

第3章

第一个 Android 应用程序

本章是掌握 Android 开发技术的开始。对于要从事 Android 开发的人员,必须熟悉本章介绍的内容。本章通过一个最简单的 Hello Android 应用程序,展开介绍相关知识点。

3.1 使用 Android Studio 工具创建项目

Hello Android 应用程序是在屏幕上显示“Hello World!”文字,如图 3-1 所示。

创建 Hello Android 应用最简单的方法是通过 Android Studio 工具提供的模板实现。具体步骤是:启动 Android Studio 工具,如图 3-2 所示,在 Android Studio 欢迎界面中选择 Start a new Android Studio project 菜单。然后进入如图 3-3 所示的配置工程对话框,在对话框的输入项目中,Application Name 项目是应用程序名,这里输入 Hello Android; Company Domain 项目是公司域名,公司域名是构成工程包名的重要组成部分。从图 3-3 可见,如果公司域名输入的是 51work6. com,则包名为域名倒置、com. a51work6. helloandroid,即:公司域名倒置+应用程序名。

提示 在 Java 中的包名命名规范是:一般都是小写字母;首字符不能是数字,包名 com. 51work6 中 51work 部分首字符是数字,这是非法的,因此 Android Studio 工具在前面添加了字母 a;另外,包名中也不能有空格,所以 Hello Android 变换为 helloandroid,即去掉空格小写所有字母。

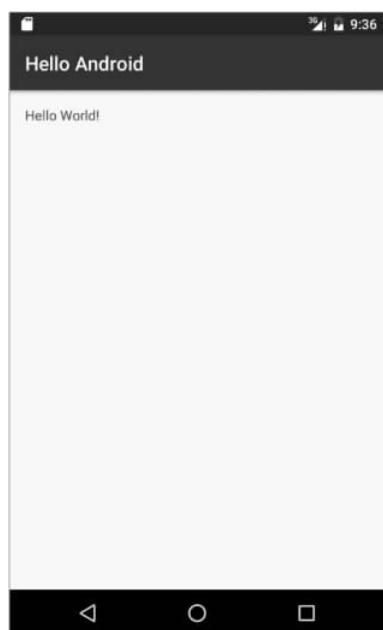


图 3-1 Hello Android 应用运行效果图



图 3-2 Android Studio 欢迎界面

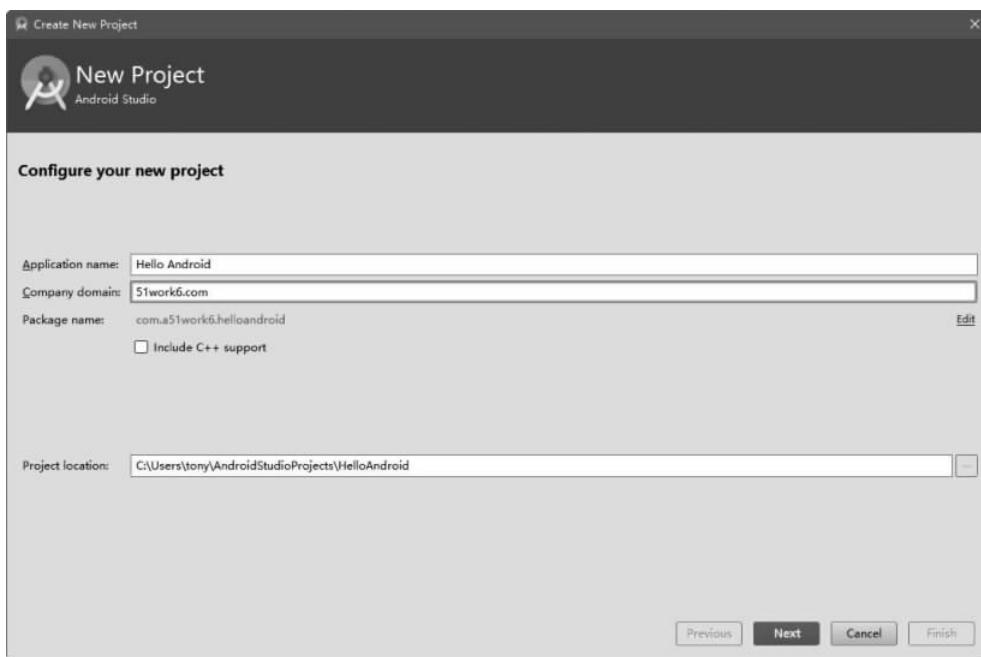


图 3-3 配置工程对话框

在图 3-3 所示的对话框中单击 Next 按钮, 进入如图 3-4 所示的对话框, 这里可以选择不同的 Android 平台和 SDK 版本。目前, Android 平台不仅仅是包括 Android 手机(Phone)和平板电脑(Tablet), 还包括手表(Wear)、电视机(TV)和车载系统(Android Auto), 本例选择 Phone and Tablet。除了选择 Android 平台, 还需要选择该应用发布所支持的 Minimum(最低的)SDK 版本, 本例选择的 API 21, 即 Android 5.0。

提示 在实际发布应用时候, Android 平台的最低 SDK 版本不应该是目前最高版本, 而应该考虑目前大部分用户所采用 Android 版本。从图 3-4 可见, API 21 用户目前不多于 40.5%, 如果不能确定选择哪一个, 可以单击 Minimum SDK 选项下面的 Help me choose 超链接来帮助选择。

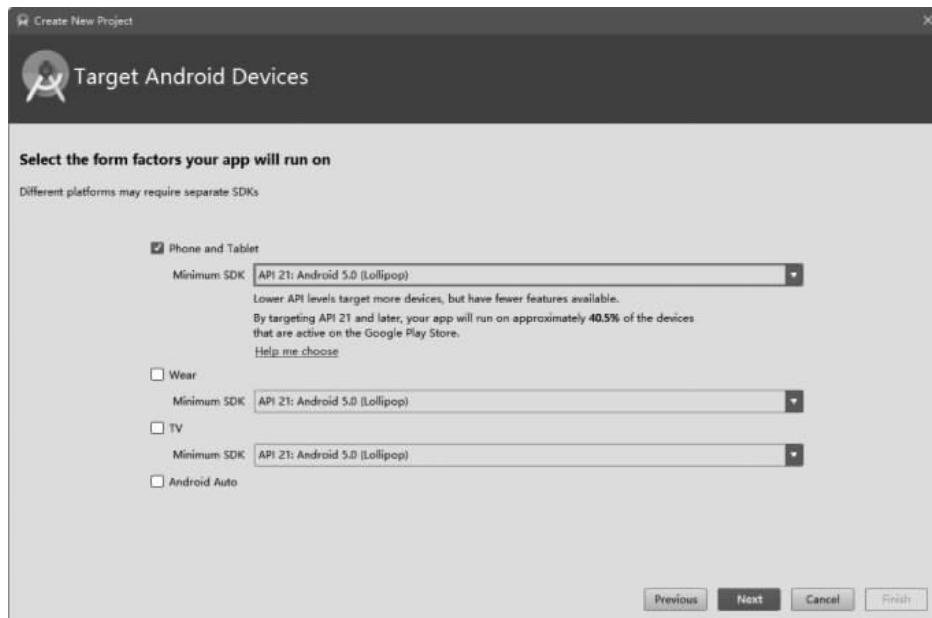


图 3-4 选择 Android 平台和 SDK 版本

在图 3-4 所示的对话框中单击 Next 按钮, 则进入如图 3-5 所示的活动(Activity)模板对话框, 这里可以选择活动(Activity)模板, 就本例而言需选择空活动(Empty Activity)模板。

提示 Activity 是 Android 应用绘制图形界面的重要组件, Activity 中能够包含若干个 View(控件)对象。本书将 Activity 翻译为“活动”。

在图 3-5 所示的对话框界面中单击 Next 按钮, 则进入如图 3-6 所示的自定义活动对话框, 其中的 Activity Name 是活动文件名, 选中 Generate Layout File 会生成布局文件, Layout Name 是布局文件名。

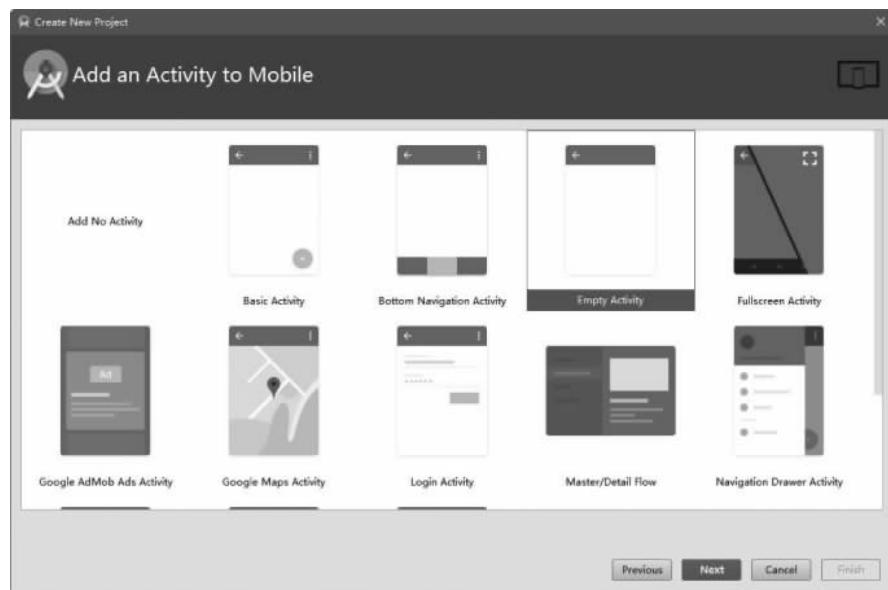


图 3-5 选择活动模板

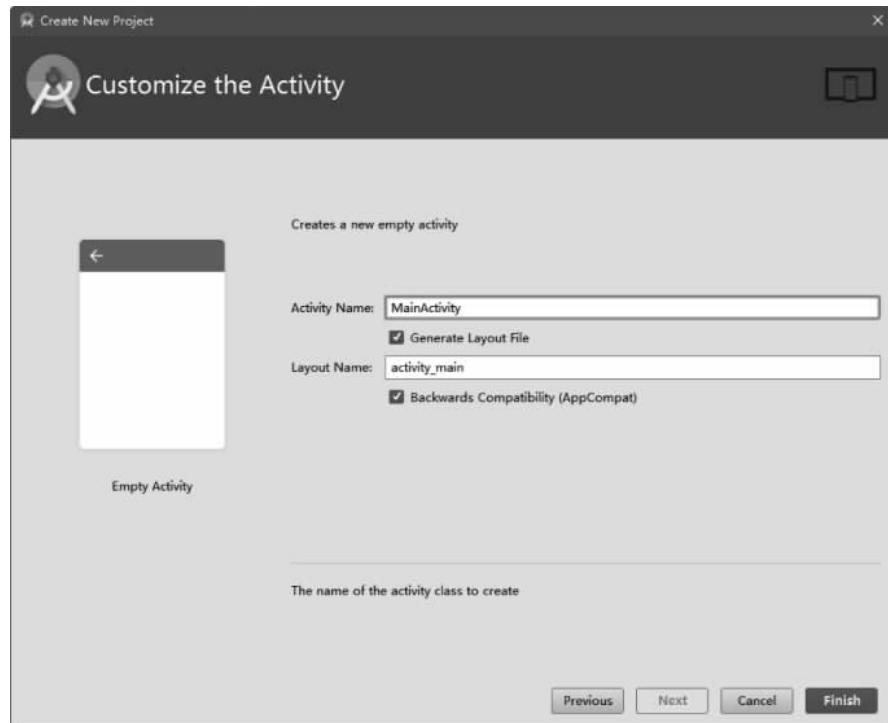


图 3-6 自定义活动

在图 3-6 所示的对话框中单击 Finish 按钮完成创建工程操作，则进入如图 3-7 所示的界面。

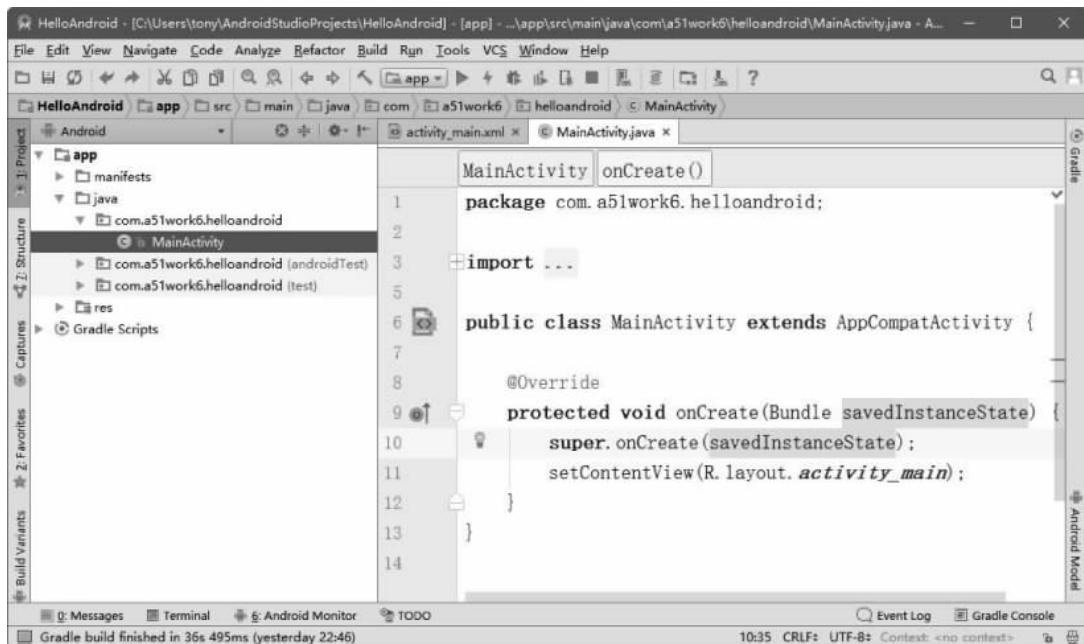


图 3-7 创建工程完成

3.2 Android 工程剖析

工程创建完成之后，需要剖析一下 Android 工程。

3.2.1 Android 工程目录结构

使用 Android Studio 工具开发 Android 应用程序，创建的工程目录结构比较复杂，开发人员应该清楚各个目录下面放置的是什么东西。工程根目录下有 app 和 Gradle Scripts，app 是应重点关注的，app 下面的主要目录有 manifests、java 和 res。

manifests 目录中的 AndroidManifest.xml 是当前 Android 应用程序的清单文件，记录应用中所使用的各种组件，java 是 Java 源代码目录，res 是资源目录。下面重点介绍一下 res 目录。

res 资源目录中存放所有程序中用到的资源文件。“资源文件”指的是布局文件、图片文件和配置文件等。子目录主要有 drawable、layout、mipmap 和 values。

- drawable。存放一些应用程序需要用的图片文件 (*.png 和 *.jpg 等)。
- layout。屏幕布局目录，layout 目录中放置的是布局文件，布局文件是 Xml 文件。

- mipmap。与 drawable 一样存放资源图片，在 Android 2.2 后增加目录，Android 系统会对 mipmap 做了一些优化，加快了图片的渲染速度，提高的图片质量，减少 GPU 的压力。
- values。参数值目录，存放软件所需要显示的各种文字和一些数据。可以在这个目录下的 strings.xml 中存放各种文字，还可以存放不同类型的数据，例如 colors.xml、dimens.xml 和 styles.xml 等。

另外，为了适配不同的设备，res 资源目录中的 drawable、layout、mipmap 和 values 等资源目录，可以分别有多个，图 3-9 是在 Windows 资源管理器中看到的目录结构，其中 mipmap 有 5 个不同的目录：

- mipmap-mdpi。放置中质量图片。
- mipmap-hdpi。放置高质量图片，是 mipmap-mdpi 尺寸的 1.5 倍。
- mipmap-xhdpi。放置超高质量图片，是 mipmap-mdpi 尺寸的 2 倍。
- mipmap-xxhdpi。放置超高质量图片，是 mipmap-mdpi 尺寸的 3 倍。
- mipmap-xxxhdpi。放置超高质量图片，是 mipmap-mdpi 尺寸的 4 倍。

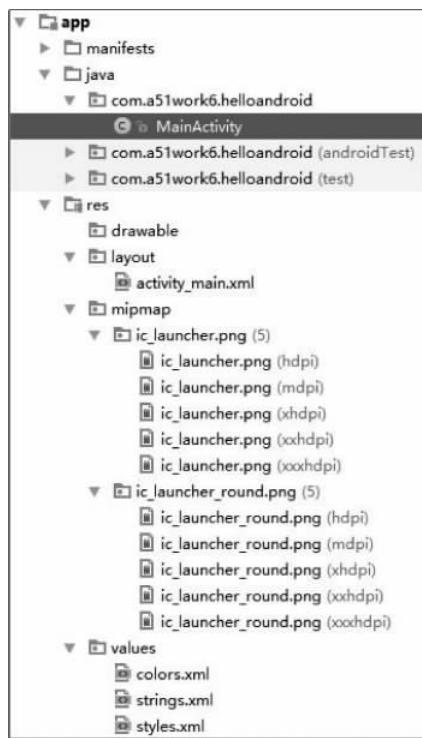


图 3-8 工程目录结构

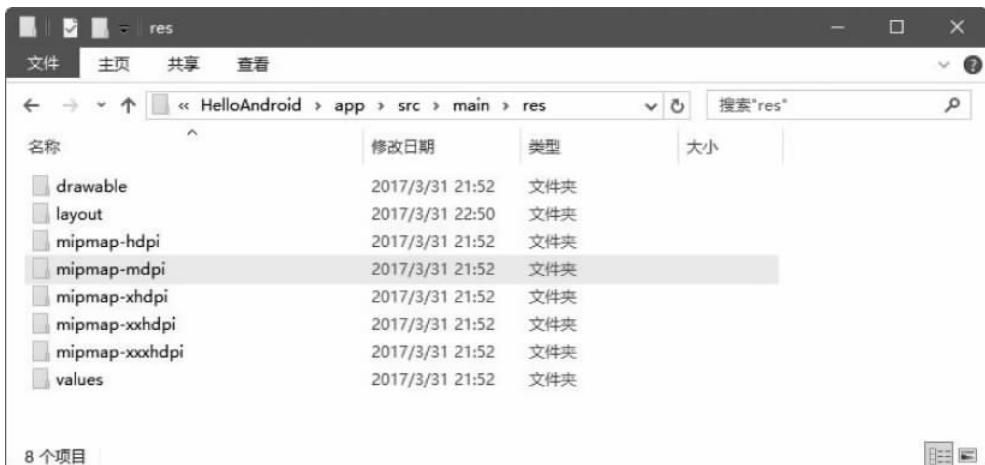


图 3-9 Windows 资源管理器目录结构

3.2.2 R.java 文件

访问 res 目录中的资源文件，并不能通过 Java IO 技术实现，而是通过 R.java 文件访问。R.java 文件是在工程编译时候自动产生的 R 类。

R.java 文件可参考如下代码：

```
package com.a51work6.helloandroid;

public final class R {
    ...
    public static final class mipmap {
        public static final int ic_launcher = 0x7f030000;
    }
    public static final class layout {
        public static final int activity_main = 0x7f030000;
    }
    public static final class string {
        public static final int app_name = 0x7f040001;
        ...
    }
}
```

R 类中包含很多静态类，且静态类的名字都与 res 中的一个目录名字对应，就像是资源字典大全，包含了用户界面、图像、字符串等对应于各个资源的标识符，R 类定义了该项目所有资源的索引。例如，在程序代码中访问 activity_main.xml 布局文件，可以通过表达式 R.layout.activity_main 访问，示例代码如下：

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
}
```

R 类还可以访问界面中的视图，如果视图在布局文件中定义 id 属性，类似代码“`android:id="@+id/textview"`”，那么在程序代码中就可以通过 `R.id.textview` 表达式访问该视图。

3.2.3 MainActivity.java 文件

Hello Android 应用只有一个屏幕，所以只有一个活动类——MainActivity.java 文件。MainActivity.java 具体代码如下：

```
package com.a51work6.helloandroid;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
```

```

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
}

```

MainActivity 是一个活动组件，MainActivity 的父类是 AppCompatActivity，AppCompatActivity 是 Activity 子类，AppCompatActivity 是支持 ActionBar 的活动类。onCreate 方法是在活动组件初始化时候调用方法。setContentView 方法是设置活动布局内容，参数是 R.layout.activity_main。

3.2.4 activity_main.xml 布局文件

布局文件 activity_main.xml 位于 res 的 layout 目录中，activity_main.xml 布局文件代码如下：

```

<?xml version = "1.0" encoding = "utf - 8"?>
<RelativeLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    xmlns:tools = "http://schemas.android.com/tools"
    android:layout_width = "match_parent"
    android:layout_height = "match_parent"
    android:paddingBottom = "@dimen/activity_vertical_margin"
    android:paddingLeft = "@dimen/activity_horizontal_margin"
    android:paddingRight = "@dimen/activity_horizontal_margin"
    android:paddingTop = "@dimen/activity_vertical_margin"
    tools:context = "com.a51work6.helloandroid.MainActivity">

    <TextView
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:text = "Hello World! " />
</RelativeLayout>

```

RelativeLayout 说明当前界面布局是相对布局，TextView 声明一个标签视图，具体内容将在后面的章节详细介绍。界面布局文件 activity_main.xml 可以使用文本工具打开，Android Studio 提供界面设计工具如图 3-10 所示，在界面设计工具中可以通过拖曳视图到设计窗口实现界面设计。

提示 在界面设计窗口的左下角有两个标签——Design 和 Text，单击 Text 标签可以切换到 Xml 文本编辑窗口。

3.2.5 AndroidManifest.xml 文件

Android 的每个应用都必须包含一个 AndroidManifest.xml 清单文件，清单文件提供

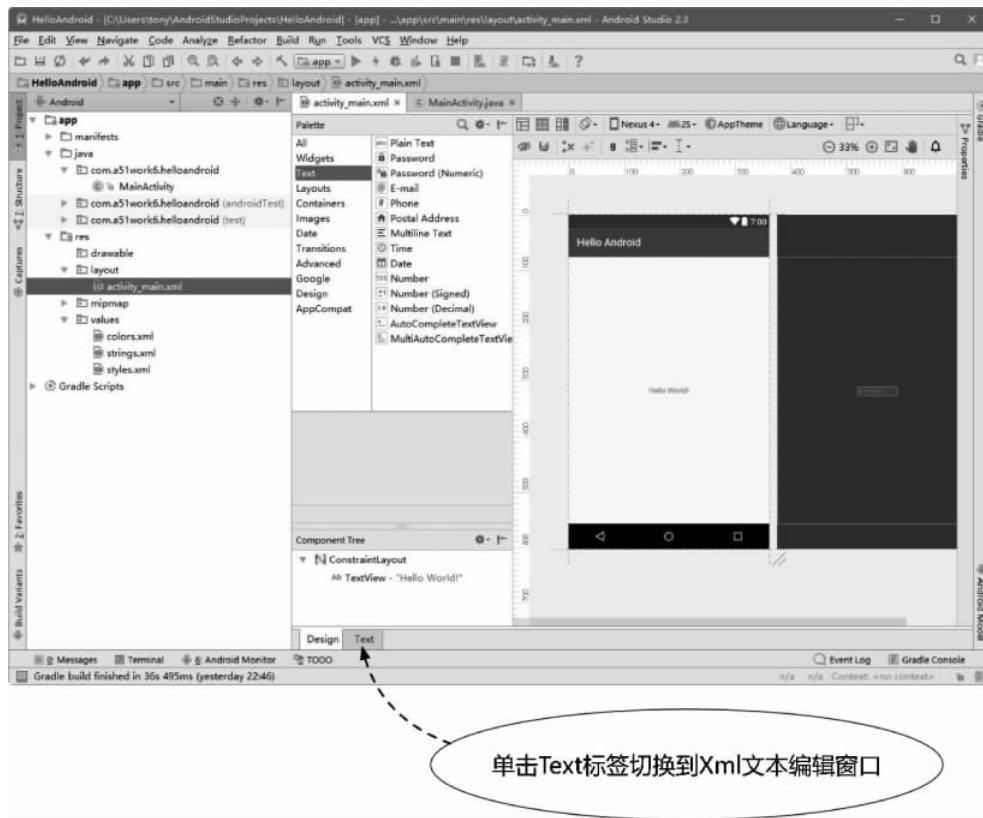


图 3-10 Android Studio 提供界面设计工具

有关当前应用的基本信息,Android 系统必须获得这些信息才能运行该应用。清单文件描述的内容如下:

- 声明应用的 Java 源代码包名,包名非常重要,它是应用的唯一标识符。
- 描述应用中的组件,即 Activity(活动)、Service(服务)、Broadcast Receiver(广播接收器)和 Content Provider(内容提供者)。
- 声明应用必须具备的权限,例如应用中使用到的服务权限(如 GPS 服务、互联网服务和短信服务等等)。
- 声明应用所需的最低 Android API 级别。
- 声明应用的安全控制和测试等信息。

注意 在 Android Studio 工程中,AndroidManifest.xml 位于 manifests 根目录下;而在操作系统(如 Windows 的资源管理器)中,AndroidManifest.xml 位于应用的根目录下,图 3-9 所示的 app/src/main 目录是应用的根目录。

AndroidManifest.xml 文件代码如下:

```

<?xml version = "1.0" encoding = "utf - 8"?>
<manifest xmlns:android = "http://schemas.android.com/apk/res/android"
    package = "com.a51work6.helloandroid"> ①

    < application
        android:allowBackup = "true"
        android:icon = "@mipmap/ic_launcher" ②
        android:label = "@string/app_name" ③
        android:supportsRtl = "true" ④
        android:theme = "@style/AppTheme">
        < activity android:name = ".MainActivity">
            < intent - filter >
                < action android:name = "android.intent.action.MAIN" /> ⑦

                < category android:name = "android.intent.category.LAUNCHER" /> ⑧
            </intent - filter >
        </activity>
    </application>

</manifest>

```

代码第①行 package = "com.a51work6.helloandroid" 是声明应用的 Java 源代码包名。清单文件中的组件声明是在标签< application >和</application >之间添加的。代码第②行 android:icon = "@ mipmap/ic_launcher" 是设置应用图标，@ mipmap/ic_launcher 是引用 res/mipmap 目录中的 ic_launcher. png 图片文件。代码第③行 android:label = "@ string/app_name" 是声明应用名，@ string/app_name 是引用 res/values/strings. xml 文件中的< string name = "app_name" ></string>标签中的内容。strings. xml 代码如下：

```

<resources>
    < string name = "app_name" > Hello Android </string>
</resources>

```

AndroidManifest. xml 文件代码第④行 android:supportsRtl = "true" 是声明应用支持从右往左书写语言习惯(主要是阿拉伯语和希伯来语)。代码第⑤行是声明应用主题为 AppTheme。

代码第⑥行声明活动组件，在活动中可以声明 Intent Filter(意图过滤器)，组件通过意图过滤器实现响应 Intent(意图)，Android 系统启动某个组件之前，需要了解该组件要处理哪些意图。清单文件中的组件声明是在标签< intent-filter >和</intent-filter >之间添加的，代码第⑦行和第⑧行是声明当前活动是主屏幕启动的活动，即应用启动的第一个界面。

3.3 运行工程

创建 Android 工程并编写代码后，就可以运行工程了。运行工程开发在 Android 模拟器或设备上运行。运行 Android Studio 工程以通过如图 3-11 所示的工具栏按钮实现。首

先选择模块(Module),一个工程可包含多个模块,但默认情况下只有一个 app 模块。模块选择完成,就可以单击运行模块按钮运行了,如果要调试程序代码,则可以单击调试模式运行模块按钮。

运行过程中会提示选择在哪个模拟器或设备上运行,如果是在设备上运行,则需要设备连接计算机才可以。如果运行成功,则会看到如图 3-1 所示的界面。



图 3-11 运行工程相关工具栏

3.4 学会使用 Android 开发者社区帮助

在开发 Android 的过程中,应该学会使用 Android 开发帮助,谷歌官方的 Android 开发者社区提供“Android SDK API 文档”、“Android SDK 开发指南”和“Android SDK 案例帮助”。

3.4.1 在线帮助文档

打开 Android 开发者社区网址 <https://developer.android.com/develop/index.html>, 页面如图 3-12 所示,在左边的导航菜单中可以找到这些帮助。

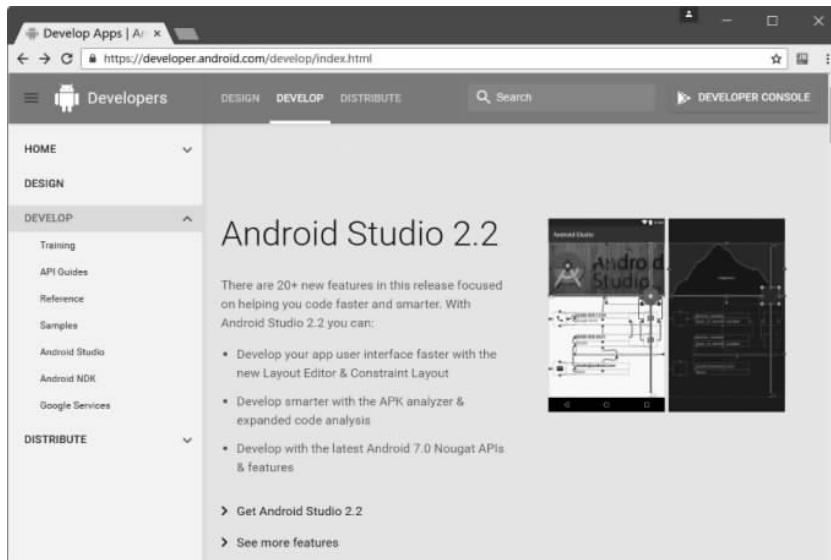


图 3-12 Android 开发者社区

3.4.2 离线帮助文档

如果要使用离线帮助文档,则需要在 SDK Manager 中下载帮助文档。如图 3-13 所示,在 SDK Manager 中选择 Documentation for Android SDK,然后安装。下载安装成功之后,

离线帮助文档会安装在<Android SDK 安装目录>\docs\目录下面，打开<Android SDK 安装目录>/docs/develop/index.html 文件，会看到类似于图 3-12 所示的页面，如图 3-14 所示。遗憾的是，在离线帮助文档中没有 Android SDK samples 信息。

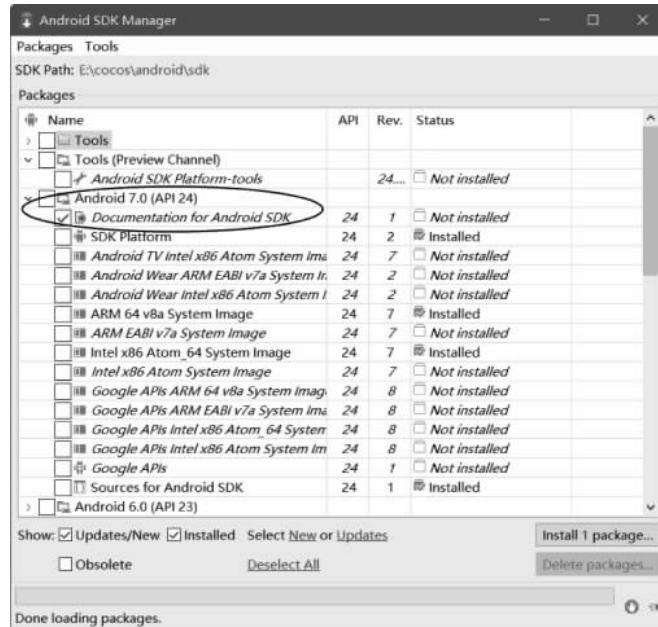


图 3-13 下载帮助文档

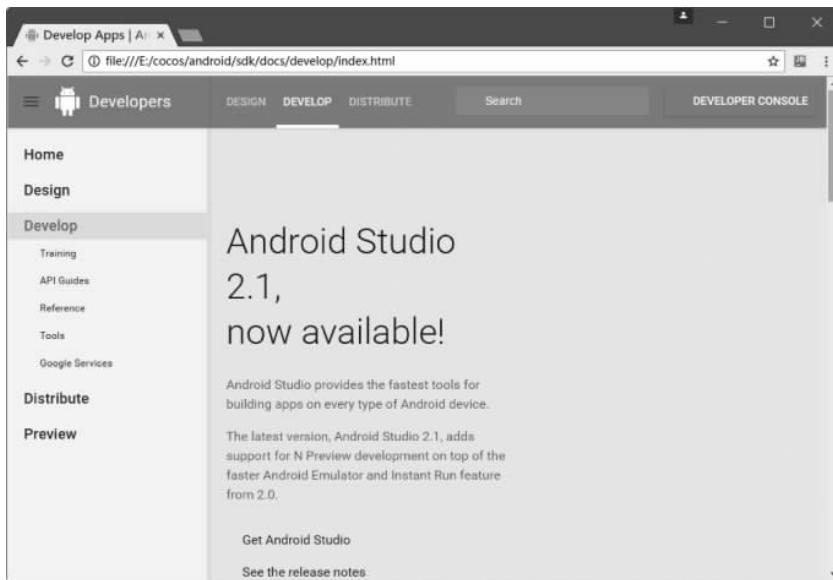


图 3-14 离线帮助文档

3.4.3 Android SDK API 文档

在图 3-12 或图 3-14 所示的页面的左边导航菜单中单击 Reference，打开 Android SDK API 文档会看到如图 3-15 所示的页面。熟悉 Java 的读者应该不陌生，非常类似于 Java 的 API 文档页面，它们的用法完全一样。

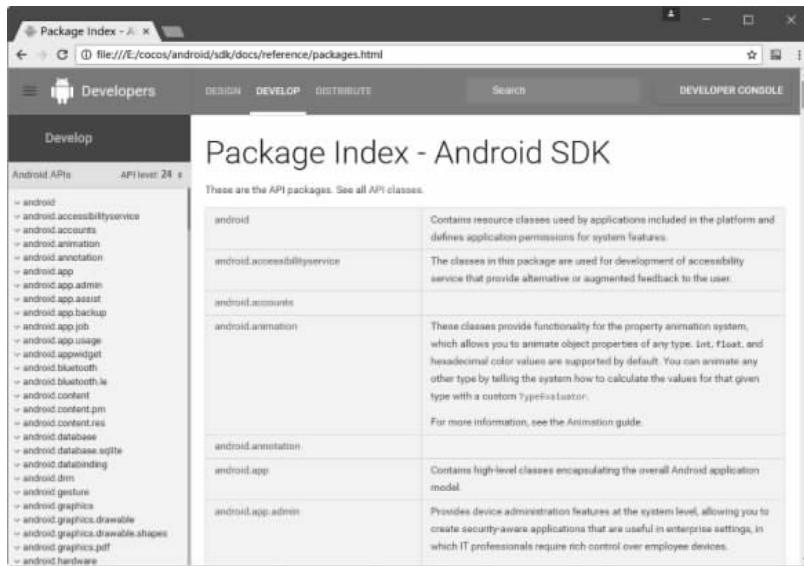


图 3-15 Android SDK API 文档

3.4.4 Android SDK 开发指南

在图 3-12 或图 3-14 所示页面的左边导航菜单中单击 API Guides，打开“Android SDK 开发指南”文档，会看到如图 3-16 所示的页面。

建议读者好好阅读一下这部分内容，在这部分内容中包含了应用开发的各个方面，主要包括框架主题、开发应用、发布应用和最佳实践等几个部分。框架主题包括用户界面相关内容、数据存储、图形技术(2D 和 3D)、意图和意图过滤器、内容提供者、多媒体、访问安全限制、蓝牙等。

3.4.5 使用 Android SDK 案例

谷歌提供了一些 Android SDK 案例，在 Android 4 之前可以通过 SDK Manager 下载，现在已经不再提供下载了。谷歌推荐现在使用 Android Studio 工具直接从 GitHub^① (<https://github.com/googlesamples/>) 导入。

^① GitHub 是一个通过 Git 进行版本控制的软件源代码托管服务。

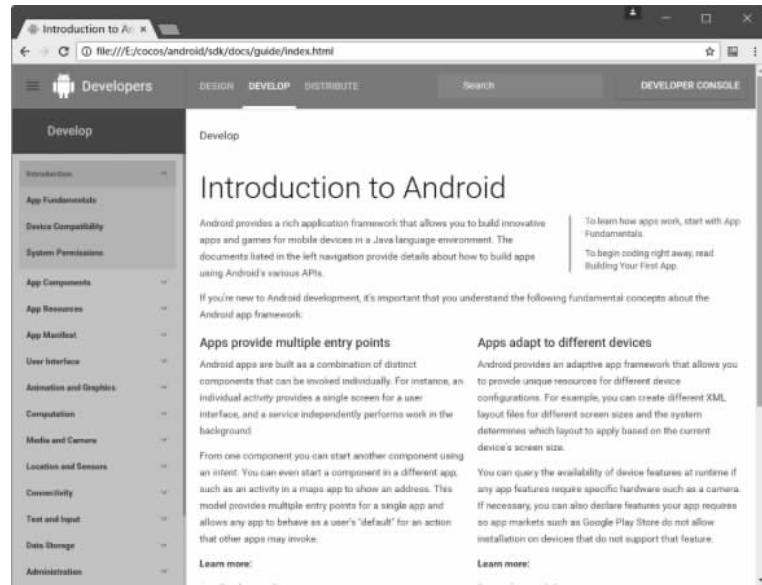


图 3-16 Android SDK 开发指南

在 Android Studio 的欢迎界面单击 Import an Android code sample 可以导入案例，在如图 3-17 所示的对话框中，选择自己需要的案例，单击 Next 按钮，进入如图 3-18 所示的对话框，在此可以选择下载之后目录，然后单击 Finish 按钮就可导入了。

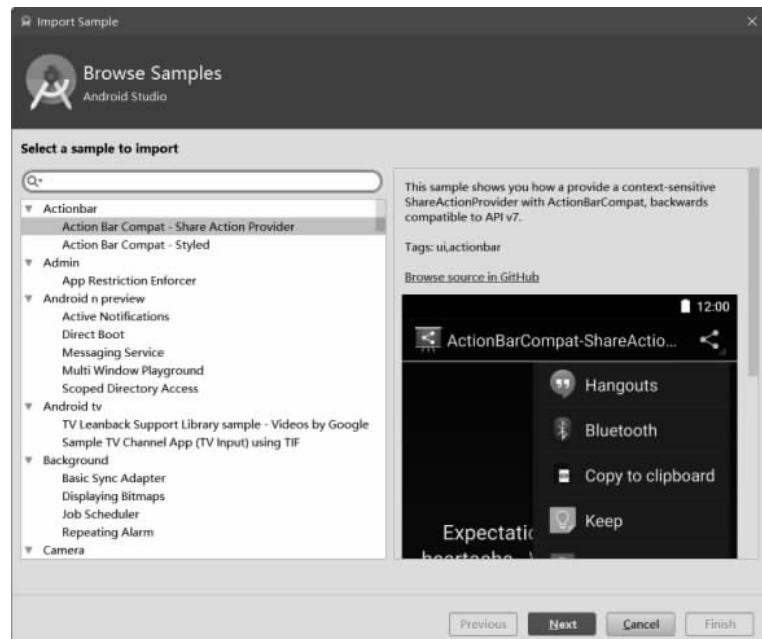


图 3-17 官方案例

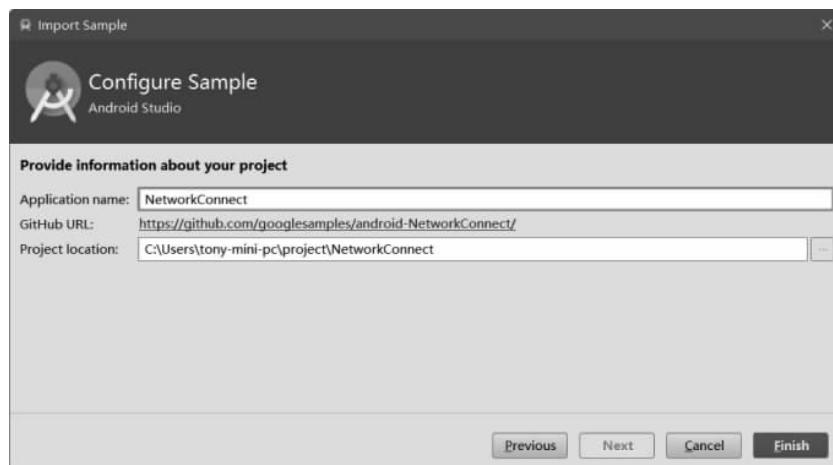


图 3-18 导入官方案例

提示 由于网络原因,有时无法连接 GitHub,需要在 Android Studio 中设置 HTTP 代理。打开 Android Studio 菜单,选择 File→Settings,打开如图 3-19 所示的对话框,在 HTTP Proxy 中选中 Auto-detect proxy settings,这样可以下载过程动态查找 HTTP 代理。

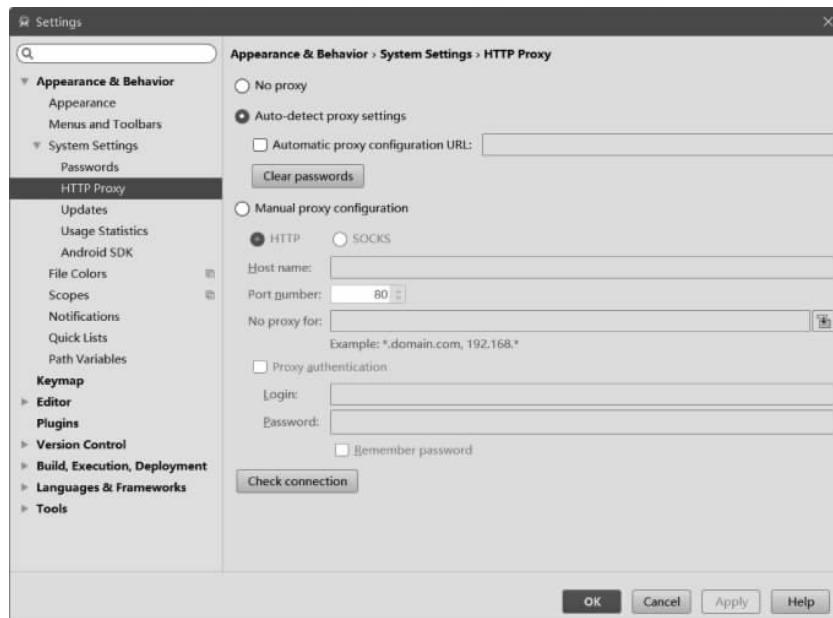


图 3-19 设置 HTTP 代理

本章总结

本章首先介绍 Android Studio 工具的 Android 工程，然后介绍 Android 工程目录结构和一些重要的文件，以及如何通过 Android Studio 运行 Android 工程。最后介绍了如何使用 Android 开发者社区帮助。