# **清单**

1. CSDN – 资源下载、教程
2. testerhome – 新闻资讯、教程、讨论
3. <http://ifeve.com/> -- 并发编程网
4. 简书 – 教程
5. [www.yiibai.com](http://www.yiibai.com) 易百教程

# **【Java笔记】**

## 【第一节：简介】

### 1.目的：熟悉java语言

（1）数据类型：byte/short/int/long/float/double/char/boolean/String,StringBuffer,StringBuilder

（2）逻辑：循环、分支

（3）异常

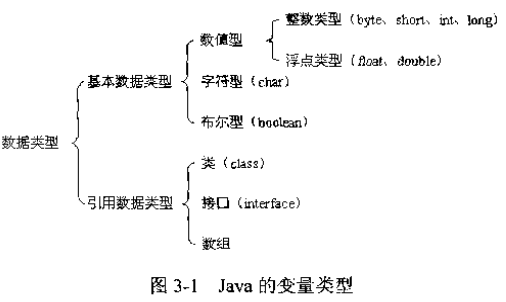
（4）常用包及函数

（5）面向对象：继承、接口、封装

（6）泛型

（7）集合

### 2.开始：需要知道的两张图



### 3.方法：从API文档入手，学习API的方法：

1.常用包：java.lang/java.util -- 基础包/工具包

2.接口（interface）：看接口中哪些类实现了该接口，则该接口中的所有方法其实现类都具有---有些类都实现了同一接口则先看这一接口（共性）---例如collection

3.以类为单位：类是你可以直接使用的，所以一切方法，属性都从类class开始

4.构造方法：创建该类对象时用到的 -- new StringBuffer(“str”)

5.静态方法：通过static关键字修饰的方法，用类名直接调用 – Arrays.sort(a);a是一个数组

6.public方法：通过实现了该类的对象来调用。--String str = “hello world”;str.toUpperCase();

7.abstract/private ---不用管

8.返回值：确定是否返回值，返回值的类型。这决定了你如何使用。同时确定它是否作用于对象本事还是返回一个新的值

9.参数：方法中需要哪些参数，是否需要参数，参数类型

## 【第二节：String、StringBuffer、StringBuilder类】--（public）

### 1.基本概念

1. String/StringBuffer/StringBuilder，大部分方法一样。
2. 不同：
   1. String创建之后不可变 -- String str = “strTest”, str = “Demo”,非法但不会报错
   2. StringBuffer相对于StringBuilder是线程安全的，可以同步访问
   3. StringBuilder相对于StringBuffer速度更快（因为非同步，非同步速度都快）

### 2.String -- 构造方法

String(char[] value)

String(String str)

String(StringBuffer/StringBuilder str)

Eg:

char[] ch = {‘h’,’l’,’l’,’o’};

String str01 = new String(ch);

String str02 = new String(“hello”);

String str03 = new String(strBuf);

String str = “hello”;

### 3.String -- Public method

|  |  |  |
| --- | --- | --- |
| 返回值类型 | 函数名 | 说明 |
| char | charAt(int index) |  |
| char[] | toCharArray() |  |
| int | compareTo(String str) |  |
| int | compareToIgnoreCase(String str) |  |
| boolean | contains(String str) | 包含 |
| boolean | isEmpty() |  |
| int | length() |  |
| String | substring(int start,int end) | 一个参数代表从这个位置到最后  返回一个新的String |
| String | toUpperCase() | 返回一个新的String |
| String | toLowerCase() | 返回一个新的String |
| String | trim() | 返回一个新的String |
| String | replace(char/String old, char/String new)  replaceFirst(old,new)  replaceAll(old,new) | 返回一个新的String |
| int | indexOf(String str) | 第一次出现的位置，没有返回0 |
| String[] | split(String str) | [\\.](file:///\\.)代表. |

### 4.转型 --- String静态方法

Static String.valueOf(boolean/char/int/… s)

1、所有类型遇到String类型自动转型为String类型

int a = 7;

String a01 = a + “”; //output：7

2、或者使用静态方法：String.valueOf(a)

3、最后：所有对象都有toString(),即返回对象的字符串表现形式

### 5.StringBuffer

1、构造方法：StringBuffer(String str)

2、public method

|  |  |  |
| --- | --- | --- |
| 返回值类型 | 函数名 | 说明 |
| StringBuffer | append(Boolean/char/char[] s) |  |
| StringBuffer | delete(int start,int end) |  |
| StringBuffer | deleteCharAt(int index) |  |
| StringBuffer | insert(int offset,double/char/float) |  |
| StringBuffer | replace(int start,int end,String str) |  |
| StringBuffer | reverse() |  |
| void | setCharAt(int index,char ch) |  |
| 均是改变原来的值，不返回新的对象，区别一下String中的方法 | | |

## 【第三节：Character类】--（static）

### 1.public method

|  |  |  |
| --- | --- | --- |
| 返回值类型 | 函数名 | 说明 |
| char | charValue() |  |
| int | compareTo(character c) |  |

### 2.static method

|  |  |  |
| --- | --- | --- |
| 返回值类型 | 函数名 | 说明 |
| static boolean | isLetter(char ch) | 是否是字母 |
| static boolean | isDigit(char ch) | 是否是数字 |
| static boolean | isWhitespace(char ch) | 是否是空格 |
| static boolean | isUpperCase(char ch) | 是否是大写 |
| static boolean | isLowerCase(char ch) | 是否是小写 |
| static char | toLowerCase(char ch) | 返回几个新的值 |
| static char | toUpperCase(char ch) | 返回一个新的值 |
| Character | valueOf(char c) | 将c转换为对象 |

### 3.构造方法：Character(char value)

## 【第四节：Arrays类】--（static）

### 1.static method

|  |  |  |
| --- | --- | --- |
| 返回值类型 | 函数名 | 说明 |
| static void | sort(long[]/char[]/int[]/… s) | 改变原来的数组顺序 |
| static int | binarySearch(Array s,s2) | 前提：排好序的数组 |
| static boolean | equals(long[] a,long[] b) | 可以是其他类型的数组 |

## 【第五节：Boolean类】--（static）

### 1.构造方法：

Boolean(Boolean value)

Boolean(String str) ------ 忽略大小写

### 2.public method

|  |  |  |
| --- | --- | --- |
| 返回值类型 | 函数名 | 说明 |
| boolean | booleanValue() |  |
| int | compareTo(Boolean b) | 比较两个对象，相同为0，不同为-1 |

### 3.static method

|  |  |  |
| --- | --- | --- |
| 返回值类型 | 函数名 | 说明 |
| static int | compare(boolean x,boolean y) |  |
| static boolean | logicalAnd(boolean a,boolean b) | 相当于&&运算 |
| static boolean | logicalOr(boolean a,boolean b) | 相当于|| |
| static boolean | parseBoolean(String str) |  |

### 4.属性

Static Boolean FALSE/TRUE -----------返回的是对象

Eg:Boolean.FALSE // 输出FALSE对象

## 【第六节：Object类】--（public）

### 1.public method

|  |  |  |
| --- | --- | --- |
| 返回值类型 | 函数名 | 说明 |
| boolean | equals(object obj) |  |
| int | hashCode() |  |
| String() | toString |  |
| class<?> | getClass() | 返回该对象运行时类 |
| 每个对象都具有的四个方法 | | |

## 【第七节：Number类】--（public）

Byte,Short,Integer,Long,Float,Double继承自Number

|  |  |  |
| --- | --- | --- |
| 返回值类型 | 函数名 | 说明 |
| byte | byteValue() |  |
| double | doubleValue() |  |
| float | floatValue() |  |
| int | intValue() |  |
| long | longValue() |  |
| short | shortValue() |  |

静态方法：static <Byte/Double/Float/Int/Long/Short> valueOf() ------🡪注意返回的是对象

属性：

static int BYTES/SIZE

static < Byte/Double/Float/Int/Long/Short > TYPE ————>>返回对象的基本类型

构造方法：

1：[Byte/Double/Float/Int/Long/Short]([Byte/Double/Float/Int/Long/Short] value)

2: [Byte/Double/Float/Int/Long/Short](String value)

3: Float/Double 特有

静态方法：static Float/Double NaN 返回一个float/double 类型的NaN（非数字）

4：Float特有构造方法

Float(double d)

Byte:以Byte类为例其他类似

|  |  |  |
| --- | --- | --- |
| 返回值类型 | 函数名 | 说明 |
| static int | compare(byte x,byte y) | 返回x-y的值 |
| static Byte | decode(String value) | 将value解码成Byte类型 |
| stati byte | parseByte(String s) |  |
| Static Byte | valueOf() | 返回的是对象 |

非静态方法：int compareTo(Byte b),返回这两个数的差

Float/Double 特有方法 static boolean isNaN(double v) static boolean isNaN()

### **总结：**

比较：--🡪返回的均是a-b的值

静态：compare(a, b)

非静态：a.compareTo(b)

**类型转换：转换为对象，转换为基本数据类型，转换为String**

静态：valueOf(value) -🡪value的类型与调用者“一样”，或者是String类型，返回的是对象

~~decode(String str)🡪返回的是对象，与调用者类型一致 🡪不常用~~

parse[Byte/Double/Float/Int/Long/Short](String str) 🡪返回的是基本数据类型(**参数只能是String类型**)且调用他的类必须跟他要转换的类型一致

Float.parseFloat(“4”) 🡪返回4;基本数据类型

非静态：[byte/double/float/int/long/short]Value() 🡪返回的是基本数据类型

向String类型转换：任何类型 + “”自动转换为String类型

toString()方法，试用与任何对象

构造方法：

1：[Byte/Double/Float/Int/Long/Short]([Byte/Double/Float/Int/Long/Short] value)

2: [Byte/Double/Float/Int/Long/Short](String value)

## 【第八节：Collection】

List list = new ArrayList();这句创建了一个ArrayList的对象后把上溯到了List。此时它是一个List对象了，有些ArrayList有但是List没有的属性和方法，它就不能再用了。   
而ArrayList list=new ArrayList();创建一对象则保留了ArrayList的所有属性。

问题的关键:   
为什么要用 List list = new ArrayList() ,而不用 ArrayList alist = new ArrayList()呢？   
问题就在于List有多个实现类，现在你用的是ArrayList，也许哪一天你需要换成其它的实现类，如 LinkedList或者Vector等等，这时你只要改变这一行就行了：   
List list = new LinkedList(); 其它使用了list地方的代码根本不需要改动。   
假设你开始用 ArrayList alist = new ArrayList(), 这下你有的改了，特别是如果你使用了 ArrayList特有的方法和属性。

## 【第九节：Math类】

### 字段：static double PI 圆周率

### 静态方法：

|  |  |  |
| --- | --- | --- |
|  |  |  |
| static double/float/int | abs(a) | 绝对值 |
| static double | ceil(double a) | 向上取整 |
| static double | floor(double a) | 向下取整 |
| static double/float/int | max(a,b) | 返回其中较大的值 |
| static double/float/int | min(a,b) | 返回其中较小的值 |
| static double | pow(a,b) | a的b次幂 |
| static double | random() | 0.0>=x x<1.0 |
| static long/int/double/float | round(a) | 返回a的四舍五入值 |
|  |  |  |

## 【第十节：System类】

## 【第十节Random类】

构造方法：

Random()

Random(long seed)

### 常用非静态方法：

|  |  |  |
| --- | --- | --- |
|  |  |  |
| boolean | nextBoolean() |  |
| double/float | nextDouble() | 0.0 到 1.0之间 |
| int | nextInt() | 随机返回一个int类型 |
| int | nextInt(int a) | 随机返回一个0~a不包括a |
| long | nextLong() |  |

## Map.Entry，遍历Map的一种方式返回类似为Set<Map.Entry<String,String>>

通过 obj.getValue(),obj.getKey(),foreach方法

## 【单例模式】

将构造方法私有化则外部类无法调用即无法实例话

1. 按需实例
2. 加锁实例
3. 提前实例
4. 解法四：饿汉式（建议使用）
5. [复制代码](javascript:void(0);)
6. public class Singleton {
7. private static Singleton instance=new Singleton();
8. private Singleton(){
10. }
11. public static Singleton getInstance(){
12. return instance;
13. }
14. }
15. 静态内部类（按需实例 + 提起实例）
16. 解法五：静态内部内。（建议使用）
17. [复制代码](javascript:void(0);)
18. public class Singleton {
19. private Singleton(){
21. }
22. private static class SingletonHolder{
23. private final static Singleton instance=new Singleton();
24. }
25. public static Singleton getInstance(){
26. return SingletonHolder.instance;
27. }
28. }

## 【继承】

在一个子类被创建的时候，首先会在内存中创建一个父[类对象](https://www.baidu.com/s?wd=%E7%B1%BB%E5%AF%B9%E8%B1%A1&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1Y3nARvP16dPWu9nhckuymv0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EPH6YP1fvP1ms)，然后在父[类对象](https://www.baidu.com/s?wd=%E7%B1%BB%E5%AF%B9%E8%B1%A1&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1Y3nARvP16dPWu9nhckuymv0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EPH6YP1fvP1ms)外部放上子类独有的属性，两者合起来形成一个子类的对象。

所以所谓的继承使子类拥有父类所有的属性和方法其实可以这样理解，子[类对象](https://www.baidu.com/s?wd=%E7%B1%BB%E5%AF%B9%E8%B1%A1&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1Y3nARvP16dPWu9nhckuymv0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EPH6YP1fvP1ms)确实拥有父类对象中所有的属性和方法，但是父类对象中的私有属性和方法，子类是无法访问到的，只是拥有，但不能使用。

总结

1. 创建子类时：先创建一个父类对象
2. 子类可以直接使用父类中的非private方法 -- （继承）
   1. 只是继承或可以直接使用非private修饰的
   2. 但是private方法或属性也是会被执行的，只是子类无法使用

拥有或执行了父类中private，但是并不能直接访问或者使用

## \*【lambda表达式】

# 【eclipse笔记】

## 问题描述：Maven项目下的SpringBoot项目无法启动

错误信息：error:Could not find or load main class Demo.java

解决思路：首先查看错误信息：Window🡪Show View🡪Other🡪Problems --应该会看到相关错误信 息 ，然后根据信息修改

Maven项目常见错误：本地jar包无法使用，或者在其他本地路径下的jar包过期，C:\Users\exifguo\.m2\repository\…………..中的某些jar文件

解决方法：删掉该包目录，右键项目🡪Maven🡪Update Project….即可

## 问题描述：SpringBoot中端口占用无法启动项目

解决方案：在resources目录下的application.properties文件中添加：server.port=1234

# 【Python笔记】

## 【第一节：常用操作】

|  |  |
| --- | --- |
| 函数 | 描述 |
| abs(number) | 返回数字的绝对值 |
| cmath.sqrt(number) | 返回平方根，也可用于负数 |
| math.sqrt(number) | 返回平方根，不可用于负数 |
| math.ceil(number) | 返回数的上入整数，返回值为浮点型 |
| math.floor(number) | 返回数的下入整数，返回值为浮点型 |
| float(object) | 将字符串或数字转换成浮点数 |
| int(object) | 将字符串或数字转换成整数 |
| long(object) | 将字符串或数字转换成长整数 |
| str(object) | 将值转换成字符串 |
| bool(object) |  |
| pow(x,y[,z]) | 返回x的y次幂(所得结果对z取模) |
| round(number[,ndigits]) | 根据给定的精度对数字进行四舍五入 |
| repr(object) | 返回值的字符串表示形式 |
| input(prompt) | 获取用户合法的输入 |
| raw\_input(prompt) | 将用户输入的字符转换为字符串 |
| help(object) | 帮助 |
| r’objct’ | 对象的原始字符 |
| \n | 换行 |
| ‘’’ | 跨多行 |
| \ | 跨多方，最后一个字符不能是反斜杠 |
| dir() | 显示模块里面的所有方法 |
| type(object) | 返回对象的类型 |
| isinstance(object,type) | 返回bool值，确认该对象是否是该类型 |

## 【第二节：序列】

基本数据结构：序列

容器：1。序列、映射（字典）

字符串和元组：str/tuple --- 不可改变

字典：key-value -- key不可变

list十种操作

|  |  |
| --- | --- |
| 函数 | 描述 |
| 修改列表中的值 | 通过索引直接修改 |
| del name[2] | 删除元素 |
| append(obj) | 在列表末尾追加 |
| count(obj) | 统计obj出现的次数 |
| extend(anoList) | 将一个list增加到另一个上 |
| index(obj) | 返回第一个匹配到obj的位置 |
| insert(obj,index) | 将对象插入到index位置 |
| pop(index) | 删除index位置的对象，默认删除最后一个，并返回被删除的对象 |
| remove(obj) | 移除列表中第一个匹配的对象 |
| reverse() reversed(listObj) | 反向操作 |
| sort() sorted(listObj) | 排序 |
| sort(key=len)/sort(reverse=True) | 高级排序 |

序列常用操作

1. 切片操作：[::]
2. 加法操作：+
3. 乘法操作：\*

序列常用函数

|  |  |
| --- | --- |
| 函数 | 描述 |
| cmp(x,y) | 比较两个值 |
| len(seq) | 返回序列的长度 |
| max(args) | 返回序列或者参数集合中的最大值 |
| min(args) | 返回序列或者参数集合中的最小值 |
| reversed(seq) | 对序列进行反向迭代（不改变原来的序列） |
| sorted(seq) | 对序列进行排序（不改变原来的） |
| list(seq) | 把序列转换成列表 |
| tuple(seq) | 把序列转换成元组 |

## 【第二节：字典】

## \*【lambda表达式】

两个构造方法

dict([(“a”,3),(“b”,6)])

dict(name=gxf,age=123)

## **【urllib模块 – Python3.x】**

字典类型的data 🡪 编码数据🡪 组装请求🡪 请求 🡪 读取(接收response) 🡪 解析

1. data = {‘name’:’gxf’,’password’:123}
2. 编码数据：newData = urllib.parse.urlencode(data).encode(‘utf-8’)

post数据必须是bytes不能是str所以需要encode

1. 组装请求：

url = r’http://www.baidu.com’

headers = {}

data = {}

requestData= urllib.request.Request(url,headers=headers,data=data)

1. html = urllib.request.urlopen(requestData) – 默认url为无参数
2. 读取：read() readlin() readlines() close() info() getcode() geturl() – page = html.read()
3. 解析：newPage = page.decode(‘utf-8’) – 依据html编码类型来解析

## 【常用数据类型：list tuple dict set】

list：list()方法可以将tuple转换为list类型

1. 有顺序，可以通过下标取值
2. 包括各种类型，里面可以有int类型同时也可以有str类型
3. 下标从0开始不能越界
4. append()尾部添加 L.append(‘Jack’)
5. insert()在指定位置添加 L.insert(2,’haha’)
6. pop()默认删除最后一个元素，可以指定删除位置，并返回删除的值

L.pop() L.pop(2)

tuple：tuple() 可以将list类型转换为tuple类型

1. 可以重复。有序可以通过下标取值
2. 不可变，没有增删改方法
3. 单元素tuple，加一个逗号防止歧义 t = (3.14,)

dict :

1. key-value 格式，没有顺序
2. key可以是str int float tuple
3. key不可以重复
4. 通过：d[‘key’]取值，如果不存在则报错。通过d.get(‘key’)如果不存在则显示None
5. key不可变，value可变

set：set() 方法可以将tuple list转换为set类型

1. 无序
2. 不包含重复的数据，其他类型转换为set会自动去除重复
3. add()添加元素，remove() 删除元素，添加重复的元素添加不上不报错，删除不存在的元素会报错

## 【常用函数：map reduce filter】

1.map()

格式：map( func, seq1[, seq2...] ) —>list，将func作用于seq中的每一个元素，并用一个列表给出返回值。

|  |  |
| --- | --- |
| 1  2  3 | map(lambda x: x \*\* 2, [1, 2, 3, 4, 5])    [1, 4, 9, 16, 25] |

2.reduce()

格式：reduce( func, seq[, init] ) —>value，将一个列表归纳为一个输出。

每次迭代，将上一次的迭代结果（第一次时为init的元素，如没有init则为seq的第一个元素）与下一个元素一同执行一个二元的func函数。在reduce函数中，init是可选的，如果使用，则作为第一次迭代的第一个元素使用。

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | reduce(lambda x, y: x + y, [2, 3, 4, 5, 6], 1)    结果为21    reduce(lambda x, y: x + y, [2, 3, 4, 5, 6])    结果为20 |

3.filter()

格式：filter( func or None, seq ) —>list, tuple, string， 对指定序列执行过滤操作。

func是一个谓词函数，接受一个参数，返回布尔值True或False。filter函数会对序列参数sequence中的每个元素调用function函数，最后返回的结果包含调用结果为True的元素。

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | def is\_even(x):      return x & 1 != 0    filter(is\_even, [1, 2, 3, 4, 5, 6, 7, 8, 9, 10])    返回结果为：  [1, 3, 5, 7, 9] |

# 【SQL笔记】

## 【第一节：简介】

## 【第二节：通配符】

# 【Django笔记】

## 【第一节：常用命令】

django-admin.py startproject mysite

manage.py startapp blog

manage.py runserver 8080

# 【Jmeter笔记】

## 问题描述：无法加载JDBC Driver驱动

解决方案：下载mysql-connector-java的jar包放置在ext文件下并在测试计划中引用

## BeanShell使用

1. 操作变量：通过内置对象vars可以对变量进行存取操作
2. vars.get(“name”)：从jmeter中获取变量名为name的值
3. vars.put(“key”,”value”)：将数据存到jmeter变量中
4. 操作属性 ：通过使用Bean shell内置对象props可以对属性进行存取操作
5. props.get(“START.HMS”)：START.HMS为属性名，在jmeter.properties中定义
6. props.put(“PROP1”,”1234”)
7. 自定义函数
8. 在Test Plan中添加一个变量：hello = kitty
9. 引用外部[**Java**](http://lib.csdn.net/base/java)文件
10. 读取java文件：source(绝对路径);
11. 其他操作（创建对象，使用。。。）与在java中一样
12. 引用外部class文件
13. 使用addClassPath(绝对路径)
14. 导入完整类名：import test.MyClass;
15. 调用。。。。与在java中一致
16. 引用外部Jar包



七、其它用法(接受参数， log等)

取值：

1）${name}

2）通过bsh.args[]按顺序提取 -----（bsh为内置对象）

内置变量：Parameters,用来保存参数集合

输出信息：log.info(字符串) log.info(Label)---输出在控件的名称

Foreach组件

配合用户定义变量来使用，相当于对一个字段进行参数化

jmeter无法处理弹窗

## 函数的使用

${\_\_javaScript("${name}" == "gxf")}直接执行js脚本，这里返回true 或者false

## 全局变量（测试计划内TestPlan）局部变量（同一线程组内ThreadGroup）

全局变量：User Defined Variables

局部变量：vars、User Parameter、CSV Data

局部变量转为全局变量${\_\_setProperty(newxfg, ${gxf},)}

取出${\_\_P(newgxf,)}

# 【SpringBoot笔记】

## 常用依赖

|  |
| --- |
| <parent>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-parent</artifactId>  <version>1.5.6.RELEASE</version>  </parent> |
| <!--web框架-->  <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-web</artifactId>  </dependency> |
| <!--日志依赖:默认此项即不需要专门添加依赖-->  <!--日志级别从低到高分为TRACE < DEBUG < INFO < WARN < ERROR < FATAL-->  <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-logging</artifactId>  </dependency> |
| <!--默认测试框架-->  <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-test</artifactId>  <scope>test</scope>  </dependency> |
| <!-- 热部署 -->  <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-devtools</artifactId>  </dependency> |
| <!--jsp支持-->  <dependency>  <groupId>org.apache.tomcat.embed</groupId>  <artifactId>tomcat-embed-jasper</artifactId>  <scope>provided</scope>  </dependency>  <dependency>  <groupId>javax.servlet</groupId>  <artifactId>jstl</artifactId>  </dependency> |
| <!--tomcat支持-->  <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-tomcat</artifactId>  </dependency> |
| <!-- 标签库 -->  <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-thymeleaf</artifactId>  </dependency> |
| <!-- 权限管理 -->  <dependency>  <groupId>org.thymeleaf.extras</groupId>  <artifactId>thymeleaf-extras-springsecurity4</artifactId>  </dependency>  <!--<dependency> -->  <!-- groupId>org.springframework.boot</groupId> -->  <!-- <artifactId>spring-boot-starter-security</artifactId> -->  <!-- </dependency> --> |
| <!--缓存caches -->  <dependency>  <groupId>org.mybatis.caches</groupId>  <artifactId>mybatis-ehcache</artifactId>  <version>1.1.0</version>  </dependency> |
| <!-- 数据库驱动-->  <dependency>  <groupId>mysql</groupId>  <artifactId>mysql-connector-java</artifactId>  </dependency> |
| <!-- jdbc-->  <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-jdbc</artifactId>  </dependency> |
|  |

## 连接MySQL配置

spring.jpa.hibernate.ddl-auto=update

spring.datasource.url=jdbc:mysql://localhost:3306/test?useSSL=true

spring.datasource.username=root

spring.datasource.password=

spring.datasource.driver-class-name=com.mysql.jdbc.Driver

server.port=1234

## ModelAndView/Model/ModelMap三者区别

使用ModelAndView首先需要new一个该对象，参数为html，默认我当前，然后装载数据

Model,ModelMap类似，不需要new，但需要在接收的时候加上该对象然后装载

## 使用RequestParam可设置默认值

@RequestParam(value="name" required=false,defaultValue="World" String naem, Model model)

## 3.@GetMapping("/") 相当于 @RequestMapping(value="/" ,method=RequestMethod.GET)

## 4.spring.jpa.hibernate.hbm2ddl-auto=update在以前表的基础上更新表结构，不影响以前的表

## 使用logger

private static final Logger logger = LoggerFactory.getLogger(Object.class);

logger.info(s + ": 目录创建成功！！！");

@Service 与 @Autowired配合使用，其中@Service用在实现类上，直接使用@Autowired来实现对象并操作实体类中的方法

## aaAttribute put 的区别

@RequestMapping("hello")

public String testVelocity(Model model,String name){

model.addAttribute("name",name);a //addAttribute不允许为空

model.put(key,value);//允许为空

return "hello";

}

## 【使用Mybatis】特别注意：如果要在映射的实体类中添加构造方法，首先一定要先重写默认的构造方法，否则无法创建实例

1-- 添加依赖：

<dependency>

<groupId>org.mybatis.spring.boot</groupId>

<artifactId>mybatis-spring-boot-starter</artifactId>

<version>1.1.1</version>

</dependency>

<dependency>

<groupId>mysql</groupId>

<artifactId>mysql-connector-java</artifactId>

</dependency>

2—添加相关配置

1）application.properties – 驱动、url、用户名、密码

spring.datasource.driverClassName **=** com.mysql.jdbc.Driver

spring.datasource.url **=** jdbc:mysql://localhost:3306/test1?useUnicode=true&characterEncoding=utf-8

spring.datasource.username **=** root

spring.datasource.password **=** root

1. 再启动类添加自动扫描：@MappingScan(“com.example.demo05.mapper;”),系统启动的时候将自动扫描该包下面的mapper

3—开发Mapper接口：

public interface UserMappper{

@Select(“select \* from users”)

List<UserEntity> getAll();

//使用参数

@Select(“select \* from users where id = #{id}”)

List<UserEntity> getAllById(Long id);

}

增删改查分别对应：@Select、@Insert、@Update、@Delete

@Result负责将实体与数据库中的字段相互对应，如果数据库中的字段与实体中的字段一样则不需要此注解

4—使用

@Autowired注解自动装载，然后调用即可

参考链接：http://blog.csdn.net/gebitan505/article/details/54929287#comments

# 【Thymeleaf标签库】

## 0.响应html或直接返回信息

响应html：@Controller @RequestMapping(value = “/”,method=Method.RequestMethod.GET)

等价：@GetMapping(“/”) @GetMapping(path=”/”)

响应信息：@Controller @RequestMapping(value = “/”) @ResponseBody

### 1.加载静态html

添加属性：<html xmlns:th=<http://www.thymeleaf.org>>

### 2.从服务器端返回到html页面数据

html： **<h1** th:text="${msg}"**></h1>**

1. 控制层：**public** String testTH(ModelMap map) {
2. map.addAttribute("msg", "Hello, rensanning! @Thymeleaf");
3. **return** "test\_th";
4. }

### 3.ModelMap中的addAttribute与put方法的区别

**结论:**  
ModelMap对象的 addAttribute,put两个方法区别是: addAttribute是不允许添加空值的key，put是允许的

### 4.控制层即接收get过来的请求，又返回html

@RequestMapping(value=”/index” , method=RequestMethod.GET)

public String index(String a, String b, ModelMap mp){

mp.put(“msg”,a+ b); //msg对应页面上<h1 th:text=”${msg}”></h1>

return “index”; //index.html,页面名称

}

可以查看依赖关系,发现spring-boot-starter-thymeleaf下面已经包括了spring-boot-starter-web,所以可以把spring-boot-starter-web的依赖去掉.

#thymeleaf start

spring.thymeleaf.mode=HTML5

spring.thymeleaf.encoding=UTF-8

spring.thymeleaf.content-type=text/html

#开发时关闭缓存,不然没法看到实时页面

spring.thymeleaf.cache=false

#thymeleaf end

## Thmeleaf常用属性

|  |  |  |
| --- | --- | --- |
| 属性 | 例子 | 说明 |
| th:action | <formid="loginform"th:action="@{/login}">...</form> | 类似form标签中的action |
| th:each |  |  |
| th:href | <a th:href="@{/logout}" ></a> | 定义超链接 |
| th:id |  | 类似html标签中的id属性 |
| th:if | th:if="${rowStat.index} == 0" |  |
| th:include |  | 见th:fragment |
| th:fragment | 声明模板片段/WEBINF/templates/footer. html  <div th: fragment=" copy" >  © 2011 The Good Thymes Virtual Grocery  </div>  引入模板片段  <div th: include=" /templates/footer : : copy" ></div>  <div th: replace=" /templates/footer : : copy" ></div> |  |
| th:replace |  | 见th:fragment |
| th:src | 相当于<script>标签中的src属性，@{} |  |
| th:text | <td th:text=”${name}”></td> |  |
| th:value |  | 类似<option>中的value |

fragment/include/replace

想要替换包含哪个标签内的内容，先确定他的父标签即fragment。例如

<head xmlns:th=<http://www.thymeleaf.org> th:fragment=”pare”>

<script th:src=”@{}”></script>

<script th:src=”@{}”></script>

</head>

在需要引用的地方使用<head th:replace=”/::para”>,其中include将包含head，replace不含，仅包括<script>

## Thmeleaf基本表达式

|  |  |
| --- | --- |
| 表达式 | 说明 |
| ${} | 获取变量 |
| \*{} | 选择表达式 |
| #{} | 通过properties获取值 |
| @{} | 超链接 |
| #maps | 工具对象表达式 |

# 【JQuery笔记】

## H5中不需要再加type=”text/javascript”

您是否很疑惑为什么我们没有在 <script> 标签中使用 type="text/javascript" ？

在 HTML5 中，不必那样做了。JavaScript 是 HTML5 以及所有现代浏览器中的默认脚本语言！

## ajax实例

$.ajax({

type:”GET”,

[url:”/page1](url:)”,

dataType:"json",

data:{name:”gxf”,age:23},

success:function(data){

$.each(data,function(i,item){

alert(item.name);

})

}

})

# 【selenium】

环境：

Python：3.6.1

selenium：3.4.3

Firefox：54.0.1

geckodriver:0.15.0

chrome：59.0.3

chromedriver：v3.1

自动化测试基础

## **1、软件测试分类**

根据项目流程：单元测试（编码）、集成测试（设计）、系统测试（需求）、验收测试

根据代码是否可见：白盒测试、黑盒测试、灰盒测试

功能测试：逻辑功能测试、界面测试、易用性测试、安装测试、兼容性测试

性能测试：负载测试、压力测试

手工测试：手工执行测试用例

自动化测试：性能自动化和功能自动化

冒烟测试：在对系统进行正式测试之前，先验证主要功能是否实现，是否具备可测性

回归测试：修改代码之后确保没有引入新的错误，或导致其他代码产生错误

随机测试：为了模拟用户的操作，测试中输入的数据都是随机的，以便发现一些边缘性的错误

探索性测试：强调主观能动性，碰到问题时主动改变策略

安全测试：验证产品是否符合安全需求定义和产品质量标准

## **2、什么项目适合自动化**

1）项目需求变动不频繁（回归测试，每日构建后的测试验证）

2）项目周期长

3）自动化测试脚本可复用

## **3、自动化测试分类和工具**

1）分类：UI自动化、接口自动化、单元测试自动化

2）工具：

QTP：企业级自动化测试工具，支持B/S，C/S架构

robot framework：Python语言编写，支持关键字驱动，分布式测试

Watir：基于web模式的功能自动化测试工具

selenium：支持多语言、多平台、多浏览器

## **4、selenium工具**

1）特点：多语言---Java、Python、PHP、ruby

                多平台---Windows、Linux、Mac

               多浏览器----Firefox、chrome、IE

                开源、免费

                简单、灵活

2）selenium1.0

selenium IDE：实现浏览器操作的简单录制和回放，支持多种语言

selenium Grid：自动化测试辅助工具

                          并行执行

                          通过一个主机控制用例在不同环境、不同浏览器下执行

selenium RC：selenium的核心，支持多种编程语言编写脚本，通过selenium服务器作为代理服务器去访问应用，从而实现测试的目的。

selenium RC：client libraries---编写测试脚本，控制selenium server的库

                        selenium server---控制浏览器行为

selenium server：selenium core---一堆js函数，嵌入到浏览器中，通过这些函数控制浏览器操                                                          作

                            launcher---启动浏览器，并将selenium core加载到浏览器中，将selenium                                                     server的http proxy设置为浏览器的代理服务器

3）selenium2.0

webdriver替代了selenium RC，为了向下兼容性，selenium RC仍然可以使用

selenium RC通过嵌入到浏览器中的js函数操作

webdriver针对各浏览器开发，通过原生浏览器支持或者浏览器扩展控制浏览器

简单自动化测试脚本

## Python webdriver API

#### **1、浏览器操作**

1）浏览器最大化

driver.maximize\_window()

2）设置浏览器宽和高

driver.set\_window\_size(400,800)

3）控制浏览器前进、后退

driver.forward()

driver.back()

#### **2、简单对象的定位**

find\_element\_by\_id()----唯一

find\_element\_by\_name()----唯一

find\_element\_by\_linx\_text()----操作对象是文字超链接

find\_element\_by\_partial\_link\_text()----操作对象是文字超链接

find\_element\_by\_class\_name()

find\_element\_by\_tag\_name()

find\_element\_by\_xpath()

暂时可以通过firepath定位xpath路径，以后再研究

find\_element\_by\_css\_selector()

这个真的不懂，firebug里面可以定位

#### **3、操作测试对象**

clear()---------------清除内容，如默认用户名和密码

click()---------------模拟鼠标点击操作

send\_keys()------向输入框输入

submit()------------提交表单

#### **4、WebElement接口常用方法**

size--------------------返回元素尺寸

text----------------------获取元素文本信息

get\_attribute(name)-------------获取元素某个属性值

is\_displayde-----------------------该元素用户是否可见

#### **5、鼠标事件**

ActionChains类操作鼠标事件

***ActionChains的使用***

from selenium.webdriver.common.actionchains import ActionChains

**ActionChains的执行原理**

调用ActionChains方法的时候，用户行为不会立刻执行，而是将所有操作放在一个队列中，当执行perform()方法时，按照放入队列的顺序先进先出执行

**ActionChains方法的书写格式**

ActionChains(driver).context\_click(操作对象).perform()

**鼠标点击**

context\_click()------------------鼠标右击

double\_click()------------------鼠标双击（应用场景：查看图片，双击选定文字）

click()-----------------------------鼠标点击

click\_and\_hold()---------------按住鼠标左键不懂

**鼠标移动**

move\_to\_element(目标元素)------------------------移动到某个元素

move\_by\_offset(xoffset,yoffset)--------------------移动到某个坐标

**鼠标拖曳**

drag\_and\_drop(source,target)------将元素从起点source移动到终点target

drag\_and\_drop\_by\_offset(source,xoffset,yoffset)-------按照坐标移动

#### **6、键盘事件**

Keys类操作键盘事件

***Keys类的使用***

from selenium.webdriver.common.keys import Keys

***常用的组合键***

send\_keys(Keys.CONTROL,'a')----------------全选

send\_keys(Keys.CONTROL,'c')----------------复制

send\_keys(Keys.CONTROL,'v')----------------粘贴

send\_keys(Keys.CONTROL,'x')----------------剪切

***常用的非组合键***

send\_keys(Keys.ENTER)--------------------------------回车键

send\_keys(Keys.BACK\_SPACE)----------------------删除键

send\_keys(Keys.SPACE)--------------------------------空格键

send\_keys(Keys.TAB)------------------------------------制表键

send\_keys(Keys.ESCAPE)------------------------------回退键

send\_keys(Keys.F5)---------------------------------------刷新键

#### **7、获得页面URL和title**

1）获得当前页面title，判断页面跳转是否符合预期

title = driver.title

2）获得当前URL，一般用来测试重定向

url = driver.current\_url

#### **8、设置等待时间**

sleep()：设置固定休眠时间。Python的time包提供sleep方法

implicitly\_wait()：webdriver提供的一个隐性等待的时间，在一个时间段内只能的等待，超时则抛出异常

WebDriverWait()：webdriver提供的另一个方法，在设置时间内，默认每隔一段时间去检测页面元素是否存在，如果超出设置时间检测不到则抛出异常。

WebDriverWait(driver, timeout, poll\_frequency=0.5, ignored\_exceptions=None)

driver - WebDriver 的驱动程序(Ie, Firefox, Chrome 或远程)

timeout - 最长超时时间，默认以秒为单位

poll\_frequency - 休眠时间的间隔（步长）时间，默认为0.5 秒

ignored\_exceptions - 超时后的异常信息，默认情况下抛NoSuchElementException 异常。

#### **9、定位一组对象**

find\_elements返回的是一个list

定位一组对象，一般应用于下列场景：

           批量操作对象，比如将页面上的CheckBox都选上

1）find\_elements\_by\_tag\_name()

2）find\_element\_by\_css\_selector()

#### **10、定位frame中的对象**

针对frame嵌套的情况，使用**switch\_to.frame(id/name/xpath)**切换到被嵌套的frame中

#### **11、多窗口处理**

要想在多个窗口之间切换，首先要获得每一个窗口的唯一标识符号（句柄）。通过获得的句柄来区别分不同的窗口，从而对不同窗口上的元素进行操作

driver.current\_window\_handle----------------------获得当前窗口的句柄

driver.window\_handles--------------------------------获得所有窗口的句柄

driver.switch\_to\_handle("句柄")---------------------切换回句柄所属的窗口

driver.close()---------------------------------------------关闭当前窗口

driver.quit()-----------------------------------------------关闭所有窗口

#### **12、下拉菜单处理**

1）传统下拉菜单

先定位到下拉菜单，再点击选项

2）下拉菜单需点击才能显示选项

有两次点击动作，第一次点击下拉菜单，第二次点击选项

3）下拉菜单不需点击，鼠标放上去就会显示选项，则可以使用move\_to\_element()方法定位

4）针对下拉菜单标签是select的

**导入Select类：from selenium.webdriver.support.select import Select**

**使用方法：Select(driver.find\_element\_by\_name("xxx")).select\_by\_index()**

选择列表：

select\_by\_index(index)---------------------------根据index属性定位选项，index从0开始

select\_by\_value(value)---------------------------根据value属性定位

select\_by\_visible\_text(text)----------------------根据选项文本值来定位

first\_selected\_option()----------------------------选择第一个选项

清除列表

deselect\_by\_index(index)---------------------------根据index属性清除选定的选项，index从0开始

deselect\_by\_value(value)---------------------------根据value属性

deselect\_by\_visible\_text(text)----------------------根据选项文本值

deselect\_all()--------------------------------------------清除所有选项

#### **13、上传文件**

上传过程一般是打开一个系统的window窗口，从窗口选择文件添加，一般会卡在如何操作Window窗口。其实，没那么复杂，只要定位上传按钮，通过send\_keys()添加文件路径就可以了。

上传控件标签为input

上传控件标签为非input

可以借助第三方工具：Autolt

#### **14、下载文件**

webdriver允许设置默认下载路径，从而可以跳过下载弹窗提示

1）chrome下载

思路：实例化一个option对象

           设置配置，并加载到option中

           启动浏览器，点击下载链接

优势：使用谷歌浏览器下载，不需要针对各种文件类型进行配置

2）Firefox下载

对于Firefox，需要我们设置其Profile：

**browser.download.dir：**指定下载路径

**browser.download.folderList：**设置成2表示使用自定义下载路径；设置成0表示下载到桌面；设置成1表示下载到默认路径

**browser.download.manager.showWhenStarting：**在开始下载时是否显示下载管理器

**browser.helperApps.neverAsk.saveToDisk：**对所给出文件类型不再弹出框进行询问

Firefox需要针对每种文件类型进行设置，这里需要我们查询对应文件的MIME类型，可以用以下链接进行查询：MIME 参考手册

#### **15、调用js和控制浏览器滚动条**

1）webdriver提供了execute\_script(script,\*args)方法调用js

2）控制浏览器滚动条

应用场景：判断法律文件是否阅读完，滚动条拉倒最下面，表示用户已看完

                   要操作的元素不在视觉范围内，需要拖动滚动条

火狐浏览器

谷歌浏览器

#### **16、处理cookie**

driver.get\_cookies()-------------------------------获得所有cookie

driver.get\_cookie(name)-------------------------获得name属性的cookie

driver.add\_cookie(cookie\_dic)-----------------添加cookie（cookie格式为字典，）

driver.delete\_cookie(name)---------------------删除特定cookie

driver.delete\_all\_cookies()----------------------删除所有cookie

#### **17、验证码问题**

跳过验证码的方法：

1）去掉验证码

2）设置万能码

3）通过cookie跳过验证码登录

#### **18、webdriver原理**

1）WebDriver 启动目标浏览器，并绑定到指定端口。该启动的浏览器实例，做为web driver 的remote server。

2）Client 端通过CommandExcuter 发送HTTPRequest 给remote server 的侦听端口（通信协议： the webriver wire protocol）

3）Remote server 需要依赖原生的浏览器组件（如：IEDriverServer.exe、chromedriver.exe），来转化转化浏览器的native 调用。

自动化测试模型

#### **1、自动化测试模型介绍**

1）模块化与类库

将脚本中重复可复用的部分拿出来写成一个公共的模块，需要的时候就调用它，可以大大提高测试人员编写脚本的效率。如将登录和退出模块化

模块化的优点：提高效率，不用编写重复脚本

                         需要修改代码时，只需要修改模块代码，调用模块的代码不需要修改

2）数据驱动

数据驱动可理解成参数化，输入数据的不同从而导致输出结果的不同

将数据与脚本分离

3）关键字驱动

通过关键字的改变引起结果的改变

QTP、robot framework 等都是以关键字驱动为主的自动化工具

#### **2、登录模块化**

#### **3、数据驱动（参数化）**

参数化方式：读取TXT文件和CSV文件、函数、字典

1）读取TXT文件

分别打开两个文件，将用户名和密码赋值给变量，传递给输入框

缺点：用户名和密码在不同的文件中，修改较麻烦

           文件中只能保存一个用户名和密码，不能循环读取

2）登录参数化（函数）

user\_info.py

登录模块中，通过2个变量来接受函数返回的值（用户名、密码）

3）登录参数化（字典）

user\_info.py

登录模块通过字典的键值对取值

4）表单参数化

通过WPS 或excel 创建表格，文件另存为选择CSV 格式

csv.reader()用于读取CSV 文件，user[0] 表示表格中第一行第一列的数据（用户名），user[1]表示表格中第一行第二列的数据（邮箱），后面类推。通过CSV 读取文件比较灵活，可以循环读取每一条数据，从而又不局限每次所读取数据的个数。

自动化测试用例设计

#### **1、手工测试用例和自动化测试用例**

**手工测试用例**

 较好的异常处理能力，能通过人为的逻辑判断校验当前步骤的功能实现正确与否。

 人工执行用例具有一定的步骤跳跃性。

 人工测试步步跟踪，能够细致的定位问题。

 主要用来发现功能缺陷

**自动化测试用例**

 执行对象是脚本，任何一个判断都需要编码定义。

 用例步骤之间关联性强。

 主要用来保证产品主体功能正确完整和让测试人员从繁琐重复的工作中解脱出来。

 目前自动化测试阶段定位在冒烟测试和回归测试。

#### **2、测试类型**

1）测试静态内容

用于验证静态性的、不变化的UI元素的存在性

如，页面底部备案信息，以及页面顶部的图片

2）测试链接

如果页面链接经常变化或文件不时被重定向，可以尝试自动化测试

3）功能测试

功能测试通常是需要自动化测试的最复杂的测试类型,但也通常是最重要的。**典型的测试是登录,注册网站账户,用户帐户操作,帐户设置变化,复杂的数据检索操作等等。**功能测试通常对应着您的应用程序的描述应用特性或设计的使用场景。

4）测试动态元素

5）ajax测试

Ajax 是一种支持动态改变用户界面元素的技术。页面元素可以动态更改,但不需要浏览器重新载入页面,如动画,RSS 源,其他实时数据更新等等。

6）断言assert和验证verify

**断言的优点和缺点：**

优点:你可以直截了当地看到检查是否通过。

缺点:当检查失败,后续的检查不会被执行,无法收集那些检查的结果状态

**验证的优点和缺点：**

优点：遇到失败时，测试不会终止

缺点：不能得到测试失败的相关反馈

**及时得到反馈会更合适,因此断言通常比验证更常使用。**

#### **3、Python异常断言**

1）异常的抛出机制：

1> 如果在运行时发生异常，解释器会查找相应的处理语句（称为handler）.

2> 要是在当前函数里没有找到的话，它会将异常传递给上层的调用函数，看看那里能不能处理。

3> 如果在最外层（全局“main”）还是没有找到的话，解释器就会退出，同时打印出traceback 以便让用户找到错误产生的原因。

注意：虽然大多数错误会导致异常，但一个异常不一定代表错误，有时候它们只是一个警告，有时候它们可能是一个终止信号，比如退出循环等。

2）捕捉异常

**try....except...**

**输出报错信息**

**try...finally...**

finally语句必须要要执行，比如：文件关闭，把数据库连接返回给连接池

如图，按Ctrl+C中断程序

finally中的语句仍然被执行

3）raise抛出异常

#### **4、webdriver截图**

**Webdriver 提供错误截图函数get\_screenshot\_as\_file()**，可以帮助我们跟踪bug，在脚本无法继续执行时候， get\_screenshot\_as\_file()函数将截取当前页面的截图保存到指定的位置

#### **5、编写自动化测试实例**

编写自动化测试用例的原则：

1、一个脚本是一个完整的场景，从用户登陆操作到用户退出系统关闭浏览器。

2、一个脚本脚本只验证一个功能点，不要试图用户登陆系统后把所有的功能都进行验证再退出系统

3、尽量只做功能中正向逻辑的验证，不要考虑太多逆向逻辑的验证，逆向逻辑的情况很多（例如手号输错有很多种情况），验证一方面比较复杂，需要编写大量的脚本，另一方面自动化脚本本身比较脆弱，很多非正常的逻辑的验证能力不强。（我们尽量遵循用户正常使用原则编写脚本即可）

4、脚本之间不要产生关联性，也就是说编写的每一个脚本都是独立的，不能依赖或影响其他脚本。

5、如果对数据进行了修改，需要对数据进行还原。

6、在整个脚本中只对验证点进行验证，不要对整个脚本每一步都做验证。

unittest单元测试框架

1、引入单元测试框架

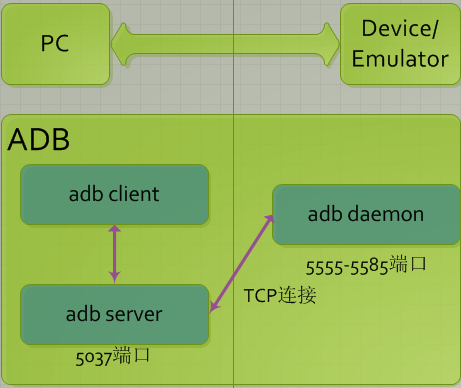
测试类：

setup()方法要在每个方法执行前执行一次

teardown()方法在每个方法执行后执行一次

# **【ADB 】Android Debug Bridge**

存放在SDK安装目录下的platform-tools文件夹中 – 直接与安卓设备进行交互



# 【Android studio + appium】

[**BetaCat 测试喵**](https://betacat.online/)

* [首页](https://betacat.online/)

* [分类](https://betacat.online/categories)

* [归档](https://betacat.online/archives)

* [标签](https://betacat.online/tags)

* [作品](https://betacat.online/works)

* [关于](https://betacat.online/about)

# **在 Windows 下搭建 Appium + Android 自动化测试环境**

 发表于 2017-05-03 |  分类于 [Tech](https://betacat.online/categories/tech/)

## 前言

本来并不打算写这么一篇文章，但是实践下来发现网上的各种教程里大致有两个问题。一是文章有些跟不上时代，目前android开发和测试的技术更新都比较快，内容有些过期。二是细节部分不是太完整，拼拼凑凑也能完成，但对新手来说就比较痛苦。那么，我也来试着总结一下自己踩过的坑。

**备注：**Android自动化测试环境和开发环境并不完全相同，测试环境可以很简单粗暴，很多工具可以不用安装，比如JDK，SDK Tools，测试脚本用C#，Python都可以完成；但是开发环境一般都是需要JDK和不少编译工具。

## 实践平台

Windows 10 Pro 64bit + Python 3.5.3

## 安装 JAVA 环境

如果只是自动化测试用，安装JRE就可以了，如果需要开发或者调试APK，那么请安装JDK。

1. JRE = JVM + Java SE 标准类库，相当于Java程序最基本的运行环境。
2. JDK = JRE + 开发工具集，包括javac编译工具等等。

JRE只有50多MB，JDK则接近200MB，请前往[Java官网](https://www.java.com/en/download/)下载最新版安装，选择64位。

## 安装 Android Studio / SDK

以前的教程里都会让你去装 Android SDK，但现在你从Google或者Bing里很难搜索到 Android SDK的下载链接，因为目前[官网](https://developer.android.com/)推荐的做法有两个：

1. 安装Android Studio （包含 Android SDK） – 推荐
2. 仅安装Android SDK 命令行工具

新版本的Android SDK和以前也不太一样，以前我们有一个 SDK Manager.exe 的工具用于更新和下载各版本的API，还有一个 AVD Manager.exe 的工具可以模拟各种型号的安卓设备。最新的 SDK 工具把UI 界面基本都去掉了，只留下命令行工具，具体的使用可以参阅官方文档：

* <https://developer.android.com/studio/releases/sdk-tools.html>

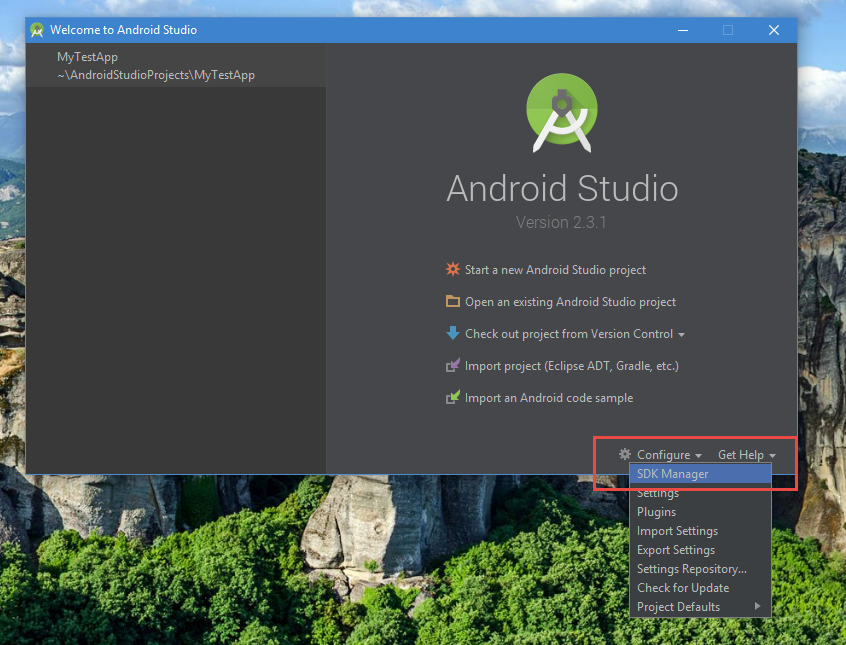
那么多一事不如少一事，我建议推荐你直接安装Android Studio，这样不仅省事，而且万一你想写个App玩玩，也是极其方便的。

[](https://tobyqin.github.io/images/android-sdk-download.png)

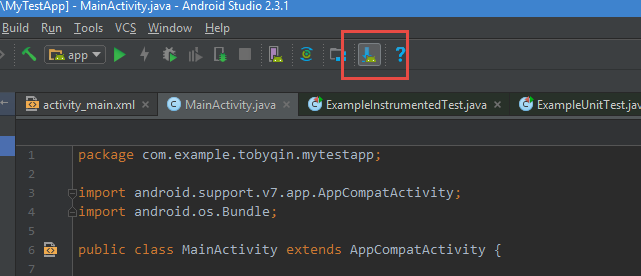
下载地址：<https://developer.android.com/studio/index.html>

安装过程非常简单，双击后一路Next，中间你也可以另外指定SDK的安装目标路径（不建议修改）。安装好之后，直接启动 Android Studio，首先会提示你联网下载一些必要的工具（你需要一个好的网络和梯子）。

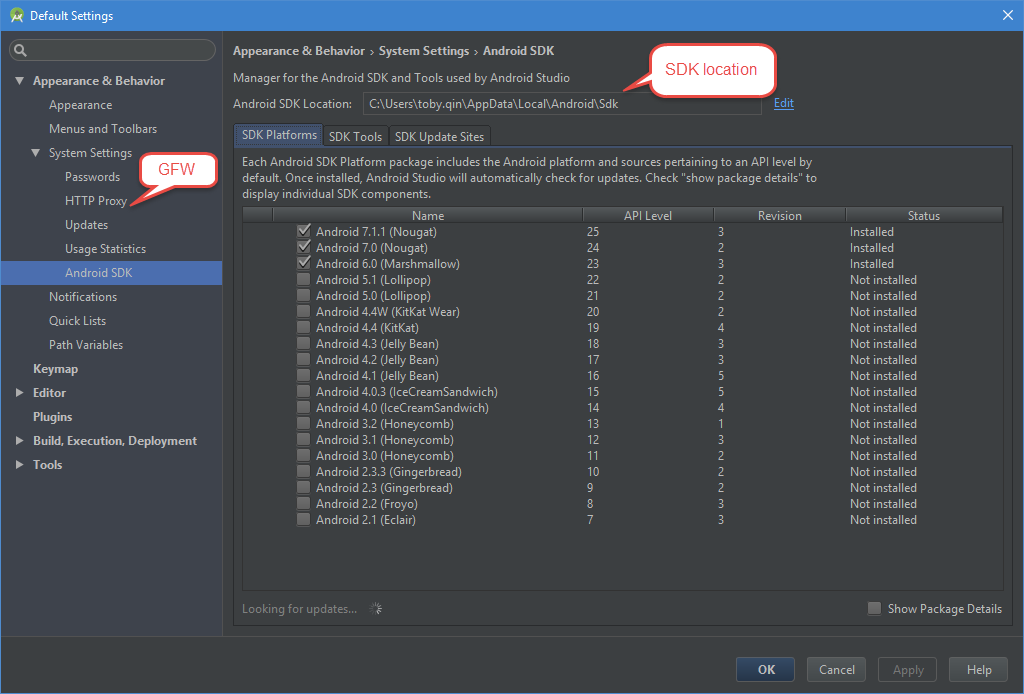
一切就绪后，你可以在启动界面就可以打开 SDK Manager。

[](https://tobyqin.github.io/images/sdk-manager-from-launch.png)

或者新建一个工程，从工程界面打开SDK Manager。

[](https://tobyqin.github.io/images/sdk-manager-from-project.png)

SDK Manager 打开之后，使用的方法就一目了然了。这就是个让你更新API和各种开发工具的工具。

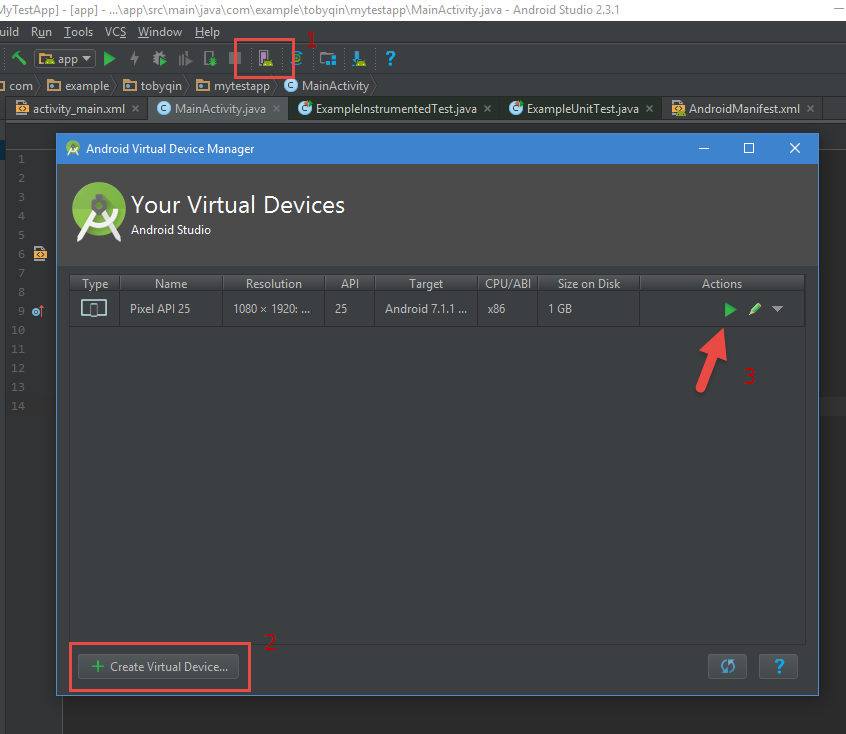
[](https://tobyqin.github.io/images/sdk-manager-usage.png)

如果只是为了自动化测试其实你只要记住SDK安装目录就可以了，不需要安装任何多余的工具。SDK的安装目录在配置环境变量时需要用到。

如果你的网络不畅通，那么需要先设置代理。接下来才能安装你想要的SDK版本和开发者工具。

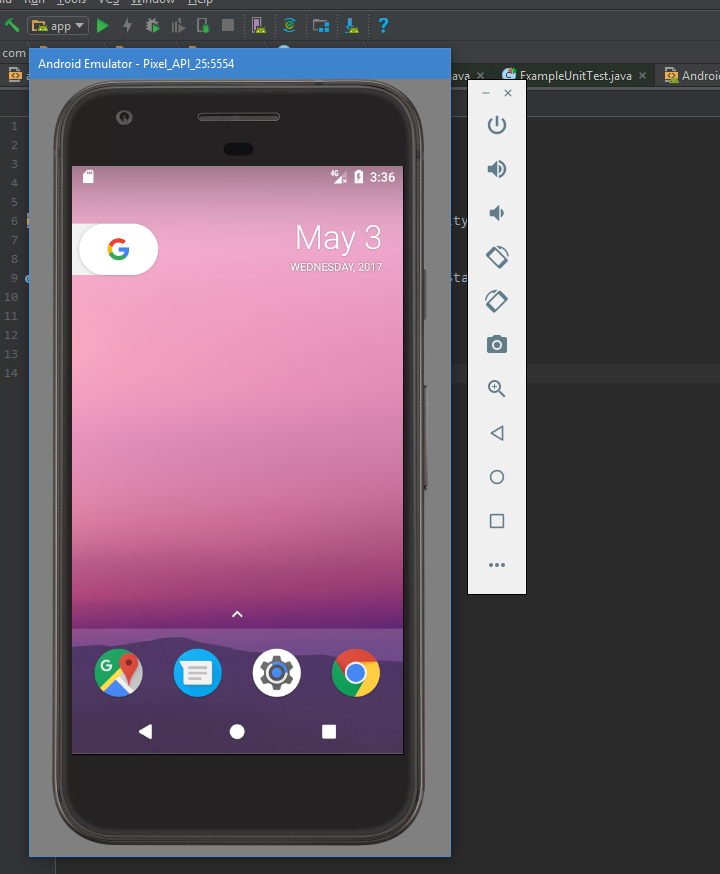
## 安装 Android 模拟器

如果你已经安装了Android Studio，模拟器也已经有了，从工程界面找到AVD Manager的按钮，按照下图三个步骤就可以添加并启动一个模拟器。

[](https://tobyqin.github.io/images/avd-manager.png)

每当你安装新版本的模拟器都是需要翻墙或者代理的，最开始启动Android Studio时它已经给你下一个最新版的Android，我当前模拟器中的Android版本是7.1.1。

如果一切顺利，那么当你启动模拟器后，你就可以看到一台虚拟的Android设备了，手动测试基本条件已经达成，自动化测试指日可待。

[](https://tobyqin.github.io/images/android-avd-start.png)

多说几句，除了使用 Android SDK 自带的模拟器外，我们还有两个选择，一是使用真机，速度和体验上会更好。另外一个选择就选择别的模拟器产品，比如 [Genymotion](https://betacat.online/posts/2017-05-03/setup-appium-automation-test-environment/)，因为是商业软件，所以对个人用户（免费）会有诸多限制，尽管如此，其成熟度也比Android SDK自带的模拟器高很多。因为这两个方案的具体实现涉及到的内容比较多，故不在本文讨论。

## 安装 Appium

Appium是开源的自动化测试框架，主要用于iOS，Android以及Windows apps等移动平台的自动化测试。官网的介绍说是它实现一套适用于移动端的WebDriver协议，所以使用Appium时用的还是依赖于Selenium，和Web自动化测试的主要区别就在Driver不一样。

举一个例子，如果我们需要用Google Chrome来跑Selenium的自动化测试，那么首先需要一个ChromeDriver。如果需要跑在IE上，那么需要一个IEDriver。那么现在需要在移动端比如Android里跑Selenium，这时候你需要一个Appium就够了，它担当了Driver的角色。Appium实现了一套标准的WebDriver，只要Appium服务起来之后，你的代码只需要和Appium交互，Appium会去告诉设备该干嘛干嘛。这里提到的设备可以是iOS或者Android，甚至是Windows Phone 和Firefox OS。

Appium 官网： <http://appium.io/>

### 安装方式一： 使用NodeJS 安装

首先到[NodeJS官网](https://betacat.online/posts/2017-05-03/setup-appium-automation-test-environment/)下载安装最新的NodeJS，Windows下属于傻瓜安装。安装好之后你可以配置NodeJS的安装源，在国内一般都用taobao的镜像，速度还不错。

在个人目录下新建一个 .npmrc 文件，写入：

|  |  |
| --- | --- |
| 1 | registry=https://registry.npm.taobao.org/ |

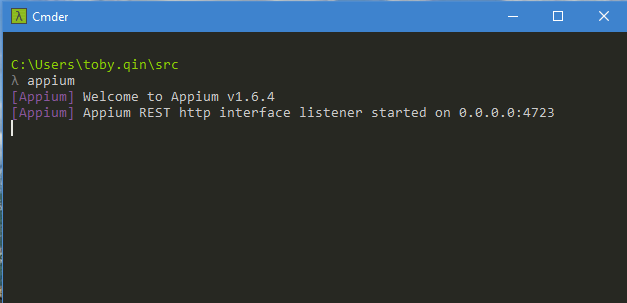
然后使用npm命令安装 Appium

|  |  |
| --- | --- |
| 1 | npm install -g appium |

建议顺便安装一下appium-doctor，通过运行appium-doctor命令可以快速检查appium的环境问题。

|  |  |
| --- | --- |
| 1 | npm install -g appium-doctor |

如果安装成功，那么就可以通过 appium 命令启动appium server。

[](https://tobyqin.github.io/images/appium-start.png)

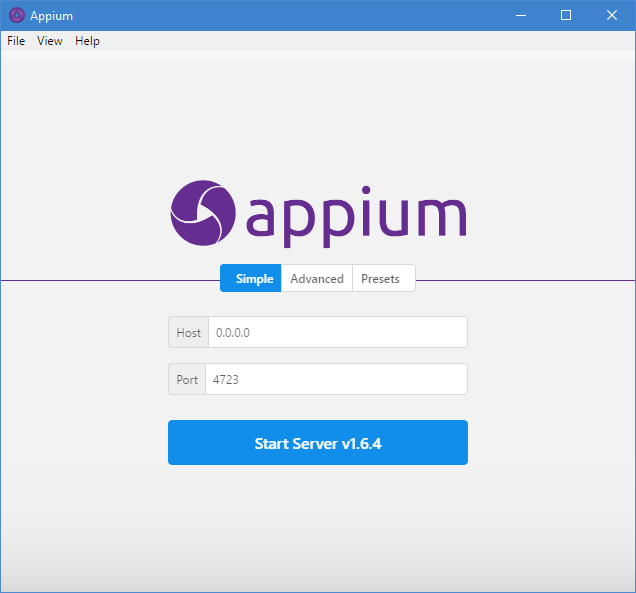
关闭命令行或者按 Ctrl - C 就可以停止 Appium Server。

### 安装方式二： 使用Appium安装包

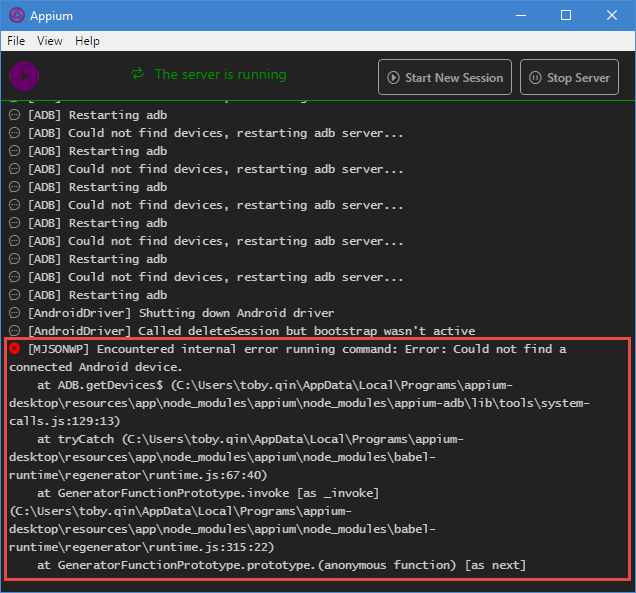
网上很多教程所引导的Appium安装包一般在百度网盘或者bitbucket里，其实这两者都不是最新的。最新版的安装包应该从官网下载，Appium 目前托管在github，正确的下载地址应该为：

* <https://github.com/appium/appium-desktop/releases>

最新版的Appium 桌面安装后启动是这样的。

[](https://tobyqin.github.io/images/appium-desktop.png)

功能上和老的版本没太大区别，但是日志和UI显示更加清晰，一旦错误发生让你更容易找到问题所在。

[](https://tobyqin.github.io/images/appium-desktop-error.png)

注意，你可以不安装Appium 桌面版，但是命令行版应该还是需要安装的，因为自动化测试运行时一般都是通过命令启动和关闭Appium，桌面版并不提供命令行功能，只是为了调试方便。

关于Appium的介绍，这里有一份非常好的文档：<http://appium.io/slate/cn/master>

### 安装 Appium-Client

本文只讨论Python实现的Appium测试，所以你只需要允许以下命令：

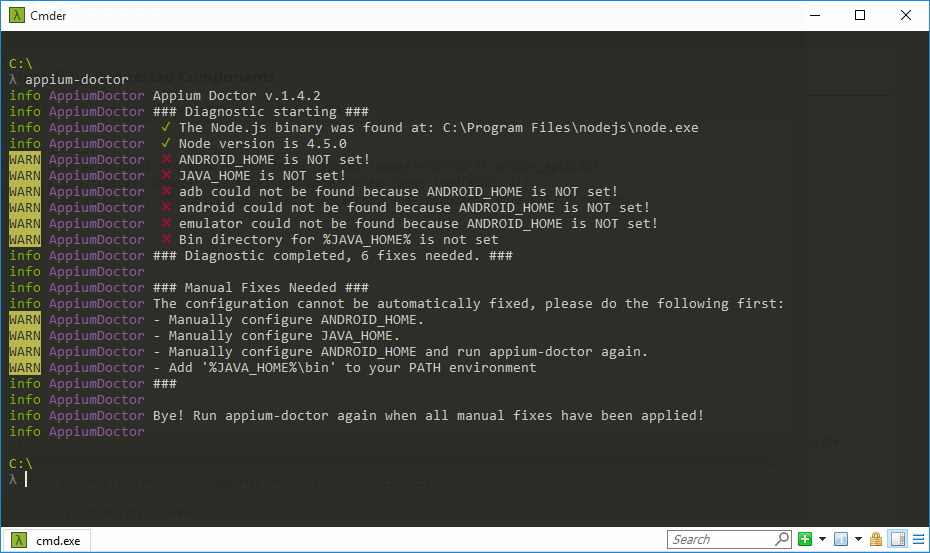
|  |  |
| --- | --- |
| 1 | pip install Appium-Python-Client |

如果需要使用其他编程语言，下表供参考：

| **语言/框架** | **Github版本库以及安装指南** |
| --- | --- |
| Ruby | <https://github.com/appium/ruby_lib> |
| Python | <https://github.com/appium/python-client> |
| Java | <https://github.com/appium/java-client> |
| JavaScript (Node.js) | <https://github.com/admc/wd> |
| Objective C | <https://github.com/appium/selenium-objective-c> |
| PHP | <https://github.com/appium/php-client> |
| C# (.NET) | <https://github.com/appium/appium-dotnet-driver> |
| RobotFramework | <https://github.com/jollychang/robotframework-appiumlibrary> |

## 必要的环境变量设置

如果你已经安装了appium-doctor，那么你只要运行appium-doctor命令就可以知道你还需要设置哪些环境变量，比如：

[](https://tobyqin.github.io/images/appium-doctor.png)

不要慌，其实你只需要设置以下环境变量：

| **环境变量** | **值** |
| --- | --- |
| ANDROID\_HOME | Android SDK 的安装位置 |
| JAVA\_HOME | JDK 或者 JRE 的安装位置 |
| 加入 PATH | %ANDROID\_HOME%\tools |
| 加入 PATH | %ANDROID\_HOME%\platform-tools |
| 加入 PATH | %JAVA\_HOME%\bin |
| 加入 PATH | %ANDROID\_HOME%\build-tools\??version?? (可选) |

最后一个环境变量是为了更方便地使用aapt这个工具，完成环境变量配置后你可以再次运行appium-doctor进行检查。不出意外，你应该能通过检查。

## 完成第一个自动化测试

终于来到了动真刀真枪的时候了，很多人一开始就放弃了，更多人开始了之后就放弃了，走到这一步真的挺不容易的。

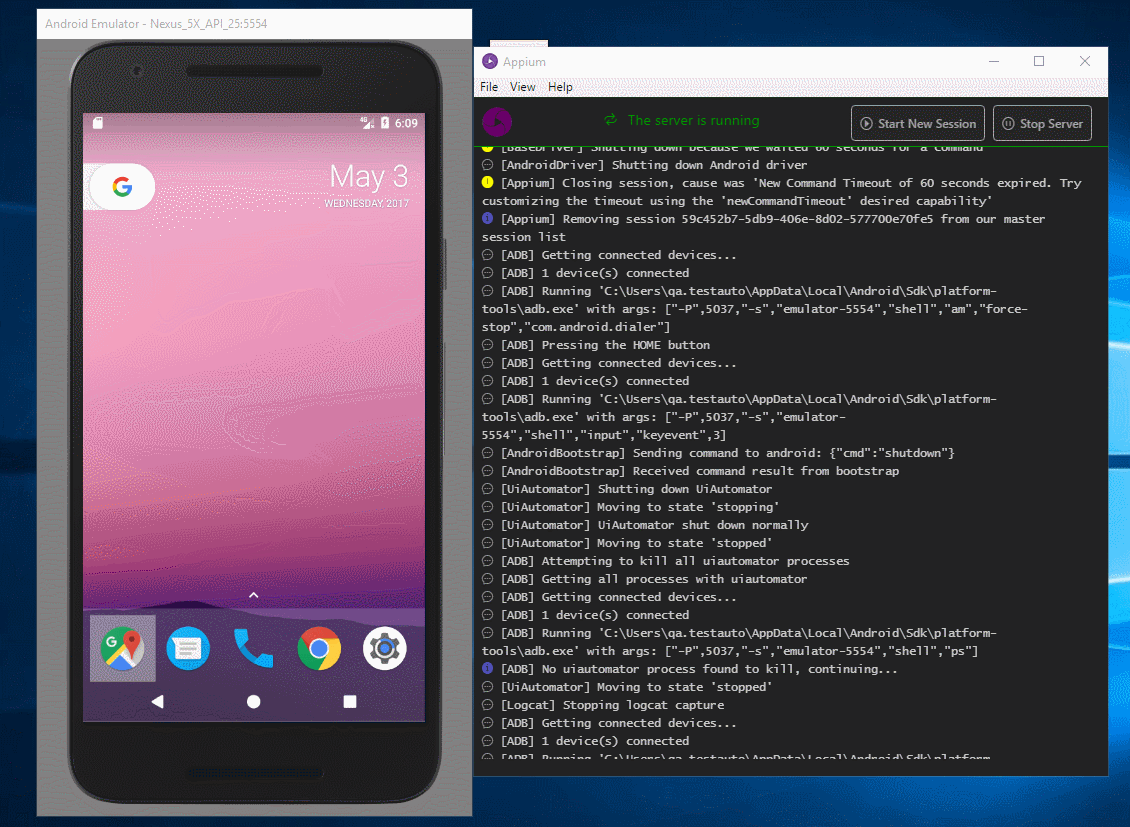
开始写代码之前有些事情你应该清楚：

1. 模拟器或者测试机必须是Ready的状态，每次启动模拟器都很费时间，所以建议模拟器不要关闭，通过代码来启动模拟器是一个办法，但是时间成本有点高。
2. 如果是调试代码阶段，建议保持Appium桌面版长期运行，远程运行时再通过代码自动启动和关闭Appium。
3. Appium会根据你的测试代码去寻找符合要求的设备，如果你启动了多台虚拟机或者连接了多台实体机，请显式地在代码中指定设备名称或者版本号。

第一个自动化测试需求很就简单，就是启动内置的拨号软件，搜索 “hello toby”。代码如下：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | from appium import webdriver  desired\_caps = {}  desired\_caps['platformName'] = 'Android'  desired\_caps['platformVersion'] = '7.1.1'  desired\_caps['deviceName'] = 'Android Emulator'  desired\_caps['appPackage'] = 'com.android.dialer'  desired\_caps['appActivity'] = 'DialtactsActivity'  driver = webdriver.Remote('http://localhost:4723/wd/hub', desired\_caps)  driver.find\_element\_by\_id('com.android.dialer:id/search\_box\_collapsed').click()  search\_box = driver.find\_element\_by\_id('com.android.dialer:id/search\_view')  search\_box.click()  search\_box.send\_keys('hello toby') |

运行效果：

[](https://tobyqin.github.io/images/appium-hello.gif)

恭喜你，你已经开始进入移动端的自动化测试的大门了！

[](https://betacat.online/images/wechat-qrcode.jpg)

欢迎到微信里去当吃瓜群众

* **本文作者：** Toby Qin
* **本文链接：** <https://betacat.online/posts/2017-05-03/setup-appium-automation-test-environment/>
* **版权声明：**本博客所有文章除特别声明外，均采用 [CC BY-NC-SA 3.0](https://creativecommons.org/licenses/by-nc-sa/3.0/) 许可协议。转载请注明出处！

[# python](https://betacat.online/tags/python/) [# selenium](https://betacat.online/tags/selenium/) [# appium](https://betacat.online/tags/appium/) [# android](https://betacat.online/tags/android/)

[如何开发一个 PyCharm 插件](https://betacat.online/posts/2017-04-01/steps-to-create-a-pycharm-plugin/)

[一款优秀的代码高亮库 - rainbow.js](https://betacat.online/posts/2017-07-17/rainbow-js-highlight-lib/)

* 文章目录

* 站点概览

1. [1. 前言](https://betacat.online/posts/2017-05-03/setup-appium-automation-test-environment/#前言)
2. [2. 实践平台](https://betacat.online/posts/2017-05-03/setup-appium-automation-test-environment/#实践平台)
3. [3. 安装 JAVA 环境](https://betacat.online/posts/2017-05-03/setup-appium-automation-test-environment/#安装-JAVA-环境)
4. [4. 安装 Android Studio / SDK](https://betacat.online/posts/2017-05-03/setup-appium-automation-test-environment/#安装-Android-Studio-SDK)
5. [5. 安装 Android 模拟器](https://betacat.online/posts/2017-05-03/setup-appium-automation-test-environment/#安装-Android-模拟器)
6. [6. 安装 Appium](https://betacat.online/posts/2017-05-03/setup-appium-automation-test-environment/#安装-Appium)
7. [7. 必要的环境变量设置](https://betacat.online/posts/2017-05-03/setup-appium-automation-test-environment/#必要的环境变量设置)
8. [8. 完成第一个自动化测试](https://betacat.online/posts/2017-05-03/setup-appium-automation-test-environment/#完成第一个自动化测试)

© 2015 - 2017  Toby Qin