。元素的名称就是组件的类型。各元素均有一组 XML 属性。属性可以看作是如何配置组件的指令。

组件属性

下面我们来看看配置组件的一些常用属性。

1. android:layout_width 和 **android:layout_height** 属性

几乎每类组件都需要 android:layout_width 和 android:layout_height 属性。它们通常被设置为以下两种属性之一。

- **match_parent**:视图与其父视图大小相同；
- **wrap_content**:视图将根据其内容自动调整大小。

根 ConstraintLayout 组件的高度与宽度属性值均为 match_parent。ConstraintLayout 虽然是根元素，但它也有父视图（View）——Android 提供该父视图用来容纳应用的整个视图层级结构。

其他包含在界面布局中的组件，其高度和宽度属性均被设置为 wrap_content。

2. android:text 属性

TextView 组件具有 android:text 属性。该属性指定组件显示的文字内容，这里就是“Hello World!”。这里我们硬编码设置了组件的文本属性，但通常 android:text 属性值不是字符串字面值，而是对字符串资源（string resources）的引用。虽然可以硬编码设置组件的文本属性，如代码清单 1-1 中：

```
android:text="Hello World!"
```

但这通常不是个好方法。将文字内容放置在独立的字符串资源 XML 文件中，然后引用它们才是好方法。在后续章节中，我们将学习如何使用字符串资源轻松实现本地化。

3. app:layout_constraintBottom_toBottomOf.....属性

这些都是 ConstraintLayout，即约束布局的内容。ConstraintLayout 作为非绑定(Unbundled)的支持库，命名空间是 app:，即来源于本地的包命名空间，在项目的

build.gradle 中声明……。就介绍到这里，否则现在的你会吃不消的。有关约束布局我们会在后续章节中详细介绍，这里只要有一点概念即可。

1.3.2.2 活动的概念

那么 activity_web_world.xml 中的 XML 元素是如何展现在屏幕上的呢？答案就在于 **Activity**。Activity（活动）是一个应用程序组件，提供一个屏幕，负责管理用户与屏幕的交互。应用的功能是通过编写一个个 Activity 子类来实现的。简单的应用可能只需要一个 Activity，而复杂的应用则会有多个 Activity。

在创建 WebWorld 项目的同时也创建了一个名为 WebWorldActivity 的 Activity 的子类。单击编辑器顶部的“**WebWorldActivity.java**”选项卡即可显示 WebWorldActivity 的内容。

WebWorldActivity 类文件存放在 app/java 目录下。该 java 目录是项目全部 java 源代码的存放处。在默认情况下，Android Studio 使用 **Android** 视图。该视图隐藏了 Android 项目的真实目录结构，因此，在很多情况下，需要借助于文件或文件夹。如图 1-10，在位于 **Project** 工具窗口顶部的 **Android** 下拉菜单中，从 **Android** 视图切换到 **Project** 视图。**Project** 视图将以实际情况显示项目中的文件和文件夹。在 **Project** 视图中依次展开 **app/java** 目录与 **com.studio.jointhope.webworld** 包，显示其中的内容。

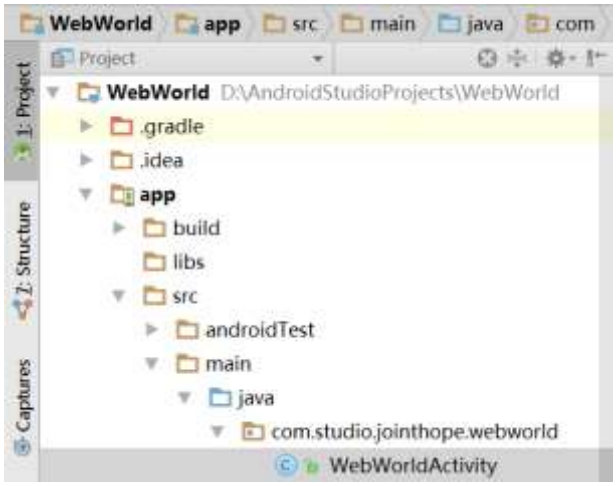

图 1-10 Project 视图中展开各级目录

然后打开 WebWorldActivity.java 文件，其代码清单 1-2 所示，仔细查看其中的代码。

代码清单 **1-2 WebWorldActivity** 的默认代码（**WebWorldActivity.java**）

```
package com.studio.jointhope.webworld;
```

```
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
public class WebWorldActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_web_world);
    }
}
```

如果不能看到全部类包导入语句，请单击第一行导入语句左边的⊕符号，从而显示全部导入语句。

该 java 类目前提供了一个 Activity 方法：**onCreate(Bundle)**。这里 **AppCompatActivity** 是 Activity 类的一个子类，用于支持与较早 Android 版本的兼容性。

Activity 子类的实例创建后，onCreate(Bundle)方法将会被调用。Activity 创建后，它需要获取并管理属于自己的用户界面。获取 activity 的用户界面调用的是如下 Activity 方法:

```
public void setContentView(int layoutResID)
```

通过传入布局的资源 ID 参数，该方法生成指定布局的视图，并将其放置在屏幕上。那么什么是资源 ID 呢?

1.3.2.3 资源与资源 ID

资源是应用中非代码形式的内容，如 XML 文件、图像文件以及音频文件等，布局也是一种资源。项目的所有资源文件都存放在目录 res/的子目录下。在 **Project** 视图中可以看到，布局 activity_web_world.xml 资源文件存放在 res/layout/目录下。

可以想象，资源就是你的邮件，而资源 ID 就是你邮件的单号，这个单号是快递公司在产生这个邮件时创建的与该邮件的对应关系的一串数字，存储在快递公司的数据中心，你不能随意改动。

资源 ID 与快递公司的邮件单号有点类似，是 Android SDK 自动生成的关于资源的一串数字。Android SDK 会自动生成一个 R 文件，所有的资源 ID 都在 R 文件中。

要看到当前 WebWorld 项目的资源 ID，展开 app/build/generated/source/r/debug 目录的内容。在这个目录中找到项目的包名，并在包中打开 R.java 文件。因为这个文件是在 Android 编译过程中自动生成的，正如是在该文件的顶部所警告的那样，不要改变它

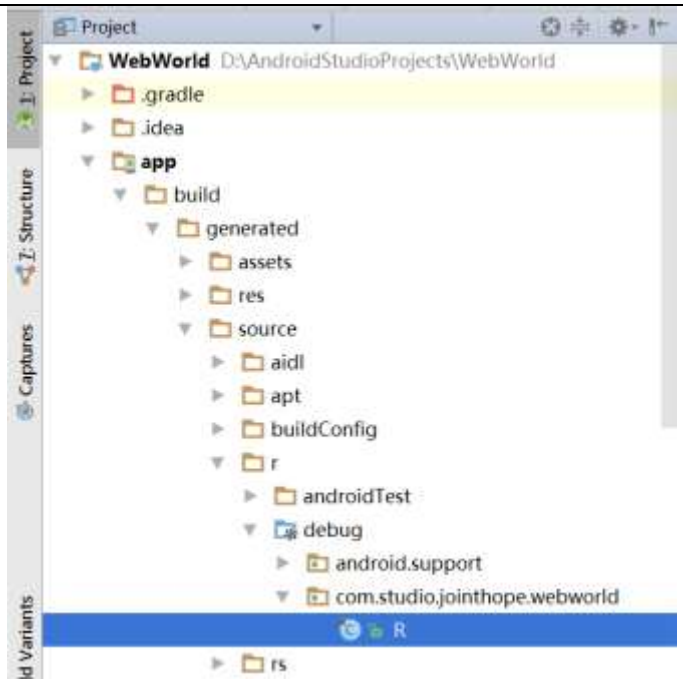

图 1-11 R 文件所在的目录

可以使用资源 ID 在代码中获取相应的资源。如 activity_web_world.xml 文件定义的布局资源 ID 为 R.layout. activity_web_world，它对应于 R 文件中的一串 16 进制数字。

```
/* AUTO-GENERATED FILE.    DO NOT MODIFY.
*
* This class was automatically generated by the
* aapt tool from the resource data it found.    It
* should not be modified by hand.
  */

packagecom.me.android.encyclopediaquiz;

public final class R {
        ......
        public static final class layout {
        ......
                public static final int activity_web_world=0x7f04001c;
                ......
        }
......
}
```

可以看到 R.layout. activity_web_world 即来自该文件。activity_web_world 是 R 的内部类 layout 里的一个整形常量名。

如果对资源进行了更改，可能无法立即看到该文件更新。针对生成的代码，Android

Studio 维护一个隐藏的 R.java 文件。这里看到的 R.java 文件是为以前的应用程序生成的。当运行应用程序时，就会看到这个文件的更新。

1.3.3 使用 WebView 浏览网页

不过代码清单 1-1 所示的默认 **TextView** 组件并不是我们所想要的。WebWorld 应用并不是仅仅为用户展现一个“Hello World!”。我们想要通过该应用展现移动互联网的精彩世界，即在 App 中内嵌 Web 网页。那么这该如何实现呢？

基本思路时留出一块屏幕用以显示 Web 页面，并加载一个 html 文件。如果你有过桌面系统 Web 浏览器的开发经历，就会知道，这不是几行代码就能搞定的。

幸好 Android 里一个叫 **WebView** 的组件可以帮我们实现这一切。

WebView 是一个展现 web 页面的控件，其作用是：

- 显示和渲染 Web 页面；
- 直接使用 html 文件（网络上或本地 assets 中）作布局；
- ……。

下面，我们首先在布局文件 activity_web_world.xml 中定义 WebView 组件。

如代码清单 1-3 所示，修改 activity_web_world.xml 文件。注意，打上删除线的是需要删除的内容，黑体显示需要添加内容。

代码清单 **1-3 WebWorldActivity** 的布局（**activity_web_world.xml**）

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
    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="com.studio.jointhope.webworld.WebWorldActivity">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello World!"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
```

```
        app:layout_constraintTop_toTopOf="parent"/>
    <WebView
        android:id="@+id/webview"
        android:layout_width="match_parent "
        android:layout_height="match_parent "/>
</android.support.constraint.ConstraintLayout>
```

我们在布局文件里添加了一个 WebView 控件，其宽度和高度布满整个屏幕。Android 为整个布局文件以及各个字符串生成资源 ID，不是所有的组件都需要资源 ID。在本章中，我们只需为 WebView 生成相应的资源 ID。要为组件生成资源 ID，在定义组件时为其添加上 android:id 属性。

正如代码清单 1-3 所示，我们增加了一行 android:id="@+id/webview"，表示向 Android 中增加一个控件 ID。

Android 中的组件需要用一个 int 类型的值来表示，这个值也就是组件标签中的 id 属性值。id 属性只能接受资源类型的值，也就是必须以@开头的值，例如本例中的 @+id/webview。

如果在@后面使用"+"，表示当修改完某个布局文件并保存后，系统会自动在 R.java 文件中生成相应的 int 类型变量。变量名就是"/"后面的值，例如，@+id/webview 会在 R.java 文件中生成 int webview = value，其中 value 是一个十六进制的数。如果 webview 在 R.java 中已经存在同名的变量，就不再生成新的变量，而该组件会使用这个已存在的变量的值。

也就是说，如果使用@+id/name 形式，当 R.java 中存在名为 name 变量时，则该组件会使用该变量的值作为标识。如果不存在该变量，则添加一个新的变量，并为该变量赋相应的值（不会重复）。

接下来我们来编码使用 WebView 组件，这只需要一个步骤：

- 引用生成的视图对象。

1.3.3.1 引用组件

在 activity 中，可以通过以下 Activity 方法引用已经生成的组件：

```
public View findViewById(int id);
```

该方法接受组件的资源 ID 作为参数，返回一个视图对象。在 WebWorldActivity.java 文件中，使用 WebView 组件的资源 ID 获取生成的对象后，赋值给对应的成员变量，如代码

清单 1-4 所示。注意，赋值前必须先将返回的 View 转型（cast）为 WebView。

代码清单 **1-4** 引用组件（**WebWorldActivity.java**）

```
package com.studio.jointhope.webworld;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.webkit.WebSettings;
import android.webkit.WebView;
import android.webkit.WebViewClient;

public class WebWorldActivity extends AppCompatActivity {
    WebView mWebView;//成员变量
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_web_world);
        WebView mWebView = (WebView) findViewById(R.id.webView); //在 Activity 的 layout 文件里添加 webview 控件
        mWebView.setWebViewClient(new WebViewClient());
        mWebView.loadUrl("http://www.baidu.com/");//加载一个网页
    }
}
```

我们调用了 WebView 的 setWebViewClient()方法，并传入了一个 WebViewClient 实例。这段代码的作用是，当需要从一个网页跳转到另一个网页时，我们希望目标网页仍然在当前 WebView 中显示，而不是打开系统浏览器。

调用 WebView 的 loadUrl()方法，并将网址传入，即可展示相应网页的内容。由于本程序使用到了网络功能，而访问网络是需要申明权限的。

1.3.4 在 manifest 配置文件中申明权限

manifest 配置文件是个包含元数据的 XML 文件，用来向 Android 操作系统描述应用。该文件以 AndroidManifest.xml 命名，在 **Project** 视图中，可在项目的 app/src/main 目录中找到它。应用的所有 activity 都必须在 manifest 配置文件中声明，这样操作系统才能够使用它们。manifest 文件如代码清单 1-5 所示。

代码清单 **1-5** 引用组件（**WebWorldActivity.java**）

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

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
          package="com.studio.jointhope.webworld">

<uses-permission android:name="android.permission.INTERNET"/>

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".WebWorldActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN"/>

                <category android:name="android.intent.category.LAUNCHER"/>
            </intent-filter>
        </activity>
    </application>

</manifest>
```

这里的 android:name 属性是必需的。属性值前面的.告诉操作系统：activity 类文件位于 manifest 配置文件头部包属性值指定的包路径下。android:name 属性值也可以设置成完整的包路径，如 android:name="com.studio.jointhope.webworld.WebWorldActivity"。

AndroidManifest.xml 配置文件包含下面这些信息：

- 应用程序的包名：包名是应用的唯一标识
- 应用程序所包含的组件：如 Activity、Service、BroadcastReceiver 和 ContentProvider 等
- 应用程序兼容的最低版本
- 应用程序使用系统所需的权限声明
- 其他程序访问该程序所需的权限声明

本项目中，创建 WebWorldActivity 时，因使用了新建应用向导，向导已自动完成声明工作。只需添加访问权限声明，即在<manifest.../>元素中添加：

```
<uses-permission android:name="android.permission.INTERNET"/>
```

如代码清单 1-5 中黑体部分所示。

1.4 运行应用

运行 Android 应用需使用硬件设备或者虚拟设备（virtual device）。

1.4.1 使用模拟器运行应用

开发工具中的 Android 设备模拟器可提供多种虚拟设备。要创建 Android 虚拟设备（AVD），在 Android Studio 中，选择 Tools→Android→AVD Manager 菜单项。AVD 管理器窗口弹出时，点击窗口左边的 Create Virtual Device...按钮。在随后弹出的对话框中，可以看到有很多配置虚拟设备的选项。对于首个虚拟设备，我们选择模拟运行 Nexus5 设备，如图 1-12 所示。点击 Next 继续。

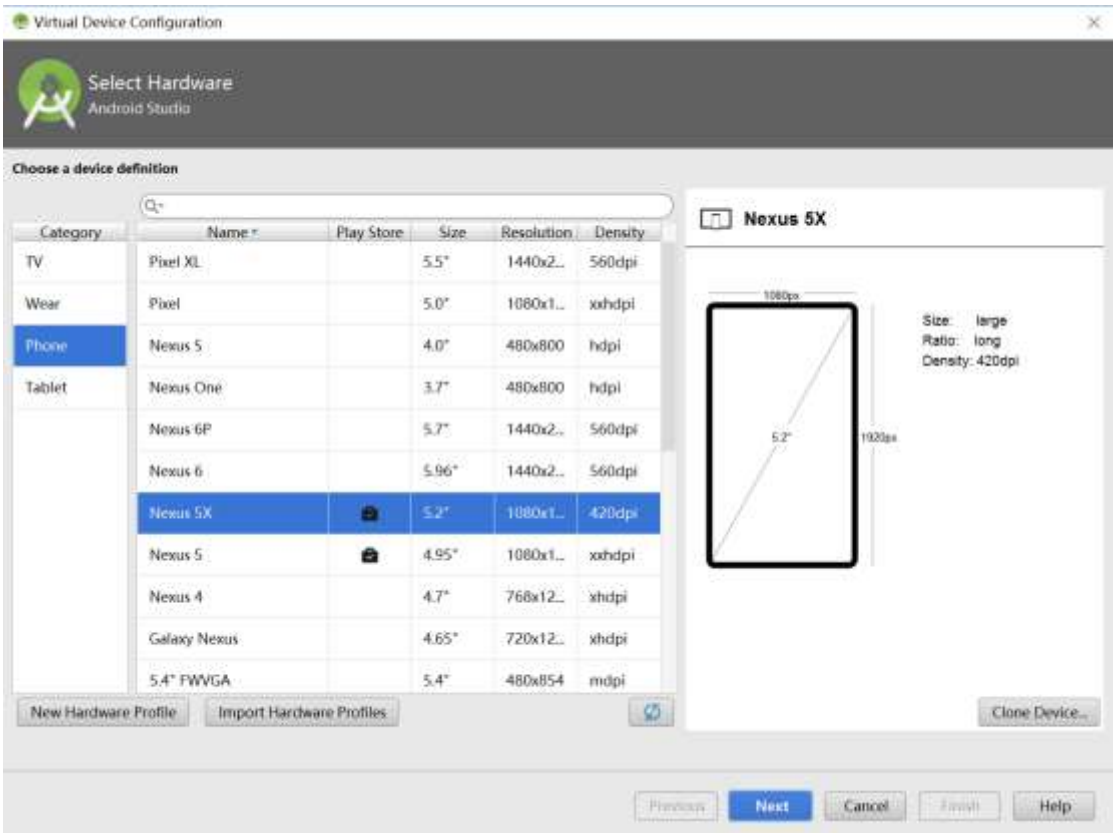

图 1-12 创建新的 AVD

接下来选择模拟器的系统镜像，如图 1-13 所示。选择 x86 Lollipop 模拟器后点击 Next 按钮继续。

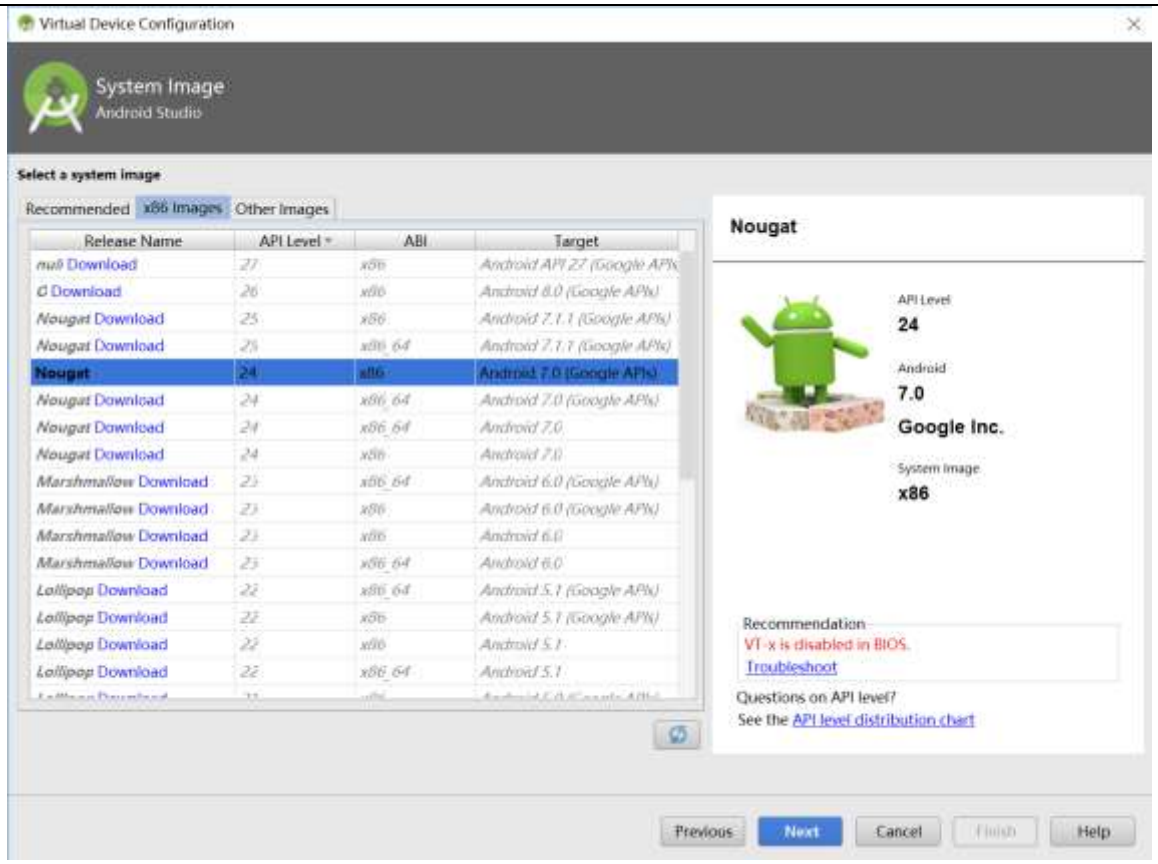

图 1-13 选择系统镜像

选择系统镜像。

需要说明的时，图 1-13 中出现了红色标注的 VT-x isdisabled inBIOS。意思就是说主机支持 VT-x，但是处于未开启状态。你需要进入 BOIS 中进行设置。

BOIS 的设置，对于不同电脑各不相同。例如对于联想电脑，开机出现 Lenovo 时按 F2 即进入 BOIS，选择 Security 选项卡，选择 Virtualization，进入后将 Intel virtualization Technology 项置为 Enable，按 F10 保存退出，启动电脑即可。其他类型，可参照相关说明，或百度查询。

最后，可以对模拟器的各项参数做最后修改并确认，如图 1-14 所示。当然，如有需要，也可以事后再编辑修改模拟器的各项参数。现在，为模拟器取个便于识别的名称，点击 Finish 按钮完成虚拟设备的创建。

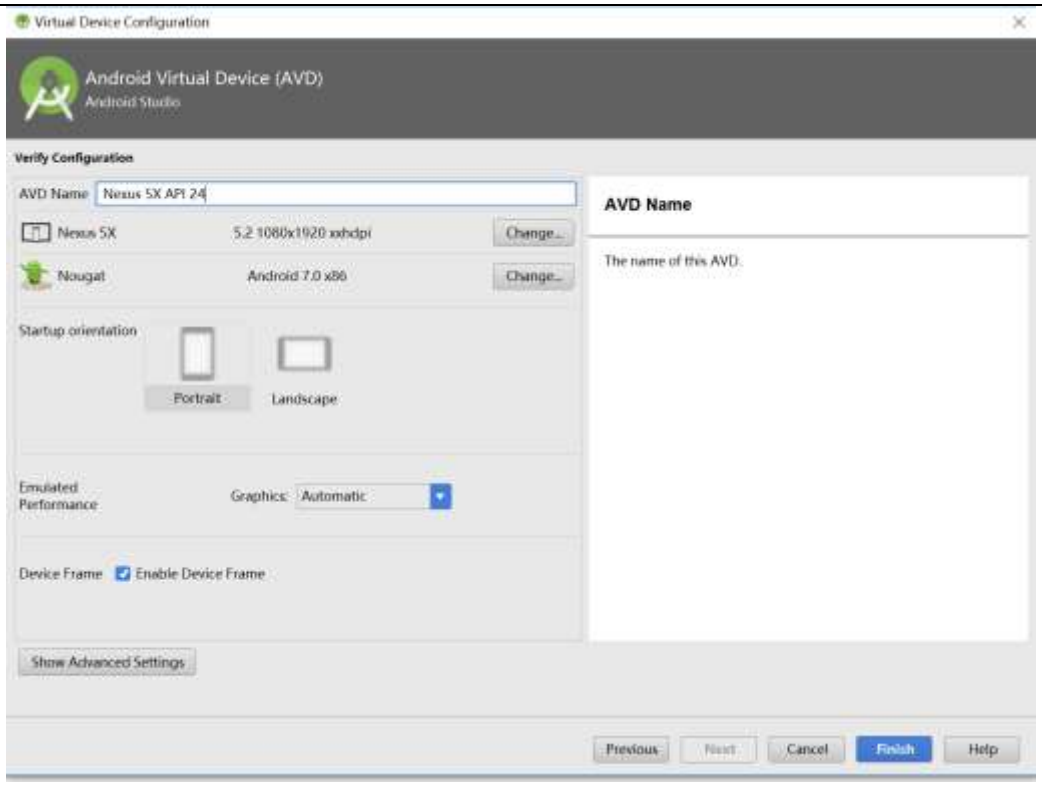

图-14 模拟参数调整

AVD 创建成功后，我们用它运行 WebWorld 应用。点击 Android Studio 工具栏上的 Run 按钮，Android Studio 会自动找到新建的虚拟设备，安装应用包（APK），然后 启动并运行应用。虚拟机启动过程非常缓慢，请耐心等待。设备启动完成，应用运行后，就可以看到如图 1-15 所示百度界面。

图 1-15 第模拟器运行 WebWorld 应用

模拟器虽然有用，但在真实设备上测试应用能获得更准确的结果。

1.4.2 使用真机运行应用

本节，我们学习如何设置系统、设备和应用，实现在硬件设备上运行 WebWorld 应用。

首先，将设备连接到系统上。如果是在 Mac 系统上开发，系统应该会立即识别出所用设备。如果是 Windows 系统，则可能需要安装 adb（Android Debug Bridger）驱动。如果 Windows 系统自身无法找到 adb 驱动，请到设备生产商的网站下载。

要在设备上测试应用，首先应打开设备的 USB 调试模式。不同版本设备的设置有较大差异。Android 4.2 或之后版本的设备，开发选项默认不可见。先选择“设定→关于平板/手机”项，通过点击版本号 7 次启用它，然后回到“设定”项，选择“开发”项，找到并勾选“**USB** 调试”选项。Android 4.0 或 4.1 版本的设备，选择“设定→开发”项，找到并勾选“USB 调试”选项。Android 4.0 版本以前的设备，选择“设定→应用项→开发”项，找到并勾选“**USB** 调试”选项。如在设置过程中遇到问题，请访问百度寻求帮助。

打开 **Devices** 视图来确认设备是否得到识别。选择 Android Studio 底部的 **Android Monitor** 工具窗口，可快速打开 Devices 视图。如图 1-16 所示，你会看到已连接设备的下拉列表，硬件设备应该已经列在其中了。

图 1-16 查看已连接设备

如果遇到设备无法识别的问题，请首先确认是否已打开设备和开发者项。如果仍然无法解决，请访问百度寻求帮助。再次运行 WebWorld 应用，Android Studio 会询问是在虚拟设备上还是在物理设备上运行应用。选择物理设备并继续。稍等片刻，WebWorld 应用应该已经在设备上运行了。如果 Android Studio 没有提供选择，应用依然在虚拟设备上运

行，请按以上步骤重新检查设备设置，并确保设备与系统已正确连接。然后，再检查运行配置是否有问题。要修改运行配置，请选择 Android Studio 窗口靠近顶部的 app 下拉列表，如图 1-17 所示。

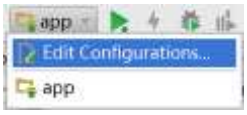

图 1-17 打开运行配置

选择 **Edit Configurations** 打开运行配置编辑窗口，如图 1-18 所示。选择窗口左边区域的 app，确 认已打开 Target Device 区域的 Show chooser dialog 选项。点击 OK 按钮并重新运行该应用，现在你会看到可以运行应用的设备选项。

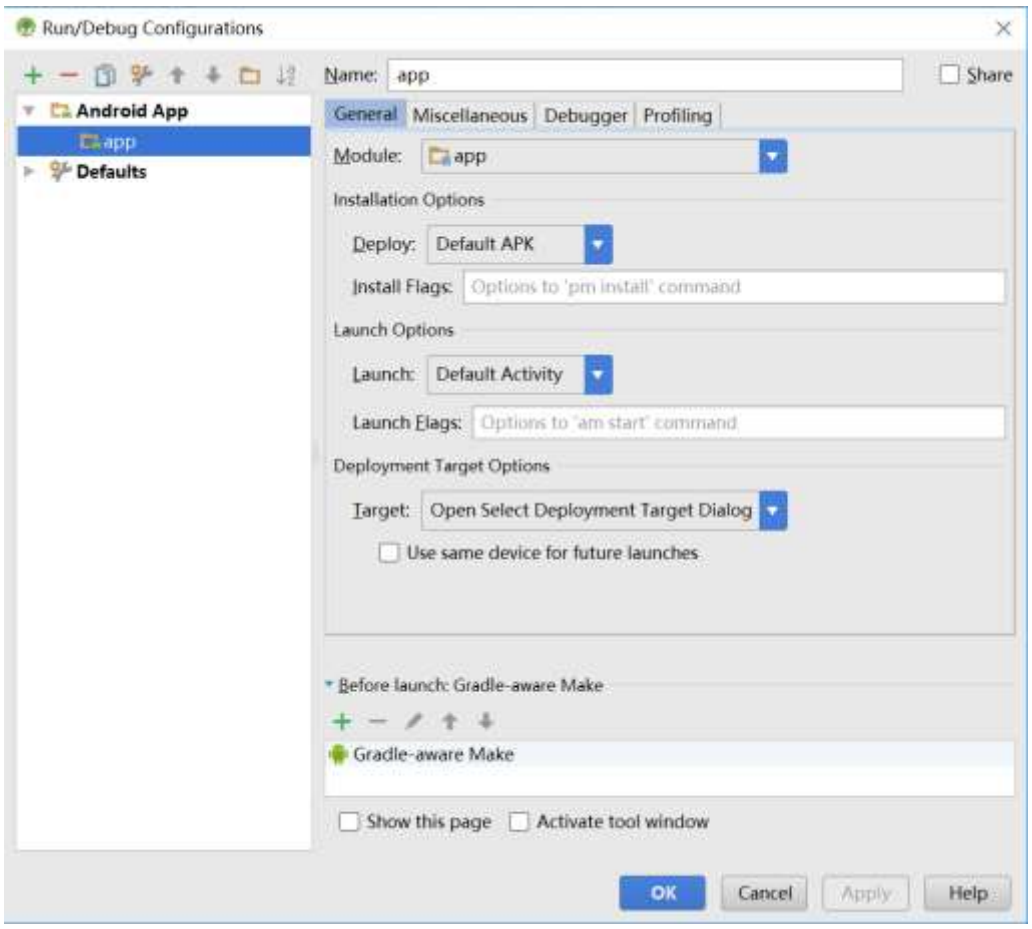

图 1-18 运行配置

运行 WebWorld 应用，确认它能正常工作。