

课程安排：

第一天：adb工具(android debug bridge安卓调试桥，电脑连接手机设备，对手机设备进行监控和操作。)，app抓包，app弱网测试，app性能测试(客户端)，app monkey测试(稳定性)。

第二天：app 操作系统简介，app测试要点，seafire 项目需求分析，seafire测试用例编写

第三天：seafire用例编写及评审，用例执行，提交bug，bug评审

回顾：

给你一个项目，怎么测试？===系统测试

1.阅读需求规格说明书，分析需求规格说明书，需求评审；

2.分析需求，提取测试要点==用xmind写思维导图

3.编写测试计划和测试方案

4.编写测试用例，并且评审

5.开发提测，进入测试执行过程：

5.1搭建测试环境===一个项目所有测试人员共用一套测试环境

5.2进行冒烟测试===保证版本的可测试性(能测，没大问题)，选取主流程，主功能20%左右用例执行，80%以上用例通过，冒烟通过

5.3按轮次进行详细全面测试==一般三轮

提交bug，跟踪bug，回归bug

第一轮：全量测试

第二轮：回归bug，交叉测试，自由测试，探索性测试

第三轮：回归bug，整体回归测试

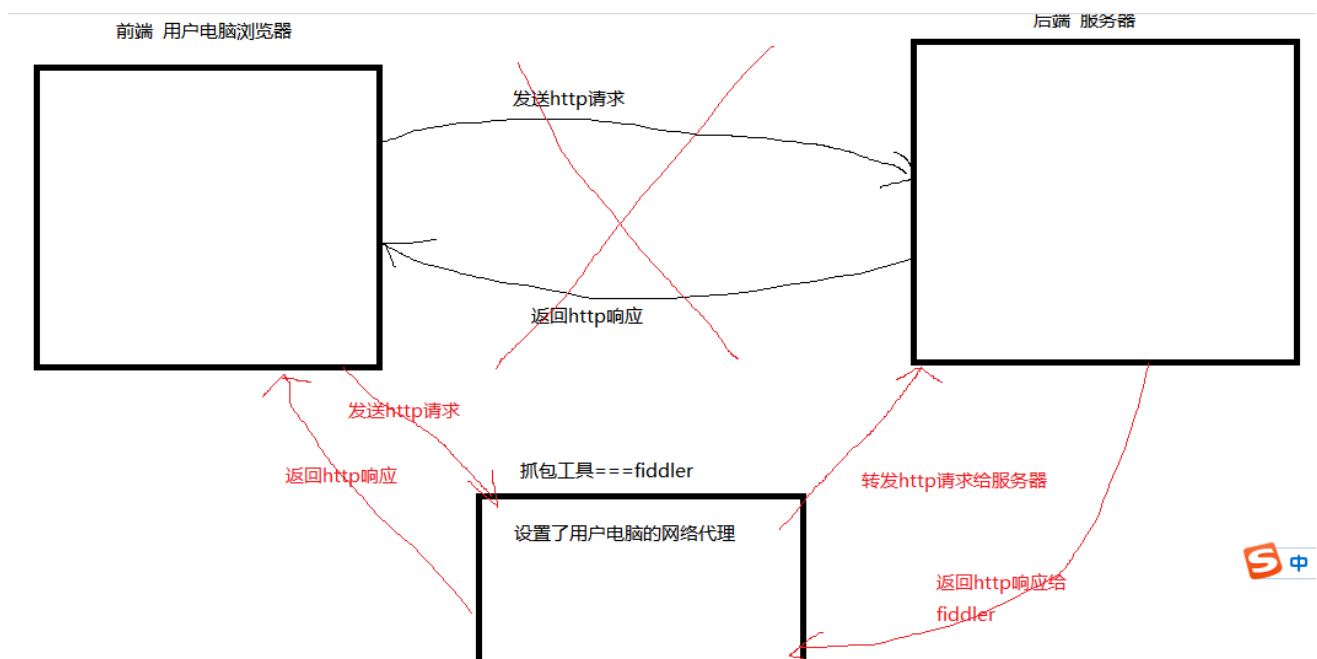
6.如果有bug遗留，要做bug评审

7.编写测试报告

8.进行验收测试

9.发布上线

fiddler工作原理：



1.adb工具

1.什么是adb工具?

adb(android debug bridge 安卓调试桥)是一个命令行工具。作用就是通过命令连接手机设备(真机/模拟

器)并且对手机设备进行操作和监控。

Android sdk (software development kit 安卓程序开发组件) 主要用来开发编译, 运行安卓程序。sdk

中包含了adb工具。

jdk (java development kit java开发组件)

2.adb工具环境搭建

1.jdk安装

由于android是用java开发的, 所以android程序的运行需要依赖jdk。

1) 安装jdk

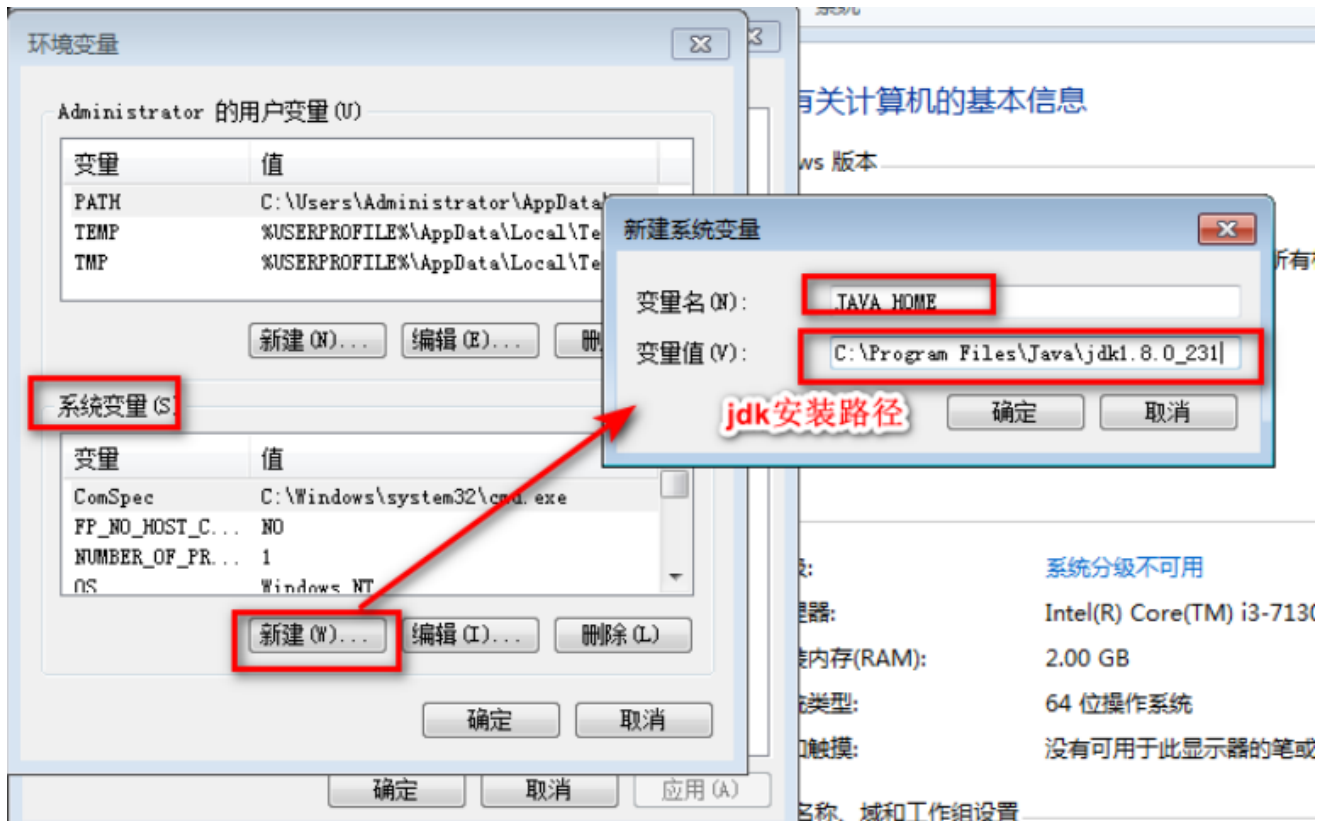
安装: jdk-8u231-windows-x64.exe

2) 配置java环境变量

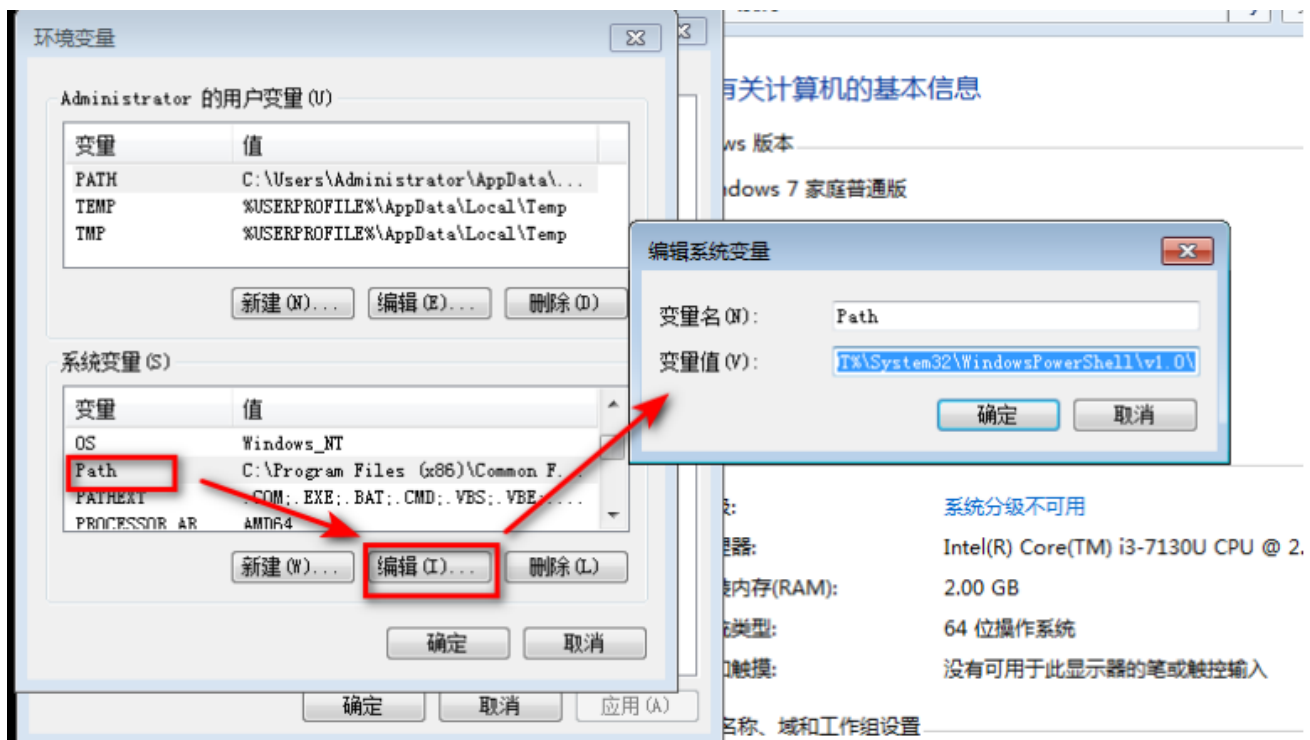
打开环境变量

(1) 新建 JAVA_HOME, 值为jdk安装路径, 如

C:\Program Files\Java\jdk1.8.0_231

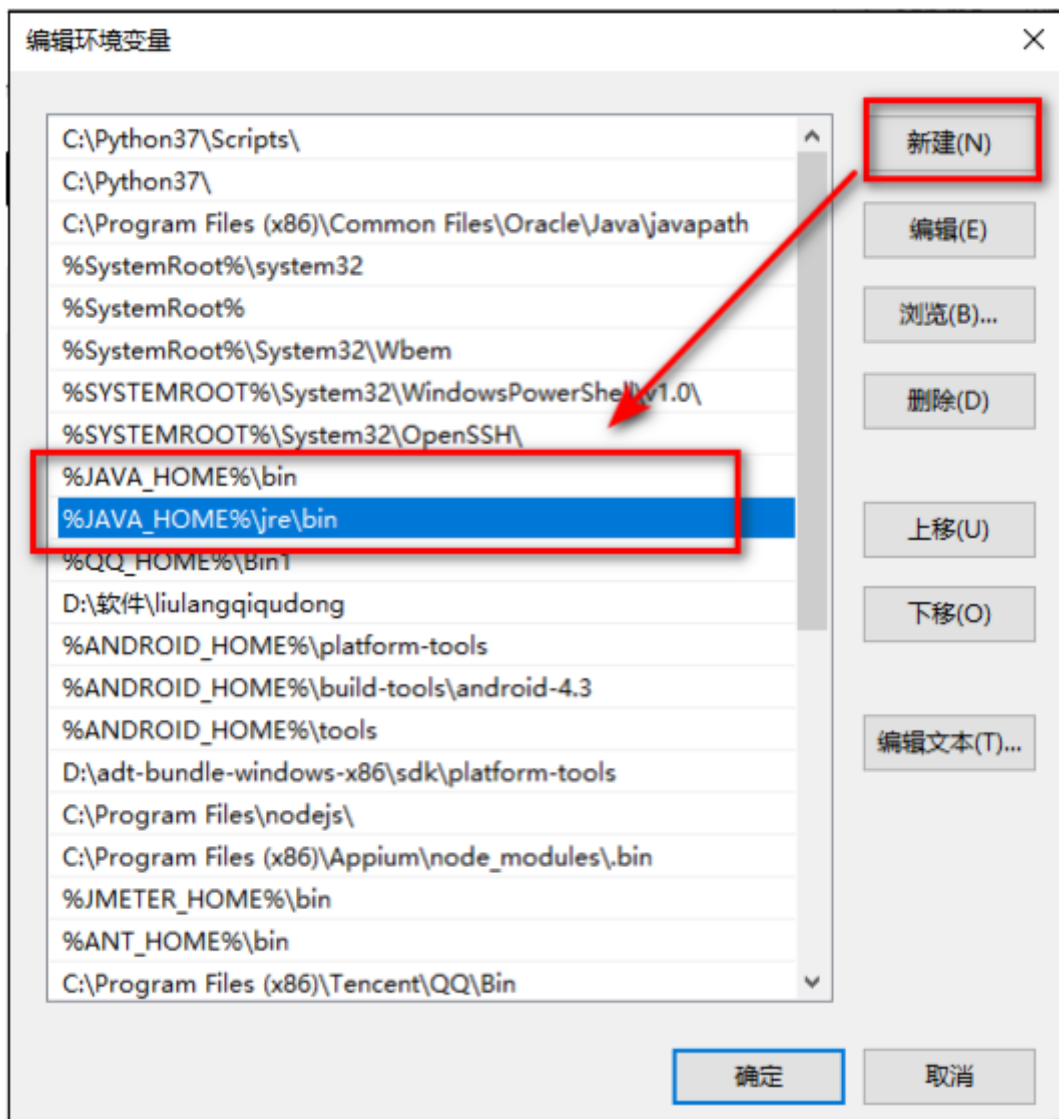


(2) 编辑 path, 在最前面加上



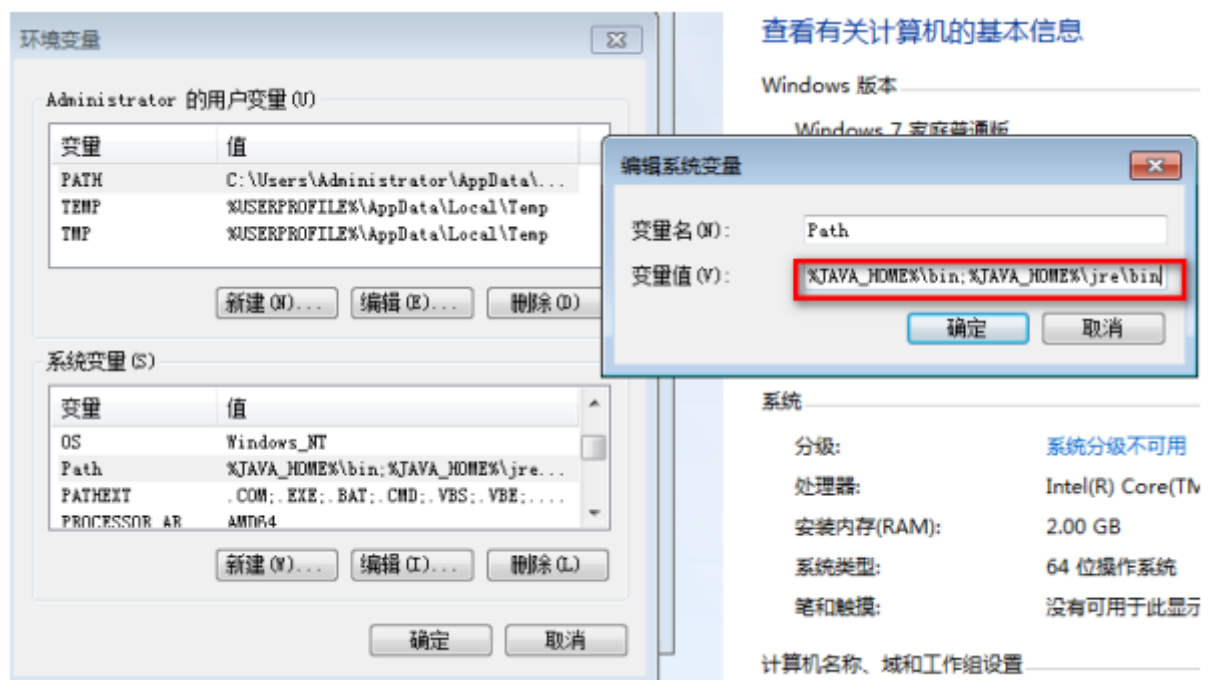
win10:

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



win7: 将上面内容添加到变量值最前面:

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



注意：最后一直点完所有的确定按钮进行保存。

3) 验证环境变量是否配置成功，注意要重新打开cmd，然后输入以下命令：

java -version

```
C:\Users\admin>java -version
java version "1.8.0_231"
Java(TM) SE Runtime Environment (build 1.8.0_231-b11)
Java HotSpot(TM) 64-Bit Server VM (build 25.231-b11, mixed mode)
```

2.安装sdk

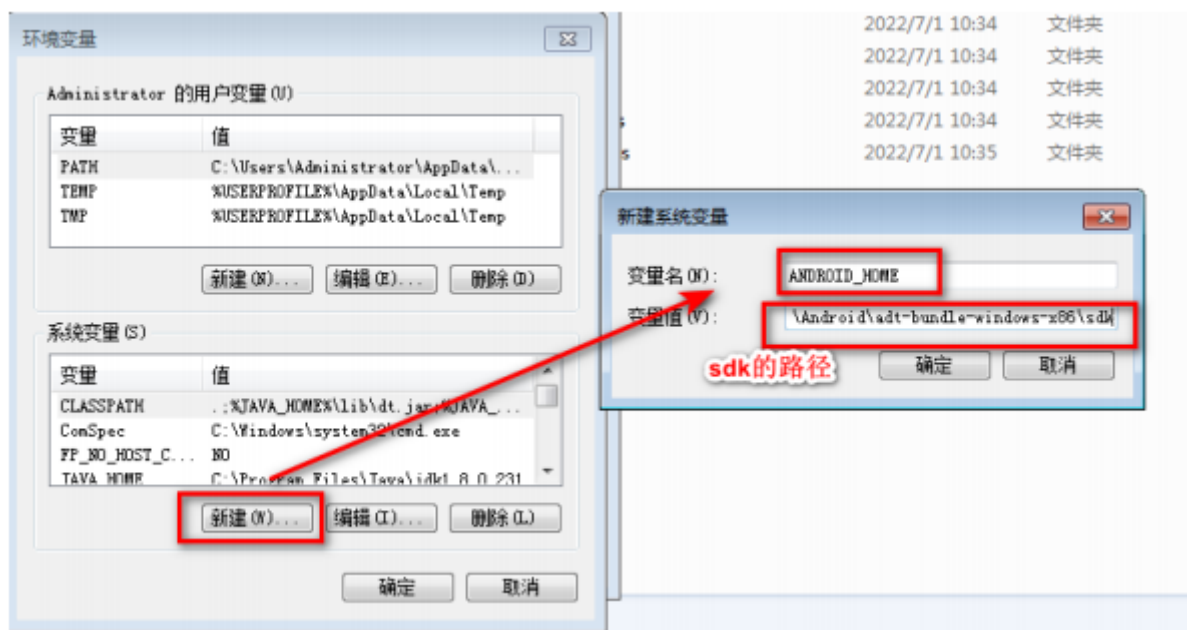
1)解压adt-bundle-windows-x86.rar

2)在D盘新建一个文件夹，名称为Android，将上一步解压后的文件夹复制到该文件夹

3) 配置android环境变量

打开环境变量

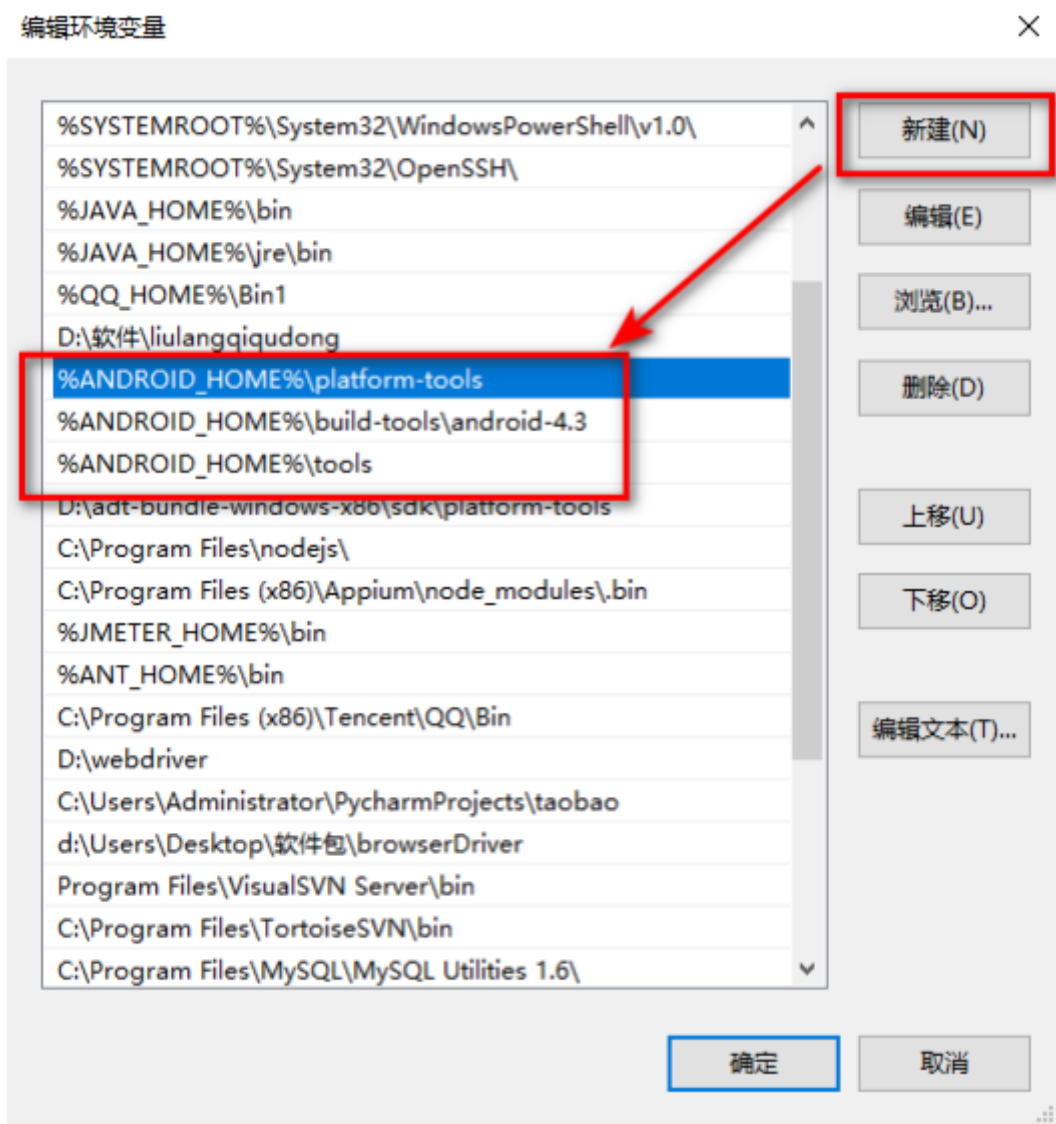
(1) 新建ANDROID_HOME，值为sdk的路径，为D:\Android\adt-bundle-windows-x86\sdk



(2) 编辑path，增加以下三个内容

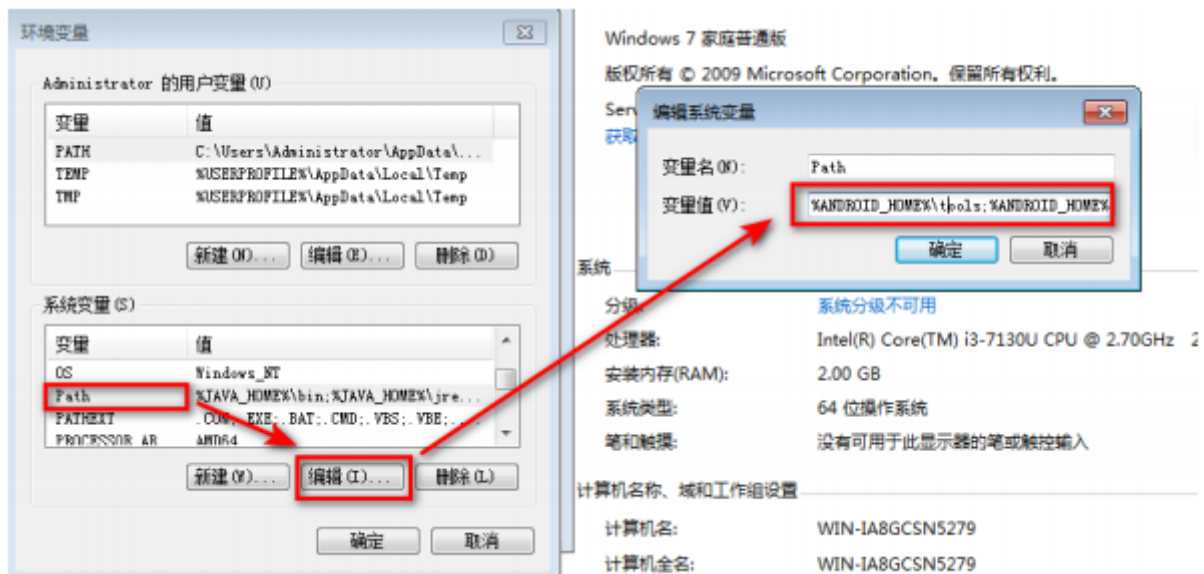
win10:

```
%ANDROID_HOME%\tools
%ANDROID_HOME%\platform-tools
%ANDROID_HOME%\build-tools\android-4.3
```



win7: 添加以下内容在path变量值的最前面

```
%ANDROID_HOME%\tools;%ANDROID_HOME%\platform-tools;%ANDROID_HOME%\build-  
tools\android-4.3;
```



注意：最后一直点完所有的确定保存。

4) 验证android环境变量是否配置成功，重新打开cmd，输入以下命令：

adb

aapt

3.安装逍遥模拟器

安装 XYAZSetup6.1.0.exe

3.常用adb命令

如果以后工作中，使用的是真机，真机连接电脑的时候，需要打开手机开发者选项中的usb调试。如果部分手机找不到开发者选项，找到手机版本号，连续点击7次版本号，也可以开启开发者选项。打开开发者选项对手机没有危害。





在windows中，进入cmd，就可以直接输入相关的adb命令运行。

1.查看当前已连接设备

```
adb devices
```

```
C:\Users\Administrator>adb devices
List of devices attached
127.0.0.1:21503 device
```

上图中表示电脑已经连接了一台模拟器。

如果连接的是模拟器，显示的ip:端口

如果真机显示的是手机的序列号

```
C:\Users\y>adb devices
List of devices attached
R28M809EQBK device
```


2.安装app

(1) 正常安装

```
adb install apk文件路径
```

```
C:\Users\Administrator>adb install C:\kaoyan3.1.0.apk
3805 KB/s (9699881 bytes in 2.489s)
    pkg: /data/local/tmp/kaoyan3.1.0.apk
Success
```

(2) 覆盖安装

如果设备上已经安装这个app，再次安装会安装失败

```
C:\Users\Administrator>adb install C:\kaoyan3.1.0.apk
3118 KB/s (9699881 bytes in 3.037s)
    pkg: /data/local/tmp/kaoyan3.1.0.apk
Failure [INSTALL_FAILED_ALREADY_EXISTS]
```

此时需要进行覆盖安装，加-r参数指定覆盖安装

```
adb install -r apk文件路径
```

```
C:\Users\Administrator>adb install -r c:\kaoyan3.1.0.apk
3377 KB/s (9699881 bytes in 2.804s)
    pkg: /data/local/tmp/kaoyan3.1.0.apk
Success
```

(3) 多个设备时安装

如果电脑连接了多个设备，需要指定操作哪一个设备，加-s参数实现指定一个设备操作。

```
adb -s ip:端口 install apk文件路径
```

例如：

```
adb -s 127.0.0.1:21503 install -r c:\kaoyan3.1.0.apk
```

```
C:\Users\Administrator>adb -s 127.0.0.1:21503 install -r c:\kaoyan3.1.0.apk
3374 KB/s (9699881 bytes in 2.807s)
    pkg: /data/local/tmp/kaoyan3.1.0.apk
Success
```

3.卸载app

```
adb uninstall app包名
```

注意：app卸载时使用的包名，包名是唯一的。

```
C:\Users\Administrator>adb uninstall com.ta1.kaoyan
Success
```

4.如何获取app包名?

1. 直接询问开发（最简单）
2. 如果有apk文件，可以获取包名

```
aapt dump badging apk文件
```

```
C:\Users\Administrator>aapt dump badging c:\kaoyan3.1.0.apk
package: name='com.tal.kaoyan' versionCode='55' versionName='3.1.0'
sdkVersion:'10'
targetSdkVersion:'21'
uses-permission:'getui.permission.GetuiService.com.tal.kaoyan'
uses-permission:'android.permission.SYSTEM_ALERT_WINDOW'
application-icon-640: res/drawable-xxhdpi-v4/ic_launcher.png
application: label='猿题库?' icon='res/drawable-mdpi-v4/ic_launcher.png'
launchable-activity: name='com.tal.kaoyan.ui.activity.SplashActivity' label='' icon='ic_launcher'
uses-feature:'android.hardware.camera'
uses-feature:'android.hardware.camera.autofocus'
```

包名

启动app时进入的activity

3. 获取当前桌面运行app的包名和当前界面activity

```
adb shell dumpsys window | findstr mCurrentFocus
```

```
C:\Users\Administrator>adb -s 127.0.0.1:21503 shell dumpsys window |findstr mCurrentFocus
mCurrentFocus=Window{24b43a09 u0 com.android.settings/com.android.settings.Settings}
C:\Users\Administrator>
```

包名

当前界面activity

5.进入设备操作系统

由于android系统内核就是linux系统，所以进入android系统可以运行linux命令。

```
adb shell
```

```
C:\Users\Administrator>adb shell
root@G011C:/ # pwd
```

命令提示符如上图，表示已进入android操作系统，此时输入的命令是linux命令。

android系统中有几个文件夹比较重要：

- 1、/data/app目录，用来放用户安装的app的apk包。
- 2、/data/data目录，用来放app的安装目录，类似于windows上的program files目录。
- 3、/sdcard目录，扩展卡目录，用来放用户数据。

输入exit或者按ctrl+c退出android系统。

6.推送文件==》将电脑上的文件发送到手机上

```
adb push 电脑上文件路径 手机存储的路径
```

```
C:\Users\Administrator>adb push d:\Users\Desktop\al.png /sdcard/test48
4686 KB/s (111791 bytes in 0.023s)
```

7.拉取文件==》将手机上的文件发送到电脑上

```
adb pull 手机上文件路径 电脑上存储的路径
```

```
C:\Users\Administrator>adb pull /sdcard/test48/test1 d:\Users\Desktop
1 KB/s (6 bytes in 0.003s)
```

注意：文件路径不要使用中文

8.获取手机上的日志

在实际工作中，很多时候发现bug之后，往往如果能提供一些报错日志信息给到开发，能够更快帮助开发定位bug。

对于web项目：前端（浏览器打开开发者工具F12），后端（服务器上，首先找到日志目录log，然后查看日志文件 cat，tac，head，tail -f，more，less等）

对于app项目：前端（android程序日志通过adb logcat命令查看），后端日志查看与web一样

```
adb logcat
```

1) 根据日志等级来获取日志

logcat日志有不同的级别：

1. V, verbose次要信息
2. I, info系统信息
3. D, debug调试信息
4. W, warning警告信息
5. E, error错误信息

作为测试工程师，如果想查看logcat日志，重点关注W和E级别的信息。可以在logcat日志中查询包名、exception、error、crash等关键字。

```
adb logcat *:W
```

获取警告以上级别的日志信息。

2)根据时间来获取日志

```
adb logcat -v time
```

3) 导出日志到文件

```
adb logcat > 日志文件
```

4) 根据关键字过滤

```
adb logcat | findstr 关键字
```

将以上几种可以一起使用：

```
adb logcat -v time *:E |findstr kaoyan >d:\kaoyan.log
```

9.屏幕截图

```
adb shell screencap 文件保存的路径
```

```
C:\Users\Administrator>adb shell screencap /sdcard/test48/a2.png  
C:\Users\Administrator>adb pull /sdcard/test48/a2.png d:\Users\Desktop  
2891 KB/s (29177 bytes in 0.009s)
```

10.无线连接设备

在工作中，有时用数据线usb连接设备会不方便操作，所以可以通过无线网络来连接设备。

前提：

- 1) 电脑和手机设备连接同一局域网；
- 2) 手机已经通过数据线能够连接上电脑；
- 3) 手机开发者选项中usb调试开启。

具体操作：

- 1) 开通电脑连接设备的端口（此时需要连接数据线）

```
adb tcpip 5555      其中端口可以自定义
```

- 2) 无线连接设备（可以拔掉数据线）

```
adb connect 设备ip:5555
```

其中设备ip地址在手机上查看wifi网络连接详细信息。

1.app抓包

使用fiddler工具进行抓包，查看app端与服务端的交互。

前提：设备（真机）必须与电脑在同一个局域网。

1.设备端设置代理

将电脑安装fiddler设置为设备的网络代理。

- 1.打开手机设置——WLAN；
- 2.找到已连接的网络，长按，选择修改网络；



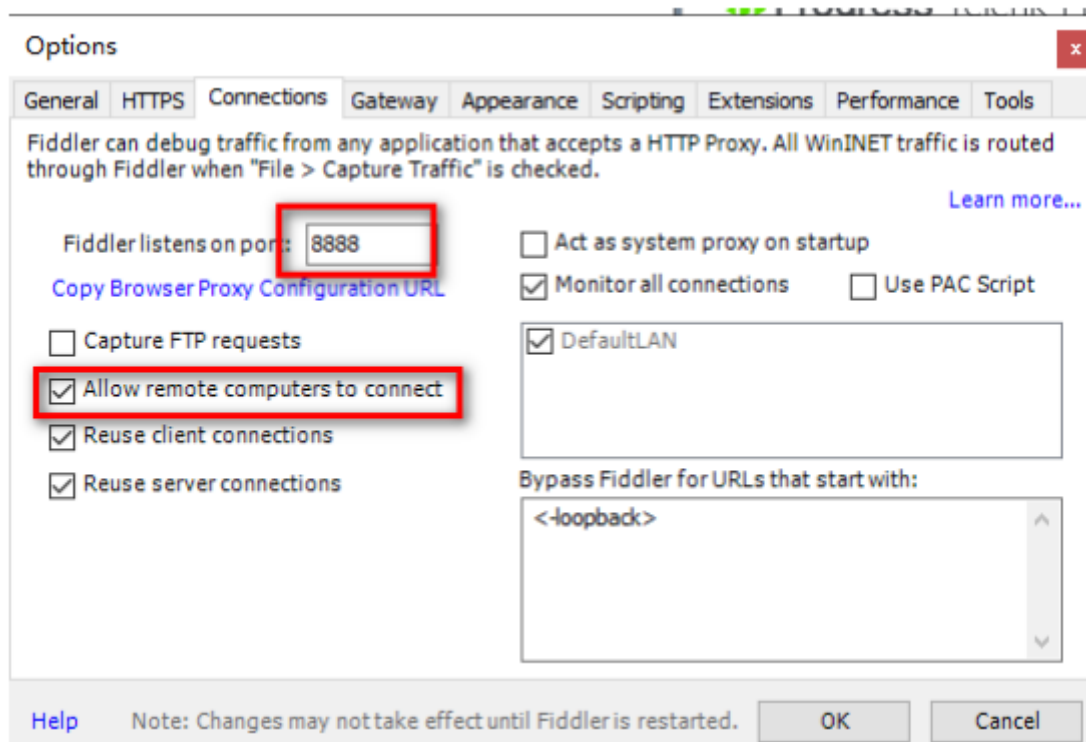
3) 勾选高级选项，设置为手动代理，并且填写代理的主机和端口，也就是对应的电脑的ip地址以及抓包

工具监听的端口。



2.fiddler抓包工具设置

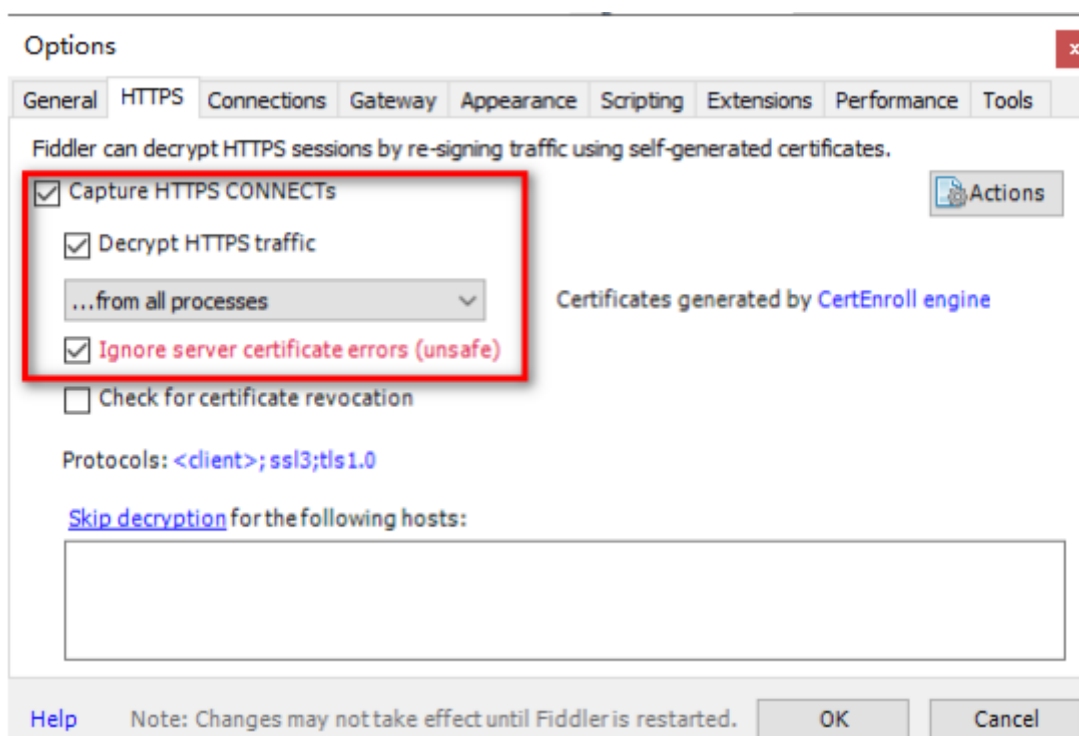
设置菜单Tools——Options——Connections，设置勾选允许远程连接，确定：



然后重启fiddler。

3.fiddler抓取https包

1)设置fiddler，在菜单Tools——Options——HTTPS中勾选允许抓取https包：



2)需要在设备上安装fiddler的证书。

打开浏览器，输入地址：电脑ip:8888



注意事项：

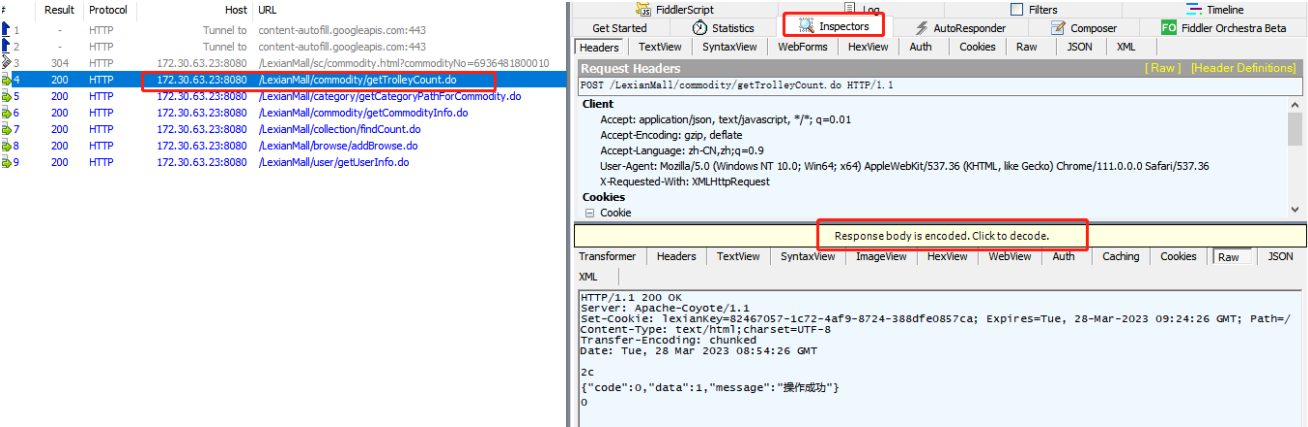
- 1) 设置fiddler完成后，需要重启fiddler。
- 2) 在设备上安装fiddler证书时，需要开启fiddler。
- 3) 在设备安装fiddler证书时，需要设置手机密码。
- 4) 安装证书时，不要关闭fiddler。
- 5) 抓包完成后，需要关闭手机代理，否则可能出现无法上网的情况。
- 6) 从android7.0开始，android系统不再信任用户级的证书，只信任系统级的证书，所以对于高版本android，需要在安装证书后，将证书从用户证书目录移动到系统证书目录下（部分机型可能需要root）

4.AutoResponder模块使用

app-----fiddler-----服务器

app-----fiddler（伪造数据）

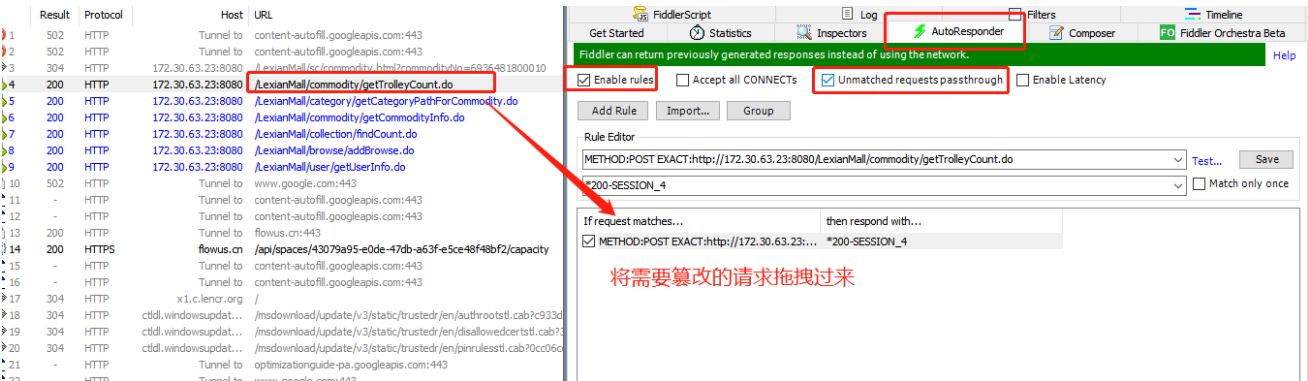
1.找到要篡改的请求响应数据



注意：点击上图中红框，进行响应数据的格式转换。

2.添加自动响应规则

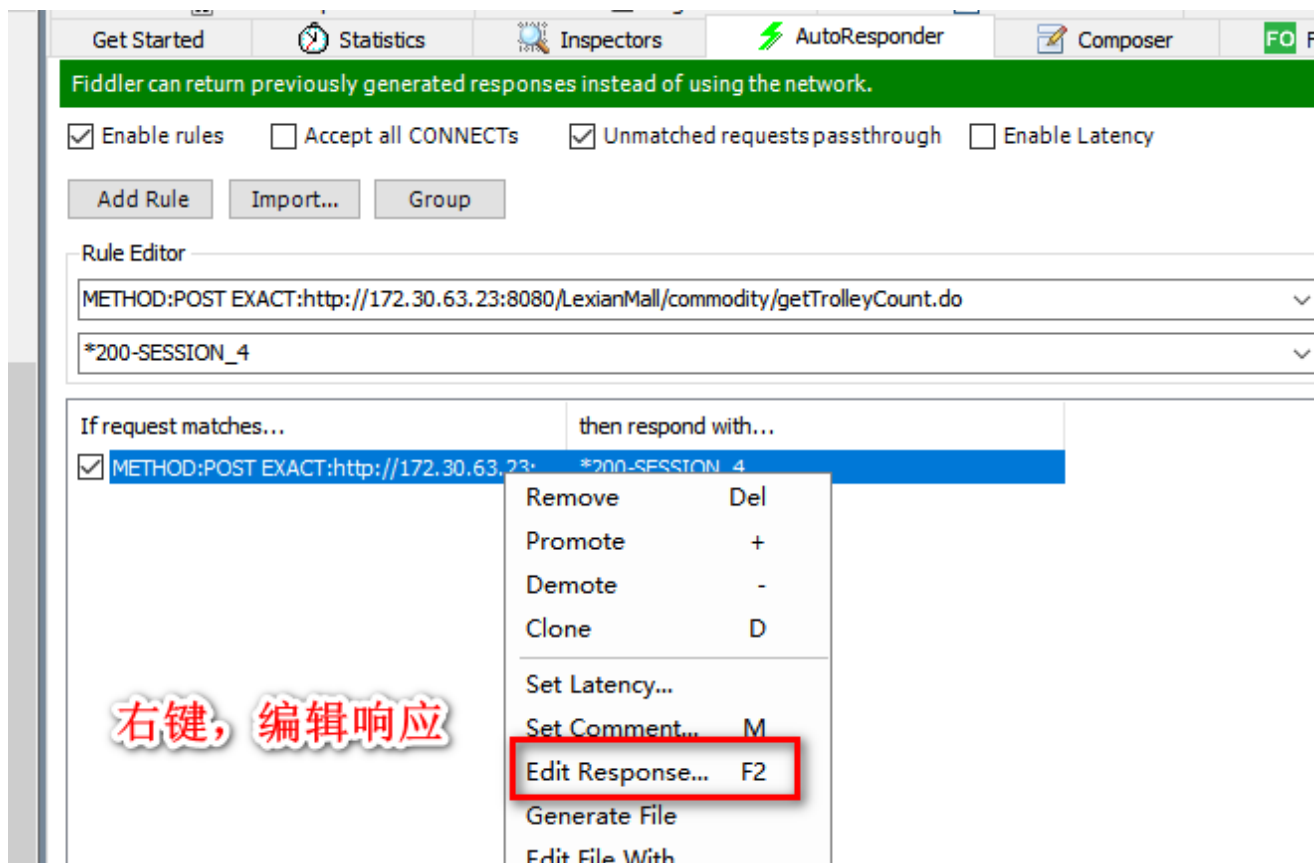
直接拖对应的请求到自动响应规则列表。



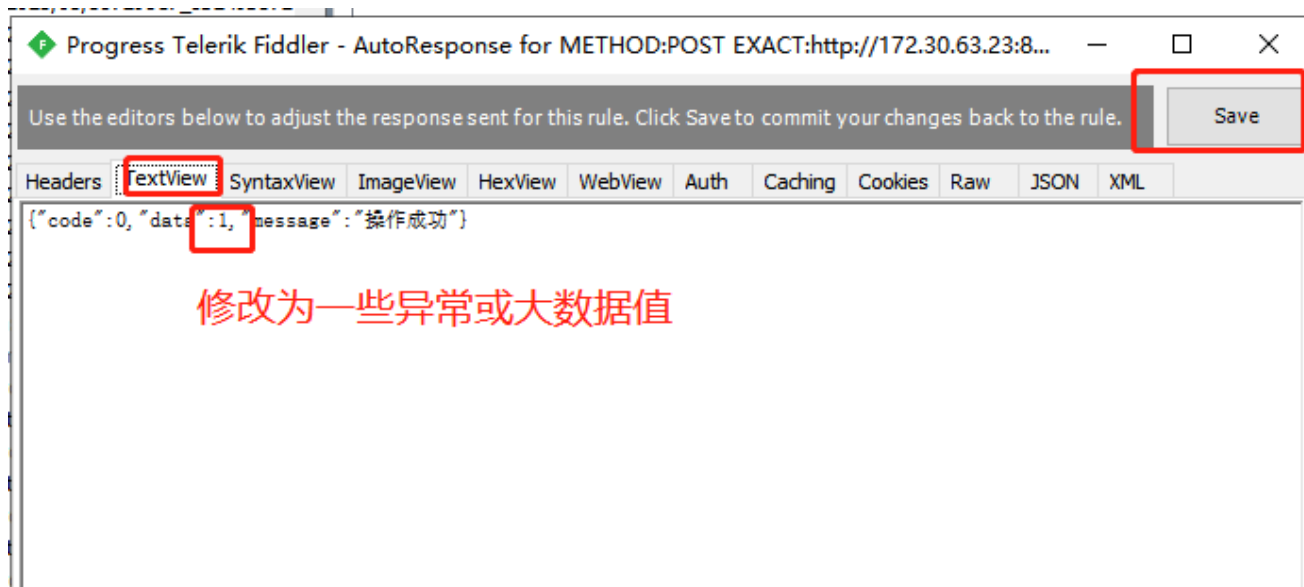
注意勾选相应选项。

3.修改响应数据

在自动响应规则上右键，点击编辑响应数据。



切换到textView标签页，修改data后面的数字为10000，点击保存。



刷新首页，可以看到首页我的购物车数量显示如下，出现显示不全的bug。



AutoResponder模块可以快速检查各种超大的数据或者异常数据。AutoResponder模块可以看成挡板或者桩。

5.fiddler设置断点

在测试过程中，比如一个购买的金额输入框，输入框前端做了限制100-1000，那么我们测试的时候，需要测试小于100的情况下。很显然前端只能输入大于100的。此时我们可以先抓到接口，修改请求参数，绕过前端，传一个小于100的数，检查服务端的功能是否正确。

可以使用fiddler进行此类操作，可以使用断点功能，修改请求或响应数据，或者直接模拟服务器响应。

使用Fiddler进行HTTP断点调试可以做到：

①修改HTTP请求头信息。例如修改请求头的UA, Cookie, Referer信息，通过“伪造”相应信息达到相应的目的（调试，模拟用户真实请求等）。

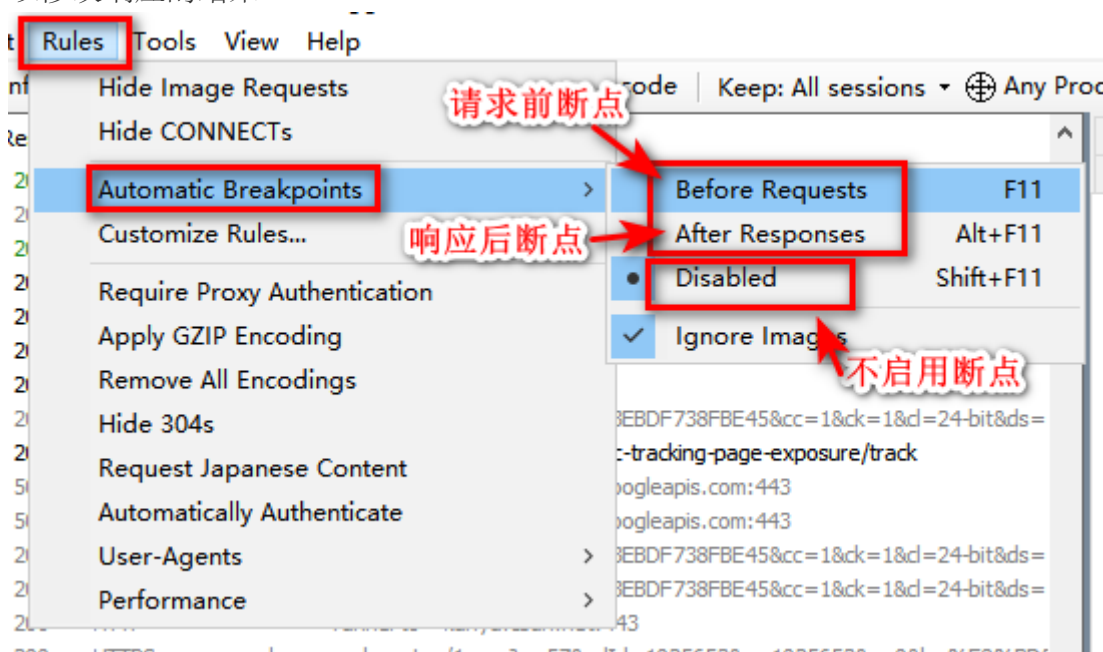
②构造请求数据，突破表单的限制，随意提交数据。避免页面js和表单限制影响相关调试。

③拦截响应数据，修改响应实体。

1.设置断点方式

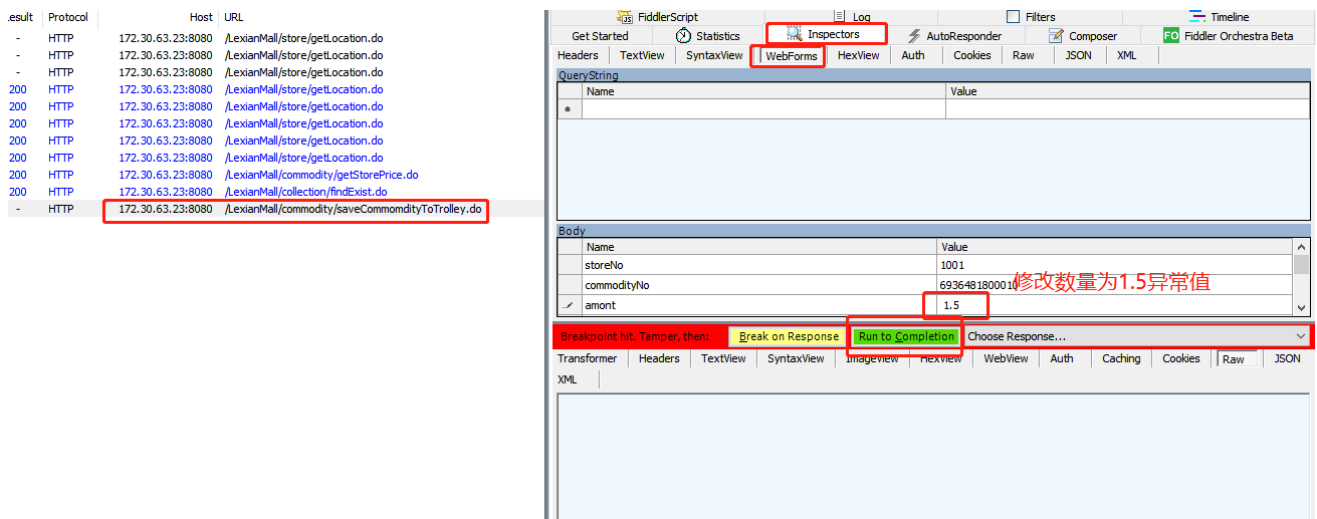
fiddler菜单栏->rules->automatic Breakpoints->选择断点方式，这种方式下设定的断点会对之后的所有HTTP请求有效。有两个断点位置：

before request。也就是发送请求之前，Fiddler代理中转之前，这时可以修改请求的数据。
after response。也就是服务器响应之后，但是在Fiddler将响应中转给客户端之前。这时可以修改响应的结果。

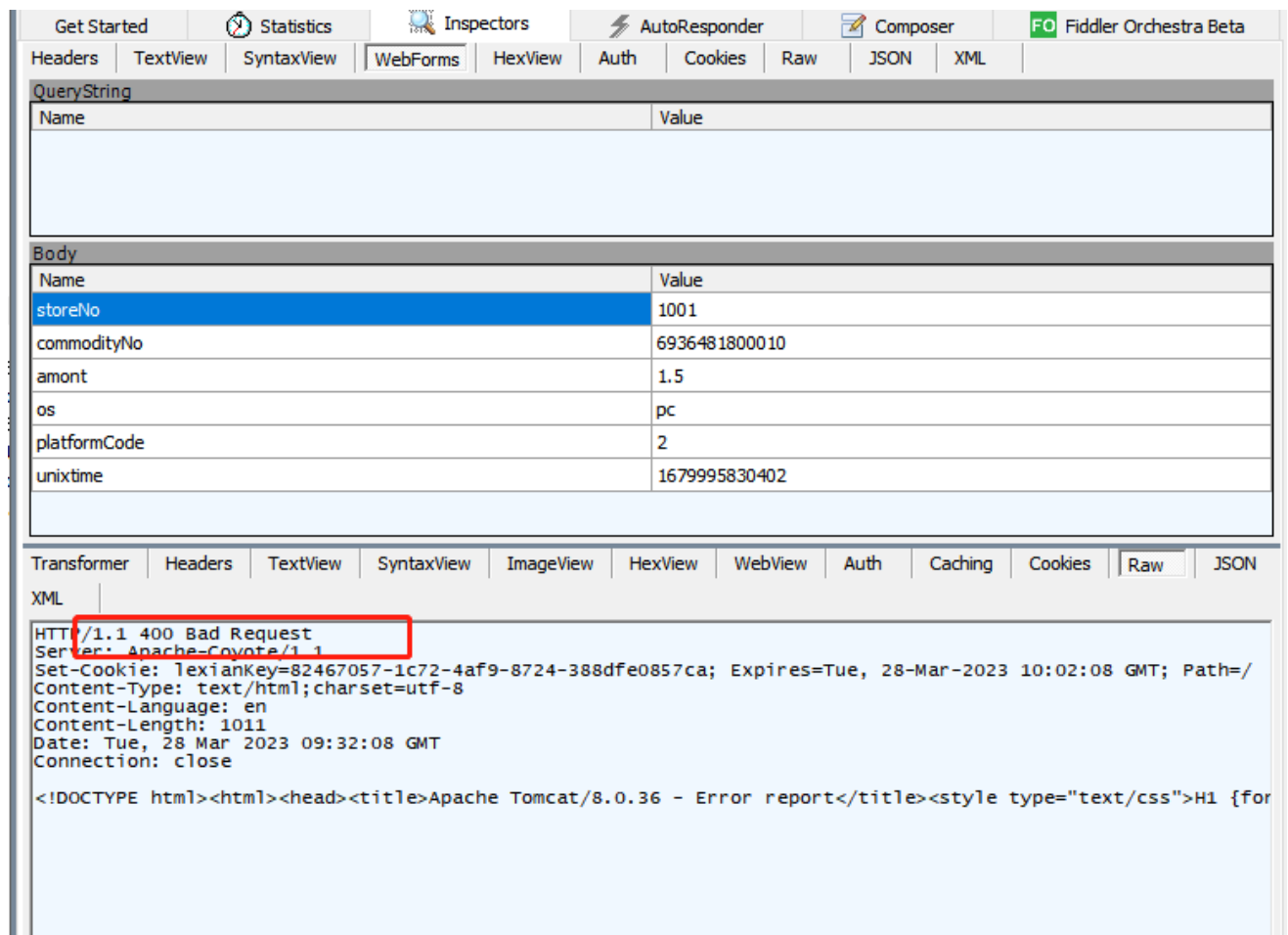


2.设置请求前断点

在左侧列表中选中请求数据。点击右侧Inspectors->WebForms，可以看到请求携带的参数，此时可以修改参数值，或添加携带参数。修改完请求参数后，点击下方【Run to Completion】，则进行消息发送。比如修改拦截加入购物车请求，修改数量为1.5

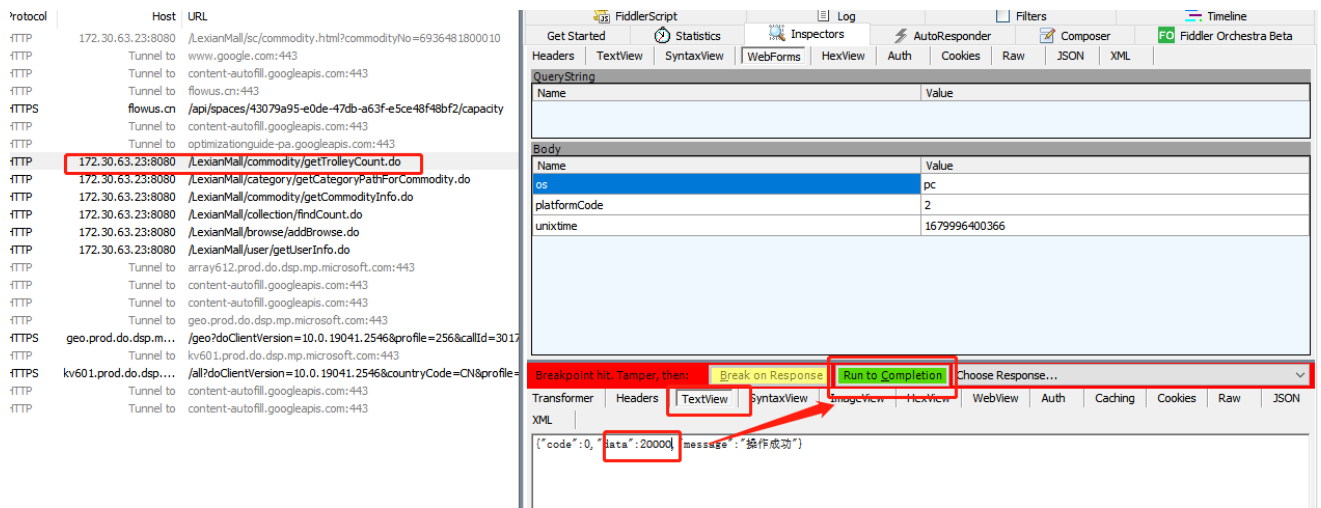


点击Run to Completion发送修改后请求，可以查看服务端返回响应码为400



3.设置响应后断点

在左侧列表中选择请求。在右侧Inspectors->TextView栏修改返回数据，然后点击【Run to Completion】，则进行消息发送



点击Run to Completion发送修改后响应后，可以查看页面上显示的购物车数量展示不全。



断点操作总结：

- 1.设置断点
- 2.页面操作
- 3.修改请求或响应
- 4.点击绿色按钮 run to completion 完成转发修改后的请求或响应

2.弱网测试

1.弱网测试背景

在移动互联网时代，用户会在各种网络状况下使用我们的APP。他们使用3G或4G网络，甚至还在用2G，现在的人们更习惯在上下班的路上去关注一些新闻，看看股市，小说，直播，玩游戏等等。那么就会面临一个问题，在地铁里，隧道，甚至是电梯,车库等等的弱网环境场景里。在这个时候，我们就需要针对这些场景，去关注一下软件的运行状态，以及弱网环境下，出现丢包、延时软件的处理机制。避免造成用户的流失。

2.测试关注点

- 用户体验

APP使用过程中，弱网的高延迟和高丢包，在实时性要求非常高的场景，容易伤害用户体验

- 非正常情况下，出现bug概率会增加

在解决日常的支持需求中，经常会遇到一些用户反馈一些无法简单复现的bug，有很大一部分的bug是由于用户自身的网络环境波动，或者是本身网络环境就较为恶劣，而App在面对这种恶劣的网络环境的健壮性不够，导致会出现一些意想不到的bug

3.弱网测试指标关注

- 丢包

丢包应该是最常见的问题。在TCP协议中，需要不停的发送请求，来确认连接状态，一旦发生丢包，就需要重传。这个时候就需要去检查产品的处理机制，给予什么提示，如果未响应怎么处理这些。

- 延时

延时也是很常见的问题。由于网络太差，产生了网络波动，导致数据包在传输的时候出现抖动。可能导致请求出现超时的现象。这个时候就需要给予相应的提示，或者是其他的处理方式。

弱网测试工具：fiddler, qnet

可以使用fiddler工具来模拟app的弱网环境。

前提：将抓包工具设置为手机的代理。

1) 使用fiddler模拟弱网

1) 设置弱网的规则，在菜单Rules——Customize Rules



```
Fiddler ScriptEditor
File Edit Go Insert View Help

if ({null != gs_ReplaceToken} && {oSession.url.indexOf(gs_ReplaceToken)>-1}) { // Case sensitive
    oSession.url = oSession.url.Replace(gs_ReplaceToken, gs_ReplaceTokenWith);
}
if ({null != gs_OverrideHost} && {oSession.host.toLowerCase() == gs_OverrideHost}) {
    oSession["x-overridehost"] = gs_OverrideHostWith;
}
if ({null!=bpRequestURI} && oSession.uriContains(bpRequestURI)) {
    oSession["x-breakrequest"]="uri";
}
if ({null!=bpMethod} && {oSession.HTTPMethodIs(bpMethod)}) {
    oSession["x-breakrequest"]="method";
}
if ({null!=uiBoldURI} && oSession.uriContains(uiBoldURI)) {
    oSession["ui-bold"]="QuickExec";
}
if (m_SimulateModem) {
    // Delay sends by 300ms per KB uploaded.
    oSession["request-trickle-delay"] = "300";
    // Delay receives by 150ms per KB downloaded.
    oSession["response-trickle-delay"] = "500";
}
if (m_DisableCaching) {
    oSession.oRequest.headers.Remove("If-None-Match");
    oSession.oRequest.headers.Remove("If-Modified-Since");
    oSession.oRequest["Pragma"] = "no-cache";
}
```

弱网规则：==默认设置

上行：300ms/KB

下行: 150ms/KB

网速: 10 kbps =====表示1s 10k bit

1Byte=8bit

如果网速上行为16kbps====> ms/KB 上行延时设置为多少?

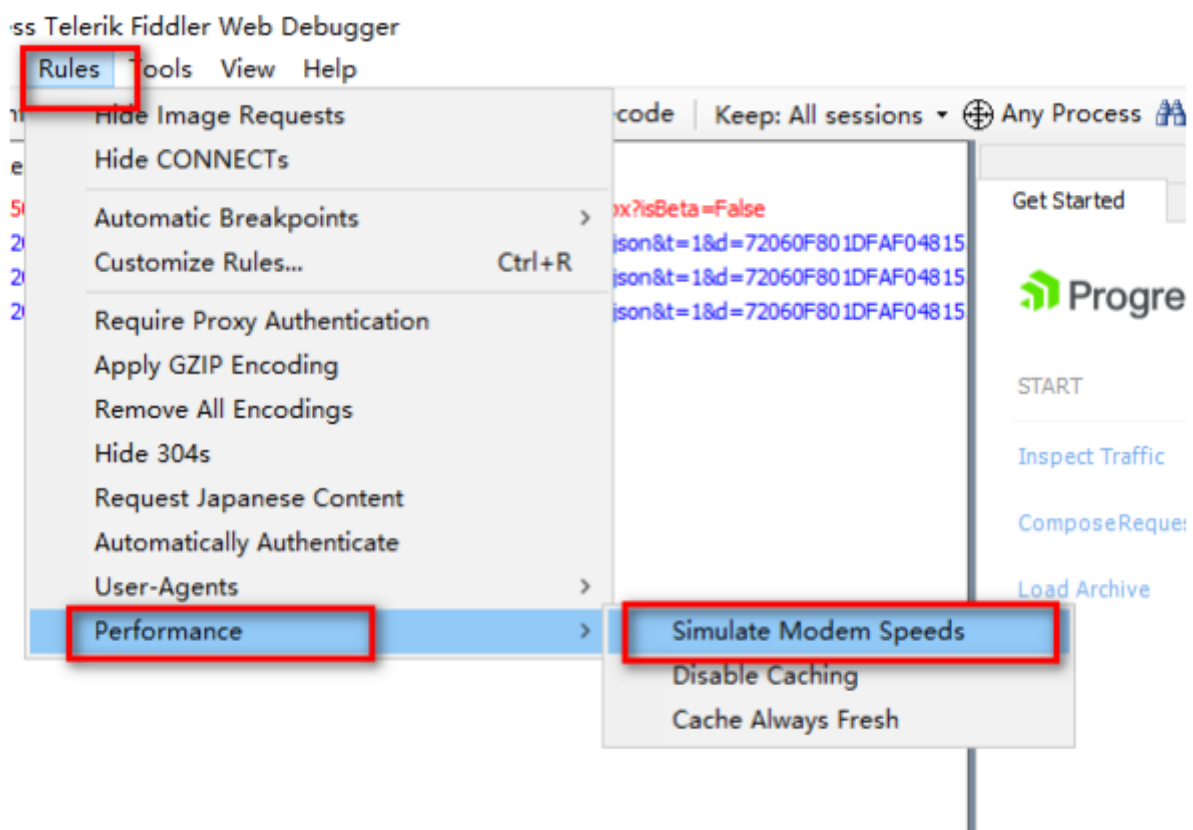
$16\text{kb/s} \Rightarrow 16\text{K}/8 = 2\text{KB/s} \Rightarrow 0.5\text{s}/\text{KB} \Rightarrow 500\text{ms}/\text{KB}$

如果网速下行为30kbps====> ms/KB 下行延时设置为多少?

$30\text{kb/s} \Rightarrow 30/8 \text{ KB/s} \Rightarrow 8/30 \text{ s}/\text{KB} \Rightarrow 8*1000/30 = 267 \text{ ms}/\text{KB}$

总结计算公式: 如果网速 x kbps ====> 延时设置为: $8*1000/x \text{ ms}/\text{KB}$

2) 打开弱网模拟开关



3.app性能测试工具

移动app性能测试工具:

- 1、emmagee, 不支持android7.0及以上版本, 网易开源的性能测试工具。
- 2、gt, 腾讯的测试工具。
- 3、solopi, 蚂蚁集团开源的手机测试工具。

4、monkey，android系统自带的工具。

5、PerfDog，腾讯的测试工具。

solopi工具可以：

- 1、录制回放，简单的自动化测试
- 2、性能测试
- 3、一机多控，用于提升兼容性测试效率

solopi工具使用：

一、安装并运行solopi。

打开solopi，点击性能测试，如出现下图提示，需要开启adb权限。



打开cmd，输入以下命令即可。

```
adb tcpip 5555
```

二、运行性能测试



三、选择被测app和性能指标



四、双击目标应用，启动被测app



五、点击绿色的箭头进行录制，录制结束后点击红色的圈停止录制。



六、点击黄色的首页图标回到solopi，再次进入性能测试，进入录制数据查看。



可以查看到各个性能指标的图表。





4.monkey测试

1.什么是monkey测试?

monkey测试：顾名思义，就是像猴子坐在电脑前会乱点一样。monkey测试是通过工具模拟用户对app进行随机滑动，拖动，点击，按键等随机动作，来完成对app的压力测试，测试app的稳定性。

monkey测试优点：

简单，快速，方便

可以发现人工很难发现的问题

缺点：

操作时随机，不能针对某一个模块进行测试

随机操作，很难重现问题。

2.monkey测试

monkey是android自带的一款压力测试的小工具，主要测试app的稳定性，测试app是否会出现闪退，崩溃等异常。

命令：

```
adb shell monkey [可选参数] 次数
```

3.monkey参数

1) -p 指定包名

```
adb shell monkey -p app的包名 次数
```

2) -v 指定日志级别

monkey日志分为三个级别：

-v	日志最简洁，只显示几倍启动，测试完成等少量信息。默认级别
-v -v	日志较为详细，还会显示activity的事件信息。
-v -v -v	日志最详细。

3) monkey日志导出

```
adb shell monkey -p 包名 -v -v -v 次数 >日志保存的路径
```

如何分析monkey日志：

1) 查看monkey是否执行完成。执行完成，最后会提示Monkey Finished，表示monkey执行结束。

2) 在导出的monkey日志中查看关键字信息：

- Crash 奔溃(闪退)
- ANR 应用程序无响应 (Application Not Response)
- Exception 错误，一般是应用程序报错。ActivityNotFoundException, NullPointerException, SQLException等

- Switch 表示页面切换，当出现错误时，可以查看switch的日志，分析是哪个页面出现问题。

```
// activityResuming(com.tal.kaoyan)
// CRASH: com.microvirt.launcher (pid 719)
// Short Msg: java.lang.NullPointerException
```

```
// Exception from procrank:
java.io.IOException: Error running exec(). Command: [procrank]
Working Directory: null Environment: null
anr traces:
```

4) --throttle 指定间隔时间

该参数用来指定随机操作之间的间隔时间，单位是毫秒。指明每次操作的间隔，一般设置为500，对应500ms（半秒），模拟人的正常速度。如果设置为更小的值，模拟人的快速点击，属于性能测试中压力测试。

```
adb shell monkey -p 包名 --throttle 100 -v -v -v 次数 >日志保存的路径
```

5) -s 设置种子数

该参数用来设置seed(种子数)，种子数的作用就是用来生成随机序列，不同的种子数会产生不同的随机序列。

那么当我们在进行monkey测试时，如果发现一个问题，可以通过设置相同的种子数来进行重复测试，用来验证缺陷是否修复。

```
adb shell monkey -p 包名 -s 种子值 --throttle 100 -v -v -v 次数 >日志保存的路径
```

6) 设置事件百分比

```
adb shell monkey -p com.tal.kaoyan --pct-motion 100 -v 1000
```

```
// Seeded: 1649893931560
// Event percentages:
// 0: 0.0%
// 1: 100.0%
// 2: 0.0%
// 3: 0.0%
// 4: -0.0%
// 5: 0.0%
// 6: 0.0%
// 7: 0.0%
// 8: 0.0%
// 9: 0.0%
// 10: 0.0%
```


事件编号	事件名称	含义	说明
0	--pct-touch	触摸事件	点击
1	--pct-motion	手势事件	直线滑动
2	--pct-pinchzoom	缩放事件	放大缩小手势
3	--pct-trackball	轨迹球事件	曲线滑动
4	--pct-rotation	屏幕旋转事件	横屏竖屏切换
5	--pct-nav	基本导航事件	物理上下左右按键
6	--pct-majornav	主要导航事件	物理菜单键、中间键
7	--pct-syskeys	系统按键事件	物理Home键、返回键、音量调节键
8	--pct-appswitch	启动Activity事件	切换界面
9	--pct-flip	键盘事件	键盘开启和关闭
10	--pct-anyevent	其他类型事件	物理字母按键、数字按键

4.移动端的操作系统

1.Android 安卓 google开发

android系统是一个基于linux的操作系统。

- Android开发语言
java
- Android客户端安装文件包 *.apk
- Android四大开发组件
 - activity，活动，可以简单理解成界面。
 - service，后台服务。比如微信APP退出后，还能收到微信的信息。
 - BroadcastReceiver，广播接收。接收Android系统发送的广播信息，当出现断网、低电量、来短信等情况下，Android系统会发送广播给要接收广播的app，app会针对广播自动处理，比如断网了，微信会提示网络已经中断。
 - ContentProvider，内容提供。一般用于app之间的数据分享，由于app之间的数据是隔离的，当一个应用想把自己的数据分享给其他应用时，就需要使用内容提供技术，例如分享，通讯录的信息可以被其他应用调用。
- Android版本

可以通过访问以下网址查看android系统的历史版本。

<https://developer.android.google.cn/about/versions>

android系统不仅仅是一款手机端或者平板上的操作系统，支持可穿戴设备、TV、汽车、物联网等。

版本号	API级别	主要特性
V4.0	14	统一了手机和平板操作系统，基于linux3.0.1内核去开发。
V5.0	21	采用全新MaterialDesign界面，用ART虚拟机替换Dalvik虚拟机，提升性能。
V6.0	23	全新的权限机制，在原有AndroidManifest.xml声明权限的基础上，新增了运行时权限动态检测。增加AndroidPay付费功能。
V7.0	24	支持多视窗，通知增强，提供配置文件指导的JIT/AOT编译。
V8.0	26	提供TensorFlowLite, 支持画中画，提供智能文本选择，提供自动填写，提供GooglePlayProtect。
V9.0	28	借助于AI提供后续操作推荐，利用应用切片引导操作，使用新的手势导航栏，支持屏幕手动旋转。

作为测试工程师，重点关注对于测试工作会产生影响的版本。

- V5.0：采用全新界面设计技术。需要考虑5.0及以后版本、5.0以前版本的适配。
- V6.0：采用全新权限机制。需要考虑6.0及以后版本、6.0以前版本的适配。
- V10.0：支持可折叠屏幕，支持5g网络。

api级别是指android系统给android应用提供的接口版本的版本。

android应用-----（api）-----android系统-----手机硬件

2.iOS 苹果===封闭

- iOS开发语言
objective-c xcode
- iso客户端文件包 *.ipa

3.HarmonyOS 鸿蒙 华为开发

5.APP应用测试的要点

对于APP项目的测试，一般是进行系统测试。测试主要从业务功能和非业务功能两个方面去考虑。

5.1 业务功能测试

根据软件的需求规格说明书，设计文档或者用户需求来验证app各项功能的实现。

回顾web测试要点:

- 功能测试
- UI测试（界面是否符合UI原型图，排版布局等）
- 兼容性测试（浏览器兼容性（5大浏览器），操作系统兼容性，分辨率兼容性）
- 易用性测试（系统是否好用，是否方便，是否吸引用户）
- 性能测试(测试服务端性能，在一定压力/负载==并发用户量情况下，检测被测试系统的响应时间以及资源消耗等指标情况，看系统是否会出现异常。jmeter loadrunner)
- 安全测试（渗透测试，借助工具对系统进行扫描，检查是否有系统漏洞。）

5.2 非业务功能测试，也叫专项测试

5.2.1 兼容性测试

- 为什么要做兼容性测试

APP在不同的机型上由于软件、硬件等不同可能出现各种各样的问题，因此需要做兼容性测试。

- 兼容性测试的关注点

1. 手机品牌型号

- 覆盖市场的主流机型
- app线上用户机型排名

2. 操作系统版本

- Android系统

不同的版本，比如7.0,8.0, 10.0, 11.0

- iOS系统

不同的版本，比如11.x、12.x、13.x、14.6

3. 屏幕的尺寸和分辨率

- 屏幕的尺寸

5.5、4.7、5.8

- 分辨率

2560×1440 超清 1920×1080 高清，1280×720标清

- 不同屏幕类型 异形屏 水滴屏 刘海屏 曲面屏

4. 网络

- 2G
- 3G
- 4G
- 5G
- WIFI

注意：兼容性测试需要在一定数量的真机上进行，通常会选择热门机型进行手工测试，除此之外还可以借助一些云测平台。

实际工作中兼容性测试主要考虑：

- 1、针对当前主流机型进行完整的功能测试和性能测试。
- 2、针对其它机型，利用云测平台来进行基本的功能测试和性能测试。

云测平台比如testin、阿里云、腾讯云、华为云等。

5.2.2 安装、卸载和升级测试

手机端app通常是C/S架构的软件，需要考虑安装、卸载和升级相关的测试。

(1)安装测试关注点

1.安装app的方式：

- 应用市场安装（华为应用市场，小米应用市场，腾讯应用宝）
- 官网/二维码/手机浏览器下载apk文件，手动安装
- 电脑手机助手下载apk安装
- adb工具安装

2.安装测试点：

- 是否可以在不同版本的手机上安装，比如考虑android10，android9的安装；
- 不同的安装方式，能否正常安装app
- 安装app后，能否覆盖安装，能否多次安装
- 卸载app后，能否重新安装
- 安装过程中出现中断是否可以恢复，比如安装过程中来电，通话结束能否继续安装；
- 安装中出现异常，恢复后是否能再次安装，比如安装过程中断电关机，手机空间不足，apk文件损坏是否提示等情况；

(2)卸载测试关注点

1.卸载app的方式：

- 应用市场/第三方工具卸载
- 桌面长按卸载
- 设置-应用程序管理功能界面卸载
- 电脑手机助手卸载
- adb工具卸载

2.卸载测试点：

- 使用不同卸载方式，能否正常卸载
- 卸载是否干净，是否有残留文件
- 卸载后，能否再次安装
- 卸载过程出现中断，比如来电等，是否能继续卸载
- 取消卸载后，软件能否仍然能正常运行；

(3) 升级测试关注点

1.升级app的方式：

- 应用市场升级
- APP应用内弹框提示升级
- 直接覆盖安装进行升级

2.升级测试点：

- 当有新版本时，要提示更新，比如有新版本更新时，当我们打开应用，是否有更新提示，点击升级，可以直接下载新版本升级；
- 升级后，版本号是否正确，功能是否正常使用，包括新功能和原有功能都正常使用；
- 是否强制升级，比如12306，部分游戏，必须升级，否则无法使用app。对于非强制升级，取消升级是否能正常使用。
- 跨版本更新时，能否更新成功等，比如直接从1.0跳过2.0直接升级到3.0。
- 向下升级，比如当前安装2.0版本，能否覆盖安装1.0版本。

5.2.3 中断测试

又叫交叉事件测试或冲突测试或者干扰测试或者中断测试。

是指一个功能正在执行过程中，另一个事件或操作对该过程进行干扰的测试。关注在app运行过程中，出现交叉事件，app本身是否运行异常。比如：在app使用过程中接听电话或者收到短信提醒等干扰。

常见交叉事件测试关注点：

- APP运行时拨打或接听电话；
- APP运行时发送/接收信息；
- APP运行时最小化到后台；
- APP运行时切换网络（4G、WiFi）；
- APP运行时低电量提示；
- APP运行时插入耳机；
- APP运行时按键，音量键，电源键；
- APP运行时虚拟导航键，比如圆圈圈
- APP运行时使用相机、计算器等手机自带的应用；
- App运行时拔插充电器。

5.2.4 PUSH测试(消息推送测试)

PUSH是指APP消息推送功能，主要用于提醒或唤醒用户，消息推送一般可以自定义推送对象，有全部推送或精确推送。

- PUSH测试关注点：
 - 用户能够正常接收到推送消息；
 - 推送到手机后，根据手机本身消息通知设置，如果设置打开，可以在手机通知栏看到消息打开消息能唤醒app查看消息详情；如果设置关闭，需要手动打开app才可以看到消息。
 - 通知内容的校验： 通知的标题，图片，查看通知进入对应的app界面

- 当用户未收到通知时，能够重新推送该通知
- **push**消息是否按指定业务规则，比如根据用户个人喜好；
- 设置不接收推送消息时，用户是否会收到**push**消息；
- 当**push**消息是针对特定用户时，检查收到**push**消息的用户是否与指定用户的身份相符；
- 用户**app**离线时，是否能收到**push**消息。
- 推送时间段验证：免打扰时间段，不推送通知；指定时间段内，能够正常推送通知。
- **push**消息推送的应用分为两种
 - 即时消息应用：**qq**、微信、钉钉、**soul**、陌陌、探探等聊天工具
 - 一般应用：美团、饿了么、头条
- **push**消息展示的形式
 - 前台：弹框提示
 - 后台：消息通知栏
- **push**消息针对不同的用户群体：全部用户/部分用户/特定用户
 - 全部用户：使用过该产品的用户
 - 热门用户：一天使用三次以上。
 - 常规用户：三天使用一次以上。
 - 沉默用户：一个月使用一次；
 - 流失用户：一个月没有使用了；
 - 小白用户：第一次使用，软件使用的引导/指导。

5.2.5 权限测试

权限是属于操作系统管理的范围

- 权限：
 - 一般权限：该权限不涉及到用户隐私数据，系统会自动授权，比如说：联网，震动
 - 危险权限：该权限涉及到用户隐私数据，需要用户手动授权
- 常见的危险权限：
 - 位置信息
 - 摄像头
 - 麦克风
 - 文件存储
 - 通讯录
 - 手机信息
 - 电话/短信
- 权限的操作：
 - 允许
 - 拒绝
 - 在使用期间允许
 - 仅本次允许

权限的常见测试点：

权限本身不是测试的重点，重点是关注依赖权限的功能

以位置信息为例：

- 允许位置信息权限，app能够正常获取到app的位置，依赖的功能能够正常使用
- 拒绝权限，app无法获取到位置信息，需要关注，在没有权限的时候，依赖该权限的功能不能报错，关注用户体验，不能闪退，白屏等
- 在使用允许，app能够在使用期间正常获取到位置信息，依赖的功能能够正常使用
- 仅本次允许，app能够在本次正常获取到位置信息，依赖的功能能够正常使用。
- 拒绝权限后，再次授权，app能够正常获取到位置信息
- 允许权限后，再拒绝权限，app在没有权限的时候，依赖该权限的功能不能报错，关注用户体验，不能闪退，白屏等

5.2.6 性能测试

性能是衡量app质量的一个重要指标。这里指的是客户端的性能测试，不包括服务端。

App性能测试常见指标：内存、CPU、流量、电量、启动速度、界面切换速度（fps=每秒帧数）等。

性能测试的关注点：

- APP的启动时间是否过长；
- APP使用时对CPU/内存的占用情况；
- APP使用时，电量流量的消耗情况；
- APP的界面切换速度是否太慢，不能有明显卡顿；
- 反复长期的操作情况下，系统资源的使用情况，会不会资源占用越来越多。

APP性能测试的常见指标：

- 时间

主要观察app的启动时间。启动时间又分为首次启动时间和非首次启动时间，非首次启动时间又分为冷启动时间（是被测app完全退出后再启动）和热启动时间（被测app未完全退出再启动）。

首次启动时间>冷启动时间>热启动时间

实际工作中主要测试的是冷启动时间。

可以利用adb命令：`adb shell am start -W packagename/MainActivity` 查看冷启动时间，例如查看考研帮的冷启动时间：

```
adb shell am start -W
com.tal.kaoyan/com.tal.kaoyan.ui.activity.SplashActivity
```

在结果中查看TotalTime的值即可：



```
CA 命令提示符
Microsoft Windows [版本 10.0.19042.1415]
(c) Microsoft Corporation。保留所有权利。

C:\Users\admin>adb shell am start -W com.tal.kaoyan/com.tal.kaoyan.ui.activity.SplashActivity
Starting: Intent { act=android.intent.action.MAIN cat=[android.intent.category.LAUNCHER] cmp=c
om.tal.kaoyan/.ui.activity.SplashActivity }
Status: ok
Activity: com.tal.kaoyan/.ui.activity.SplashActivity
ThisTime: 783
TotalTime: 783 冷启动时间为783毫秒
WaitTime: 788
Complete

C:\Users\admin>
```

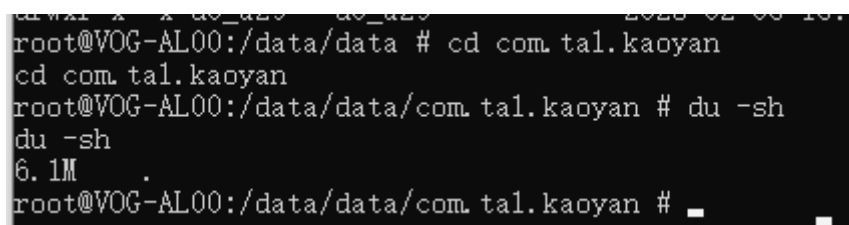
- 存储占用

存储包括外存储和内存储，外存储比如硬盘、u盘、光盘等，用于保存数据。内存储就是常说的内存，用于执行程序。对于手机而言，无论是外存储还是内存储都是比较紧张的，因此测试移动app都需要关注。

- 1)外存储占用

外存储占用包含：

- 1、apk文件大小，保存在/data/app目录下
- 2、app的安装目录大小，安装目录在/data/data目录下，用du -sh来进行计算。



```
root@VOG-AL00:/data/data # cd com.tal.kaoyan
cd com.tal.kaoyan
root@VOG-AL00:/data/data/com.tal.kaoyan # du -sh
du -sh
6.1M .
root@VOG-AL00:/data/data/com.tal.kaoyan #
```

- 2)内存储占用

可以使用一些Linux命令比如top命令来查看内存储占用。

实际工作中还是使用测试工具来检查内存占用。

- CPU占用

网站服务器cpu占用和移动app对cpu占用要求不同，服务器cpu占用希望是高峰期时超过90%，移动app的cpu占用希望是峰值不要超过40%。实际工作中通过测试工具来进行测试。

- 流量耗用/流量耗用

可以从以下几个场景下来进行测试：

- 1、安装时
- 2、前台运行时
- 3、后台运行时

流量耗用和电量耗用也一般利用测试工具来检查。

- 界面的切换速度

通过fps (每秒帧数) 来衡量, 如果出现fps的大幅波动, 就可能存在卡顿。

高于60帧: 显示更加流畅

低于60帧: 页面有卡顿的情况

常用客户端性能测试工具: perfdog (性能狗), solopi, 腾讯gt, 云测平台

1.web测试和app测试的区别?

web是属于B/S架构, app项目属于C/S架构, 测试过程都是从业务功能和非业务功能两个方面来进行测试的。

业务功能的测试都一样, 没有区别, 对照需求去分析, 编写测试用例进行测试。

非业务功能测试方面, 兼容性, 易用性, 性能等这些方面测试内容不同, 同时app测试还有一些专项测试, web测试是没有的。

兼容性测试: web测试主要测试浏览器, 操作系统, 分辨率几方面兼容性, app测试主要测试机型, 操作系统, 屏幕, 网络等几个方面兼容, 在测试时候可以借助一些云测平台来进行兼容性;

易用性: web测试在网页上主要是鼠标的操作, 而app主要是手势操作, 所以易用性上考虑也不同;

性能测试: web性能测试主要测试服务端, 在不同并发条件, 系统的性能指标如何(响应时间, 事务成功率, 资源占用等), 以及系统是否稳定; app性能测试除了同样要测试服务端以外, 还需要关注客户端的性能, 关注cpu, 内存, 电量, 流量, 启动时间, 界面切换速度等指标。

app专项类测试: 安装/卸载/升级测试, 中断测试, 消息推送测试, 权限测试, 弱网测试等。

2.app测试发现bug, 如何定位是前端bug, 还是后端bug? 如何获取日志?

首先看是否有交互, 如果没有交互, 都是前端的bug, 比如界面排版, 错别字, app安装等错误.....

如果有交互, 可以通过抓包的方式查看交互的请求和响应的正确性:

如果请求错误, 前端bug;

如果请求正确, 响应错误, 后端bug;

如果请求响应都正确, 但是界面错误, 也是前端bug。

一般界面, 兼容性等大部分都是前端bug, 数据正确性, 性能, 安全等大部分是后端bug。

前端日志: 通过adb logcat命令查看日志

后端日志: 连接服务器, 找到日志目录, 查看日志文件, 命令tail -f, more, less等.....

5.2.7 其他类型的测试

专项测试除了以上列出的内容，我们还要关注一些其他的测试点。比如用户体验，极限，权限等相关内容的测试。

- 用户体验
 - 界面是否美观，布局是否合理；
 - 是否可以保持登录；
- 极限（边界）
 - 内存满的时候安装APP；
 - 运行APP时手机断电（电量耗尽）；
 - 没有SD卡/双SD卡
 - 飞行模式
 - 系统时间有误（晚于和早于标准时间）
 - 第三方依赖（比如我们的App依赖第三方App，但是现在第三方App没有安装或者版本过低的测试情况）。

6.APP测试环境和发布平台

一、APP的项目环境

一般公司内部开发、测试人员使用不同的环境进行测试，以隔离测试过程中彼此之间的干扰，同时，线上用户会使用单独环境。

- 开发环境

指开发人员进行开发时调试用的环境

- 测试环境

指测试人员进行上线前测试的环境

- uat环境

验收环境，进行验收测试

- 预生产环境

测试环境到生产环境的过渡，与生产环境的配置和版本保持一致

- 生产环境（线上环境）

指正式提供对外服务的环境，产品的实际用户使用的环境

系统开发流程及对应环境的关系：

开发（开发环境）-->测试（测试环境）-->验收（uat环境）-->准备上线（预生产环境）-->上线（生产环境）

二、APP应用的发布

app完成开发之后，开发人员就会打出应用程序包，由测试人员安装测试。

- Android: APK测试包(weixin.apk)
- iOS: IPA测试包(weixin.ipa)

应用线上发布平台-release

产品测试完成后要在线上发布，让用户下载使用。安卓和iOS常用的发布平台和渠道：

- Android: 各类手机品牌应用商城、豌豆荚、应用宝、360手机助手等，提交apk到应用市场，审核通过后，即可上线；不通过打回修改再审核。
- iOS: AppStore、iTools

三、移动APP灰度发布策略

为什么要做灰度发布？

灰度发布一般用在APP拥有的用户数量有一定积累，某些重大功能改动的版本发布前。通过先邀请部分用户对新版本进行试用，用来降低产品在推向正式市场的风险。目的就是在发布前通过部分用户的使用，来验证我们发布的版本是否存在一些不可接受的问题，如果有，则发布补丁版本，然后再给部分用户使用，直到版本稳定为止。最终推向市场的是这个经过灰度测试的稳定版本。

灰度策略主要要考虑以下几点

1. 选取平台、比例

一般选取Android作为灰度平台，ios要做灰度很难绕开appstore的发版规则，由于appstore不支持灰度功能，所以手段要么是选取越狱设备，要么是testflight作为灰度包的安装渠道，但是明显实现成本都很高，盖的用户群很受限制。比例一般可根据产品用户数量来决定，10%、20%、30%甚至50%，视产品具体阶段而定。抽取规则可以通过用户id、手机号、设备id的尾号来抽取。还有一种方式是选取某一渠道投放灰度包，但是这样有几个缺点：

- 渠道的大小决定了覆盖用户量，但是这个很难做到精细比例的控制。
- 容易被其他渠道抓包，导致比例不可控，同时干扰正式版的正常发布。

2. 做好数据打点

决定灰度包要不要推广到市场，最直观快捷的方案是观测数据。所以针对灰度包的打点要保证全面，同时要能够与正式版本区分开。用来对比数据。

3. 做好版本控制

由于灰度版本是针对部分用户的beta版本，功能难免会不稳定，所以最好不要占用正常的版本号，而是单独细分颗粒度更细的版本号用在灰度版本上。

4. 灰度版本一定要具有回收能力

灰度包要具有能把发布出去的版本全部回收、即清洗为正式版本的能力。这样做是为了保证一旦灰度包出现重要bug，不会有部分用户的版本停留在有bug的版本导致后续整体的数据表现受到这部分bug的影响。一般手段为强制用户升级。