

广告彩铃培训笔记

1 开发环境搭建

1.1 JDK1.8安装

以下以window系统安装java环境为例

下载JDK

首先需要下载java开发工具包JDK1.8，下载地址：






<http://www.oracle.com/technetwork/java/javase/downloads/index.html>，点击如下下载按钮：










Java SE 8u241

Java SE 8u241 includes important bug fixes. Oracle strongly recommends that all Java SE 8 users upgrade to this release.

- Documentation
- Installation Instructions
- Release Notes
- Oracle License
 - Binary License
 - Documentation License
 - BSD License
- Java SE Licensing Information User Manual
 - Includes Third Party Licenses
- Certified System Configurations
- Readme Files
 - JDK ReadMe
 - JRE ReadMe

Oracle JDK

-  JDK Download
-  Server JRE Download
-  JRE Download
-  Documentation Download
-  Demos and Samples Download

Linux x64 RPM Package	170.65 MB	 jdk-8u241-linux-x64.rpm
Linux x64 Compressed Archive	185.53 MB	 jdk-8u241-linux-x64.tar.gz
macOS x64	254.06 MB	 jdk-8u241-macosx-x64.dmg
Solaris SPARC 64-bit (SVR4 package)	133.01 MB	 jdk-8u241-solaris-sparcv9.tar.Z
Solaris SPARC 64-bit	94.24 MB	 jdk-8u241-solaris-sparcv9.tar.gz
Solaris x64 (SVR4 package)	133.8 MB	 jdk-8u241-solaris-x64.tar.Z
Solaris x64	92.01 MB	 jdk-8u241-solaris-x64.tar.gz
Windows x86	200.86 MB	 jdk-8u241-windows-i586.exe
Windows x64	210.92 MB	 jdk-8u241-windows-x64.exe

下载后JDK的安装根据提示进行，还有安装JDK的时候也会安装JRE，一并安装就可以了。

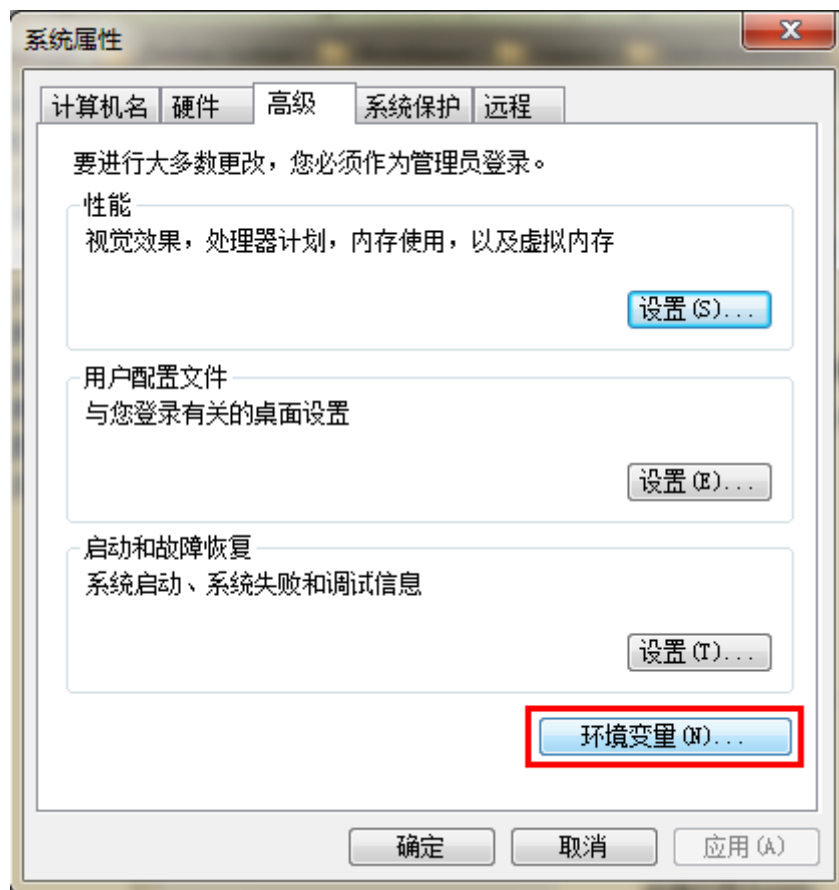
安装JDK，安装过程中可以自定义安装目录等信息，例如我们选择安装目录为 **C:\Program Files (x86)\Java\jdk1.8.0_241**。

配置环境变量

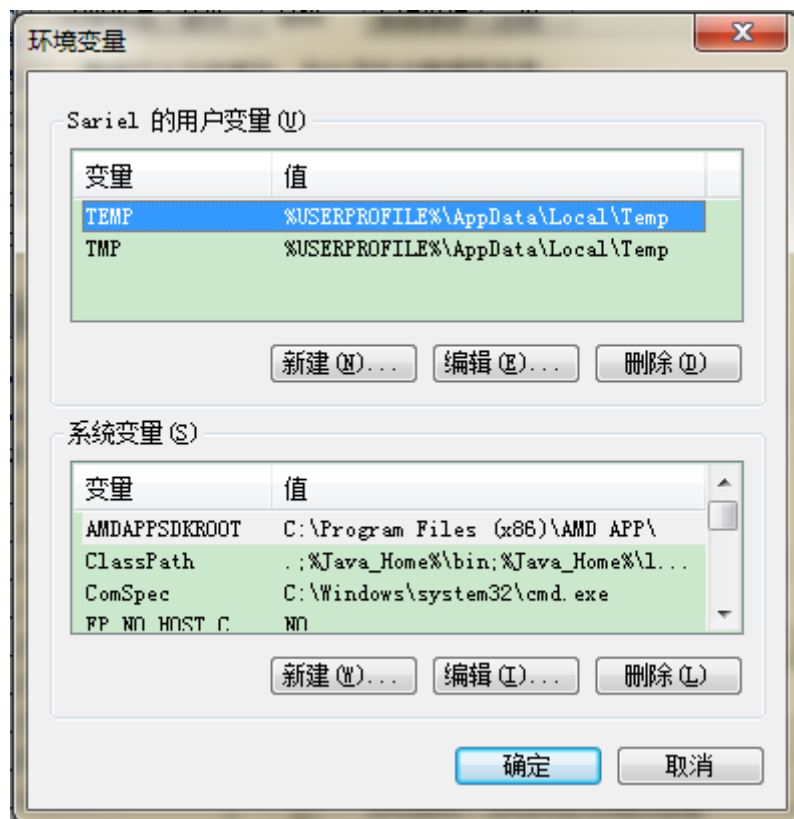
1.安装完成后，右击"我的电脑"，点击"属性"，选择"高级系统设置"；



2.选择"高级"选项卡，点击"环境变量"；



然后就会出现如下图所示的画面：



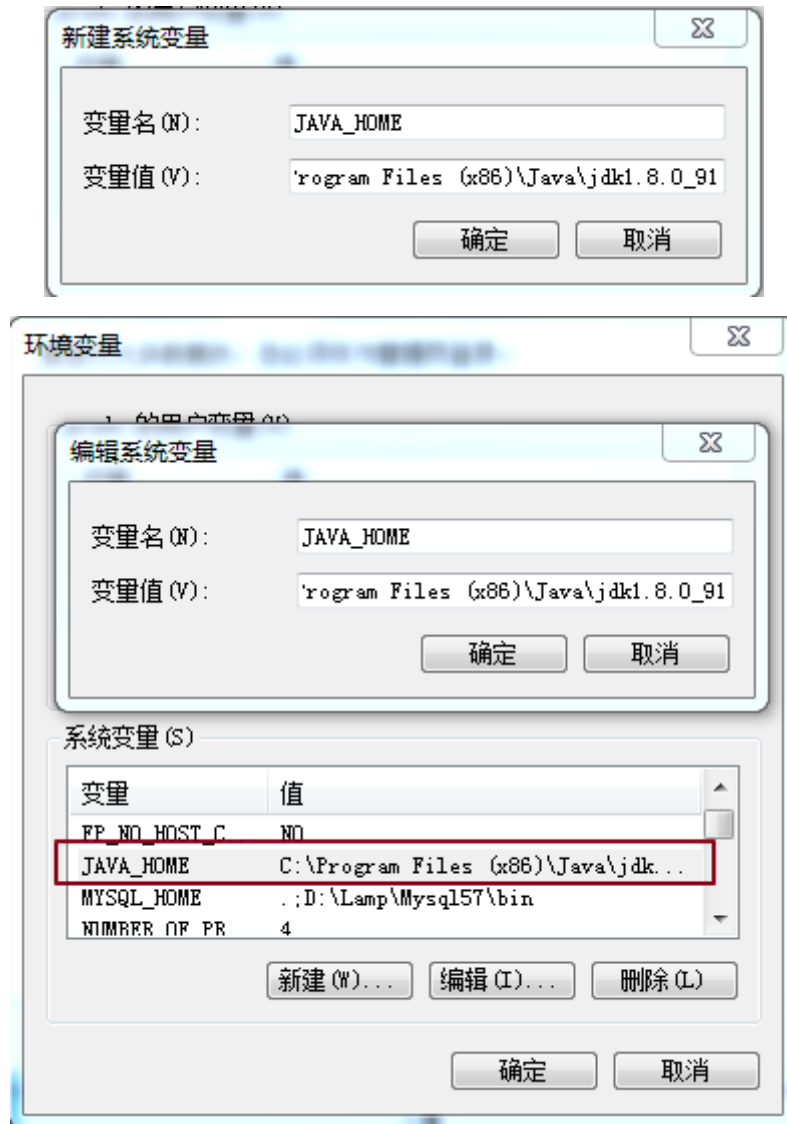
在 "系统变量" 中设置 3 项属性, JAVA_HOME、PATH、CLASSPATH(大小写无所谓),若已存在则点击"编辑", 不存在则点击"新建".

注意：如果使用 1.5 以上版本的 JDK, 不用设置 CLASSPATH 环境变量, 也可以正常编译和运行 Java 程序。

变量设置参数如下：

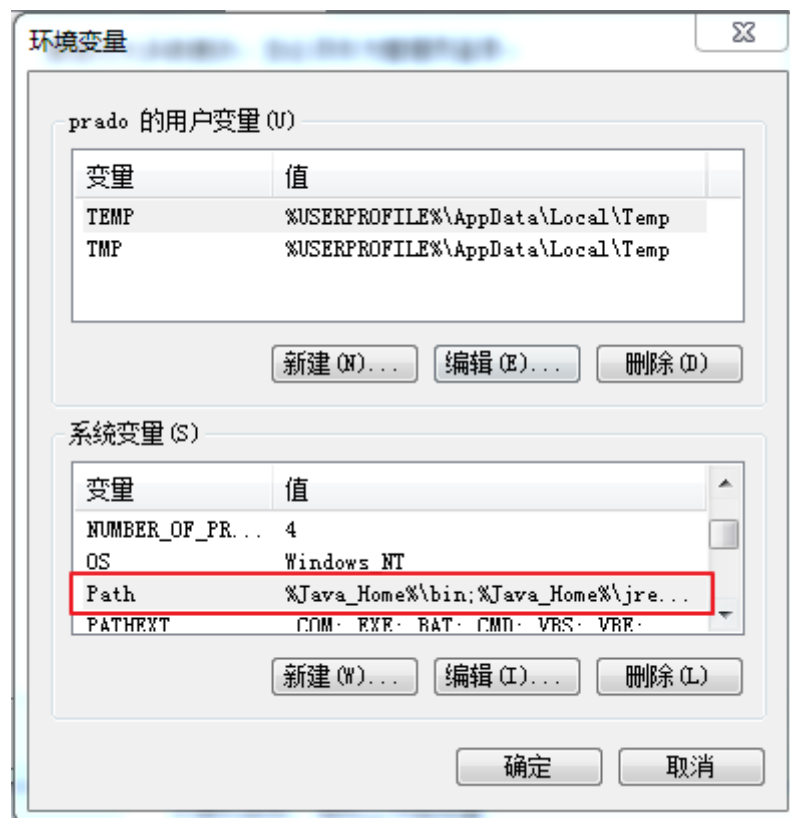
- 变量名: JAVA_HOME
- 变量值: C:\Program Files (x86)\Java\jdk1.8.0_91 // 要根据自己的实际路径配置
- 变量名: CLASSPATH
- 变量值: .;%JAVA_HOME%\lib\dt.jar;%JAVA_HOME%\lib\tools.jar; //记得前面有个"."
- 变量名: Path
- 变量值: %JAVA_HOME%\bin;%JAVA_HOME%\jre\bin;

JAVA_HOME 设置



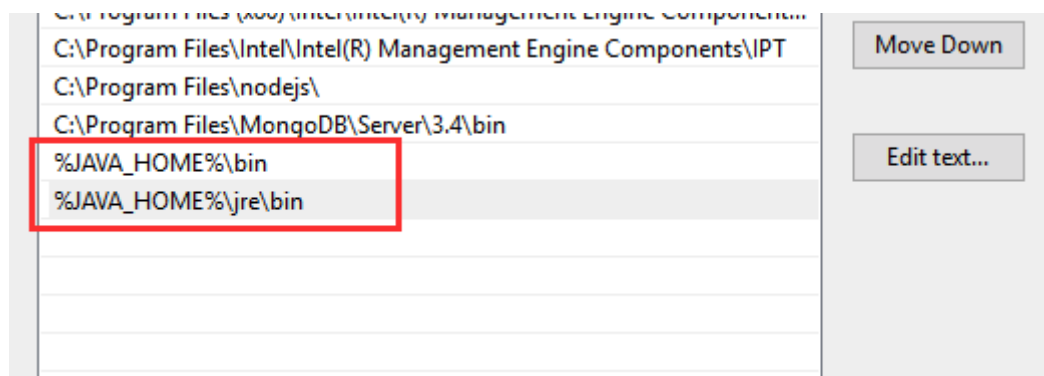
PATH设置





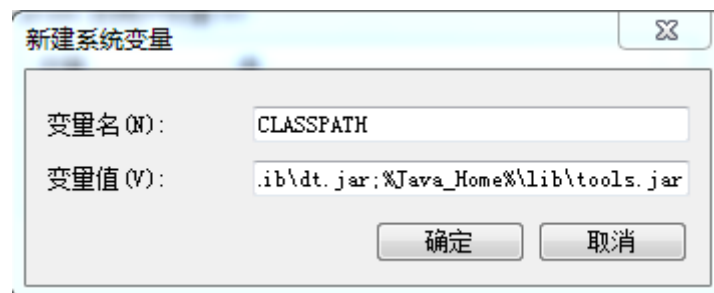
注意：在 Windows10 中，Path 变量里是分条显示的，我们需要将
%JAVA_HOME%\bin;%JAVA_HOME%\jre\bin; 分开添加，否则无法识别：

```
%JAVA_HOME%\bin;
%JAVA_HOME%\jre\bin;
```



更多内容可参考：[Windows 10 配置Java 环境变量](#)

CLASSPATH 设置



这是 Java 的环境配置，配置完成后，你可以启动 Eclipse 来编写代码，它会自动完成java环境的配置。

测试JDK是否安装成功

- 1、"开始"->"运行"，键入"cmd"；
- 2、键入命令: **java -version**、**java**、**javac** 几个命令，出现以下信息，说明环境变量配置成功；

```
C:\Users\prado>java -version
java version "1.8.0_91"
Java(TM) SE Runtime Environment (build 1.8.0_91-b14)
Java HotSpot(TM) 64-Bit Server VM (build 25.91-b14, mixed mode)
```

1.2 Android studio 安装

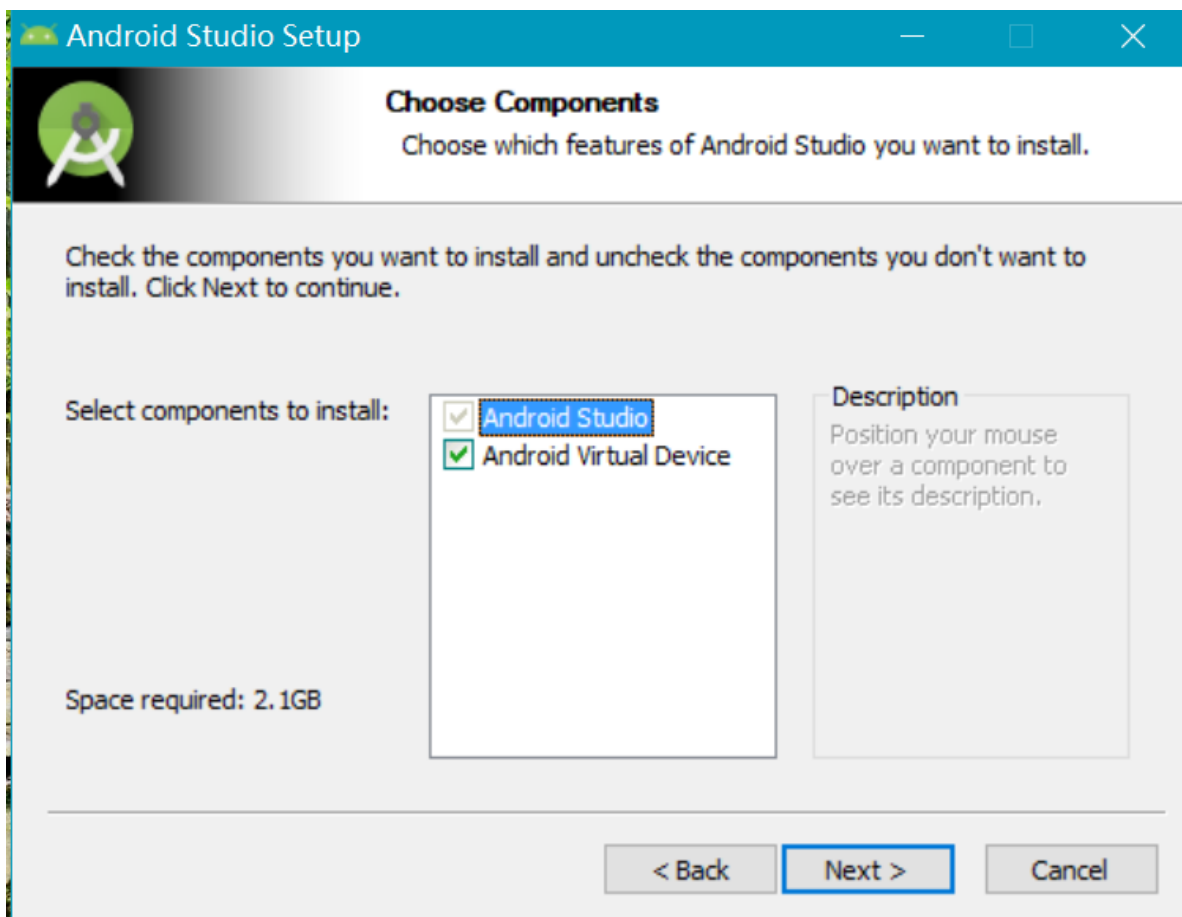
下载地址：

<http://www.android-studio.org/>

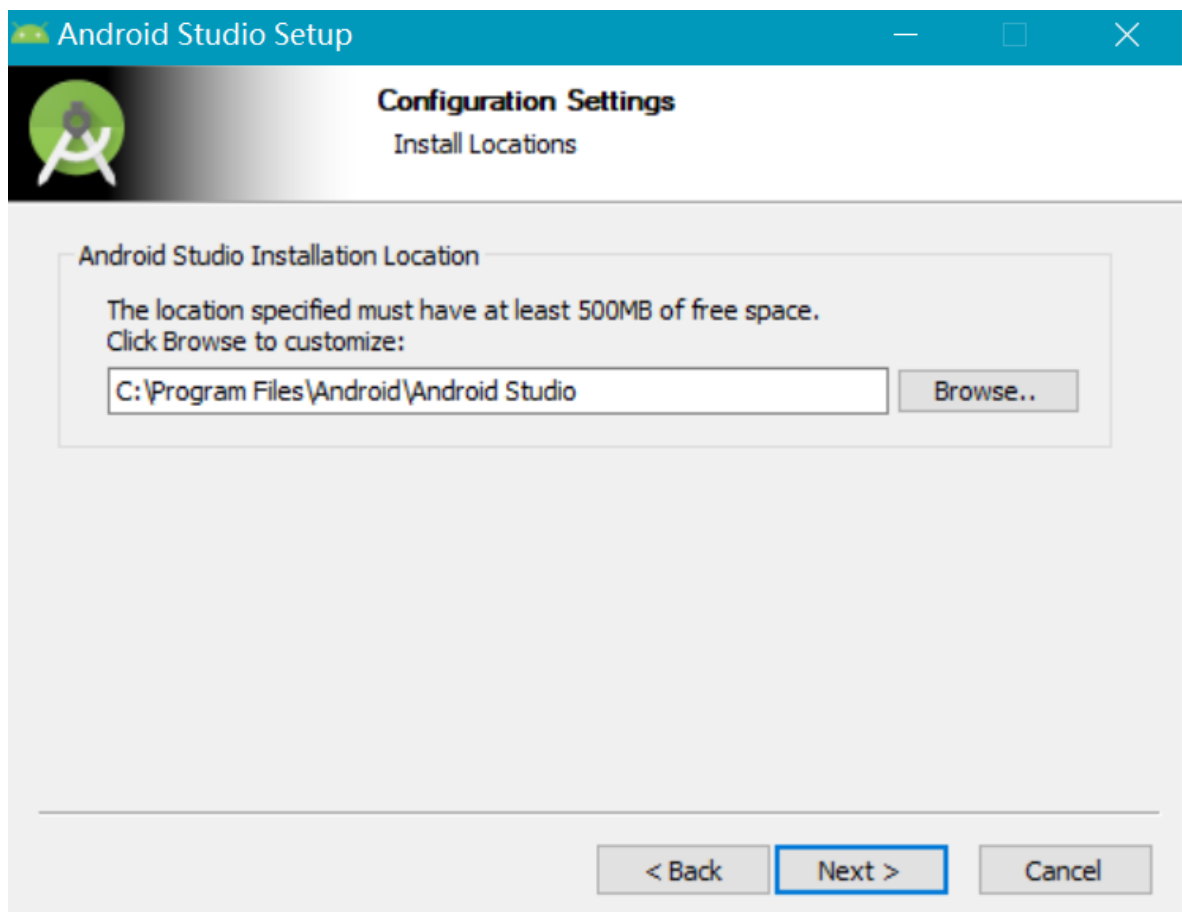
选择安装的插件

第一个是Android Studio主程序，必选。

第三个和第四个是虚拟机和虚拟机的加速程序，如果你要在电脑上使用虚拟机调试程序，就勾上。



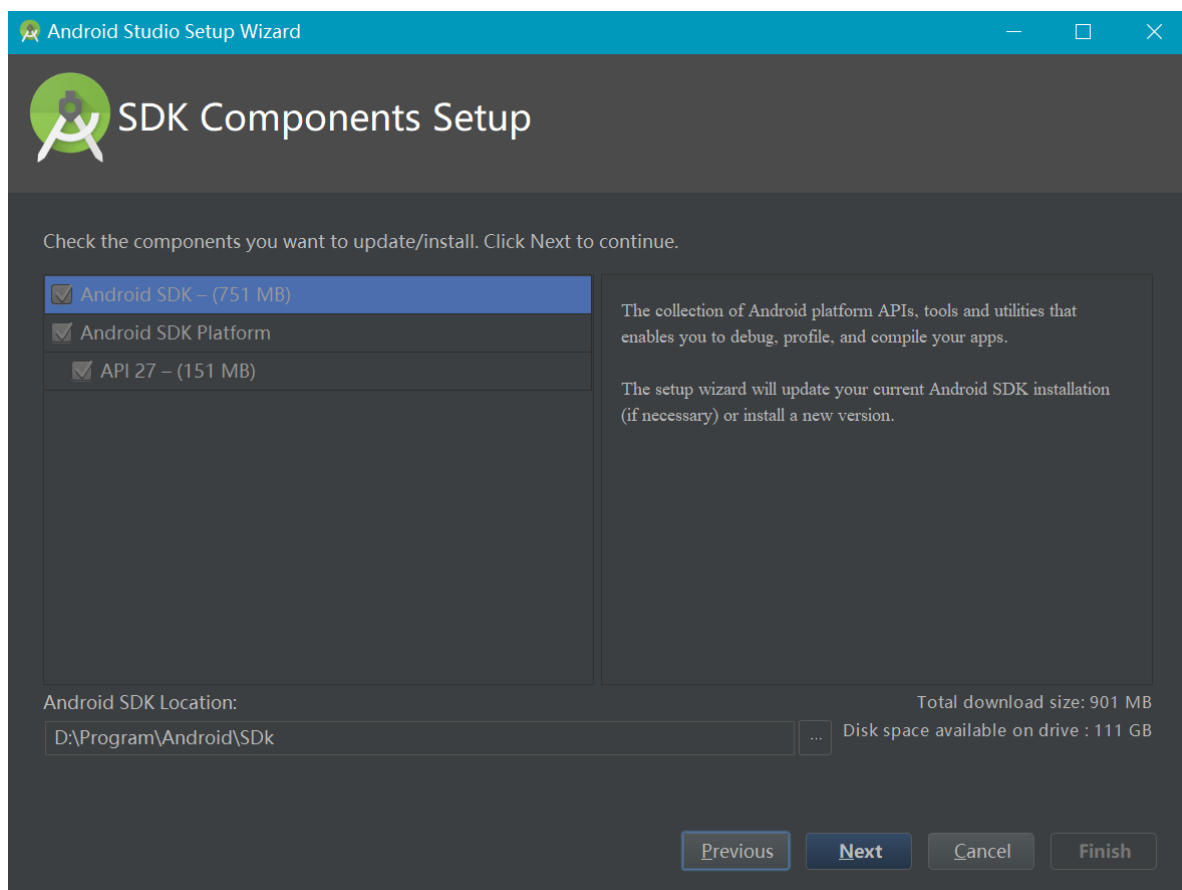
选择Android studio和安装目录



建议自己选择安装路径，放在D盘。

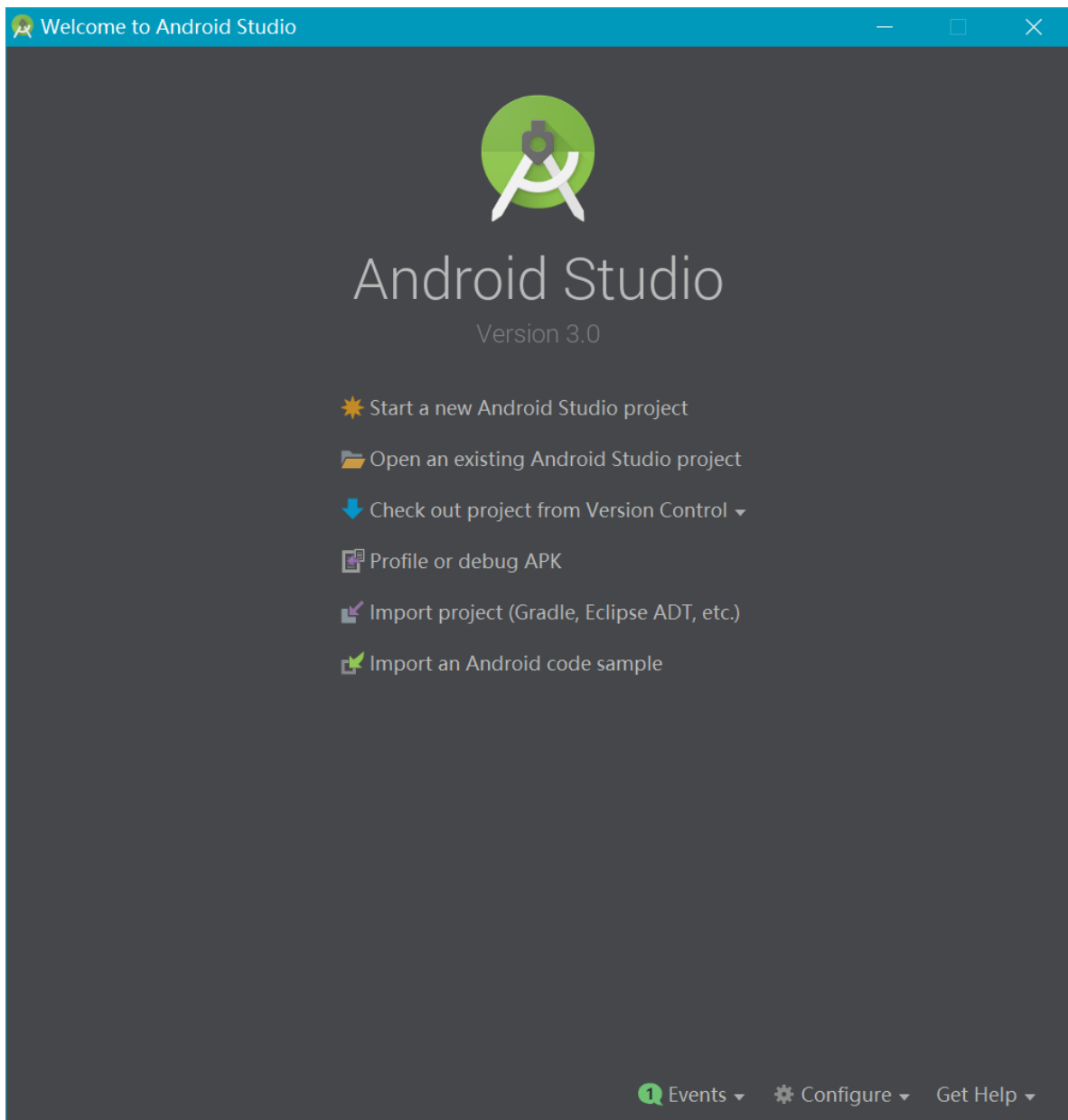
自动安装

剩下和其他安装过程并无不同，不停的下一步就可以。



这一步是要选择的，安装SDK组件，有些教程建议安装完后，SDK单独下载，此处通过Android studio直接安装即可。

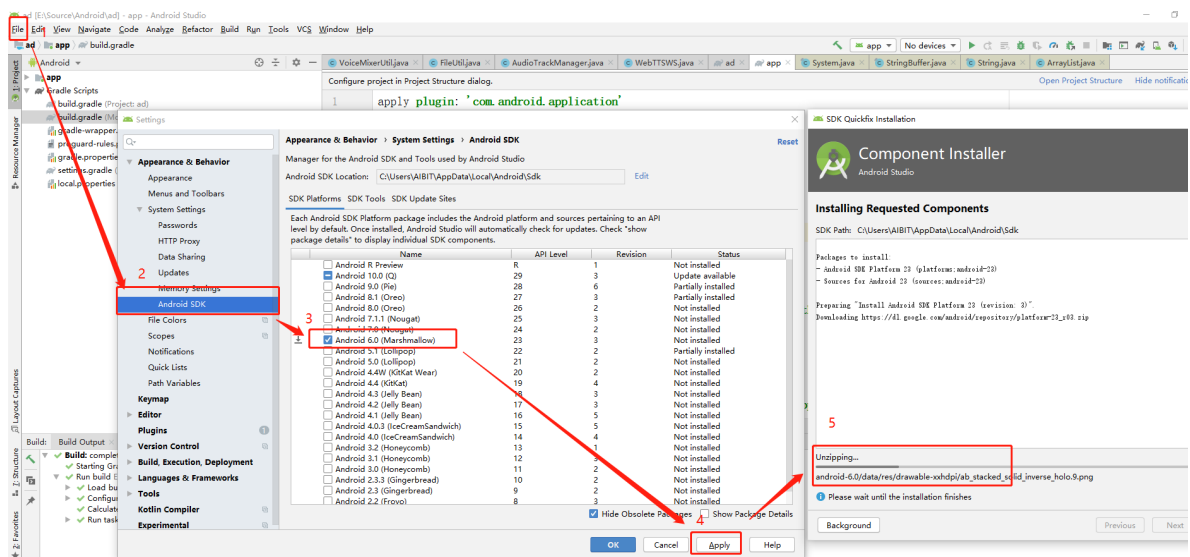
安装成功后界面



可以创建一个新的工程，进行开发了

1.3 Android SDK 下载

如下图所示，打开file，找到settings，在弹出的对话框中选择Android SDK，选择一个目前大部分Android设备都支持的系统，Android 6.0，API版本为23，选择Apply后，等待下载完成，就可以在工程中使用了。



1.4 Android手机或者模拟器环境

1.4.1 手机

1.4.1.1 手机硬件准备

手机系统	手机版本	手机数据线	adb debug功能
Android 系统	6.0 或以上	数据线, 非充电线	支持, 需要用户打开手机的adb debug功能, 一般打开的方式是找到系统版本连续点击7次后, 在设置中找到adb功能打开, 并且需要打开adb debug功能

1.4.1.2 adb常用命令

查看本机连接上的adb设备

```
adb devices
```

进入android设备命令行, 如果有服务没有启动可以使用adb kill-server , adb start-server来重启

```
adb shell
```

安装apk应用

```
adb install -r name.apk
```

adb下载命令: 把/storage/emulated/0/tts.pcm文件拷贝到电脑的当前目录下

```
adb push /storage/emulated/0/tts.pcm .
```

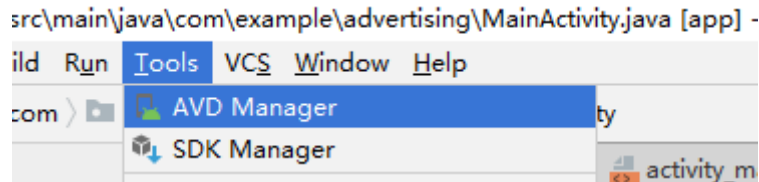
adb上传命令: 把当前目录的test.pcm文件拷贝到手机的/storage/emulated/0/目录下

```
adb pull ./test.pcm /storage/emulated/0/
```

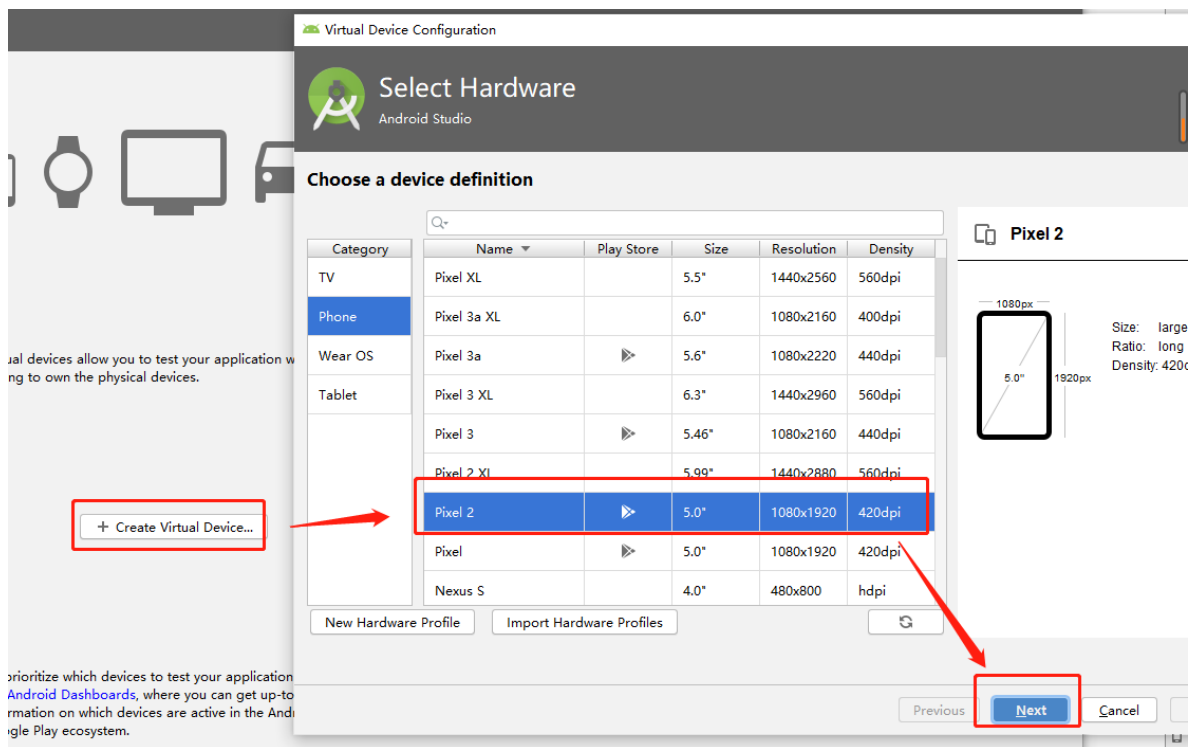
1.4.2 模拟器

模拟器的功能类似android手机的功能，但是有些sensor信号无法模拟，比如重力加速度传感器等

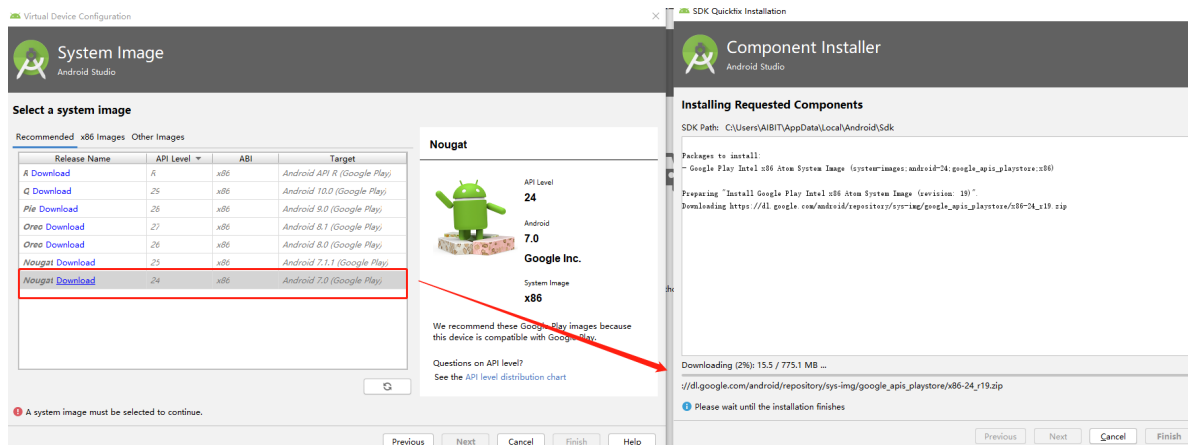
1.4.2.1 点击Tools的AVD Manager创建模拟器



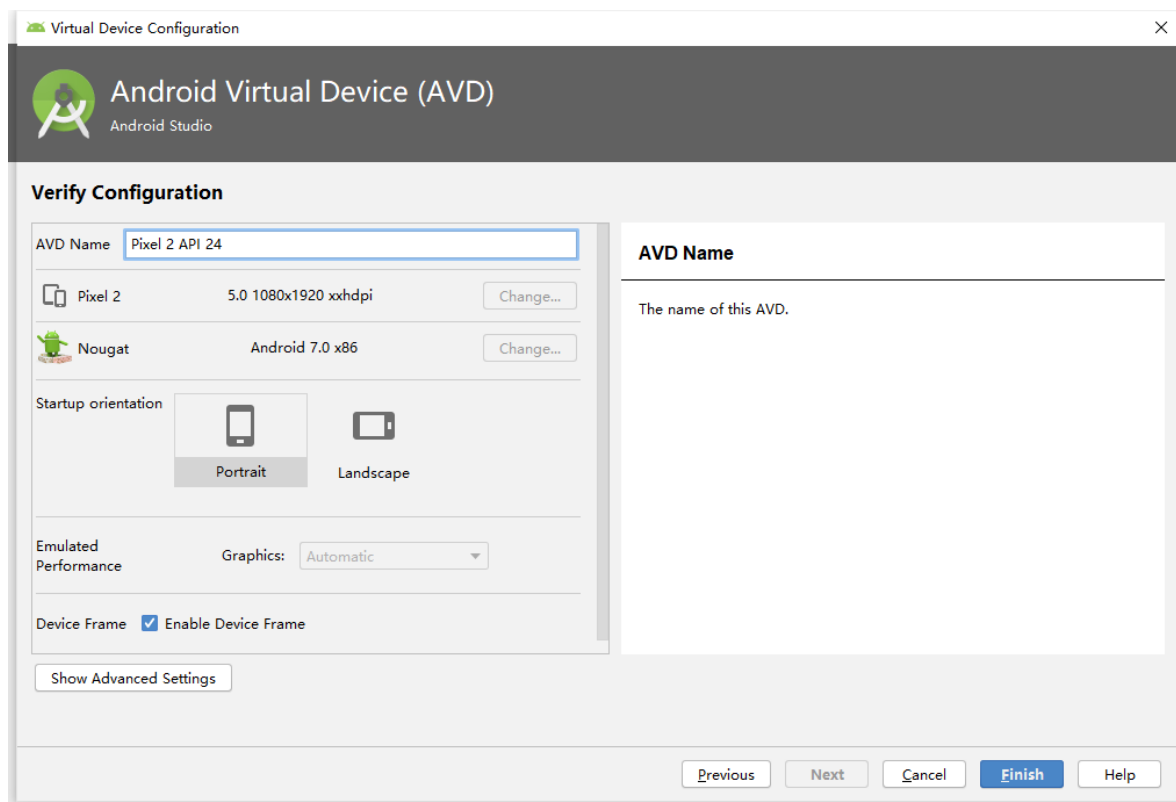
1.4.2.2 选择一个模拟器外观



1.4.2.3 选择模拟器的系统版本，点击download等待下载完成

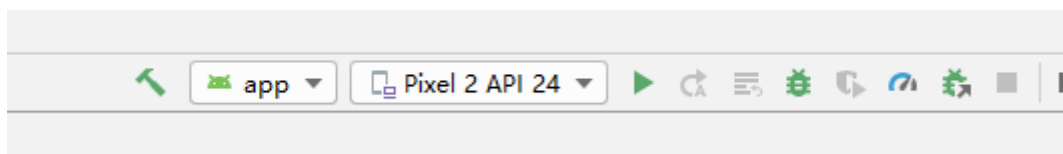


1.4.2.4 选择AVD虚拟机的名称和显示方式后点击finish完成



1.4.2.5 查看虚拟机状态

安装成功后，IDE的运行调试区能够看到虚拟机的名字



1.5 讯飞应用账号建立

调用讯飞AI能力接口，需要注册账号，并且创建应用，如下

1.5.1 注册账号

打开www.xfyun.cn，在右侧点击注册，输入手机号完成注册



1.5.2 创建webapi版本的应用

登录成功后，点击控制台进入应用管理界面，点击创建新应用，创建一个webapi应用，名称可以随便，不要重复就可以

我的应用 > 创建应用

* 应用名称

AIUI的测试应用

* 应用分类

应用-教育学习-学习

* 应用功能描述

AIUI的测试

提交

[返回我的应用](#)

创建成功后，点击创建的应用程序进入应用管理界面，点击在线语音合成一栏，查看每日调用次数是否有问题，另外需要注意右侧的APPID、APISecret、APIKey，在程序中会使用到

讯飞开放平台

平台首页

消息 28

< 返回

AIUI的测试应用

语音识别

语音合成

在线语音合成 (流式版)

离线语音合成 (普通版)

离线语音合成 (高品质版)

实时用量

用量预警通知 已关闭 管理

今日实时服务量

0

剩余服务量

500

购买服务量

历史用量

2020-02-20 - 2020-03-20

日视图

月视图

导出Excel

服务接口认证信息

APPID

APISecret

APIKey

*SDK调用方式只需APPID，APIKey或APISecret适用于WebAPI调用方式。

在线语音合成 (流式版) SDK

SDK名称

版本

2 功能实现

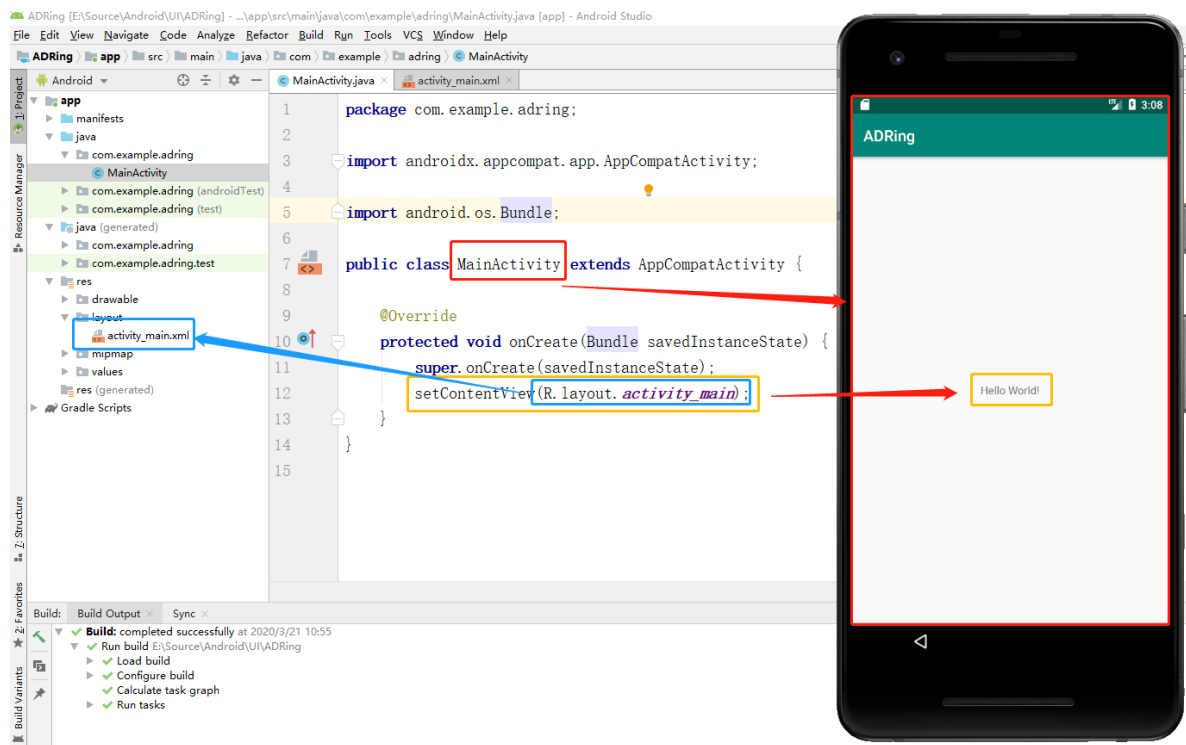
2.1 从0到1创建一个android应用

2.1.1 最终效果



了解Android界面显示的基本原理：

MainActivity.java显示了一个界面，调用setContentView方法显示Hello World! 标签，如果想在界面中配置更多的界面元素，可以通过R.layout.activity_main这个资源索引找到res/layout/activity_main.xml来显示不同的内容。



res/layout/activity_main.xml的布局文件内容如下，其中有一个叫做TextView的文本控件。

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.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=".MainActivity">

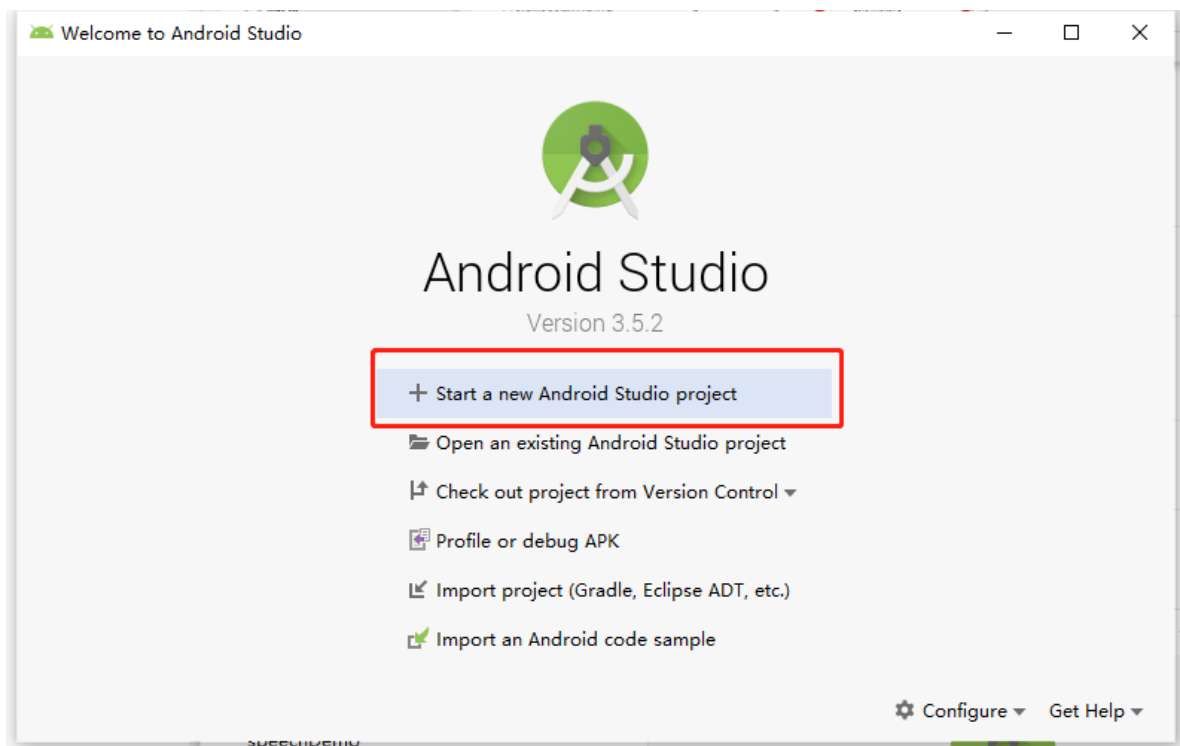
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello World!"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
   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

</androidx.constraintlayout.widget.ConstraintLayout>
```

2.1.2 操作步骤

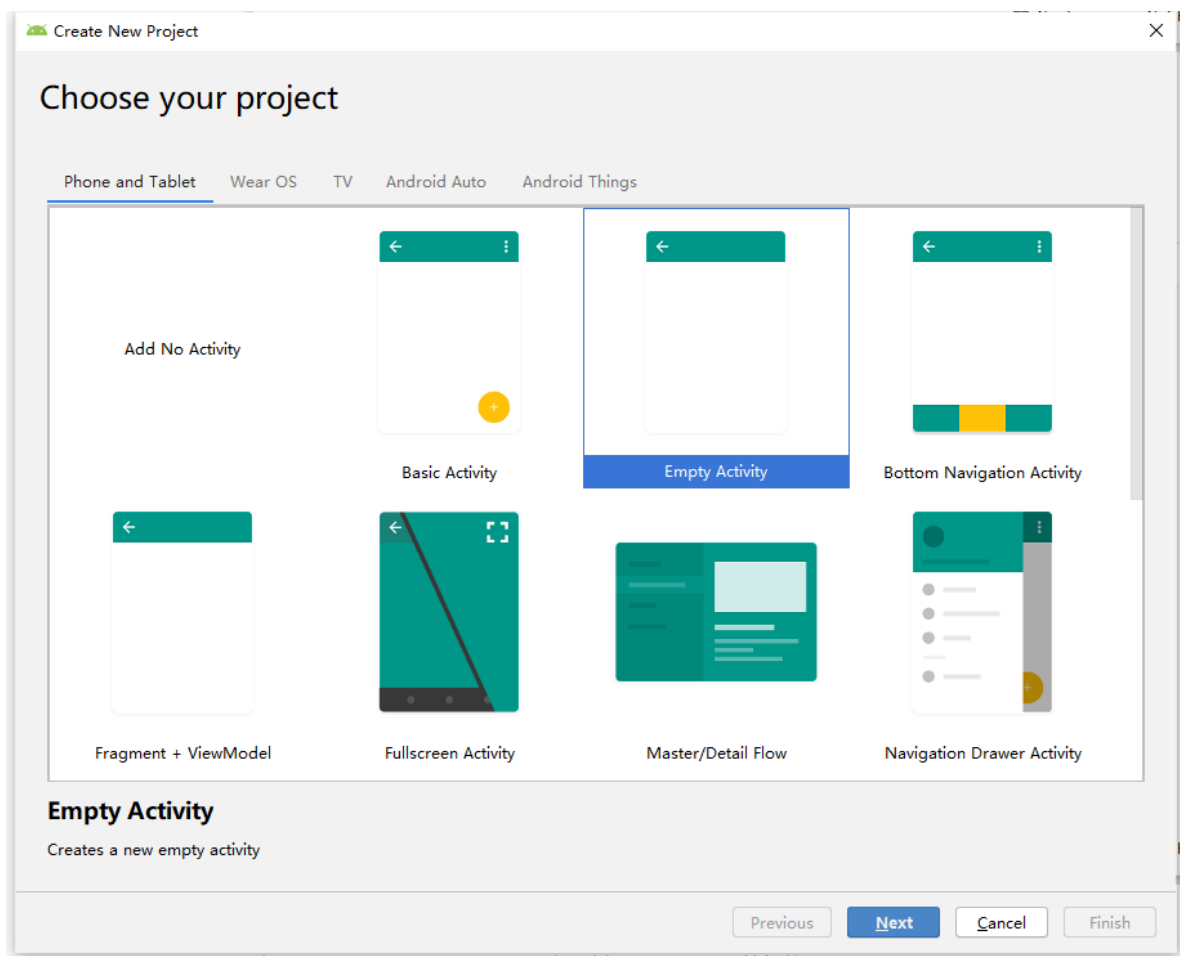
2.1.2.1 创建一个新的android工程

点击start a new android studio project工程



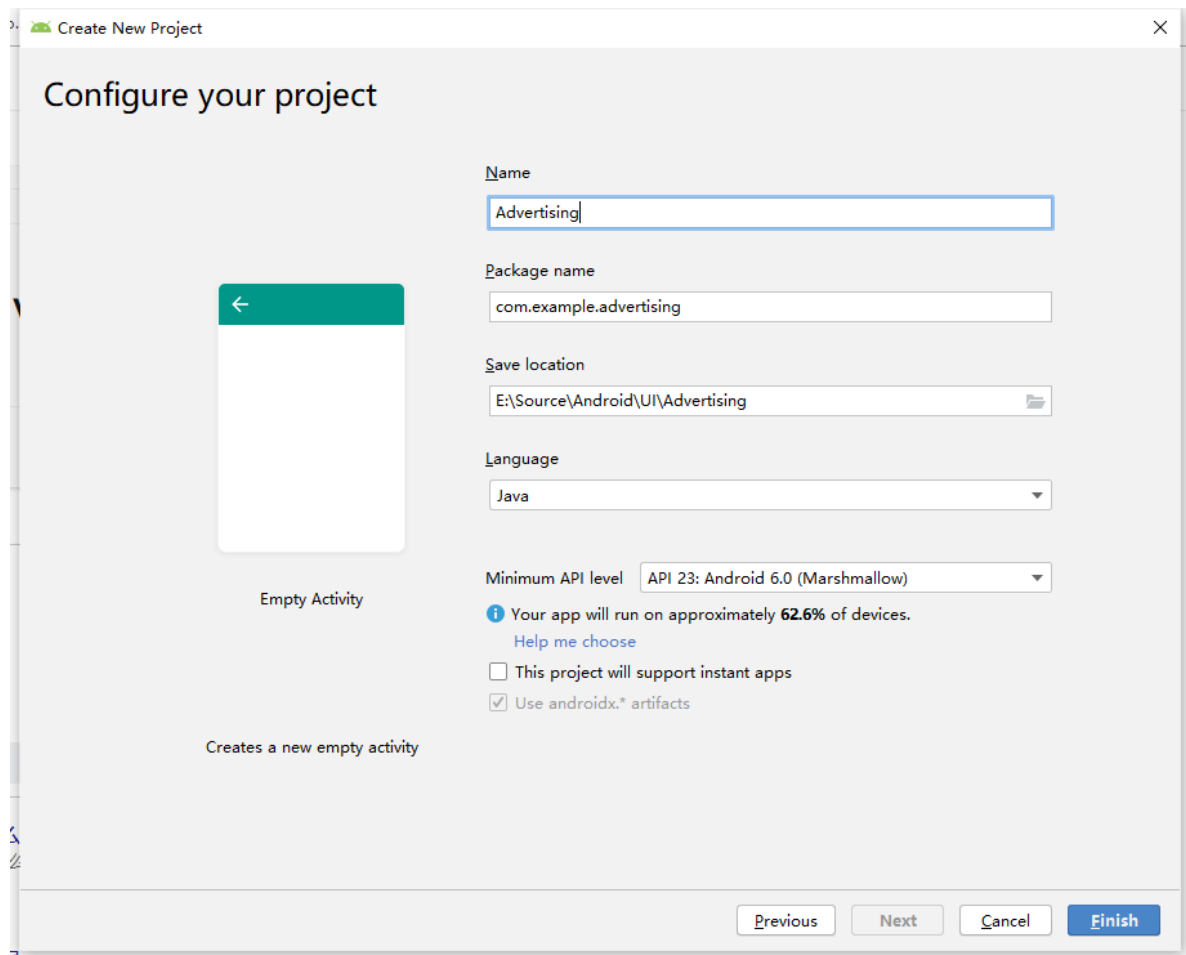
2.1.2.2 选择工程类型

选择empty activity，activity是android中用于显示界面的组件，然后点击next



2.1.2.3 配置工程

在Name中选择一个工程的英文名字，不要带特殊字符，保存的Save location也需要是英文路径，使用的Language默认选择java，API版本推荐Android6.0，兼容性会好一点，然后点击Finish



2.1.2.4 认识IDE界面

工程目录与管理窗口

包含代码gradle编译管理文件、java源码文件、res资源文件、manifest.xml工程管理文件等

代码编辑区

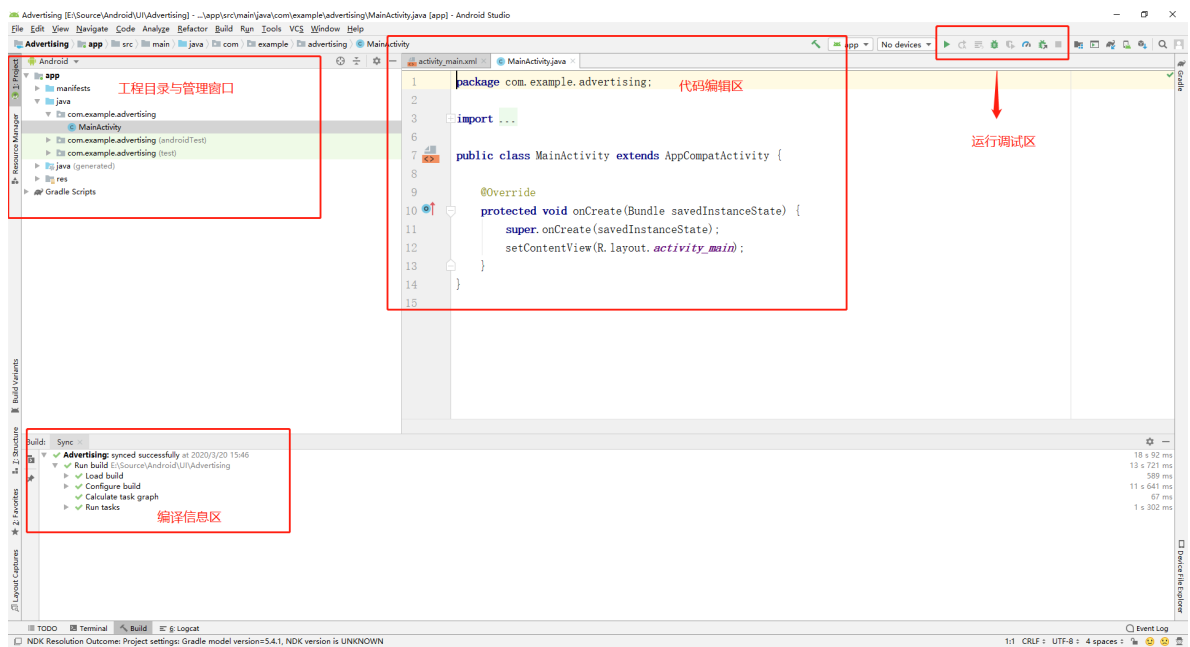
代码编辑区域，也可以编辑界面配置文件xml，用于配置activity的显示

编译信息区

android默认会在后台自动编译工程，注意查看这部分的状态信息，看是否有工程错误的地方

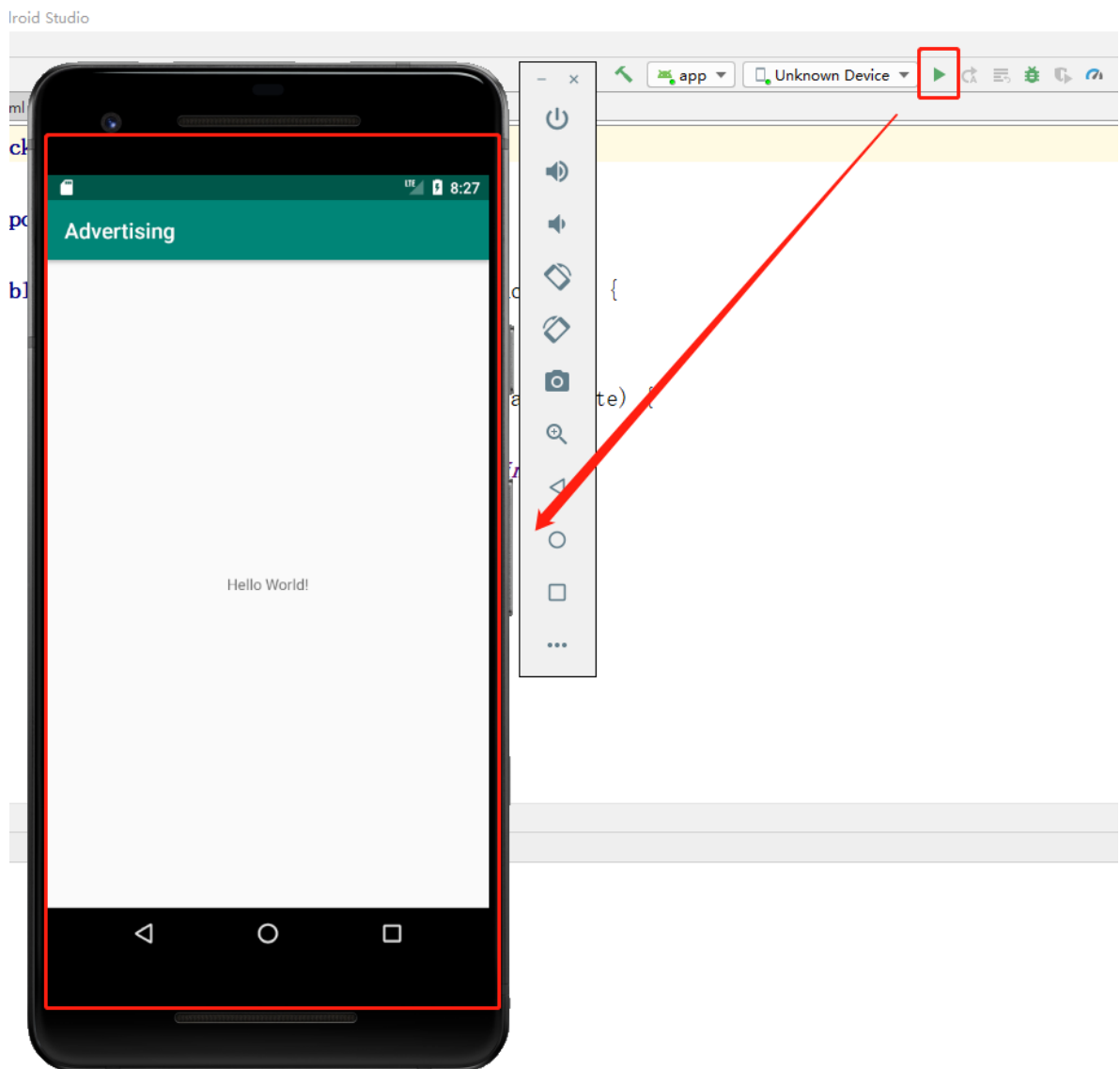
运行调试区

有运行、调试、单步调试等功能，方便代码的运行调试



2.1.2.5 运行程序查看结果

点击右上角的三角运行图标，会在设备上安装当前的应用程序。下图是模拟器，如果使用手机，需要使用usb线连接上手机，并且保证手机的adb debug已经打开，并选择运行的设备。



2.2 从0到1完成一个语音合成应用

2.2.1 演示效果

演示效果的界面部分同上，不同的是完成了重要的功能，讯飞开发平台的语音合成调用，并播放了出来，也是课程中最重要的部分。并且掌握这一节的知识，公司其他的讯飞开放平台能力调用也基本上都能够掌握了。

2.2.2 知识点

- AIUI语音合成文档分析
- websocket okhttp3接口使用
- 常用加密库base64/sha1/md5的使用

2.2.3 掌握技能

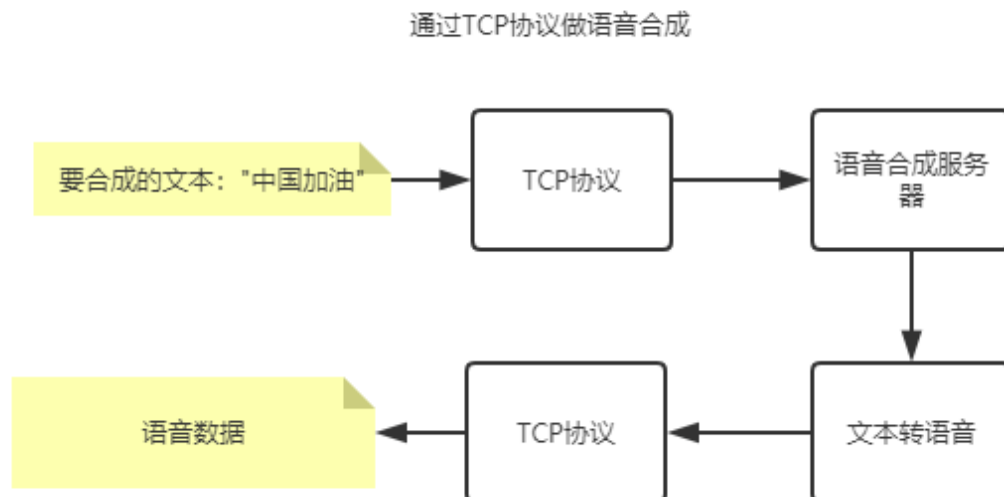
- 能够分析AIUI语音合成文档，完成语音合成的调用

2.2.4 实现

语音合成服务的基本功能是要把用户要合成的文本得到后，转化为语音再传输给客户，使用的协议可以有TCP/HTTP/Websocket等。

2.2.4.1 TCP方式的语音合成

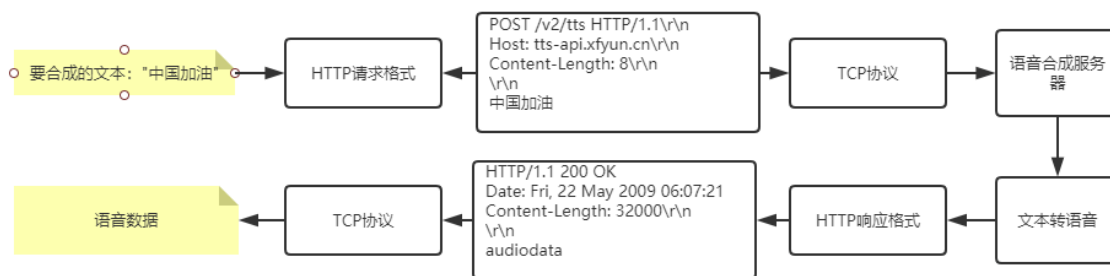
使用TCP的问题是不能只发送"中国加油",没有控制部分，另外tcp是流，不保证数据的单次接收完整。比如说你发的是"中国加油"，而服务端可能收到两次，第一次收到"中国",第二次收到"加油"，会导致服务端只合成部分音频信息，解决这个问题必须定义TCP私有业务协议，比如最简单的length+json+checksum。



2.2.4.2 HTTP方式的语音合成

使用这种方式是满足要求的，百度语音合成目前使用的是HTTP的接口方式。

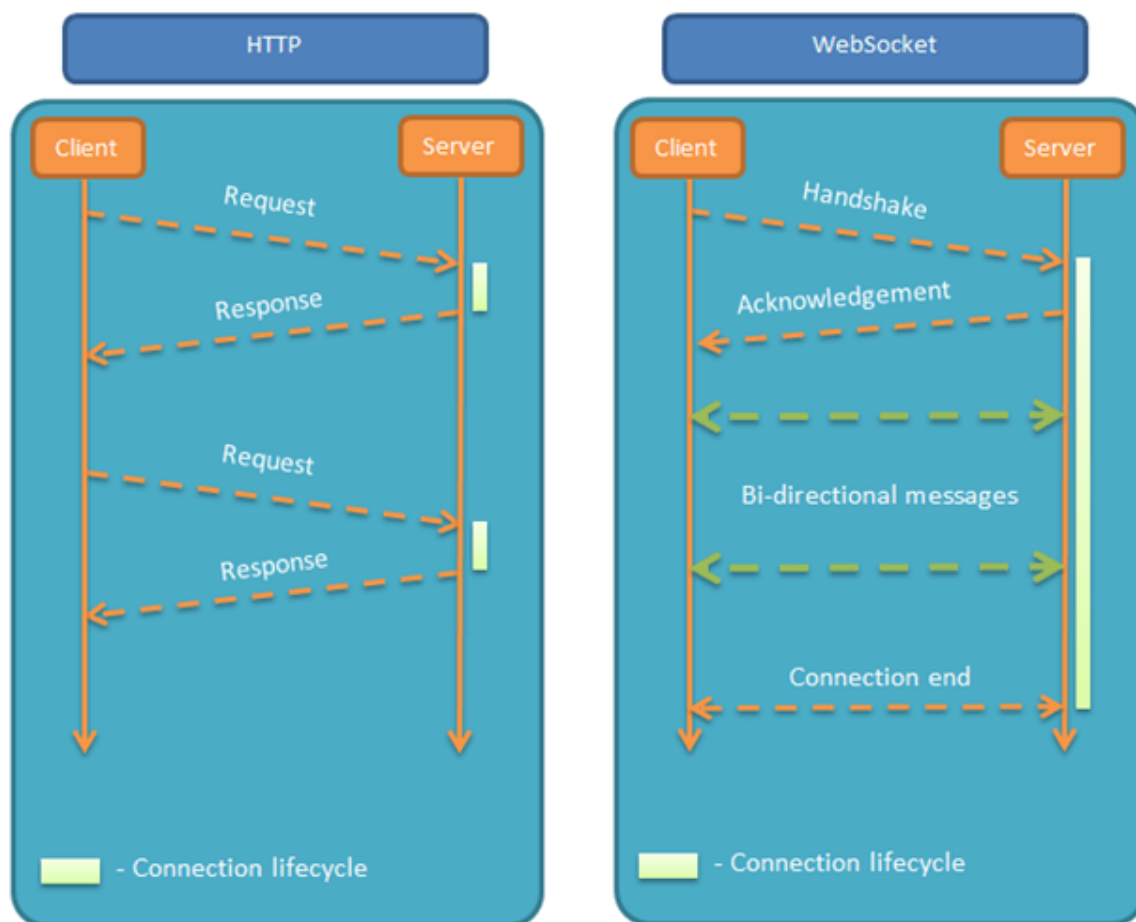
通过HTTP协议做语音合成



2.2.4.3 Websocket方式

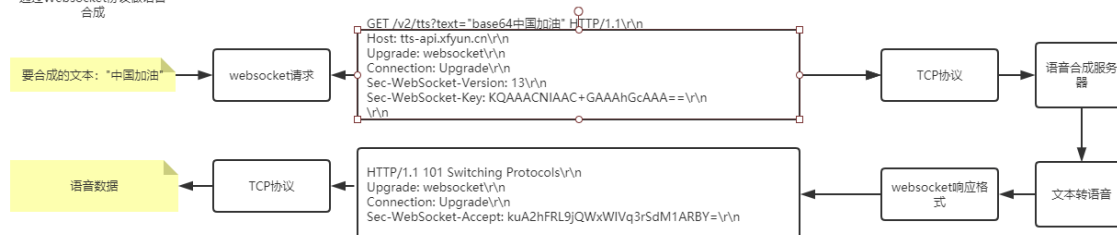
这种方式可以多次交互，讯飞开放平台目前使用的是Websocket的接口方式。

首先看下http和websocket在传输过程中的对比图，会发现websocket是多次交互，http仅一次交互



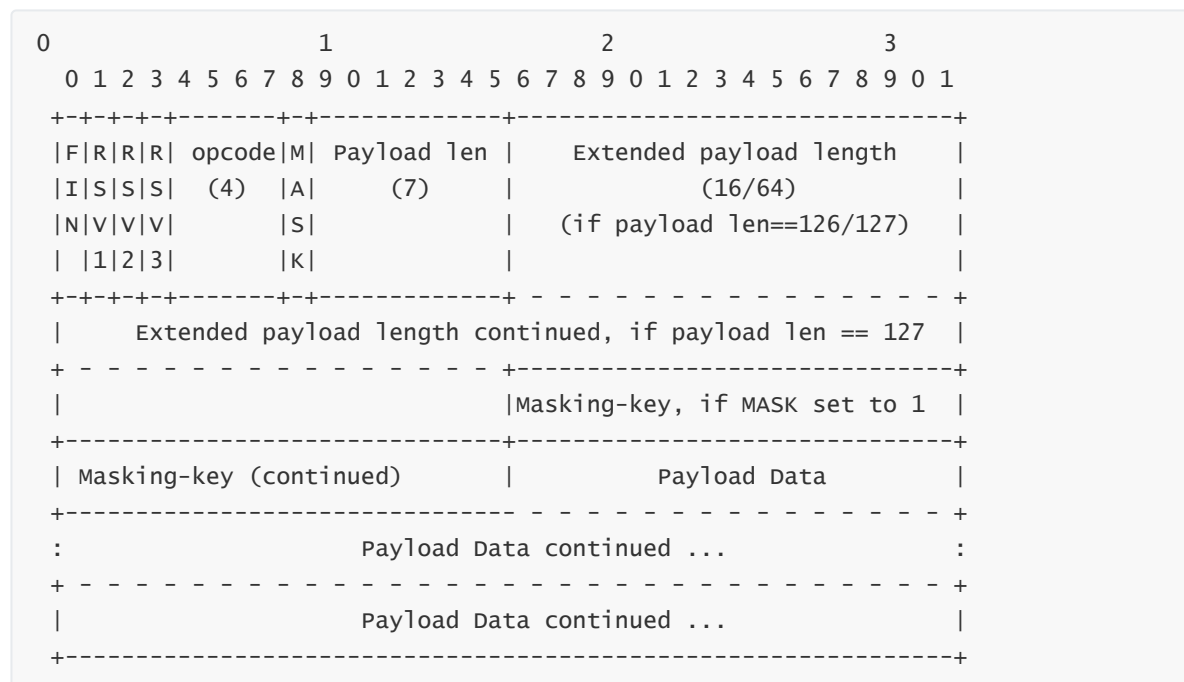
再来看，完成一个文本到语音合成的websocket通信过程：

通过Websocket协议做语音合成



以上的是websocket握手。因此可以看到，websocket有两个部分，一个部分使用http完成握手，一个部分完成数据传输。websocket数据传输可以像TCP一样多次进行，因此需要一个传输的协议格式，，这个格式就是：

数据传输格式(先了解):



websocket握手消息举例:

```
GET /v2/tts?text="base64中国加油" HTTP/1.1\r\n
Host: tts-api.xfyun.cn\r\n
Upgrade: websocket\r\n
Connection: Upgrade\r\n
Sec-WebSocket-Version: 13\r\n
Sec-WebSocket-Key: KQAAACNIAAC+GAAAhGCAAA==\r\n\r\n
```

服务端返回101表示服务端升级websocket协议成功, 客户端检验key之类的没有问题后, 可以后续通信:

```
HTTP/1.1 101 Switching Protocols\r\n
Upgrade: websocket\r\n
Connection: Upgrade\r\n
Sec-WebSocket-Accept: kuA2hFRL9jQWxWIVq3rSdM1ARBY=\r\n\r\n
```

协议这种东西不太直观, 但是由上面的分析可知, 使用讯飞开放平台, 必须使用websocket, 目前Android平台比较好用的是okhttp3这个类库, 提供了websocket的握手和多次交互的接口。

2.2.4.3 okhttp3-websocket类库的使用

首先需要在工程中加入okhttp3库的依赖, build.gradle(Module:app)构建文件的dependencies中增加内容:

```
implementation 'com.squareup.okhttp3:okhttp:3.14.1'
```

build.gradle(Module:app)构建文件中的android增加内容, okhttp3强制要使用JDK8:

```
compileOptions {
    targetCompatibility = "8"
    sourceCompatibility = "8"
}
```

使用okhttp3完成一个简单的websocket请求是怎样的，其实比较简单，分为两步。

1. 创建一个OkHttpClient对象，然后调用newWebSocket完成连接；
2. 然后实现onOpen、onMessage、onClosing、onClosed、onFailure就可以了

onOpen就是收到了服务端的http response，如果返回的值是101，则表示服务端认可，可以继续下面的数据传输了，这个类似http返回的200。数据传输在onMessage中，比如语音合成服务器audio数据传递到客户端后，就可以在onMessage中处理audio数据，然后拿出去播放了，但是要小心onFailure，可能服务器因为我们传递的数据不合法，而拒绝提供服务，并关闭连接，下面是实例代码。

```
OkHttpClient client = new OkHttpClient.Builder().build();
String url = "ws://tts-api.xfyun.cn/v2/tts";
Request request = new Request.Builder().url(url).build();
WebSocket webSocket = client.newWebSocket(request, new WebSocketListener() {
    @Override
    public void onOpen(WebSocket webSocket, Response response) {
        super.onOpen(webSocket, response);
    }
    @Override
    public void onMessage(WebSocket webSocket, String text) {
        super.onMessage(webSocket, text);
    }
    @Override
    public void onMessage(WebSocket webSocket, ByteString bytes) {
        super.onMessage(webSocket, bytes);
    }
    @Override
    public void onClosing(WebSocket webSocket, int code, String reason) {
        super.onClosing(webSocket, code, reason);
        System.out.println("socket closing");
    }
    @Override
    public void onClosed(WebSocket webSocket, int code, String reason) {
        super.onClosed(webSocket, code, reason);
        System.out.println("socket closed");
    }
    @Override
    public void onFailure(WebSocket webSocket, Throwable t, Response response) {
        super.onFailure(webSocket, t, response);
        System.out.println("connection failed:" + response.message());
    }
}
```

接下来我们查看语音合成文档，套用以上的代码完成语音合成的调用吧。

2.2.4.4 语音合成文档分析

根据文档的鉴权部分说明文档可知，只要能够把服务端需要的url地址组装好，就可以完成语音合成的功能了，url=wss://tts-api.xfyun.cn/v2/tts?host=tts-api.xfyun.cn&date=GMT时间&authorization=base64(string)

URL分解如下：

服务地址	wss://tts-api.xfyun.cn/v2/tts
参数host	请求主机，是字符串tts-api.xfyun.cn
参数date	当前的时间戳，格式为Thu, 01 Aug 2019 01:53:21 GMT，协议规范编号为RFC1123
参数authorization	使用base64编码的签名相关信息

date: GMT时间（格林尼治标准时间）一般指世界时，即0时区的区时，比北京时间（东8区）晚8小时；所以GMT时间+8小时所得结果就是北京时间

authorization:

```
authorization = base64(api_key="$api_key",algorithm="hmac-sha256",headers="host
date request-line",signature="$signature")
```

signature:

```
signature=base64(signature_sha)
```

signature_sha:

```
signature_sha=hmac-sha256(signature_origin,$apiSecret)
```

signature_origin:

```
host: tts-api.xfyun.cn
date: Thu, 01 Aug 2019 01:53:21 GMT
GET /v2/tts HTTP/1.1
```

因此只要能够组装好authorization基本上也就完成了URL的组装，再套用okhttp3接口就能完成语音合成了。

2.2.4.5 代码开发

按照以上的分析，使用getAuthUrl来完成URL的组装，其中大部分的代码是在拼接authorization：

```
/**
 * 语音合成URL拼接，满足调用需求
 * @param hostUrl(string) host的http地址
 * @param apiKey(string) 应用的apikey
 * @param apiSecret(string) 应用的apiSecret
 * @return 服务端要求的URL
 */
public static String getAuthUrl(String hostUrl, String apiKey, String apiSecret)
throws Exception {
    URL url = new URL(hostUrl);
    // 设置时间格式，满足date的要求
```

```

SimpleDateFormat format = new SimpleDateFormat("EEE, dd MMM yyyy
HH:mm:ss z", Locale.US);
// 设置时间为GMT时间
format.setTimeZone(TimeZone.getTimeZone("GMT"));
// 基于以上的配置信息获取到当前的满足格式的GMT时间
String date = format.format(new Date());

// 拼接signature_origin
StringBuilder builder = new StringBuilder("host:
").append(url.getHost()).append("\n").//
    append("date: ").append(date).append("\n").//
    append("GET ").append(url.getPath()).append(" HTTP/1.1");

// 拼接signature_sha
Charset charset = Charset.forName("UTF-8");
Mac mac = Mac.getInstance("hmacsha256");
SecretKeySpec spec = new SecretKeySpec(apiSecret.getBytes(charset),
"hmacsha256");
mac.init(spec);
byte[] hexDigits = mac.doFinal(builder.toString().getBytes(charset));
// 拼接signature
String sha = Base64.encodeToString(hexDigits, Base64.NO_WRAP);
// 拼接authorization
String authorization = String.format("hmac username=\"%s\",
algorithm=\"%s\", headers=\"%s\", signature=\"%s\"", apiKey, "hmac-sha256",
"host date request-line", sha);
// 拼接URL
HttpUrl httpUrl = HttpUrl.parse("https://" + url.getHost() +
url.getPath()).newBuilder().//
    addQueryParameter("authorization",
Base64.encodeToString(authorization.getBytes(), Base64.NO_WRAP)).//
    addQueryParameter("date", date).//
    addQueryParameter("host", url.getHost()).//
    build();
return httpUrl.toString();
}

```

使用拼接好的URL，调用okhttp3接口完成语音合成的服务器通信：

```

/**
 * 获取TTS语音数据
 * @param text(string) 需要合成的文本
 * @return 服务端要求的URL
 * @note 使用到了gson类库，在build.gradle(Module:app)中增加以下内容
 *       implementation 'com.google.code.gson:gson:2.8.6'
 */

private static final String hostUrl = "http://tts-api.xfyun.cn/v2/tts";
private static final String appid = "xxxxxxx";
private static final String apiSecret = "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx";
private static final String apiKey = "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx";
private static final String vcn = "xiaoyan";
public static ArrayList<byte[]> mArrayList = new ArrayList<byte[]>();
public static void getTTSDData(String text) throws Exception {
    // 获取到服务端要求的URL请求
    String authUrl = getAuthUrl(hostUrl, apiKey, apiSecret);
    // 构造http客户端

```



```

OkHttpClient client = new OkHttpClient.Builder().build();
// 把schema中的http(s)替换成ws(s)
String url = authUrl.toString().replace("http://",
"ws://").replace("https://", "wss://");
// 构造连接请求request
Request request = new Request.Builder().url(url).build();
// 发起websocket连接, 异步接收结果
WebSocket webSocket = client.newWebSocket(request, new WebSocketListener() {
    @Override
    public void onOpen(WebSocket webSocket, Response response) {
        super.onOpen(webSocket, response);
        mArrayList.clear();
        // 调用到这个地方, 说明服务端返回了101消息, 类似http返回的200消息, 这里可以发送
        请求了

        // 查看语音合成文档, 组装消息
        // 注意官方提供的请求有问题, 请使用第二个format
        //String format = "{\"common\":{\"app_id\":\"%s\"},\"business\":
        {\"vcn\":\"%s\", \"aue\":\"raw\", \"speed\":\"50\", \"data\":
        {\"status\":2, \"encoding\":\"UTF8\", \"text\":\"%s\"}}";
        String format = "{\"common\":{\"app_id\":\"%s\"},\"business\":
        {\"aue\":\"raw\", \"tte\":\"UTF8\", \"ent\":\"intp65\", \"vcn\":\"%s\", \"pitch\":50
        , \"speed\":50}, \"data\":{\"status\":2, \"text\":\"%s\"}}";
        String reqData = String.format(format, appid, vcn,
        Base64.encodeToString(text.getBytes("utf8"), Base64.NO_WRAP));
        try {
            webSocket.send(reqData);
        } catch (UnsupportedEncodingException e) {
            e.printStackTrace();
        }
    }
    @Override
    public void onMessage(WebSocket webSocket, String text) {
        super.onMessage(webSocket, text);
        System.out.println("receive=> " + text);
        // 当服务器有数据返回时, 调用到这里, 注意服务器的audio数据可能分多次返回, 当
        status为2时代表时最后一块音频数据, websocket连接也可以关闭了。
        ResponseData resp = null;
        Gson gson = new Gson();
        try {
            resp = gson.fromJson(text, ResponseData.class);
        } catch (Exception e) {
            e.printStackTrace();
        }
        if (resp.getData() != null) {
            // 先把每次获取到的音频base64 decode后放入mArrayList
            String result = resp.getData().audio;
            byte[] audio = Base64.decode(result, Base64.NO_WRAP);
            mArrayList.add(audio);
            // 看文档得知, 当status为2时, 为最后一块音频
            if (resp.getData().status == 2) {
                int length = 0;
                for (int i = 0; i < mArrayList.size(); i++) {
                    length += mArrayList.get(i).length;
                }

                // 通过以下代码, pcm就是服务端返回给客户端的合成语音了, 调用播放接口播
                放
                byte[] pcm = new byte[length];

```

```

        int curLength = 0;
        for (int i = 0; i < mArrayList.size(); i++) {
            System.arraycopy(mArrayList.get(i), 0, pcm, curLength,
mArrayList.get(i).length);
            curLength += mArrayList.get(i).length;
        }
        AudioManager.getInstance().startPlay(pcm);
        // 数据接收完毕，关闭连接，释放资源
        websocket.close(1000, "");
    }
}
@Override
public void onMessage(Websocket websocket, ByteString bytes) {
    super.onMessage(websocket, bytes);
}
@Override
public void onClosing(Websocket websocket, int code, String reason) {
    super.onClosing(websocket, code, reason);
    System.out.println("socket closing");
}
@Override
public void onClosed(Websocket websocket, int code, String reason) {
    super.onClosed(websocket, code, reason);
    System.out.println("socket closed");
}
@Override
public void onFailure(Websocket websocket, Throwable t, Response
response) {
    super.onFailure(websocket, t, response);
    System.out.println("connection failed:" + response.message());
}
});
}
}

```

语音合成请求参数（这个是开放平台提供，合成有问题）：

```

// 以下json请求官方提供的有问题，speed不能为字符串，另外encoding也被放到了business的tte
{
    "common": {
        "app_id": "123456"
    },
    "business": {
        "vcn": "xiaoyan",
        "aue": "raw",
        "speed": "50"
    },
    "data": {
        "status": 2,
        "encoding": "UTF8",
        "text": "exSI6ICJlbiIsCgkgICAgInBvc2l0aw9uIjogImZhbnNlIgoJf"
    }
}

```

修改后

```
{
  "common": {
    "app_id": "123456"
  },
  "business": {
    "aue": "raw",
    "tte": "UTF8",
    "ent": "intp65",
    "vcn": "xiaoyan",
    "pitch": 50,
    "speed": 50
  },
  "data": {
    "status": 2,
    "text": "exSI6ICJlbiIsCgkgICAgInBvc2l0aw9uIjogImZhbnNlIgoJf"
  }
}
```

ResponseData内容:

```
public static class ResponseData {
    private int code;
    private String message;
    private String sid;
    private Data data;
    public int getCode() {
        return code;
    }
    public String getMessage() {
        return this.message;
    }
    public String getSid() {
        return sid;
    }
    public Data getData() {
        return data;
    }
}

public static class Data {
    private int status; //标志音频是否返回结束 status=1, 表示后续还有音频返回,
    status=2表示所有的音频已经返回
    private String audio; //返回的音频, base64 编码
    private String ced; // 合成进度
}
```

MainActivity.java中调用语音合成的通信接口:

```
package com.example.advertising;

import androidx.appcompat.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);
    try {
        webTTSWS.getTTSDData(MainActivity.this, "我们是共产主义接班人",
responseResult);
    } catch (Exception e) {
        e.printStackTrace();
    }
}

webTTSWS.IResponseResult responseResult = new webTTSWS.IResponseResult() {
    @Override
    public void getAudio(byte[] audio) {
        AudioManager.getInstance().startPlay(audio);
    }
};
}

```

AudioTrackManager.java文件:

```

package com.example.advertising;

import android.media.AudioFormat;
import android.media.AudioManager;
import android.media.AudioRecord;
import android.media.AudioTrack;

import java.io.DataInputStream;
import java.io.File;
import java.io.FileInputStream;

public class AudioTrackManager {
    public static final String TAG = "AudioTrackManager";
    private AudioTrack audioTrack;
    private DataInputStream dis;
    private Thread recordThread;
    private boolean isStart = false;
    private static AudioTrackManager mInstance;
    private int bufferSize;
    private byte[] audioPcm = null;

    public AudioTrackManager() {
        bufferSize = AudioTrack.getMinBufferSize(16000,
AudioFormat.CHANNEL_OUT_MONO, AudioFormat.ENCODING_PCM_16BIT);
        audioTrack = new AudioTrack(AudioManager.STREAM_MUSIC, 16000,
AudioFormat.CHANNEL_OUT_MONO, AudioFormat.ENCODING_PCM_16BIT, bufferSize * 2,
AudioTrack.MODE_STREAM);
    }

    /**
     * 获取单例引用
     *
     * @return
     */
    public static AudioTrackManager getInstance() {

```

```

        if (mInstance == null) {
            synchronized (AudioTrackManager.class) {
                if (mInstance == null) {
                    mInstance = new AudioTrackManager();
                }
            }
        }
        return mInstance;
    }

    /**
     * 销毁线程方法
     */
    private void destroyThread() {
        try {
            isStart = false;
            if (null != recordThread && Thread.State.RUNNABLE ==
recordThread.getState()) {
                try {
                    Thread.sleep(500);
                    recordThread.interrupt();
                } catch (Exception e) {
                    recordThread = null;
                }
            }
            recordThread = null;
        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            recordThread = null;
        }
    }

    /**
     * 启动播放线程
     */
    private void startThread() {
        destroyThread();
        isStart = true;
        if (recordThread == null) {
            recordThread = new Thread(recordRunnable);
            recordThread.start();
        }
    }

    /**
     * 播放线程
     */
    Runnable recordRunnable = new Runnable() {
        @Override
        public void run() {

            android.os.Process.setThreadPriority(android.os.Process.THREAD_PRIORITY_URGENT_
AUDIO);

            byte[] tempBuffer = new byte[bufferSize];
            int len = 0;
            int readCount = 0;

```

```

        if (null != audioPcm && (audioPcm.length > 0)) {
            int totalCount = audioPcm.length;
            while (readCount < totalCount) {
                if (readCount + tempBuffer.length > totalCount) {
                    len = totalCount - readCount;
                } else {
                    len = tempBuffer.length;
                }
                System.arraycopy(audioPcm, readCount, tempBuffer, 0,
len);

                audioTrack.play();
                audioTrack.write(tempBuffer, 0, len);
                readCount += len;
            }
        } else {
            while (dis.available() > 0) {
                readCount= dis.read(tempBuffer);
                System.out.println("readCount:" + readCount);
                if (readCount == AudioTrack.ERROR_INVALID_OPERATION ||
readCount == AudioTrack.ERROR_BAD_VALUE) {
                    continue;
                }
                if (readCount != 0 && readCount != -1) {
                    audioTrack.play();
                    audioTrack.write(tempBuffer, 0, readCount);
                }
            }
        }
        stopPlay();
    } catch (Exception e) {
        e.printStackTrace();
    }
}

};

/**
 * 播放文件
 *
 * @param path
 * @throws Exception
 */
private void setPath(String path) throws Exception {
    File file = new File(path);
    dis = new DataInputStream(new FileInputStream(file));
}

/**
 * 启动播放
 *
 * @param path
 */
public void startPlay(String path) {
    try {
        setPath(path);
        startThread();
    } catch (Exception e) {
        e.printStackTrace();
    }
}

```

```

    }
}

public void startPlay(byte[] pcm) {
    try {
        audioPcm = new byte[pcm.length];
        System.arraycopy(pcm, 0, audioPcm, 0, pcm.length);
        startThread();
    } catch (Exception e) {
        e.printStackTrace();
    }
}

/**
 * 停止播放
 */
public void stopPlay() {
    try {
        destroyThread();
        if (audioTrack != null) {
            if (audioTrack.getState() == AudioRecord.STATE_INITIALIZED) {
                audioTrack.stop();
            }
            // if (audioTrack != null) {
            //     audioTrack.release();
            // }
        }
        if (dis != null) {
            dis.close();
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
}
}

```

另外要注意，播放需要申请Android的录音权限，网络也要申请对应的权限，这些权限可以在manifest.xml中增加，如下，但是Android6.0之后，需要运行时动态获取，为了不增加负担，先在设置中找到对应的应用程序，把相关的录音等权限打开

```

<!-- 录音权限 -->
<uses-permission android:name="android.permission.RECORD_AUDIO" />
<!-- 网络相关权限 -->
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
<uses-permission android:name="android.permission.CHANGE_NETWORK_STATE" />
<!-- 外部存储读写权限 -->
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />

```

2.3 带用户交互界面的语音合成

2.3.1 演示效果



2.3.1 知识点

- android layout布局
- Button、EditText界面元素
- 按钮的事件处理逻辑

2.3.1 掌握技能

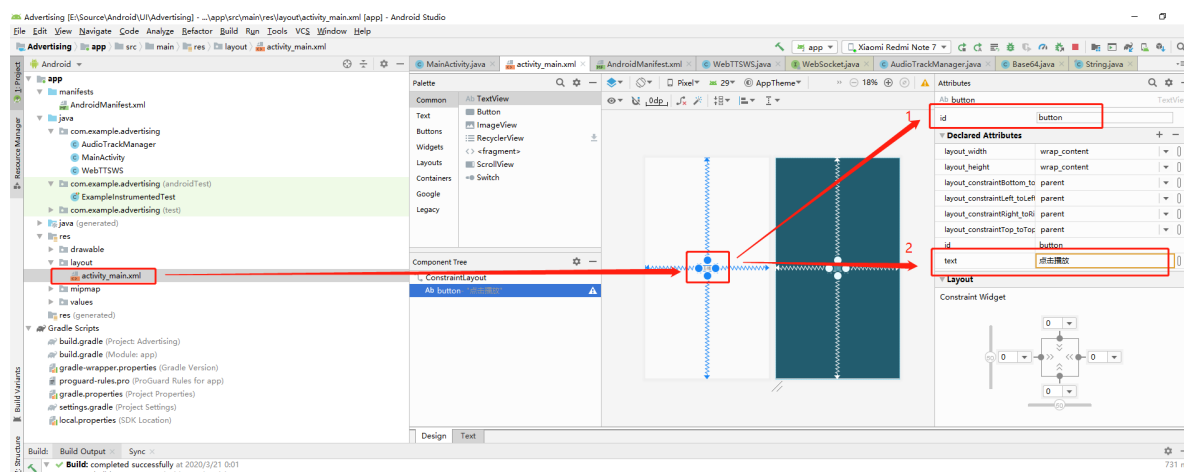
- 能够通过界面控制合成功能

2.3.4 实现

2.3.4.1 界面配置

先在界面上画一个按钮，找到这个Activity的布局文件R.layout.activity_main，然后把之前中间显示的helloworld的小东西改造一下😊

如图所示，把中间的显示控件TextView增加一个id名称，方便在代码中可以通过findviewbyid()找到这个显示控件，并能够控制它。然后给它重新起一个名字，方便让用户知道能够点击它。



2.3.4.2 代码调用

代码中需要给它再增加一个单击的事件，当事件发生时，我们让它触发播放的事件就可以了。

```
protected void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.activity_main);  
    textView = findViewById(R.id.button);  
    textView.setOnClickListener(new View.OnClickListener(){  
        @Override  
        public void onClick(View v) {  
            try {  
                webTTWS.getTTData("我们是共产主义接班人", responseResult);  
            } catch (Exception e) {  
                e.printStackTrace();  
            }  
        }  
    });  
}
```

2.3.4.3 代码优化

由于播放的代码放到了WebTTWS中，显得不太优雅，代码也不大容易看得懂，可以在WebTTWS中增加一个接口，异步回调到MainActivity中，代码的逻辑会显得更加完整。

WebTTWS中增加变量和接口类：

```

private static IResponseResult mResponseResult;
public interface IResponseResult {
    void setAudio(byte[] audio);
}
// getTTSDData增加接口参数permissionsResult
public static void getTTSDData(Activity c, String text, @NonNull IResponseResult
permissionsResult) {
    mResponseResult = permissionsResult;
    // .....
    // 音频调用的代码去掉，采用接口的方式
    // AudioManager.getInstance().startPlay(pcm);
    mResponseResult.setAudio(pcm);
}

```

MainActivity.java中增加参数调用

```

WebTTSWS.getTTSDData("我们是共产主义接班人", responseResult);
WebTTSWS.IResponseResult responseResult = new WebTTSWS.IResponseResult() {
    @Override
    public void setAudio(byte[] audio) {
        AudioManager.getInstance().startPlay(audio);
    }
};

```

2.4 一个广告彩铃应用

2.4.1 演示效果

和上一章相比，增加了背景音乐合成的功能，完成广告彩铃的基础功能的一个应用。

2.4.2 知识点

混音的原理

pcm的格式及混音

2.4.3 掌握技能

能够通过控制混音功能，通过界面控制完成广告彩铃的应用

2.4.4 实现

音频混合原理: 量化的语音信号的叠加等价于空气中声波的叠加

2.4.4.1 直接叠加法

A (A1,A2,A3,A4) 和B (B1,B2,B3,B4) 叠加后求平均值，得到C ((A1+B1) , (A2+B2) , (A3+B3) , (A4+B4))

这种情况，输出的音频中A和B音频数据都可以以相同声音大小播放，但是可能出现溢出的情况。假设A音频指定时间点的某段采样数据是 (23,67,511,139,307) ， B音频对应该时间点的采样数据是 (1101,300,47,600,22) ，那么两者直接叠加的话，得到的采样数据是 (1124,367,558,739,329) ，这个短采样数据就是两者声音混合的数据了。

2.4.4.2 叠加后求平均值

A (A1,A2,A3,A4) 和B (B1,B2,B3,B4) 叠加后求平均值, 得到C ((A1+B1) /2, (A2+B2) /2, (A3+B3) /2, (A4+B4) /2)

这样可以避免出现溢出的情况, 但是会出现两者声音会比之前单独的声音小了一半, 比如人声和背景音乐混合, 导致输出的音频中, 人声小了一半, 背景音乐也小了一半, 这种情况可能就不是想要的效果, 特别是多段音频混合的情况。

2.4.4.3 归一化混音(自适应加权混音算法)

使用更多的位数(32 bit)来表示音频数据的一个样本, 混完音后在想办法降低其振幅, 使其仍旧分布在 16 bit所能表示的范围之内, 这种方法叫做归一法。

为避免发生溢出, 使用一个可变的衰减因子对语音进行衰减。这个衰减因子也就代表语音的权重, 衰减因子随着音频数据的变化而变化, 所以称为自适应加权混音。当溢出时, 衰减因子较小, 使得溢出的数据在衰减后能够处于临界值以内, 而在没有溢出时, 又让衰减因子慢慢增大, 使数据较为平缓的变化。

在资源res目录下创建raw, 把预备好的一个bg1.pcm文件放进去, 并调用 VoiceMixerUtil.normalizationMix的归一化混音接口完成混音, 得到混音后的mixpcm, 并且送到 AudioTrackManager中进行播放

```
WebTTSWS.IResponseResult responseResult = new WebTTSWS.IResponseResult() {
    @Override
    public void getAudio(byte[] audio) {
        byte[] bgpcm = null;
        try {
            InputStream in = getResources().openRawResource(R.raw.bg1);
            //获取文件的字节数
            int lenght = in.available();
            //创建byte数组
            bgpcm = new byte[lenght];
            //将文件中的数据读到byte数组中
            in.read(bgpcm);
        } catch (Exception e) {
            e.printStackTrace();
        }
        byte[][] mix = {audio, bgpcm};
        byte[] mixpcm = VoiceMixerUtil.normalizationMix(mix);
        AudioTrackManager.getInstance().startPlay(mixpcm);
    }
};
```

VoiceMixerUtil列出了几种混音的算法, 其中归一化方法效果较好,VoiceMixerUtil.java文件参考如下:

```
package com.example.ad.utils;

public class VoiceMixerUtil {
    /**
     * 噪音太大
     * @param data1
     * @param data2
     * @return
     */
    public static byte[] mixVoice(byte[] data1, byte[] data2) {
        int length1 = data1.length;
        int length2 = data2.length;
```

```

        int count = length1 <= length2 ? length1 : length2;
        int size = length1 >= length2 ? length1 : length2;
        byte[] data = new byte[size];
        int i = 0;
        for (; i < count; i++) {
//            data[i] = (byte) (data1[i]+data2[i]-(data1[i]*data2[i]>>0x10));
            if (data1[i] < 0 && data2[i] < 0) {
                data[i] = (byte) (data1[i] + data2[i] - (data1[i] * data2[i] / -
(Math.pow(2, 16 - 1) - 1)));
            } else {
                data[i] = (byte) (data1[i] + data2[i] - (data1[i] * data2[i] /
(Math.pow(2, 16 - 1) - 1)));
            }
        }
        if (i == length1) {
            for (int j = i; j < length2; j++) {
                data[j] = data2[j];
            }
        } else if (i == length2) {
            for (int j = i; j < length1; j++) {
                data[j] = data1[j];
            }
        }
        return data;
    }

/**
 * 较好
 * 每一行是一个音频的数据
 */
    public static byte[] averageMix(byte[][] bMulRoadAudios) {

        if (bMulRoadAudios == null || bMulRoadAudios.length == 0)
            return null;

        byte[] realMixAudio = bMulRoadAudios[0];

        if (bMulRoadAudios.length == 1)
            return realMixAudio;

        for (int rw = 0; rw < bMulRoadAudios.length; ++rw) {
            if (bMulRoadAudios[rw].length != realMixAudio.length) {
                return null;
            }
        }

        int row = bMulRoadAudios.length;
        int coloum = realMixAudio.length / 2;
        short[][] sMulRoadAudios = new short[row][coloum];

        for (int r = 0; r < row; ++r) {
            for (int c = 0; c < coloum; ++c) {
                sMulRoadAudios[r][c] = (short) ((bMulRoadAudios[r][c * 2] &
0xff) | (bMulRoadAudios[r][c * 2 + 1] & 0xff) << 8);
            }
        }

        short[] sMixAudio = new short[coloum];

```

```

        int mixVal;
        int sr = 0;
        for (int sc = 0; sc < coloum; ++sc) {
            mixVal = 0;
            sr = 0;
            for (; sr < row; ++sr) {
                mixVal += sMulRoadAudioes[sr][sc];
            }
            sMixAudio[sc] = (short) (mixVal / row);
        }

        for (sr = 0; sr < coloum; ++sr) {
            realMixAudio[sr * 2] = (byte) (sMixAudio[sr] & 0x00FF);
            realMixAudio[sr * 2 + 1] = (byte) ((sMixAudio[sr] & 0xFF00) >> 8);
        }

        return realMixAudio;
    }

    /**
     * 较好
     * 归一化混音
     */
    public static byte[] normalizationMix(byte[][] allAudioBytes) {
        if (allAudioBytes == null || allAudioBytes.length == 0)
            return null;

        byte[] realMixAudio = allAudioBytes[0];

        //如果只有一个音频的话，就返回这个音频数据
        if (allAudioBytes.length == 1)
            return realMixAudio;

        //row 有几个音频要混音
        int row = realMixAudio.length / 2;
        //
        short[][] sourecs = new short[allAudioBytes.length][row];
        for (int r = 0; r < 2; ++r) {
            for (int c = 0; c < row; ++c) {
                sourecs[r][c] = (short) ((allAudioBytes[r][c * 2] & 0xff) |
                    (allAudioBytes[r][c * 2 + 1] & 0xff) << 8);
            }
        }

        //coloum第一个音频长度 / 2
        short[] result = new short[row];
        //转成short再计算的原因是，提供精确度，高端的混音软件据说都是这样做的，可以测试一下不
        //转short直接计算的混音结果
        for (int i = 0; i < row; i++) {
            int a = sourecs[0][i];
            int b = sourecs[1][i];
            if (a < 0 && b < 0) {
                int i1 = a + b - a * b / (-32768);
                if (i1 > 32767) {
                    result[i] = 32767;
                } else if (i1 < -32768) {
                    result[i] = -32768;
                } else {

```

```

        result[i] = (short) i1;
    }
} else if (a > 0 && b > 0) {
    int i1 = a + b - a * b / 32767;
    if (i1 > 32767) {
        result[i] = 32767;
    } else if (i1 < -32768) {
        result[i] = -32768;
    } else {
        result[i] = (short) i1;
    }
} else {
    int i1 = a + b;
    if (i1 > 32767) {
        result[i] = 32767;
    } else if (i1 < -32768) {
        result[i] = -32768;
    } else {
        result[i] = (short) i1;
    }
}
}
return toByteArray(result);
}

public static byte[] toByteArray(short[] src) {
    int count = src.length;
    byte[] dest = new byte[count << 1];
    for (int i = 0; i < count; i++) {
        dest[i * 2 + 1] = (byte) ((src[i] & 0xFF00) >> 8);
        dest[i * 2] = (byte) ((src[i] & 0x00FF));
    }
    return dest;
}

/**
 * 较好
 * @param bMulRoadAudios
 * @return
 */
public static byte[] mixRawAudioBytes(byte[][] bMulRoadAudios) {

    if (bMulRoadAudios == null || bMulRoadAudios.length == 0)
        return null;

    byte[] realMixAudio = bMulRoadAudios[0];

    if (bMulRoadAudios.length == 1)
        return realMixAudio;

    for (int rw = 0; rw < bMulRoadAudios.length; ++rw) {
        if (bMulRoadAudios[rw].length != realMixAudio.length) {
            return null;
        }
    }

    int row = bMulRoadAudios.length;
    int coloum = realMixAudio.length / 2;

```

```

short[][] sMulRoadAudioes = new short[row][coloum];

for (int r = 0; r < row; ++r) {
    for (int c = 0; c < coloum; ++c) {
        sMulRoadAudioes[r][c] = (short) ((bMulRoadAudioes[r][c * 2] &
0xff) | (bMulRoadAudioes[r][c * 2 + 1] & 0xff) << 8);
    }
}

short[] sMixAudio = new short[coloum];
int mixVal;
int sr = 0;

for (int sc = 0; sc < coloum; ++sc) {
    mixVal = 0;
    sr = 0;
    for (; sr < row; ++sr) {
        mixVal += sMulRoadAudioes[sr][sc];
    }
    sMixAudio[sc] = (short) (mixVal / row);
}

for (sr = 0; sr < coloum; ++sr) {
    realMixAudio[sr * 2] = (byte) (sMixAudio[sr] & 0x00FF);
    realMixAudio[sr * 2 + 1] = (byte) ((sMixAudio[sr] & 0xFF00) >> 8);
}

return realMixAudio;
}
}

```

3 作业

使用申请的APPID账号，完成2.3节的例子

(如果需要可以提供2.3节的代码，但是没有getAuthUrl部分和业务逻辑部分)

资源下载路径 (ppt、文档、作业、源码)：

<https://pan.ifytek.com:443/#/link/29373155624001D60544A922BF0A5721>

访问密码：woo8

有效期限截止：2020-5-20