# Code Page ,字符集，字符编码

字符集的定义其实就是字符的集合，而字符编码则是指怎么将这些字符变成字节用于保存、读取和传输。

字符编码用于解决保存的问题，codepage用于解决显示的问题。

Unicode字符的编码方案一般被称为Unicode Translation Format，即UTF。

字体（库）是与字符集有关系的，而所谓的乱码，则是与字符编码有关系的。现在好像还没有什么字体能涵盖unicode字符集。部分中文字体只能涵盖GB2312字符集。所以所谓的“宋体-方正超大字符集”大家应该明白是什么意思了。

“Unicode编码”到底指什么？  
“Unicode编码”又是一个容易搞混的概念。大体上，Unicode编码可以指：  
1、字符在Unicode字符集内的序列编码，这是一个32位的二进制数（不过包括规划的字符，24位都足够了）。以前这是一个16位的二进制数（古老的事情）。  
2、UTF-16编码方案，这是一种使用两到四个字节表示Unicode字符的编码方案（因为Unicode字符现在连24位二进制的表都摆不满）。值得注意的是UTF-16编码方案有个版本，一个叫大尾序，一个叫小尾序，鉴于网上能普遍找到的东西我就懒得写了，所以就此略过。UnicodeEncoding类就是这种编码方式的实现。  
  
这两者看起来什么关系都没有，怎么会都被叫做Unicode编码呢？因为对于大多数常见的Unicode字符而言，UTF-16编码方案两个字节连起来就是Unicode序号。

Win-dos窗口输入中文乱码，可使用chcp 65001（utf-8编码）、chcp 936（简体中文gbk编码）

## ASCII编码

计算机中所有的信息最终都是表示为一个二进制字符串，每个二进制位（bit）有0、1两种状态，8位可有256中状态称为一个字节（byte）

美国制定的一套字符编码，对英语字符与二进制做了统一对应关系🡺ASCII码

ASCII码一共规定了128个字符编码（包含32个不能打印出来的控制符号），值占用7位，最前面的1位统一规定为0

## Unicode

它是一种所有符号的编码（中英文、符号、汉字……）

需要注意的是：Unicode只是一个符号集，它只规定了符号的二进制代码，却没有规定这个二进制代码应该如何存储。

造成的结果：出现Unicode的多种存储方式，即有许多种不同的二进制格式可以用来表示Unicode。

## UTF-8

Utf-8是Unicode的实现方式之一。

特点：它是一种变长的编码方式，可以使用1-4个字节表示一个字符，根据不同的符号而变化字节长度。

## 记事本的几种编码

ANSI—对于英文文件是ASCII编码，对于简体中文是GB2312

Unicode编码—UCS-2编码方式，即直接用两个字节存入字符的Unicode码，这个选项用的little endian格式

Unicode big endian编码—与上一个相对应

Utf-8

以汉字"严"为例，Unicode码是4E25，需要用两个字节存储，一个字节是4E，另一个字节是25。存储的时候，4E在前，25在后，就是Big endian方式；25在前，4E在后，就是Little endian方式。

Unicode规范中定义，每一个文件的最前面分别加入一个表示编码顺序的字符，这个字符的名字叫做"零宽度非换行空格"（ZERO WIDTH NO-BREAK SPACE），用FEFF表示。这正好是两个字节，而且FF比FE大1。

如果一个文本文件的头两个字节是FE FF，就表示该文件采用大头方式；如果头两个字节是FF FE，就表示该文件采用小头方式。

UTF-8：编码是六个字节"EF BB BF E4 B8 A5"，前三个字节"EF BB BF"表示这是UTF-8编码，后三个"E4B8A5"就是"严"的具体编码，它的存储顺序与编码顺序是一致的

# Win-Dos常用命令：

关机：shutdown /r /f /t 0 r-restart ,f-force ,t-time /s-shutdown

选择开机自启项：msconfig

查看目录文件：dir ，dir /w 紧凑方式显示, dir /p 分页显示

磁盘整理：defrag path /u /v

显示进程：tasklist

杀进程：taskkill /f /im pid

# Windows远程桌面

1. 在防火墙上设置允许TCP 3389端口上的访问；

在【防火墙】界面选择【高级设置】，弹出防火墙高级设置界面，在点击入站规则。

【入站规则】中选择新建规则。

【新建规则】：1-选取端口；

在特定本地端口框中输入我们需要绑定局域网可访问端口，即：2000；

允许连接；

命名为test，点击完成。

1. 为管理员设置密码，以便访问者访问；
2. 我的电脑-右键属性-远程设置-远程-选上“允许远程协助远程这台计算机、允许运用任意版本远程”。
3. 控制面板-管理工具-服务-开启Remote Desktop Services。

# 安卓内存查看

adb shell dumpsys meminfo packagename or pid （adb shell dumsys meminfo com.tencen.qq）//查看指定应用或进程的内存情况

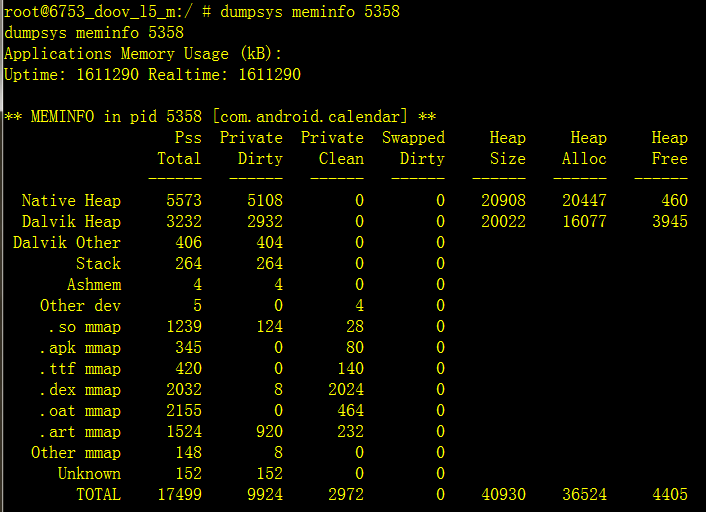
adb shell dumpsys meminfo system//查看系统内存

adb shell procrank//查看所有进程内存使用情况

adb shell cat /proc/meminfo//查看内存快照文件

android程序内存被分为2部分：native和dalvik，dalvik就是我们平常说的java堆，我们创建的对象是在这里面分配的，而bitmap是直接在native上分配的，对于内存的限制是 native+dalvik 不能超过最大限制。android程序内存一般限制在16M，当然也有24M的。

其中size是需要的内存，而allocated是分配了的内存，对应的2列分别是native和dalvik，当总数也就是total这一列超过单个程序内存的最大限制时，OOM就很有可能会出现了。



# Adb命令、logcat

从4.2.2开始，adb连接新增了认证确认功能，即用USB连接时在手机上会出现需要确认的对话框。用户点击确认后才可继续。在用户确认之前或出现其它认证相关的问题时，adb devices显示的状态是**unauthorized**。确认之后，android手机会在/data/misc/adb/adb\_keys这个文件上添加一行key数据，据推测这个key应是运行adb的PC用户的RSA公钥。之后这个PC的用户运行adb连接这个手机时就不会再出现认证确认对话框了。

## 通用adb

1. adb shell dmesg//打印出内核的调试信息

adb forward tcp:5555 tcp:8000//发布端口

adb get-product、adb get-serialno//获取设备ID和序列号

adb shell sqlite3//访问数据库

adb reboot//重启手机

adb reboot recovery//恢复出厂

adb shell screencap –p /sdcard/pic.png //截屏

adb shell screenrecord /sdcard/video.mp4 //录屏 –verbose输出详情，--time-limit录制时长，--size 视频尺寸大小

重新挂载/system分区：需root权限，adb shell 🡪su🡪mount🡪mount -o remount,rw -t yaffs2 file\_path

Adb shell cat /data/misc/wifi/\*.conf //查看已连过的wifi密码

Adb shell date –s 20160101122020 //修改系统时间

Adb shell svc wifi enable/disable //开启关闭wifi开关

adb connect ip:port//在同一局域网内使用ip地址连接手机

adb –s serialnumber shell//指定某序列号的手机

## adb shell pm

adb shell pm list packages//列出手机安装的应用的包名package manager

adb shell pm list packages -3 //只列出第三方应用

adb shell pm list permissions //列出所有应用获取的权限

adb shell pm clear package\_name //清空应用数据

## adb shell am

adb shell am start –n package\_name/.launch\_class (com.tencent.mobileqq/.activity.SplashActivity)//启动应用

adb shell force-stop packagename//强制退出某应用

aapt d badging xxx.apk //查看某应用包名

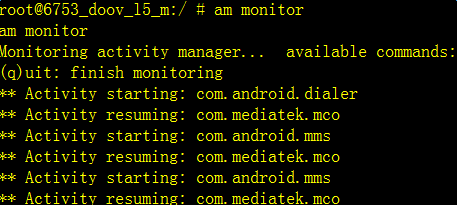
adb shell am startservice uid //开启某服务

adb shell am stopservice uid //停止某服务

adb shell am broadcast uid content //发送一条广播消息

adb shell am instrument //调用android测试框架

adb shell am monitor //监视应用启动，可查看应用包名



## uiautomator

adb shell uiautomator runtest mtbf.jar -c com.yunos.mtbf.tests.aging.SendSMS -e runner com.yunos.test.uitest.framework.AliUiTestRunner //调用uiautomator执行测试用例

## service、dumpsys

service list //列出当前的服务

常用服务：

cpu信息： cpuinfo

内存信息： meminfo

输入法信息： input\_method

手机当前的活动窗口： window

音频信息： media.audio\_flinger

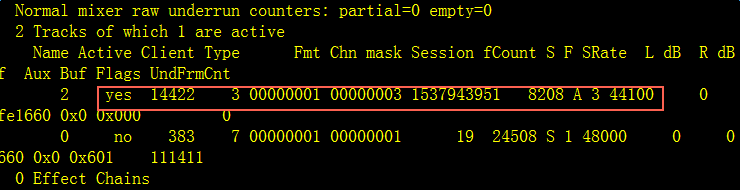
fps信息：surfaceflinger （需要root）

input 等

**通用dumpsys xxx 如：dumpsys cpuinfo**

dumpsys meminfo package\_name //指定应用的内存情况

dumpsys media.audio\_flinger //音频 查看pid对应session id –->yes 表示正在播放



dumpsys batterystats //查看电量情况

dumpsys batterystats > package\_name //查看指定应用的电量情况

## monkey

系统目录：system/bin/monkey

adb shell monkey -s 123 –p package\_name --pct-touch 100 --thrrotle 400 –vvv --kill-process-after-error --ignore-timeouts --ignore-security-exceptions --ignore-crashes 120000

## tcpdump

* tcpdump host 10.63.64.21 //只打印与指定主机之间的通信
* tcpdump -i eth0 src host hostname //截获主机发出的数据
* tcpdump -i eth0 dst host hostname //截获主机接收到的数据
* tcpdump tcp port 23 and host 210.27.48.1 //指定主机和端口，23为telnet
* tcpdump tcp -i eth1 -t -s 0 -c 100 and dst port ! 22 and src net 192.168.1.0/24 -w ./target.cap //

(1)tcp: ip icmp arp rarp 和 tcp、udp、icmp这些选项等都要放到第一个参数的位置，用来过滤数据报的类型

(2)-i eth1 : 只抓经过接口eth1的包

(3)-t : 不显示时间戳

(4)-s 0 : 抓取数据包时默认抓取长度为68字节。加上-S 0 后可以抓到完整的数据包

(5)-c 100 : 只抓取100个数据包

(6)dst port ! 22 : 不抓取目标端口是22的数据包

(7)src net 192.168.1.0/24 : 数据包的源网络地址为192.168.1.0/24

(8)-w ./target.cap : 保存成cap文件，方便用ethereal(即wireshark)分析

## Input

Adb shell input text xxxx //输入内容，需焦点在文本框，输入法为开启状态

Adb shell input tap 120 220 //点击坐标位置

Adb shell input swipe 120 220 400 520 //滑动屏幕

## logcat

1. adb logcat ActivityManager:I MyApp:D \*:S//logcat分级过滤

V-verbs（最低优先级），D-debug，I-info，W-warning，E-error，F-fault，S-silent（最高优先级）

export ANDROID\_LOG\_TAGS="ActivityManager:I MyApp:D \*:S"//设置默认过滤器。

1. **控制日志的输出格式：-v加以下字段：**

brief-显示优先级、标记和原始进程的PID

time-显示日期，优先级、标记、原始进程PID

thread-仅显示进程，线程和优先级/标记

tag-仅显示优先级/标记

process-仅显示进程PID

long —显示所有的元数据字段并且用空行分隔消息内容

raw — 显示原始的日志信息，没有其他的元数据字段。

只能在-v后加一种格式。

1. Android日志系统为日志消息保持了多个循环缓冲区，而且不是所有的消息都被发送到默认缓冲区，要想查看这些附加的缓冲区，可以使用-b 选项，以下是可以指定的缓冲区：

radio — 查看包含在无线/电话相关的缓冲区消息

events — 查看事件相关的消息

main — 查看主缓冲区 (默认缓冲区)

1. **内存占用：一般看PSS的值**

VSS - Virtual Set Size 虚拟耗用内存（包含共享库占用的内存）

RSS - Resident Set Size 实际使用物理内存（包含共享库占用的内存）

PSS - Proportional Set Size 实际使用的物理内存（比例分配共享库占用的内存）

USS - Unique Set Size 进程独自占用的物理内存（不包含共享库占用的内存）

# Android知识图：

[android结构图](http://blog.csdn.net/xyz_lmn/article/details/41411355)

# Jmeter

# 32位64位操作系统

32位和64位操作系统是指：CPU一次处理数据的能力是32位还是64位。现在市场上的CPU一般都是64位的，但是这些CPU并不是真正意义上的64 位CPU，里面依然保留了大部分32位的技术，只是进行了部分64位的改进。32位和64位的区别还涉及了内存的寻址方面，32位系统的最大寻址空间是2 的32次方= 4294967296（bit）= 4（GB）左右，而64位系统的最大寻址空间的寻址空间则达到了2的64次方= 4294967296（bit）的32次方=数值大于1亿GB。换而言之，就是说32位系统的处理器最大只支持到4G内存，而64位系统最大支持的内存高 达亿位数。

# 专业名词

1. SMTP（Simple Mail Transfer Protocol）即简单邮件传输协议,它是一组用于由源地址到目的地址传送邮件的规则，由它来控制信件的中转方式。SMTP协议属于TCP/IP协议簇，它帮助每台计算机在发送或中转信件时找到下一个目的地。
2. Tcp/IP协议（Transmission Control Protocol/Internet Protocol）即传输控制协议/因特网互联协议，又名网络通讯协议。
3. MD5（Message Digest Algorithm）即消息摘要算法第五版，为计算机安全领域广泛使用的一种散列函数，用以提供消息的完整性保护，作用是让大容量信息在用数字签名软件签署私人密钥前被"压缩"成一种保密的格式（就是**把一个任意长度的字节串变换成一定长的十六进制数字串**）
4. Ajax（Asychronous Javascript And XML）异步javascript和xml，是指一种创建交互式网页应用和网页开发的技术。通过在后台与服务器进行少量数据交换，AJAX 可以使网页实现异步更新。这意味着可以在不重新加载整个网页的情况下，对网页的某部分进行更新。
5. HTTP（Hyper Text Transfer Protocol）超文本传输协议，互联网上应用最为广泛的一种网络协议，所有www文件都必须遵守这个标准。
6. HTTPS即http的安全版本（Hyper Text Transfer Protocol over Secure Socket Layer）以安全为目标的HTTP通道，在HTTP下加入SSL层。
7. IPC（Inter-Process Communication）进程间通信，指至少两个进程或线程间传输数据或信号的一些技术或方法。
8. Base64,网络上最常见的用于传输8Bit字节码的编码方式之一，用做HTTP表单和URL参数中，采用Base64编码不仅比较简短，同时也具有不可读性，即所编码的数据不会被人用肉眼所直接看到。
9. JNI（Java Native Interface）提供了若干的API，实现了java和其他语言的通信交互。（主要是C&C++）。
10. MemCache是一套分布式的高速缓存系统，常用于网站提升网站访问速度，接收请求后会先检查客户端的请求数据是否在MemCache中，以减少对系统数据库的访问。
11. Socket 套接字，网络上两个程序通过一个双向的通信连接实现数据的交换，这个连接的一端称为一个socket，其本质为编程接口。网络化的应用程序在开始任何通信之前必须创建套接字。
12. CSS Cascading Style Sheets层叠样式表，是一种用来表现HTML或XML等文件样式的计算机语言。样式定义如何显示 HTML 元素，通常存储在样式表中；把样式添加到 HTML 4.0 中，是为了解决内容与表现分离的问题；外部样式表可以极大提高工作效率，通常存储在 CSS 文件中；多个样式定义可层叠为一。
13. HTML HyperText Markup Language 超文本标记语言，HTML 是用来描述网页的一种语言；它不是一种编程语言，而是一种标记语言 (markup language)，标记语言是一套标记标签 (markup tag)；使用标记标签来描述网页。
14. UTF-8--8-bit Unicode Transformation Format）是一种针对Unicode的可变长度字符编码，又称万国码。
15. Unicode Unicode 是为了解决传统的字符编码方案的局限而产生的，它为每种语言中的每个字符设定了统一并且唯一的二进制编码，以满足跨语言、跨平台进行文本转换、处理的要求。
16. ASCII-- American Standard Code for Information Interchange，美国信息交换标准代码。标准的单字节字符编码方案，用于基于文本的数据，主要用于显示现代英语和其他西欧语言。它是现今最通用的单字节编码系统。
17. Json –javascript object notation是一种轻量级的数据交换格式，采用完全独立于语言的文本格式，使得json称为理想的数据交换语言。
18. CDN（Content Delivery Network）内容分发网络。CDN是构建在网络之上的内容分发网络，依靠部署在各地的边缘服务器，通过中心平台的负载均衡、内容分发、调度等功能模块，使用户就近获取所需内容，降低网络拥塞，提高用户访问响应速度和命中率。CDN的关键技术主要有内容存储和分发技术。CDN的基本原理是广泛采用各种缓存服务器，将这些缓存服务器分布到用户访问相对集中的地区或网络中，在用户访问网站时，利用全局负载技术将用户的访问指向距离最近的工作正常的缓存服务器上，由缓存服务器直接响应用户请求。
19. PAC-proxy auto-config自动切换代理服务器。一个函数判断访问某个主机是否走代理。

# SVN

## 下载安装

<http://tortoisesvn.net/downloads>

## 概念

subversion，版本控制+备份服务器，称用来存放上传档案的地方就做**Repository**；上传已经存在且修改过的档案就叫做**commit，**即提交修改给SVN server；每次的commit，SVN server都会赋予他一个**新的版本**。同时也会把每次上传的**时间记录**下来，可以随时取得某一个时刻的版本。SVN的存放档案方式是采用**差异备份**的方式，只会备份有不同的地方。

## 在widows下建立SVN Repository

1. 在磁盘下新建空目录如：svn\_repo
2. 右键文件TortoiseSVN->Create repository here，可选择FSFS就可以，ok。访问目录类似：[http://svn.**simba.taobao.com**/svn/tianyun\_dept/MobileProduct/CloudPlatform/FunctionTest/AutoCase/codebaseTestCase/systemteam/mtbf\_script/](http://svn.simba.taobao.com/svn/tianyun_dept/MobileProduct/CloudPlatform/FunctionTest/AutoCase/codebaseTestCase/systemteam/mtbf_script/)
3. 或者file:///D:/svn\_repo

## 创建working目录

可以在working目录的icon上按，也可进入working目录后，在空白的地方右键 TortoiseSVN->checkout；输入已经创建的repository地址；SVN会在您的工作目录下，以及其子目录下建立这个.svn的子目录（隐藏文件，不可更改）

## 提交内容

* 如果新增文件—在文件名右键TortoiseSVN->Add；这个Add的动作并未真正的将档案放到Repository中，仅仅是告知SVN准备要在Repository中放入这些档案。
* 让这些档案真正 的放入到 Repository中，要执行commit动作，右键SVN commit，需要添加commit信息。
* 同步文件update
* 需要回溯至特定的日期或是版本，这时就可以利用SVN的Update to revision的功能。Show log查看之前版本的信息。

## 分支branch

很多时候您会希望有另外一个复制的目录来进行新的编修。等到确定这个分支的修改已经完毕了，再合并到原来的主要开发版本上。

1. 新建分支目录，选择Branch/Tag…这个项目。
2. 通常我们会将所有的branch集中在一个目录下面，**确认From WC at URL，**在**To URL**需要输入您要的目录即可。目录不存在时，会由SVN帮您建立。需要注意SVN为**斜线作为目录分隔字符**，而非反斜线。
3. 进入分支目录，需要在branch子目录下面进行SVN update就可以看到这个新增的目录。
4. 修改完成后要merge回trunk目录中， 在D:\working\my\_prj\trunk目录空白处，按下鼠标右键，选择**Merge**。选择From跟To的URL字段，可以通过**Dry run**按钮，试作一次Merge。这个merge只会显示一些讯息，不会真正的更新到trunk的目录去。
5. 确认merge没有问题，就可以直接使用**commit**来将被修改的档案commit回**SVN repository**上。

## 制作Tag或是Release

Tag与Release的作法与Branch完全相同。只是Branch可能会需要merge回原来的trunk中，而tag及 release大部分都不需要merge回trunk中。

* 在Trunk目录上面，按下鼠标右键，选择Branch/Tag的项目，新建tag/1.0

# Fiddler抓包工具

## 下载安装

1. [fiddler4\_4.6.2.0window版本](http://dlsw.baidu.com/sw-search-sp/soft/47/10963/fiddler4_4.6.2.0_setup.1453708442.exe)
2. 能记录所有客户端和服务器的http和https请求，允许你**监视**，**设置断点**，甚至**修改输入输出数据**。
3. 同类的工具有: **httpwatch, firebug, wireshark**
4. Fiddler2启动的时候默认IE的代理设为了127.0.0.1:8888，而其他浏览器是需要手动设置的，所以将Firefox的代理改为127.0.0.1:8888就可以监听数据了：Tools -> Options, 在Options 对话框上点击Advanced tab - > network tab -> setting
5. 需要手动开启捕获https会话：Fiddler Tool->Fiddler Options->HTTPS tab

## 应用

### 常见命令有

help--打开官方的使用页面介绍，所有的命令都会列出来；cls--清屏 (Ctrl+x 也可以清屏)； select--选择会话的命令；?.png--用来选择png后缀的图片；bpu--截获request

### Fiddler中设置断点修改Request

设置断点后，可以修改httpRequest的任何信息包括host, cookie或者表单中的数据。

* 打开Fiddler 点击Rules-> Automatic Breakpoint ->Before Requests(这种方法会**中断所有的会话**)；如何消除命令？点击Rules->Automatic Breakpoint ->Disabled
* 在命令行中输入命令: bpu www.baidu.com (这种方法只会中断[www.baidu.com](http://www.baidu.com))；如何消除命令？在命令行中输入命令bpu

### Fiddler中设置断点修改Response

1. 打开Fiddler 点击Rules-> Automatic Breakpoint ->After Response (这种方法会中断所有的会话)；消除：点击Rules-> Automatic Breakpoint ->Disabled
2. 第二种:在命令行中输入命令: bpuafter www.baidu.com (这种方法只会中断[www.baidu.com](http://www.baidu.com))；消除：在命令行中输入命令bpuafter

### Fiddler中创建AutoResponder规则

Fiddler 的AutoResponder tab允许你**从本地返回文件**，而不用将http request 发送到服务器上。

**看个实例：**

1. 打开博客园首页，把博客园的logo图片保存到本地，并且对图片做些修改。
2. 打开Fiddler 找到logo图片的会话， http://static.cnblogs.com/images/logo\_2012\_lantern\_festival.gif，把这个会话拖到AutoResponer Tab下。
3. 选择Enable automatic reaponses 和Unmatched requests passthrough
4. 在下面的Rule Editor 下面选择 Find a file... 选择本地保存的图片. 最后点击Save 保存下。
5. 再用IE博客园首页， 你会看到首页的图片用的是本地的。

### 过滤

Fiddler中有过滤的功能，右键Filters tab中，里面有很多选项。右键标签可移除

### Fiddler中会话比较功能

选中2个会话，右键然后点击Compare，就可以用WinDiff来比较两个会话的不同了（需要安装windiff）

### Fiddler中提供的编码小工具

点击Fiddler 工具栏上的TextWizard, 这个工具可以Encode和Decode string

### Fiddler中查询、保存会话

* 用快捷键Ctrl+F打开Find Sessions的对话框，输入关键字查询你要的会话；选择你想**保存的会话**，然后点击File->Save->Selected Sessions
* 默认的时候Fiddler是不能嗅探到localhost的网站。不过只要在localhost后面加个点号“.”，Fiddler就能嗅探到。

### 高级脚本

[参考官网文档](http://docs.telerik.com/fiddler/configure-fiddler/tasks/configurefiddler)

首先先安装SyntaxView插件，Inspectors tab->Get SyntaxView tab->Download and Install SyntaxView Now... 安装成功后Fiddler 就会多了一个Fiddler Script tab。

# Python编程中的注意事项

## 函数默认参数使用可变对象：

def use\_mutable\_default\_param(idx=0, ids=[]):  
 ids.append(idx)  
 print(idx)  
 print(ids)  
  
use\_mutable\_default\_param(idx=1)  
use\_mutable\_default\_param(idx=2)

append这类方法会直接修改对象，所以下次调用此函数时，其绑定的**默认参数**已经不再是空list了，调用两次后，列表ids的值变为[1,2]

**正确解法：**

def donot\_use\_mutable\_default\_param(idx=0, ids=None):  
 if ids is None:  
 ids = []  
 ids.append(idx)  
 print(idx)  
 print(ids)

## try…except

应该尽量少的使用try…except，这样可以在开发期尽早的发现问题。即使要使用try…except，也应该尽可能的指定出要**捕获的具体异常**，并在except语句中将**异常信息记入log**，或者处理完之后，再直接**raise出来**。

## Dict冗余代码

**冗余版：**

d = {}  
datas = [1, 2, 3, 4, 2, 3, 4, 1, 5]  
for k in datas:  
 if k not in d:  
 d[k] = 0   
 d[k] += 1

**精简版：(使用collections.defaultdict数据结构)**

From collection import defaultdict

default\_d = defaultdict(lambda: 0)  
datas = [1, 2, 3, 4, 2, 3, 4, 1, 5]  
for k in datas:  
 default\_d[k] += 1

**另外：**

d.setdefault(list, []).append(x)

**另外：**

可使用enumerate()函数同时处理一个序列的**索引和值**的访问。for i , k in enumerate(list): enumerate其实还有个第二参数，表示序号从几开始。如果想要序号从1开始数起，可以使用**enumerate(list, 1)**。

## 使用for…else

search\_list = ['Jone', 'Aric', 'Luise', 'Frank', 'Wey']  
for s in search\_list: #处理C开头的内容  
 if s.startswith('C'):  
 *# do something when found*  
 print('Found')  
 break  
else: #处理剩余的内容  
 *# do something when not found*  
 print('Not found')

## 过度使用tuple unpacking（多变量赋值）

Human=(‘Lily’,23,180)

Name,age,height=human

如果human改变，对其赋值会有影响。

## 尽可能少的使用import \*

这样会污染当前的命名空间，对后续的debug不利。应具体指明要使用的函数

## 对 文件的操作，使用with open（） as f 形式

这样可以避免文件操作异常

# Python积累

## 协程

* Coroutine，又称作协程。从字面上来理解，即协同运行的例程，它是比是线程（thread）更细量级的用户态线程，特点是允许用户的主动调用和主动退出，挂起当前的例程然后返回值或去执行其他任务，接着返回原来停下的点继续执行（通过调用yield方法）。操作系统的Interrupts和Traps机制为这种实现提供了可能性。
* 与return不同，**yield**在函数中返回值时会保存函数的状态，使下一次调用函数时会从上一次的状态继续执行，即从yield的下一条语句开始执行。
* 将一个函数改为generator，如下就可以通过for循环遍历函数值：

def fib(max):

n,a,b=0,0,1

while n<max:

**yield b** #print b

a,b=b,a+b

n+=1

for i in fib(5):

print i,

* **generator与coroutine的区别：**

generator总是生成值，一般是迭代的序列

coroutine关注的是消耗值，是数据(data)的消费者

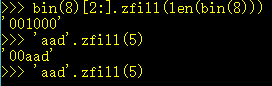
coroutine不会与迭代操作关联，而generator会

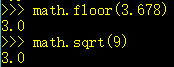
coroutine强调协同控制程序流，generator强调保存状态和产生数据

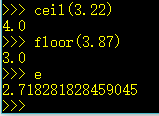
**相似的**是，它们都是不用return来实现重复调用的函数/对象，都用到了yield(中断/恢复)的方式来实现。

## 小函数

zfill(int)在字符前补全0



**对数：**

http://b.hiphotos.baidu.com/baike/s%3D162/sign=c26ae4e5cfea15ce45eee40f84013a25/b8389b504fc2d56244645a8de11190ef76c66c66.jpg

## Python内置高阶函数：

1. Map：map函数有两个参数，一个是函数，另一个是列表，返回值为对传入的列表每一个元素执行传入的函数操作得到的列表。map(func , list)

>>> map(lambda x:x\*2 ,[ri(1,99) for i in range(9)])

[94, 4, 124, 176, 6, 14, 12, 8, 16]

>>> map(lambda x,y:x+y, [1,2,3],[4,5,6])

[5, 7, 9]

>>> map(lambda x,y:(x+y,x-y),[4,3,2],[1,2,3])

[(5, 3), (5, 1), (5, -1)]

>>> map(None,[3,2,1],['q','w','e'])

[(3, 'q'), (2, 'w'), (1, 'e')]



1. Reduce: reduce函数也有两个参数，一个是函数，另一个是列表，返回值为对list的每一个元素反复调用函数f，得到最终结果.



1. Filter: filter函数接受函数参数f和列表参数lst，f对lst元素进行判断，返回lst的元素中调用f函数结果为true的元素组成的列表（将不满足f函数条件的元素过滤掉）；接受一个bool函数。



1. Sorted: sorted函数接受一个列表lst和一个函数参数f，f为自定义的比较lst元素大小的函数，返回值为lst中元素按f函数排列的列表。sorted(iterable[,cmp,[,key[,reverse=True]]])—>Return a new sorted list from the items in iterable.

>>> lis.sort(key=sum,reverse=True)

>>> lis

[(4, 3, 7, 8, 9, 2), (6, 5, 8, 3), (4, 3, 2), (1, 2)]

>>> lis.sort(key=lambda x:x[-1]\*5,reverse=True)

>>> lis

[(6, 5, 8, 3), (4, 3, 7, 8, 9, 2), (4, 3, 2), (1, 2)]

>>> lis=[(-1,5,3),(-5,2,1),(1,2,1,5,4,2)]

>>> lis.sort(cmp=lambda x,y:1 if abs(x)>abs(y) else -1,key=lambda x:x[0])

>>> lis

[(1, 2, 1, 5, 4, 2), (-1, 5, 3), (-5, 2, 1)]

* 查看模块内容：dir() ,\_\_file\_\_ ,\_\_doc\_\_ ,\_\_all\_\_
* python的子进程模块subprocess()可以操作执行电脑的其他程序，执行命令，call方法会等待子进程执行完毕，Popen方法需要加p.wait() Popen可以传入sdtin, sdtout参数来获取执行命令后的结果，注意shell=True ,sdtout=subprocess.PIPE p.sdtout是一个文件类型，可以读取p.sdtout.readline()用作后续的处理。
* 快速计算文件大小：import os; os.stat(file).st\_size
* File.seek(0)可将读取文件的游标返回值起始位置。
* 除法：divmod()
* 四舍五入：round()

# Python格式输出

[详解点击](http://blog.csdn.net/wchoclate/article/details/42297173)

%s ,%d ,%f ,%e ,%g ,%x ,%o

1.‘%g’根据实际情况输出，默认情况下最多保留**6位有效数字**。

2.‘%g’在保证六位有效数字的前提下，使用小数方式，否则使用科学计数法。

3. %g、%G输出一样，对应e、E，也可自定义精度

>>> '%g'%0.000001234

'1.234e-06'

>>> '%g'%12345678

'1.23457e+07'

>>> '%G'%12345678

'1.23457E+07'

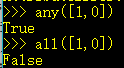
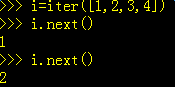
>>> '%G'%0.000001234

'1.234E-06'

# Python的条件与循环

1. python中由于使用缩进强制使代码对齐，不会出现像C语言那样的else悬挂问题（多个if语句中，无法判断else属于哪一个if语句，C语言中将else归属于最近的if语句中）。
2. 适当情况下可以用字典替代多个if-elif语句：default=’invalid choice…try again’ action=msg.get(user.cmd, default)
3. 条件表达式（三元操作符）：x if c else y



1. 许多通信服务器的客户端/服务器系统使用**无线循环**来工作（一直等待客户端的连接）。
2. For循环，迭代序列的三种基本方法：1.通过序列项迭代，2.通过序列索引迭代（range(len(list))），3.项和索引迭代（enumerate函数）
3. 与序列相关的内建函数：sorted([]) reversed([]) enumerate([]) zip([],[])
4. 当遇到continue语句时，程序终止当前循环，并忽略剩余的语句，然后回到循环的顶端，开始下一次循环。
5. Pass语句用于代码结构的开发与调试或者异常处理中（不想对非致命的错误采取任何措施可用pass语句）
6.  any() all() 内建函数，接收可迭代对象，判断迭代对象中是否全为真。
7. 迭代器iter()函数：，可对**序列**、**字典**（iterkeys、itervalues、iteritems）、**文件**（可直接for i in file: 按行读取，文件对象生成的迭代器会自动调用readline()方法）进行迭代。

## 列表解析式

**条件过滤：**





**生成矩阵（3\*5）**



**计算文件单词个数：**



## 生成器表达式

* 与列表解析式不同，它并不真正创建数字列表，而是返回一个生成器（可迭代对象），可节省内存,这个生成器在计算出一个条目后，把这个条目产生出来（**yield**）
* 如计算文件中所有单词长度的总和：使用**列表解析式**，sum([llen(word) for line in file for word in line.split()]) ;使用**生成器表达式**，sum([llen(word) for line in file for word in line.split()]).
* **寻找某文件中最长的行**：return max(len(x.strip()) for x in open(‘log.txt’,’r’))

# Python文件输入输出

* 文件支持的所有方法：

['close', 'closed', 'encoding', 'errors', 'fileno', 'flush', 'isatty', 'mode', 'name', 'newlines', '

next', 'read', 'readinto', 'readline', 'readlines', 'seek', 'softspace', 'tell', 'truncate', 'write'

, 'writelines', 'xreadlines']

1. 可使用内建函数open() file()打开一个文件，返回文件对象，打开模式有：r-只读，w-写，a-追加，b-二进制（rb、ab、wb），+（r+、w+、a+）-可读可写；文件名可以是相对路径或绝对路径。
2. 主要文件方法四类：**输入**：read()-直接读取，默认读取至文件末尾；readline()-读取整行；readlines()-读取所有行作为字符串列表返回；输出：write()-与read、readline相反；writelines()-接收一个字符串列表作为参数，写入文件；
3. 在使用read、readline方法读取文件时，python不会删除行结束符，可通过line.strip()完成；同样写入文件时write、writelines也不会自动加入行结束符，应手动添加\r\n(\n)。
4. 文件内移动：seek()方法默认值为0代表文件开头，1代表从当前位置算起，2代表从文件末尾算起；text()方法告诉当前文件指针在文件中的位置。

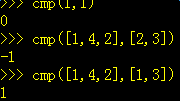
## 文件属性、系统模块

* 文件属性：file.mode ,file.name ,file.softspace ,file.closed ,file.newlines ,file.encoding
* 标准文件：只要程序一运行就可以访问3个标准文件，stdin、stdout、stderr，可通过sys模块访问这些文件的句柄，sys.stdin
* 命令行参数：sys.argv；如果程序需要用到一些简单的命令行参数，可以使用getopt模块，若需要提供复杂的选项可以使用optparser模块。
* 文件系统：主要的两个模块os、os.path
* Os模块属性：sep ,linesep ,curdir ,pardir ,pathsep
* Os模块：文件处理，mkfifo() ,remove() ,rename() ,stat()；函数symlink() ,utime() ,tmpfile() ,walk() ；目录/文件夹，chdir() ,chroot() ,listdir() ,getcwd() ,mkdir() ,rmdir() ；访问/权限，access() ,chmod() ,chown() ,umask() ；文件描述符操作，open() ,read()/write() ,dup() ；设备号，makedev() ,major()/minor()
* Os.path模块：分隔，basename() ,dirname() ,join() ,split() ,splitdrive() ,splitext() ；信息，getatime() ,getctime() ,getmtime() ,getsize() 查询，exists() ,isabs() ,isdir() ,isfile() ,islink() ,ismount() ,samefile()
* 永久存储模块：marshal、pickle模块的区别在于marshal只能处理简单的python对象（数字、序列、映射以及代码对象），而pickle可以处理递归对象、被不同地方多次引用的对象、用户自定义的类和实例。（两个主要函数：dump() ,load()）；shelve模块，提供了python对象的序列化和存储转换，以及类似字典和文件的对象，可以完成python对象的永久性存储。
* 与文件相关的其他模块：base64—提供二进制字符串与文本字符串之间的编解码；shutil—提供高级文件访问功能；zipfile—用于读取zip归档文件的工具。

# Python的sorted()函数

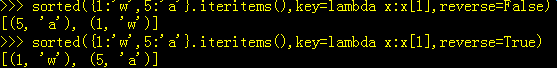
sorted(iterable[,cmp,[,key[,reverse=True]]])—>Return a new sorted list from the items in iterable.

* cmp指定一个定制的比较函数，这个函数接收两个参数（iterable的元素），默认值为None。



* key指定一个接收一个参数的函数，这个函数用于从每个元素中提取一个用于比较的关键字。默认值为None。（如：lambda x:x[1]）
* reverse是一个布尔值。如果设置为True，列表元素将被倒序排列。通常来说，key和reverse比一个等价的cmp函数处理速度要快。这是因为对于每个列表元素，cmp都会被调用多次，而key和reverse只被调用一次。

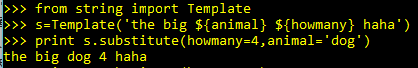
**按字典的value大小排序：**



# python字符串：

序列的一些通用方法：max() min() len() cmp() zip() enumerate()

字符串模板Template



字符串函数：raw\_input() str() Unicode() isinstance() chr() unichr() ord()

**字符串类型内建函数：string.fun()**

**Capitalize() count() decode() encode() endswith() find() index() isalnum() isalpha() isdigit() isspace() join() strip() just() lower() upper() partition() replace() split() splitlines() swapcase() startswith() zfill()**





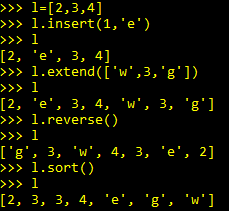


**列表类型内建方法：list.fun()**

append() count() remove() extend() index() insert() pop() reverse() sort()

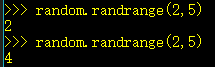
那些可以改变对象值的可变对象的方法是没有返回值的。方法在列表中原地执行操作。

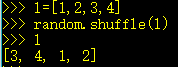
而不可变对象的方法是必须有返回值的。

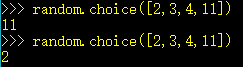


# Python random模块

['choice', 'randint', 'random', 'randrange', 'sample', 'seed', 'shuffle', 'uniform']



# Python网络编程

## Socket

* Socket是应用层与TCP/IP协议族通信的中间软件抽象层，它是一组接口。在设计模式中，Socket其实就是一个门面模式，它把复杂的TCP/IP协议族隐藏在Socket接口后面。

1. 有两种：Unix套接字，其家族名为AF\_UNIX（AF表示地址家族）或者AF\_LOCAL，python中使用AF\_UNIX；另一种是基于网络的AF\_INET是使用最广泛的一个。
2. Python中只支持AF\_UNIX、AF\_NETLINK、AF\_INET（用于网络编程）
3. 一个网络地址由网络通信所必须的主机和端口号组成。
4. 合法端口号范围0~65535，其中小于1024的端口号为系统保留端口。
5. 套接字的两种类型：面向连接，提供顺序、可靠、不重复的数据传输，实现这中连接的主要是TCP，在创建TCP套接字时指定套接字类型为SOCK\_STREAM； 无连接类型，实现这种连接的主要是UDP，无顺序、不可靠、可重复的数据传输，但可提供更好的性能，指定套接字类型为SOCK\_DGRAM。
6. 语法：socket(socket\_family ,socket\_type ,protocol=0) 例如:tcpSock=socket.socket(socket.AF\_INET,socket.SOCK\_STREAM) ;可通过from socket import \* 来简化代码：tcpSock=socket.socket(AF\_INET, SOCK\_STREAM)
7. 方法：服务器端函数：bind() listen() accept() ；客户端函数：connect() connect\_ex()//出错时返回出错码，而不是抛异常； 公共用途函数：recv() send() sendall() recvfrom() sendto()
8. 通过创建一个新的线程或进程来完成与客户端的通信是一种常用的手段，SocketServer模块是一个基于socket模块的高级别的套接字通讯模块，支持新的线程或进程来处理客户端请求。
9. 在服务器要退出的时候，要记得调用close()函数。

## 常用模块

SocketServer模块：在创建服务器方面提供了高层次的功能，甚至提供了把线程或进程集成到服务器中的功能。

标准库中唯一的异步开发支持库—async；也可选择相对标准库更现代、更强大的第三方库---Twisted

## 网络客户端编程

数据交换的参与者是一个服务提供者和一个服务的使用者。

三个网际协议：FTP NNTP POP3

### 文件传输

* 常见的文件传输网际协议：HTTP（web获取网页，文件下载）、 FTP（文件远程传输需用户登录或匿名登录）、 scp/rsync（Unix远程传输文件，需登录）
* FTP一般的客户端超过15分钟不活动之后，连接就会被关闭。客户端服务器使用两个套接字来通讯：一个是控制和命令端口21，另一个是数据端口（有时是20号端口），FTP有两种模式：主动和被动。
* Python编写ftp客户端使用ftplib模块，实例化ftplib.FTP类对象。
* 一般的FTP通讯中，要使用到的指令有：login() cwd() dir() pwd() stor\*() retr\*()

### Usnet新闻组

Usenet新闻组服务内容取决于服务提供者，很多都是可供公众访问的。

网络新闻传输协议（NNTP）

### 电子邮件

* 只有邮件头式必要的
* 组件：消息传输代理MTA（message transport agent）、MTA之间通讯使用的协议是消息传输系统MTS
* 简单邮件传输协议：SMTP（simple mail transfer protocol）是MTS的一种。
* 方法：sendmail(from ,to ,msg[,mopts ,ropts])、quit() //退出 、login(user,passwd) //在SMTP服务器有SMTP-AUTH时使用。
* 邮件用户代理MUA（mail user agent）
* 邮局协议：POP让用户的工作站可以访问邮箱服务器里的邮件。邮件要能从工作站通过简单传输协议SMTP发送到邮件服务器。
* 交互式邮件访问协议IMAP ，比POP更为复杂，Microsoft Exchange使用IMAP作为其下载机制。
* POP3类方法：user(login) pass\_(passwd) stat() list() retr() quit()

# Python模块

1. 模块是用来组织python代码的方法，而包则是用来组织模块的。
2. 一个模块可以被看作一个文件，模块的文件名就是模块的名字加扩展名.py。与其他可以导入类的语言不同，python中导入的是模块或模块属性。
3. 出现ImportError:No module named…原因模块不在搜索路径中，默认搜索路径是在编译或是安装时指定的。它可以在一个或两个地方修改。可以导入sys import sys; sys.path; sys.path.append(‘xxx’)在path中添加你要导入的模块的路径即可。
4. 名称空间是名称（标识符）到对象的映射；名称空间分：局部名称空间、全局名称空间、内建名称空间； Python解释器首先加载内建名称空间，它由**\_\_builtins\_\_**模块中的名字构成，该模块包含内建函数、异常以及其他属性。
5. Import导入，如果在一个模块的顶层导入，那它的作用域就是全局的，如果在函数中导入，那它的作用域就是局部的。不提倡 from module import \*
6. from module import xxx as ddd
7. 加载（load）只在第一次导入（import）时发生。多次导入不会被重复加载。
8. 显示的导入新特性：from \_\_future\_\_import new\_feature
9. 警告异常类的集合，warning直接从exception继承，作为所有警告的基类，包括：UserWarning、DeprecationWarning、SyntaxWarning、RuntimeWarning
10. Import语句调用\_\_import\_\_()函数完成导入工作，因此可以sys=\_\_import\_\_(‘sys’)导入sys模块。
11. Locals() globals() 返回调用者局部、全局名称空间的**字典，不传入参数。**
12. Reload(module) 重新加载
13. \_\_init\_\_.py初始化模块，from-import语句导入子包时需要用到它。
14. \_\_init\_\_.py文件可以为空，在文件中加入\_\_all\_\_变量，它包含在执行from xxx import \*这样的语句时应该导入的模块的名字，由一个模块名字符串列表组成。

# Python logging模块

默认情况下，logging将日志打印到屏幕，级别为WARNING。

日志级别为：CRITICAL>ERROR>WRANING>INFO>DEBUG>NOTSET，也可自定义级别。



## 配置日志输出格式和方式：logging.basicConfig()

logging.basicConfig(level=logging.DEBUG,  
                format='%(asctime)s %(filename)s[line:%(lineno)d] %(l evelname)s %(message)s',  
                datefmt='%a, %d %b %Y %H:%M:%S',  
                filename='myapp.log',  
                filemode='w')

format: 指定输出的格式和内容，format可以输出很多有用信息，如上例所示:

%(levelno)s: 打印日志级别的数值

%(levelname)s: 打印日志级别名称

%(pathname)s: 打印当前执行程序的路径，其实就是sys.argv[0]

%(filename)s: 打印当前执行程序名

%(funcName)s: 打印日志的当前函数

%(lineno)d: 打印日志的当前行号

%(asctime)s: 打印日志的时间

%(thread)d: 打印线程ID

%(threadName)s: 打印线程名称

%(process)d: 打印进程ID

%(message)s: 打印日志信息

datefmt: 指定时间格式，同time.strftime()

level: 设置日志级别，默认为logging.WARNING

stream: 指定将日志的输出流，可以指定输出到sys.stderr,sys.stdout或者文件，默认输出到sys.stderr，当stream和filename同时指定时，stream被忽略.

## 将日志同时输出到屏幕和文件中：

1. 配置日志输出格式：logging.basicConfig()
2. 定义一个StreamHandler

#**定义一个StreamHandler**，将INFO级别或更高的日志信息打印到标准错误，并将其添加到当前的日志处理对象#  
console = logging.StreamHandler()  
console.setLevel(logging.INFO)  
formatter = logging.Formatter('%(name)-12s: %(levelname)-8s %(message)s')  
console.setFormatter(formatter)  
logging.getLogger('').addHandler(console)

## Logging日志回滚

import logging  
from logging.handlers import RotatingFileHandler  
  
#############################################################################定义一个RotatingFileHandler，最多备份5个日志文件，每个日志文件最大10M  
Rthandler = RotatingFileHandler('myapp.log', maxBytes=10\*1024\*1024,backupCount=5)  
Rthandler.setLevel(logging.INFO)  
formatter = logging.Formatter('%(name)-12s: %(levelname)-8s %(message)s')  
Rthandler.setFormatter(formatter)  
logging.getLogger('').addHandler(Rthandler)  
############################################################################

logging有一个日志处理的主对象，其它处理方式都是通过addHandler添加进去的。  
**logging的几种handle方式如下：**

logging.StreamHandler: 日志输出到流，可以是sys.stderr、sys.stdout或者文件  
logging.FileHandler: 日志输出到文件

日志回滚方式，实际使用时用RotatingFileHandler和TimedRotatingFileHandler  
logging.handlers.BaseRotatingHandler  
logging.handlers.RotatingFileHandler  
logging.handlers.TimedRotatingFileHandler

logging.handlers.SocketHandler: 远程输出日志到TCP/IP sockets  
logging.handlers.DatagramHandler:  远程输出日志到UDP sockets  
logging.handlers.SMTPHandler:  远程输出日志到邮件地址  
logging.handlers.SysLogHandler: 日志输出到syslog  
logging.handlers.NTEventLogHandler: 远程输出日志到Windows NT/2000/XP的事件日志  
logging.handlers.MemoryHandler: 日志输出到内存中的制定buffer  
logging.handlers.HTTPHandler: 通过"GET"或"POST"远程输出到HTTP服务器。

由于StreamHandler和FileHandler是常用的日志处理方式，所以直接包含在logging模块中，而其他方式则包含在logging.handlers模块中。

# Python多线程

## 线程和进程

* 进程是程序的一次执行，每个进程都有自己的地址空间、内存、数据栈及记录其运行轨迹的辅助数据。进程之间只能使用进程间通信IPC，不能直接共享信息。
* 所有的线程运行在同一个进程中，共享相同的运行环境。它有开始、顺序执行、结束三部分，有一个自己的指令指针，记录自己运行到什么地方。各个线程之间共享同一片数据，一般是并发执行的，容易出现竞争。
* 计算机的核心是CPU，就像一座工厂时刻在运行；进程好比工厂的车间，代表CPU处理的单个任务；线程好比车间里的工人，一个进程可以包括多个线程。防止多个线程同时读写某一块内存区域（比如车间的厕所，一次只能容一个人）需要加互斥锁做保护。

## Python thread模块

['allocate', 'allocate\_lock', 'error', 'exit', 'exit\_thread', 'get\_ident', 'interrupt\_main', 'stack\_size', 'start\_new', 'start\_new\_thread']

Thread模块不支持守护线程，当主线程结束后，所以子线程都会强制终止掉，没有警告也没有正常的清理工作。因此推荐使用threding模块。

1. python 是一般解释性的语言，也就是我们的代码要经过解释器解释后才能运行，而一个python的进程只有一个python解释器。

## Python threading模块

Threading模块支持守护线程：守护线程是一个等待客户请求的服务器，如果没有客户请求，就会一直等着，如果设定一个线程为守护线程，就表示这个线程是不重要的，在进程退出时不用等待这个线程退出。

**Threading模块中的对象：**

['BoundedSemaphore', 'Condition', 'Event', 'Lock', 'RLock', 'Semaphore', 'Thread', 'ThreadError', 'Timer']

**threading模块中的方法：**

['active\_count', 'currentThread', 'current\_thread', 'enumerate', 'local', 'setprofile', 'settrace', 'stack\_size', 'warnings']

**Thread类中的方法：**

['daemon', 'getName', 'ident', 'isAlive', 'isDaemon', 'is\_alive', **'join'**, 'name', 'run', 'setDaemon', 'setName', 'start']

join()—等待线程结束，可以完全不用调用，一旦线程启动就会一直运行，直到线程的函数结束，退出为止。如果主线程除了等待线程结束外，还有其他事情做就不用调用join()，只有在等待线程结束的时候才要调用。

## python 的 GIL

1. GIL也就是全局解释器锁。这个锁保证了同一时刻只能有一个线程运行。
2. 在多线程环境中，Python 虚拟机按以下方式执行：

　　1.设置 GIL

　　2.切换到一个线程去运行

　　3.运行：a. 指定数量的字节码指令，或者b. 线程主动让出控制（可以调用 time.sleep(0)）

　　4.把线程设置为睡眠状态

　　5.解锁 GIL

　　6.再次重复以上所有步骤

1. 在进行文件读写操作，或者爬虫在等待网页下载的时候，就比较适合使用多线程了。I/O 密集型的 Python 程序比计算密集型的程序更能充分利用多线程环境的好处。

## 方法函数：

1. **allocate\_lock()** -> lock object创建一个lock object。
2. **exit()**和 raise SystemExit 同义，将退出一个线程，除非进行了异常捕获。线程退出的方法：thread.exit() sys.exit() 或者抛出SystemExit异常。
3. **get\_ident()**->integer返回一个非零的整数,惟一地标识当前线程和其他线程同时存在。这个可以用来识别每个线程资源。
4. **interrupt\_main()**在主线程中抛出 KeyboardInterrupt 异常。子线程可以使用这个方法来中断主线程。
5. **stack\_size**([size]) -> size返回创建新线程时使用的线程堆栈的大小。可以使用 size 参数来指定堆栈大小(以字节为单位)，然后再创建相应的线程。而且必须是0(使用平台或配置默认)或一个正整数，其值至少32768(32 k)。如果不支持更改线程堆栈大小，则触发 ThreadError 异常。如果指定的大小是无效的，将触发 ValueError，同时不修改其大小。32 k字节是目前支持最小的堆栈大小值，其目的是为了保证有足够的堆栈空间翻译本身。注意,一些平台可能有特殊的栈大小限制，例如要求最小堆大于32 kb或要求是系统内存页大小的倍数，详情查看该系统平台是说明文档。通常每页 4 kb 是最常见的，在没有说明文档的时候可以尝试使用 4kb 的倍数。
6. **start\_new\_thread(function, args[, kwargs])** :开始一个新的线程,并返回其标识符（通常是个数字）。线程将调用相应的函数 function 并将 args 元祖中的元素作为位置参数，字典 kwargs 作为关键字参数传入。当函数 return 的时候，线程结束，但是其返还值是忽略的。当函数内部触发未处理的异常时，线程将结束，同时打印堆栈跟踪信息，当然触发 SystemExit 就无痕退出了。要求一定要有前两个参数。所以，就算我们想要运行的函数不要参数，我们也要传一个空的元组。Exmple: thread.start\_new\_thread(loop1, ())

## LockType

1. 主线程不会等待子线程的执行，而主线程执行完毕后，进程退出，子线程也就消失了。而且通常情况下我们要执行的函数需要的时间是无法确定的，而且根据 cpu 的性能不同，所需要的时间也就不同。因此，需要使用线程锁，也就是使用 **allocate\_lock()**方法。这个方法返回一个 **LockType** 类型锁对象。
2. **acquire([waite])->bool** 尝试获取锁对象。

　　当没有参数的时候，会加锁，等待另一个线程解锁，如果已经锁定了，则返回True。

　　当给定参数时，只有当参数的布尔值为真才会加锁，同时返回一个数值反应是否成功加锁。

加锁后若线程运行该代码块时，不进行解锁操作，将阻塞该线程。阻塞是不可中断的。

1. **release()** 释放锁，让另一个被阻塞的线程获得线程锁。锁对象必须在锁定状态，但是它不必被同一个线程解锁。
2. **locked()->bool** 判断线程是否在锁定状态。

loops = [4, 2]

def loop(nloop, nsec, lock):

print '子线程', nloop, '开始:', ctime()

sleep(nsec)

print '子线程', nloop, '结束:', ctime()

lock.release() # 解锁

def main():

print '主线程开始:', ctime()

locks = [] # 一个用来存放锁对象的列表

nloops = range(len(loops))

for i in nloops:

lock = thread.allocate\_lock() # 创建一个进程锁对象

lock.acquire() # 尝试获得这个锁对象，并加上锁

locks.append(lock) # 将获得的锁对象放到一个列表中

for i in nloops:

thread.start\_new\_thread(loop, (i, loops[i], locks[i])) # 每次循环启动一个新线程

for i in nloops:

while locks[i].locked(): # 查看是否所有线程锁都以释放，否则一直循环，防止主线程停止导致子线程退出

pass

print '主线程结束:', ctime()

if \_\_name\_\_ == '\_\_main\_\_':

main()

注意：

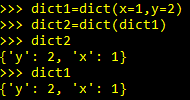
我们说过acquire([wait])后，如果不用release()解锁的话，会阻塞当前线程，但是这个例子中，我们为每个函数都开启了一个线程，也就是说，如果我在函数内不释放锁，并不会影响其他线程的执行，只会影响 locked() 函数的返回值。但是，如果我们限制了线程的数量，比如说我的 cpu 是双核4线程的，我希望只开4个线程，在 threading 模块中能够做到。如果我只开了4个线程，但是我的功能函数有很多，我希望他们排队执行，此时若有一个函数锁定了以后老是不释放，那么这个函数所在的线程就会被阻塞掉，其他函数就不能继续在这个线程运行了，只能等待。

# Python字典：

**zip():**



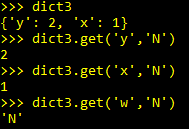
字典的浅拷贝：这种拷贝方法比copy速度慢，推荐使用copy()方法。dict2=dict1.copy()

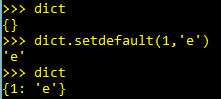
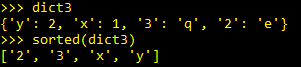
## 字典方法：dict.func()

Clear() fromkeys() get() items() keys() values() update() pop() setdefault()//检查字典中是否含有某键。

Itertems() iterkeys() itervalues()//返回迭代对象，用于节省内存（类似xrange()函数），用做循环迭代。

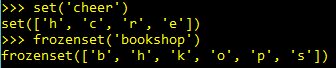
 



# Python集合

Set() frozenset()



## 集合方法：set.func()

Add() update() remove()

集合的比较：> < - == != in not in >= <= & | ^//异或

共有方法：issubset() issuperset() union() intersection() difference() symmetric\_difference() copy()

可变集合的方法： update() intersection\_update() difference\_update() symmetric\_difference\_update() add() remove() discard() pop() clear()

# Python函数式编程

逻辑复用

## 参数

关键字参数、默认参数、参数组

函数调用完整的语法：func(positional\_args ,keyword\_args ,\*tuple\_grp\_args ,\*\*dict\_grp\_args)

## 属性

可用句点属性标识来直接增加函数属性，bar.version=1.0；类似于self.value=xxx

## 装饰器

接受函数对象参数，它定义了某个函数并在定义内的某处嵌入了对目标函数的调用或者至少一些引用。可以在装饰器中置入通用的代码来降低程序复杂度，如：引入日至；增加计时逻辑来监测性能；给函数加入事务的能力。@decorator func()对紧挨的的函数进行装饰。

## 传递函数

* 所有的对象都是通过引用来传递的，函数也一样。def foo(): 可以进行传递bar=foo之后，可以通过bar()调用函数，与foo()一样的效果。明确**函数对象的引用**（foo）和**函数对象的调用**（foo()）的区别。
* 也可以把函数作为参数传入其他函数来进行调用。

def convert(func,seq):

return [func(eachNum) for eachNum in seq]

myseq=(123,12.22,-2.2e3,1111111111111L)

print convert(int ,myseq)

print convert(long, myseq)

print convert(float ,myseq)

## formal arguments

* **位置参数**：传入的参数个数、顺序必须与声明的一致，否则报错，无法执行。
* **默认参数**：python默认参数的语法规定，所有的位置参数必须出现在任何一个默认参数之前。Default=xxx
* **可变长参数：**\*nkwargs ,\*\*kwargs分别表示用元组、字典接收位置参数、关键字参数。

## 偏函数

>>> from operator import add ,mul

>>> **from functools import partial**

>>> add1=partial(add,1)

>>> mul10=partial(mul,10)

>>> add1(3)

4

>>> mul10(3)

30

>>> baseTwo=partial(int,base=2)

>>> baseTwo('11010')

26

## Globa语句

声明全局变量：globa a ，在函数内声明全局变量。

## 生成器

* 生成器是一个带**yield语句**的函数，当生成器的next()方法被调用的时候，它会准确的从离开的地方继续。

from random import randint

def randGen(aList):

while len(aList)>0:

yield aList.pop(randint(0,len(aList)))

for item in randGen([‘vivo’,’oppo’,’apple’]):

print item

* 加强的生成器特性：可以使用send()方法传入值；close()方法关闭。

## 递归算法

**递归算法3个原则：**

1、 一个递归算法必须有一个基本状态，达到这个基本状态时递归函数就停止递归。

2、 递归必须改变其状态并向基本状态移动。

3、 递归函数必须调用自身（调用的参数值不一样）。

# Python面向对象编程

## 定义说明

1. 面向对象主要有两个主题：类和类的实例。类（相当于工厂制造玩具的机器）是对象的定义，而实例（相当于制造出来的玩具）是真正的实物，它存放了类中所定义的对象的具体信息。
2. 新式类必须继承至少一个父类，可以是单继承或多继承。经典类可以不显示写继承，默认继承object，推荐尽可能使用新式类。
3. 创建一个实例的过程称作实例化：MyFirstObject=MyNewObjectType()赋值给变量，即为实例分配的内存。类的属性是动态的，不需要在构造器或者其他任何地方为他们预先声明或者赋值。
4. 方法：所有的方法声明中都存在self参数，代表实例对象本身，当使用实例调用方法时，由解释器传递给方法，所以不需要自己传递self进来，因为它是自动传入的。相当与其他语言的this。Python中一般的方法会需要这个实例（self），而静态方法或类方法不会。
5. \_\_init\_\_()方法类似构造器，是一个特殊的方法，在创建一个新的对象时被调用。Python中主要是在实例被创建后，实例化调用返回这个实例之前，去执行某些特定的任务或者设置。
6. 创建子类：在\_\_init\_\_()方法中，需要显式的写出基类的init方法，在调用基类的构造方法时要显式的传入self参数，因为我们不是在该实例中而是在一个子类实例中调用那个方法。
7. 命名规范：类名使用大写字母区分单词，方法名使用小写字母+下划线区分。

## 类属性

决定类的属性：dir() ,\_\_dict\_\_ ,\_\_doc\_\_ ,\_\_module\_\_ ,\_\_file\_\_ ,\_\_name\_\_ ,\_\_bases\_\_ ,\_\_class\_\_

## 实例

1. \_\_init\_\_()构造器方法、\_\_new\_\_()构造器方法、\_\_del\_\_()解构器方法
2. 在调用\_\_del\_\_()方法时：首先调用父类的\_\_del\_\_()，
3. \_\_init\_\_()返回None，否则导致TypeError异常。
4. 类属性可以直接用类访问，也可以用实例访问。

## 类属性和实例属性

>>> class C(object):

... version=1.1

...

>>> c=C()

>>> c.version

1.1

>>> C.version

1.1

>>> c.version=1.0 实例属性赋值，类属性不变，类属性和实例属性是独立的，只是属性名相同，但实例可以访问类属性，不可以改变类属性。

>>> c.version

1.0

>>> C.version

1.1

>>> del c.version 删除实例属性后，访问的是类属性

>>> c.version

1.1

>>> C.version

1.1

>>> del C.version 删除类属性，类属性、实例属性都没有了

>>> C.version

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

AttributeError: type object 'C' has no attribute 'version'

>>> c.version=1.0 重新赋值实例属性，不会影响类属性

>>> C.version

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

AttributeError: type object 'C' has no attribute 'version'

>>> C.version=2.0 重新赋值类属性，二者独立存在

>>> c.version

1.0

>>> C.version

2.0

>>>

**若属性为可变值如列表、字典，则二者相同，实例属性可改变类属性的值。**

>>> class D(object):

... x={'w':1,'l':2}

...

>>> d=D()

>>> d.x['s']=3

>>> D.x

{'s': 3, 'l': 2, 'w': 1}

>>> d.x

{'s': 3, 'l': 2, 'w': 1}

**类属性（静态成员）的修改会影响到所有的实例，修改类属性需要类名，而不是实例名。**

## 调用非绑定方法

不经常用到，主要在派生一个子类，而且要覆盖父类的方法，这是需要调用那个父类中想要覆盖掉的构造方法。\_\_init\_\_()

## 静态方法和类方法

1. 使用staticmethod() , classmethod()内建函数。foo=classmethod(foo) 将普通方法foo转为类方法foo。
2. 使用函数修饰符：@staticmethod @classmethod

## 通过继承覆盖方法

class C(P):

def foo(self):

P.foo(self) #调用父类方法

print 'Hi, I am C-foo()'

class C(P):

def foo(self):

super(C,self).foo() #使用内建函数super调用父类方法

print 'Hi, I am C-foo()'

* 新式类的多重继承：查找与子类更亲近的父类。可通过新式类的属性\_\_mro\_\_查看其查找顺序。
* 经典类，查找父类使用深度优先算法，新式类使用MRO（方法解释顺序），在多重继承中会出现菱形效应。

## 内建函数

**issubclass(sub,sup)**

**isinstance(obj1,obj2)** //obj2是一个类，也可以是类型对象，用于obj1是否是obj2类型如isinstance(4 , int)，obj也可以是一个元组。

**Hasattr() getattr() setattr() delattr()**

>>> class myClass(object):

... def \_\_init\_\_(self):

... self.foo=100

... def move(self):

... print 'hello'

...

>>> myInst=myClass()

>>> hasattr(myInst,'move')

True

>>> hasattr(myInst,'foo')

True

>>> getattr(myInst,'foo')

100

>>> getattr(myInst,'move')

<bound method myClass.move of <\_\_main\_\_.myClass object at 0x00000000024D7198>>

>>> getattr(myInst,'go','oops!') #如果找不到则返回oops

'oops!'

>>> setattr(myInst,'bar','attr') #

>>> dir(myInst)

['\_\_class\_\_', '\_\_delattr\_\_', '\_\_dict\_\_', '\_\_doc\_\_', '\_\_format\_\_', '\_\_getattribute\_\_', '\_\_hash\_\_', '\_

\_init\_\_', '\_\_module\_\_', '\_\_new\_\_', '\_\_reduce\_\_', '\_\_reduce\_ex\_\_', '\_\_repr\_\_', '\_\_setattr\_\_', '\_\_size

of\_\_', '\_\_str\_\_', '\_\_subclasshook\_\_', '\_\_weakref\_\_', 'bar', 'foo', 'move']

>>> delattr(myInst,'bar')

>>> dir(myInst)

['\_\_class\_\_', '\_\_delattr\_\_', '\_\_dict\_\_', '\_\_doc\_\_', '\_\_format\_\_', '\_\_getattribute\_\_', '\_\_hash\_\_', '\_

\_init\_\_', '\_\_module\_\_', '\_\_new\_\_', '\_\_reduce\_\_', '\_\_reduce\_ex\_\_', '\_\_repr\_\_', '\_\_setattr\_\_', '\_\_size

of\_\_', '\_\_str\_\_', '\_\_subclasshook\_\_', '\_\_weakref\_\_', 'foo', 'move']

>>>

**dir() ,vars() vars()与dir()相似，只是给定的对象参数都必须有一个\_\_dict\_\_属性，返回一个字典。**

**super()**

## 包装授权

1. 当在实例或类字典中的完整搜索失败后，就调用getattr()来查找一个成功的匹配。对于包装后的对象的属性的访问是通过getattr()方法，授权给对象。实现\_\_getattr\_\_(self,attr)函数
2. 对于某些特殊行为不在类型的方法列表中（如列表的切片、索引取值），因为他们不是属性，可以通过get()方法获取到对象后对其进行操作：xxx.get()[3]

## 新式类特性

1. \_\_slot\_\_属性：字典\_\_dict\_\_会占据大量内存，如果你有少量的属性可以用\_\_slot\_\_代替字典，\_\_slot\_\_=(‘foo’ , ‘bar’)，这种特性的主要目的是节约内存，防止了用户动态的增加\_\_slot\_\_之外的实例属性。
2. \_\_getattribute\_\_()特殊方法，替代\_\_getattr\_\_()方法，可以一直调用，而不局限于不能找到的情况，若二者同时存在，默认优先调用\_\_getattribute\_\_()方法。
3. \_\_get\_\_(self,obj,typ=None) \_\_set\_\_(self,obj,val) \_\_delete\_\_(self,obj)

## 属性property

1. 一般情况下当我们使用点属性符号来处理一个实例属性时，其实是在修改这个实例的\_\_dict\_\_属性。
2. 可以用property代替实现那些特殊方法\_\_getattr\_\_() \_\_setattr\_\_() ,来处理属性（get set delete）尤其是在处理大量实例属性时更有效。
3. Property(fget=None ,fset=None ,fdel=None ,doc=None)

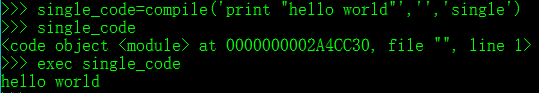
## 与类相关的模块

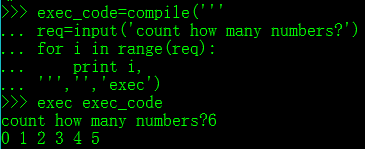
1. UserList 提供一个列表对象的封装类
2. UserDict 提供一个字典对象的封装类
3. UserString 提供一个字符串对象的封装类
4. types 定义所有python对象的类型在标准python解释器中的名字
5. operator 标准操作符的函数接口

# python执行环境

## 可调用对象（通过变量名后一对圆括号调用）

1. Python 有 4 种可调用对象： 函数， 方法， 类， 以及一些类的实例
2. Python有3种不同类型函数对象。第一种：内建函数（BIFs）使用c/c++写的，编译后写入python解释器，在\_\_builtins\_\_模块中。从内部机制来看， BIFs 和内建方法(BIMs)属于相同的类型。第二种：用户定义的函数（UDF）；第三种：lambda 匿名函数。
3. 调用类的结果便是创建了实例，即大家所知道的实例化。
4. python 给类提供了名为\_\_call\_\_的特别方法，该方法允许程序员创建可调用的对象（实例）。实现了\_\_call\_\_方法后，可以调用实例化对象。功能上类似\_\_init\_\_；只有定义类的时候实现了\_\_call\_\_方法，类的实例才能成为可调用的。
5. 代码对象可以作为函数或者方法调用的一部分来执行，也可用 exec 语句或内建函数eval()来执行。
6. 可执行对象和内建函数：callable() ,compile() ,eval() ,input() , exec 'max(1,8,5)' ,
7. **compile(string,file, type)**，第一参数代表了要编译的 python 代码。第二个字符串，虽然是必需的，但通常被置为空串。该参数代表了存放代码对象的文件的名字(字符串类型） compile 的通常用法是动态生成字符串形式的 Python 代码， 然后生成一个代码对象——代码显然没有存放在任何文件。最后的参数是个字符串，它用来表明代码对象的类型。有三个可能值：eval ，single ，exec ；eval\_code=compile(‘100+200’, ‘’ ,’eval’) eval(eval\_code)





* eval()对表达式求值，后者可以为字符串或内建函数 complie()创建的预编译代码对象。这是eval()第一个也是最重要的参数.......这便是你想要执行的对象。第二个和第三个参数，都为可选的，分别代表了全局和局部名字空间中的对象。如果给出这两个参数，globals 必须是个字典，locals可以是任意的映射对象，比如，一个实现了\_\_getitem\_\_()方法的对象。(在 2.4 之前，local 必须是一个字典）如果都没给出这两个参数，分别默认为 globals()和 locals()返回的对象，如果只传入了一个全局字典，那么该字典也作为 locals 传入。**eval(source[, globals[, locals]]) -> value**

exec 还可以接受有效的 python 文件对象。>>> f = open('xcount.py') # open the file

>>> exec f # execute the file

* 内建函数 input()是 eval()和 raw\_input()的组合，等价于 eval(raw\_input())

# Python time模块

## 方法：

time.time()//返回时间戳；

time.localtime()//返回事件元组，可以传入时间戳参数；time.strftime(‘%Y-%m-%d %H: %M: %S’)//格式化输出，可传入时间元组；

time.mktime()//返回时间戳，可以传入时间元组参数；

time.asctime() time.ctime()//返回时间字符串'Sat Jul 25 20:08:32 2015'；time.strptime(str,fmt)//传入时间字符串、输出格式，返回时间元组；

# Python requests模块

## 请求方法：get、post、put、delete

r=requests.get(url ,\*\*kwargs)

r=requests.post(url, data=None,json=None,\*\*kwargs)

获取response对象后可以查看，status\_code ,text ,content ,headers ,json() ,encoding ,raw, cookies还可以设置编码方式：r.encoding=‘utf-8'

post方法可以传入一个字典类型的data参数，类似网页表单形式，或者直接传入一个文件{‘file’: ‘open(‘content.xml’,’rb')'}

## 响应状态码：

status\_code 为了方便requests还附带了一个内置的状态码查询对象：r.status\_code==requests.codes.ok 返回True；如果发送错误请求，要抛出异常response.raise\_for\_status(),当没有异常发生是返回None

响应头是大小写不敏感的content-type和Content-Type一样。

发送cookies到服务器可以直接传入参数，cookies=dict(cookies\_are=‘working’); r=requests.get(url ,cookies=cookies)

## 重定向：

默认情况下除了head，requests会自动处理所有重定向，可以通过response.history追踪，也可以在请求方法中传入allow\_redirects=False禁止重定向。

## 超时：

timeout参数可以设置在规定时间内服务器没有应答就停止等待响应，timeout不对下载过程做限制。

## 异常：

网络问题会抛出ConnectionError异常；返回了不正确的状态码response.raise\_for\_status()会抛出HTTPError异常；请求超时抛出Timeout异常；请求超过了最大重定向次数会抛出TooManyRedirects异常；所有requests抛出的显式异常都继承自requests.exception.RequestException

## Session

Session可以跨请求保持某些参数。通过session的方法设定参数会被跨请求保持，如果是方法级别的参数不会被保持。比如：s.headers.update(’test’:’pass’); s.get(url, headers={’test1’:’user’}) 两个header参数会被一起发送。

可以使用with requests.session() as s: 确保请求退出后会话被关闭，即使发生异常也一样。

要获取服务器响应头部信息使用：response.headers      如果是要获取发送到服务器的请求头部使用：response.request.headers，如：r=requests.get(url,headers=headers)

准备的请求：PreparedRequest，当从API或会话调用中收到一个response对象时，request属性其实是使用了PreparedRequest，当需要在发送请求之前对body、header做一些处理时就要在Prepared对象中做一些改变：prepared=s.prepare\_request(req)对prepared对象prepared.body 、prepared.headers做改变。

from requests import Request, Session  
  
s = Session()  
req = Request('GET',  url,  
    data=data  
    headers=headers  
)  
  
prepped = s.prepare\_request(req)  
  
# do something with prepped.body  
# do something with prepped.headers  
  
resp = s.send(prepped,  
    stream=stream,  
    verify=verify,  
    proxies=proxies,  
    cert=cert,  
    timeout=timeout  
)

print(resp.status\_code)

 SSL证书验证：对于https可以在请求方法中添加verify=True参数，默认为True，也可以传递一个文件路径，多个文件的话写在一个元组中cert=(‘/path/server.crt’,’/path/key')

CA证书，Requests默认自带一套它信任的根证书，可以使用系统中的certifi包，建议经常更新certifi。

为了提高效率，使用requests的stream参数可以推迟下载响应体直到访问response.content属性。stream=True，可以使用response.iter\_content  response.iter\_lines方法来控制工作流；如果设置stream=True要记得关闭response.close，所以可以使用contextlib 模块的closing with closing(requests.get(url,stream=True)) as r:

只有所有响应体数据被读取完毕连接才会被释放为连接池，所以应该确保将stream设置为False或读取response对象的content属性。

requests支持流式上传，允许发送大的数据流或文件，with open(‘file’,’rb') as f: requests.post(url ,data=f)

支持块编码请求：就是请求参数data传入一个生成器。对于块编码请求，对response的处理应用response.iter\_content()对其数据进行迭代。

事件挂钩hook参数hooks=dict(hook\_name=callback\_function)

## 自定义身份验证：

auth参数，requests在requests.auth中提供了两种常见的身份验证方案：HTTPBasicAuth和HTTPDigestAuth。from requests.auth import AuthBase

## 代理：

可以为请求方法传入proxies参数（参数为一个字典形式）

socks代理，安装依赖库使用和HTTP代理一样

requests请求参数有：url  params  data  json  headers  cookies files  auth  timeout  allow\_redirects  proxies verify  stream  cert

>>> **import** requests  
>>> req = requests.request**(**'GET'**,** '<http://httpbin.org/get>'**)**

<Response [200]>

get post head put delete patch

会话：session

# Http相关

1 web服务器资源：MIME类型是一种文本标记，表示一种主要的对象类型和一个特定的子类型，中间由一条斜杠来分隔。image/jpeg

2 URI 统一资源标识符，有两种形式：URL(统一资源定位符) URN(统一资源名)

3 事务：请求+响应

4 方法：常用GET PUT DELETE POST HEAD TRACE OPTION

5 状态码：200 文档正确返回，302 重定向到其他地方获取资源，401 需要用户名密码，403 服务器拒绝访问，404 无法找到这个资源，405 请求不支持的方法，406 没有匹配客户端的资源，408 请求超时，413 实体太大，414 URL太长，502 bad gateway ，503 当前无法提供服务，504 网关超时，505 协议版本不支持

6 请求报文和响应报文：起始行（说明意图）、首部字段（名字+值，用冒号分隔，以空行结束）、主体（包含所有类型的数据）

7 先通过URL（由DNS进行域名转换）确定服务器资源的ip地址及端口号，建立tcp连接，通过Http协议传输资源信息。

8 Telnet程序可以将键盘连接到某个TCP端口，并将此端口的输出回送到显示屏上。常用于远程终端会话。

9 web的结构组件：代理（位于客户端与服务器之间的HTTP中间实体）、缓存（HTTP仓库）、网关（连接其他应用程序的特殊web服务器）、隧道（对HTTP通信报文进行盲转发的特殊代理）、Agent代理（发起自动HTTP请求的半智能web客户端）

10 对下层使用了TCP协议的HTTP来说，默认端口为80

11 URL格式：方案//：用户名：密码@主机名/路径;参数/路径？查询组件/路径#片段

12 URL的快捷方式，相对引用，自动扩展

13 方案格式：http https(默认端口443)       mailto ftp rtsp rtspu(实时流传输协议，音视频媒体资源的标识符u代表udp协议) file news telnet

14 状态码：200-299成功，300-399资源被移走 重定向，400-499客户端请求出错，500-599服务器出错

15 方法：安全方法GET HEAD不会对服务器产生动作

16 扩展方法：LOCK MKCOL(创建资源) COPY MOVE

17 TCP为HTTP提供了一条可靠的比特传输管道。

18 HTTPS就是在HTTP和TCP之间插入了一个(SSL或TLS)密码加密层

19 IP可以将你连接到正确的计算机上，端口号可以将你连接到正确的应用程序上。

20 每一条TCP连接通过4个值识别，源ip、源端口号、目的ip、目的端口号

21 TCP慢启动、持久连接、Nagle算法和TCP\_NODELAY、TIME\_WAIT端口耗尽

22 HTTP优化：并行连接、持久连接(connection首部keep-alive )、管道化持久连接、

23 connection首部，keep-alive、proxy-Authenticate、Proxy-Connection、Transfer-Encoding、Update都不能被代理转发或作为缓存响应使用的首部。

24 客户端不应该以管道化方式传送非幂等请求(比如POST)

25 关闭通道：全关闭、半关闭，应当前关闭输出通道，再检查确认没有数据输入将其关闭。

**HTTP结构**

1 代理连接的两个或多个使用相同协议的应用程序，而网关是协议转换器的运用。

2 代理可以改善安全性、提高性能、节省监控流量、实现有用的增值web服务

3 代理配置：手动配置浏览器选项、自动配置需设置pac文件的uri地址，JavaScript脚本。

4 服务器与代理的不同，完整与部分URl地址

5 网关、资源网关

6 使用web隧道的原因是在HTTP连接中嵌入非HTTP流量。位于客户端和网关之间

7 web机器人，网页爬虫、蜘蛛，用于搜索引擎查找网页

8 为解决HTTP协议的复杂性、可扩展性、性能、传输依赖性有关的一些问题。HTTP-NG：模块化及功能增强

9 模块化分层：报文传输层（新增WEBMUX协议）、远程调用层（请求/响应功能）、Web应用层（HTTP方法）

10 cookie

11 HTTP质询/响应机制：基本认证（与SSL配合使用，base-64加密相当与明文，不安全）、摘要认证（不在网络上直接传输密码，而是通过MD5摘要函数转换成的一个128位的摘要）

# Android知识

## Android框架

1. **应用层application**：用java编写，运行在虚拟机上的程序，如电话、联系人、短信、浏览器，开发可对其进行修改。
2. **应用程序框架application framework**：编写核心应用时所使用的api框架，开发可访问核心应用所使用的api框架来开发自己的应用；framework层提供了很多服务和系统来支持上层应用程序，包括视图view system（list、button、text、内置浏览器等）、内容提供器content provider（共享数据或者访问其他程序的数据）、资源管理器resources manager（图形、布局文件等）、通知管理器notification manager（可在状态栏显示自定义的消息提醒）、活动管理器activity manager（管理应用程序生命周期，并提供导航回退功能）、窗口管理器widow manager（管理所有程序的活动窗口）、包管理器package manager（android系统内的程序包管理）。
3. **系统运行库（library androidruntime）**：包含C/C++库，供android系统不同组件使用。重要的库有：Bionic系统C库 ；媒体库（media framework）；图层管理surface manager；浏览器引擎libwebcore；小型关系型数据库SQLite；android系统中每个应用程序都运行在自己的Dalvik虚拟机中执行.dex文件，每个执行的app都被分配一个虚拟机，使得android可以后台同时运行多个app。
4. **最底层是Linux 内核层（Linux kernel）**：主要提供与硬件操作的驱动，显示驱动、键盘驱动、照相机驱动、内存、蓝牙wifi、音频、Binder、电源管理

**1、Linux Kernel**

Android基于Linux 2.6提供核心系统服务，例如：安全、内存管理、进程管理、网络堆栈、驱动模型。Linux Kernel也作为硬件和软件之间的抽象层，它隐藏具体硬件细节而为上层提供统一的服务。 如果你学过计算机网络知道OSI/RM，就会知道分层的好处就是使用下层提供的服务而为上层提供统一的服务，屏蔽本层及以下层的差异，当本层及以下层发生了变化不会影响到上层。 也就是说各层各司其职，各层提供固定的SAP（Service Access Point），专业点可以说是高内聚、低耦合。如果你只是做应用开发，就不需要深入了解Linux Kernel层。

**2、Android Runtime**

Android 包含一个核心库的集合，提供大部分在Java编程语言核心类库中可用的功能。每一个Android应用程序是Dalvik虚拟机中的实例，运行在他们自己 的进程中。Dalvik虚拟机设计成，在一个设备可以高效地运行多个虚拟机。Dalvik虚拟机可执行文件格式是.dex，dex格式是专为Dalvik 设计的一种压缩格式，适合内存和处理器速度有限的系统。 大多数虚拟机包括JVM都是基于栈的，而Dalvik虚拟机则是基于寄存器的。 两种架构各有优劣，一般而言，基于栈的机器需要更多指令，而基于寄存器的机器指令更大。dx 是一套工具，可以將 Java .class 转换成 .dex 格式。一个dex文件通常会有多个.class。由于dex有時必须进行最佳化，会使文件大小增加1-4倍，以ODEX结尾。 Dalvik虚拟机依赖于Linux 内核提供基本功能，如线程和底层内存管理。

**3、Libraries**

Android包含一个C/C++库的集合，供Android系统的各个组件使用。这些功能通过Android的应用程序框架（application framework）暴露给开发者。

下面列出一些核心库：

a)：系统C库——标准C系统库（libc）的BSD衍生，调整为基于嵌入式Linux设备

b)：媒体库——基于PacketVideo的OpenCORE。这些库支持播放和录制许多流行的音频和视频格式，以及静态图像文件，包括MPEG4、 H.264、 MP3、 AAC、 AMR、JPG、 PNG

c)：界面管理——管理访问显示子系统和无缝组合多个应用程序的二维和三维图形层

d)：LibWebCore——新式的Web浏览器引擎,驱动Android 浏览器和内嵌的web视图

e)：SGL——基本的2D图形引擎

f)：3D库——基于OpenGL ES 1.0 APIs的实现。库使用硬件3D加速或包含高度优化的3D软件光栅

g)：FreeType ——位图和矢量字体渲染

h)：SQLite ——所有应用程序都可以使用的强大而轻量级的关系数据库引擎

**4、Application Framework**

通过提供开放的开发平台，Android使开发者能够编制极其丰富和新颖的应用程序。开发者可以自由地利用设备硬件优势、访问位置信息、运行后台服务、设置闹钟、向状态栏添加通知等等，很多很多。 开发者可以完全使用核心应用程序所使用的框架APIs。应用程序的体系结构旨在简化组件的重用，任何应用程序都能发布他的功能且任何其他应用程序可以使用这些功能（需要服从框架执行的安全限制）。这一机制允许用户替换组件。 所有的应用程序其实是一组服务和系统，包括：

a)：视图（View）——丰富的、可扩展的视图集合，可用于构建一个应用程序。包括包括列表、网格、文本框、按钮，甚至是内嵌的网页浏览器

b)：内容提供者（Content Providers）——使应用程序能访问其他应用程序（如通讯录）的数据，或共享自己的数据

c)：资源管理器（Resource Manager）——提供访问非代码资源，如本地化字符串、图形和布局文件

d)：通知管理器（Notification Manager）——使所有的应用程序能够在状态栏显示自定义警告

e)：活动管理器（Activity Manager）——管理应用程序生命周期,提供通用的导航回退功能

**5、Applications**

Android装配一个核心应用程序集合，包括电子邮件客户端、SMS程序、日历、地图、浏览器、联系人和其他设置。所有应用程序都是用Java编程语言写的。更加丰富的应用程序有待我们去开发！

## Android GC触发

* **Dalvik GC**

在对象的分配中会导致GC，第一次分配对象失败我们会触发GC但是不回收Soft的引用，如果再次分配还是失败我们就会将Soft的内存也给回收，前者触发的GC是GC\_FOR\_MALLOC类型的GC，后者是GC\_BEFORE\_OOM类型的GC。而当内存分配成功后，我们会判断当前的内存占用是否是达到了GC\_CONCURRENT的阀值，如果达到了那么又会触发GC\_CONCURRENT。

这个阀值又是如何来的呢，上面我们说到的一个目标利用率，GC后我们会记录一个目标值，这个值理论上需要再上述的范围之内，如果不在我们会选取边界值做为目标值。虚拟机会记录这个目标值，当做当前允许总的可以分配到的内存。同时根据目标值减去固定值（200~500K), 当做触发GC\_CONCURRENT事件的阈值。

* **ART GC**

ART在GC上做的比Dalvik好太多了，不光是GC的效率，减少Pause时间，而且还在内存分配上对大内存的有单独的分配区域，同时还能有算法在后台做内存整理，减少内存碎片。对于开发者来说ART下我们基本可以避免很多类似GC导致的卡顿问题了。另外根据谷歌自己的数据来看，ART相对Dalvik内存分配的效率提高了10倍，GC的效率提高了2-3倍。

# Web server

1. web service是一种应用程序，向外界暴露可以通过web调用的API，向调用者（客户端）返回数据。
2. web service有三要素：**SOAP  WSDL  UDDI**；soap用于描述传递信息的格式；wsdl用于描述如何访问具体接口；uddi用来管理、分发、查询webservice。
3. SOAP 是简单对象访问协议，简单、轻量、基于XML的交换数据的一种协议。
4. 在webservice中所有的访问都通过soap进行，用wsdl定义的接口封装，通过uddi进行目录查找，可以动态的改变一个服务的提供方而无需影响客户端的配置。
5. 对于webservice一个更详细的定义：web service是一个平台独立、低耦合、自包含的、基于编程的web应用程序。可使用开放的XML标准来描述、发布、发现、协调和配置这些应用程序，用于开发分布式的互操作的应用程序。
6. wsdl 网络服务描述语言，wsdl端口元素相当于一个函数库有可执行的操作和相关消息；wsdl消息元素相当于函数调用的参数；wsdl types元素定义webservice使用的数据类型；wsdl bindings为每个端口定义消息格式和协议细节，相当于接口文档、目录
7. uddi 通用描述、发现集成服务。用于对webservice的发布、发现、查找；uddi是一种目录服务，企业可通过uddi来注册和搜索web service。
8. web service的体系结构：提供者、请求者、中介、发布、发现、绑定。

# Mvc、mvp、mvvm

1. Android三个耳熟能详的框架：mvc、mvp、mvvm
2. mvc，全称model、view、controller，就是用controller控制model中的数据在view中显示。工作原理即当用户触发事件时，view通知controller层，然后controller区通知model层更新数据，model更新完数据后直接在view层显示。

对应到Android项目上是什么：view相当于Android的各种布局文件layout.xml，controller相当于各种activity，按钮监听器监听用户动作，主要是逻辑实现；剩下的java bean还有一些repository类（各种类文件）就相当于model层。

1. **mvc的缺点是什么**：view层的功能太弱，可以实现的任务太少，比如动态改变页面背景、动态隐藏显示按钮这类简单的功能也无法实现，全部写在controller层，就导致controller层既是controller层又是view层，代码实现相当复杂，不好维护。还有就是view层与model层是耦合的，耦合对于程序是非常不利的，不好维护。因此出现了MVP、mvvm两种框架。
2. **mvp框架**：model层与view层不再相互可知，由presenter做二者之间的桥梁，各种activity、fragment都属于view层；controller只负责监听转发动作。操作view层的事件传递到presenter层，presenter层通知model更新数据，并将数据的更新返回到view层，view与presenter之间通过接口实现相互调用，不存在耦合。
3. mvvm框架只是将presenter层换做viewmodel层，且view与view model之间是绑定的，使得viewmodel层的数据更新，会影响view的变动。

# Mysql练习

* select city\_name,**count(order\_no)** as count from order\_info where city\_name='廊坊市' **group by** city\_name;
* select city\_name,count(order\_no) as count from order\_info **group by** city\_name **order** **by** count;
* #聚合函数

**avg() sum() min() max() count()**

* #嵌套查询

select count(t.counts) from (select id,count(\*) counts from 表名 group by id) t

* select 类别, 摘要, sum(数量) as 数量之和 from A **group by all** 类别, 摘要
* **where** 子句的作用是在对查询结果进行分组前，将不符合where条件的行去掉，即在分组之前过滤数据，where条件中不能包含聚和函数，使用where条件过滤出特定的行。
* **having** 子句的作用是筛选满足条件的组，即在分组之后过滤数据，条件中经常包含聚和函数，使用having 条件过滤出特定的组，也可以使用多个分组标准进行分组。
* select 类别, sum(数量) as 数量之和 from A group by 类别 **having** sum(数量) > 18

## 查询语句

### Sutdent表的定义

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **字段名** | **字段描述** | **数据类型** | **主键** | **外键** | **非空** | **唯一** | **自增** |
| Id | 学号 | INT(10) | 是 | 否 | 是 | 是 | 是 |
| Name | 姓名 | VARCHAR(20) | 否 | 否 | 是 | 否 | 否 |
| Sex | 性别 | VARCHAR(4) | 否 | 否 | 否 | 否 | 否 |
| Birth | 出生年份 | YEAR | 否 | 否 | 否 | 否 | 否 |
| Department | 院系 | VARCHAR(20) | 否 | 否 | 是 | 否 | 否 |
| Address | 家庭住址 | VARCHAR(50) | 否 | 否 | 否 | 否 | 否 |

### Score表的定义

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 字段名 | 字段描述 | 数据类型 | 主键 | 外键 | 非空 | 唯一 | 自增 |
| Id | 编号 | INT(10) | 是 | 否 | 是 | 是 | 是 |
| Stu\_id | 学号 | INT(10) | 否 | 否 | 是 | 否 | 否 |
| C\_name | 课程名 | VARCHAR(20) | 否 | 否 | 否 | 否 | 否 |
| Grade | 分数 | INT(10) | 否 | 否 | 否 | 否 | 否 |

### 1.创建student和score表

CREATE  TABLE  student (

id  INT(10)  NOT NULL  UNIQUE  PRIMARY KEY  ,

name  VARCHAR(20)  NOT NULL ,

sex  VARCHAR(4)  ,

birth  YEAR,

department  VARCHAR(20) ,

address  VARCHAR(50)

);

创建score表。SQL代码如下：

CREATE  TABLE  score (

id  INT(10)  NOT NULL  UNIQUE  PRIMARY KEY  AUTO\_INCREMENT ,

stu\_id  INT(10)  NOT NULL ,

c\_name  VARCHAR(20) ,

grade  INT(10)

);

### 2.为student表和score表增加记录

向student表插入记录的INSERT语句如下：

INSERT INTO student VALUES( 901,'张老大', '男',1985,'计算机系', '北京市海淀区');

INSERT INTO student VALUES( 902,'张老二', '男',1986,'中文系', '北京市昌平区');

INSERT INTO student VALUES( 903,'张三', '女',1990,'中文系', '湖南省永州市');

INSERT INTO student VALUES( 904,'李四', '男',1990,'英语系', '辽宁省阜新市');

INSERT INTO student VALUES( 905,'王五', '女',1991,'英语系', '福建省厦门市');

INSERT INTO student VALUES( 906,'王六', '男',1988,'计算机系', '湖南省衡阳市');

向score表插入记录的INSERT语句如下：

INSERT INTO score VALUES(NULL,901, '计算机',98);

INSERT INTO score VALUES(NULL,901, '英语', 80);

INSERT INTO score VALUES(NULL,902, '计算机',65);

INSERT INTO score VALUES(NULL,902, '中文',88);

INSERT INTO score VALUES(NULL,903, '中文',95);

INSERT INTO score VALUES(NULL,904, '计算机',70);

INSERT INTO score VALUES(NULL,904, '英语',92);

INSERT INTO score VALUES(NULL,905, '英语',94);

INSERT INTO score VALUES(NULL,906, '计算机',90);

INSERT INTO score VALUES(NULL,906, '英语',85);

### 3.查询student表的所有记录

mysql> SELECT \* FROM student;

+-----+--------+------+-------+------------+--------------+

| id  | name   | sex  | birth | department | address      |

+-----+--------+------+-------+------------+--------------+

| 901 | 张老大 | 男   |  1985 | 计算机系   | 北京市海淀区 |

| 902 | 张老二 | 男   |  1986 | 中文系     | 北京市昌平区 |

| 903 | 张三   | 女   |  1990 | 中文系     | 湖南省永州市 |

| 904 | 李四   | 男   |  1990 | 英语系     | 辽宁省阜新市 |

| 905 | 王五   | 女   |  1991 | 英语系     | 福建省厦门市 |

| 906 | 王六   | 男   |  1988 | 计算机系   | 湖南省衡阳市 |

+-----+--------+------+-------+------------+--------------+

### 4.查询student表的第2条到4条记录

mysql> SELECT \* FROM student LIMIT 1,3;

+-----+--------+------+-------+------------+--------------+

| id  | name   | sex  | birth | department | address      |

+-----+--------+------+-------+------------+--------------+

| 902 | 张老二 | 男   |  1986 | 中文系     | 北京市昌平区 |

| 903 | 张三   | 女   |  1990 | 中文系     | 湖南省永州市 |

| 904 | 李四   | 男   |  1990 | 英语系     | 辽宁省阜新市 |

+-----+--------+------+-------+------------+--------------+

### 5.从student表查询所有学生的学号（id）、姓名（name）和院系（department）的信息

mysql> SELECT id,name,department FROM student;

+-----+--------+------------+

| id  | name   | department |

+-----+--------+------------+

| 901 | 张老大 | 计算机系   |

| 902 | 张老二 | 中文系     |

| 903 | 张三   | 中文系     |

| 904 | 李四   | 英语系     |

| 905 | 王五   | 英语系     |

| 906 | 王六   | 计算机系   |

+-----+--------+------------+

### 6.从student表中查询计算机系和英语系的学生的信息

mysql> SELECT \* FROM student WHERE department IN ('计算机系','英语系');

+-----+--------+------+-------+------------+--------------+

| id  | name   | sex  | birth | department | address      |

+-----+--------+------+-------+------------+--------------+

| 901 | 张老大 | 男   |  1985 | 计算机系   | 北京市海淀区 |

| 904 | 李四   | 男   |  1990 | 英语系     | 辽宁省阜新市 |

| 905 | 王五   | 女   |  1991 | 英语系     | 福建省厦门市 |

| 906 | 王六   | 男   |  1988 | 计算机系   | 湖南省衡阳市 |

+-----+--------+------+-------+------------+--------------+

### 7.从student表中查询年龄18~22岁的学生信息

mysql> SELECT id,name,sex,2013-birth AS age,department,address

    -> FROM student

    -> WHERE 2013-birth BETWEEN  18 AND 22;

+-----+------+------+------+------------+--------------+

| id  | name | sex  | age  | department | address      |

+-----+------+------+------+------------+--------------+

| 905 | 王五 | 女   |   22 | 英语系     | 福建省厦门市 |

+-----+------+------+------+------------+--------------+

mysql> SELECT id,name,sex,2013-birth AS age,department,address

    -> FROM student

    -> WHERE 2013-birth>=18 AND 2013-birth<=22;

+-----+------+------+------+------------+--------------+

| id  | name | sex  | age  | department | address      |

+-----+------+------+------+------------+--------------+

| 905 | 王五 | 女   |   22 | 英语系     | 福建省厦门市 |

+-----+------+------+------+------------+--------------+

### 8.从student表中查询每个院系有多少人

mysql> SELECT department, COUNT(id) FROM student GROUP BY department;

+------------+-----------+

| department | COUNT(id) |

+------------+-----------+

| 计算机系   |         2 |

| 英语系     |         2 |

| 中文系     |         2 |

+------------+-----------+

9.从score表中查询每个科目的最高分

mysql> SELECT c\_name,MAX(grade) FROM score GROUP BY c\_name;

+--------+------------+

| c\_name | MAX(grade) |

+--------+------------+

| 计算机 |         98 |

| 英语   |         94 |

| 中文   |         95 |

+--------+------------+

### 10.查询李四的考试科目（c\_name）和考试成绩（grade）

mysql> SELECT c\_name, grade

    ->      FROM score WHERE stu\_id=

    ->  (SELECT id FROM student

    ->    WHERE name= '李四' );

+--------+-------+

| c\_name | grade |

+--------+-------+

| 计算机 |    70 |

| 英语   |    92 |

+--------+-------+

### 11.用连接的方式查询所有学生的信息和考试信息

mysql> SELECT student.id,name,sex,birth,department,address,c\_name,grade

    -> FROM student,score

    ->  WHERE student.id=score.stu\_id;

+-----+--------+------+-------+------------+--------------+--------+-------+

| id  | name   | sex  | birth | department | address      | c\_name | grade |

+-----+--------+------+-------+------------+--------------+--------+-------+

| 901 | 张老大 | 男   |  1985 | 计算机系   | 北京市海淀区 | 计算机 |    98 |

| 901 | 张老大 | 男   |  1985 | 计算机系   | 北京市海淀区 | 英语   |    80 |

| 902 | 张老二 | 男   |  1986 | 中文系     | 北京市昌平区 | 计算机 |    65 |

| 902 | 张老二 | 男   |  1986 | 中文系     | 北京市昌平区 | 中文   |    88 |

| 903 | 张三   | 女   |  1990 | 中文系     | 湖南省永州市 | 中文   |    95 |

| 904 | 李四   | 男   |  1990 | 英语系     | 辽宁省阜新市 | 计算机 |    70 |

| 904 | 李四   | 男   |  1990 | 英语系     | 辽宁省阜新市 | 英语   |    92 |

| 905 | 王五   | 女   |  1991 | 英语系     | 福建省厦门市 | 英语   |    94 |

| 906 | 王六   | 男   |  1988 | 计算机系   | 湖南省衡阳市 | 计算机 |    90 |

| 906 | 王六   | 男   |  1988 | 计算机系   | 湖南省衡阳市 | 英语   |    85 |

+-----+--------+------+-------+------------+--------------+--------+-------+

### 12.计算每个学生的总成绩

mysql> SELECT student.id,name,SUM(grade) FROM student,score

    -> WHERE student.id=score.stu\_id

    -> GROUP BY id;

+-----+--------+------------+

| id  | name   | SUM(grade) |

+-----+--------+------------+

| 901 | 张老大 |        178 |

| 902 | 张老二 |        153 |

| 903 | 张三   |         95 |

| 904 | 李四   |        162 |

| 905 | 王五   |         94 |

| 906 | 王六   |        175 |

+-----+--------+------------+

### 13.计算每个考试科目的平均成绩

mysql> SELECT c\_name,AVG(grade) FROM score GROUP BY c\_name;

+--------+------------+

| c\_name | AVG(grade) |

+--------+------------+

| 计算机 |    80.7500 |

| 英语   |    87.7500 |

| 中文   |    91.5000 |

+--------+------------+

### 14.查询计算机成绩低于95的学生信息

mysql> SELECT \* FROM student

    -> WHERE id IN

    -> (SELECT stu\_id FROM score

    -> WHERE c\_name="计算机" and grade<95);

+-----+--------+------+-------+------------+--------------+

| id  | name   | sex  | birth | department | address      |

+-----+--------+------+-------+------------+--------------+

| 902 | 张老二 | 男   |  1986 | 中文系     | 北京市昌平区 |

| 904 | 李四   | 男   |  1990 | 英语系     | 辽宁省阜新市 |

| 906 | 王六   | 男   |  1988 | 计算机系   | 湖南省衡阳市 |

+-----+--------+------+-------+------------+--------------+

### 15.查询同时参加计算机和英语考试的学生的信息

mysql> SELECT \*  FROM student

    ->  WHERE id =ANY

    ->  ( SELECT stu\_id FROM score

    ->  WHERE stu\_id IN (

    ->          SELECT stu\_id FROM

    ->          score WHERE c\_name=  '计算机')

    ->  AND c\_name= '英语' );

+-----+--------+------+-------+------------+--------------+

| id  | name   | sex  | birth | department | address      |

+-----+--------+------+-------+------------+--------------+

| 901 | 张老大 | 男   |  1985 | 计算机系   | 北京市海淀区 |

| 904 | 李四   | 男   |  1990 | 英语系     | 辽宁省阜新市 |

| 906 | 王六   | 男   |  1988 | 计算机系   | 湖南省衡阳市 |

+-----+--------+------+-------+------------+--------------+

mysql> SELECT a.\* FROM student a ,score b ,score c

    -> WHERE a.id=b.stu\_id

    -> AND b.c\_name='计算机'

    -> AND a.id=c.stu\_id

    -> AND c.c\_name='英语';

+-----+--------+------+-------+------------+--------------+

| id  | name   | sex  | birth | department | address      |

+-----+--------+------+-------+------------+--------------+

| 901 | 张老大 | 男   |  1985 | 计算机系   | 北京市海淀区 |

| 904 | 李四   | 男   |  1990 | 英语系     | 辽宁省阜新市 |

| 906 | 王六   | 男   |  1988 | 计算机系   | 湖南省衡阳市 |

+-----+--------+------+-------+------------+--------------+

### 16.将计算机考试成绩按从高到低进行排序

mysql> SELECT stu\_id, grade

    ->  FROM score WHERE c\_name= '计算机'

    ->  ORDER BY grade DESC;

+--------+-------+

| stu\_id | grade |

+--------+-------+

|    901 |    98 |

|    906 |    90 |

|    904 |    70 |

|    902 |    65 |

+--------+-------+

### 17.从student表和score表中查询出学生的学号，然后合并查询结果

mysql> SELECT id  FROM student

    -> UNION

    -> SELECT stu\_id  FROM score;

+-----+

| id  |

+-----+

| 901 |

| 902 |

| 903 |

| 904 |

| 905 |

| 906 |

+-----+

### 18.查询姓张或者姓王的同学的姓名、院系和考试科目及成绩

mysql> SELECT [student.id](http://student.id), name,sex,birth,department, address, c\_name,grade

    -> FROM student, score

    -> WHERE

    ->  (name LIKE  '张%'  OR name LIKE  '王%')

    ->  AND

    ->  student.id=score.stu\_id ;

+-----+--------+------+-------+------------+--------------+--------+-------+

| id  | name   | sex  | birth | department | address      | c\_name | grade |

+-----+--------+------+-------+------------+--------------+--------+-------+

| 901 | 张老大 | 男   |  1985 | 计算机系   | 北京市海淀区 | 计算机 |    98 |

| 901 | 张老大 | 男   |  1985 | 计算机系   | 北京市海淀区 | 英语   |    80 |

| 902 | 张老二 | 男   |  1986 | 中文系     | 北京市昌平区 | 计算机 |    65 |

| 902 | 张老二 | 男   |  1986 | 中文系     | 北京市昌平区 | 中文   |    88 |

| 903 | 张三   | 女   |  1990 | 中文系     | 湖南省永州市 | 中文   |    95 |

| 905 | 王五   | 女   |  1991 | 英语系     | 福建省厦门市 | 英语   |    94 |

| 906 | 王六   | 男   |  1988 | 计算机系   | 湖南省衡阳市 | 计算机 |    90 |

| 906 | 王六   | 男   |  1988 | 计算机系   | 湖南省衡阳市 | 英语   |    85 |

+-----+--------+------+-------+------------+--------------+--------+-------+

### 19.查询都是湖南的学生的姓名、年龄、院系和考试科目及成绩

mysql> SELECT [student.id](http://student.id), name,sex,birth,department, address, c\_name,grade

    -> FROM student, score

    -> WHERE address LIKE '湖南%'   AND

    ->  student.id=score.stu\_id;

+-----+------+------+-------+------------+--------------+--------+-------+

| id  | name | sex  | birth | department | address      | c\_name | grade |

+-----+------+------+-------+------------+--------------+--------+-------+

| 903 | 张三 | 女   |  1990 | 中文系     | 湖南省永州市 | 中文   |    95 |

| 906 | 王六 | 男   |  1988 | 计算机系   | 湖南省衡阳市 | 计算机 |    90 |

| 906 | 王六 | 男   |  1988 | 计算机系   | 湖南省衡阳市 | 英语   |    85 |

+-----+------+------+-------+------------+--------------+--------+-------+

## 插入、更新、删除

### Food表的定义

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **字段名** | **字段描述** | **数据类型** | **主键** | **外键** | **非空** | **唯一** | **自增** |
| Id | 编号 | INT(10) | 是 | 否 | 是 | 是 | 是 |
| Name | 食品公司 | VARCHAR(20) | 否 | 否 | 是 | 否 | 否 |
| Company | 生产厂商 | VARCHAR(30) | 否 | 否 | 是 | 否 | 否 |
| Price | 价格(单位:圆) | FLOAT | 否 | 否 | 否 | 否 | 否 |
| Produce\_time | 生产年份 | YEAR | 否 | 否 | 否 | 否 | 否 |
| Validity\_time | 保质期(单位:年) | INT(4) | 否 | 否 | 否 | 否 | 否 |
| Address | 厂址 | VARCAHR(50) | 否 | 否 | 否 | 否 | 否 |

mysql> CREATE TABLE food(

    ->  id INT(10) PRIMARY KEY NOT NULL UNIQUE AUTO\_INCREMENT,

    ->  name VARCHAR(20) NOT NULL,

    ->  company VARCHAR(30) NOT NULL,

    ->  price FLOAT,

    ->  produce\_time YEAR,

    ->  validity\_time INT(4),

    ->  address VARCHAR(50)

->  );

### 1.将下边的记录插入到food表中

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Id** | **Name** | **Company** | **Price** | **Produce\_time** | **Validity\_time** | **address** |
| 1 | AA饼干 | AA饼干厂 | 2.5 | 2008 | 3 | 北京 |
| 2 | CC牛奶 | CC牛奶厂 | 3.5 | 2009 | 1 | 河北 |
| 3 | EE果冻 | EE果冻厂 | 1.5 | 2007 | 2 | 北京 |
| 4 | FF咖啡 | FF咖啡厂 | 20 | 2002 | 5 | 天津 |
| 5 | GG奶糖 | GG奶糖厂 | 14 | 2003 | 3 | 广东 |

第一种方法不指定具体的字段：

INSERT  INTO  food  VALUES(1,'AA饼干','AA饼干厂', 2.5 ,'2008', 3 ,'北京');

第二种方法是依次指定food表的字段：

INSERT  INTO  food( id, name, company, price, produce\_time, validity\_time, address)

VALUES(2,'CC牛奶','CC牛奶厂', 3.5 ,'2009', 1 ,'河北');

第三种方法是同时插入多条记录：

INSERT  INTO  food  VALUES

(NULL,'EE果冻','EE果冻厂', 1.5 ,'2007', 2 ,'北京') ,

(NULL,'FF咖啡','FF咖啡厂', 20 ,'2002', 5 ,'天津') ,

(NULL,'GG奶糖','GG奶糖', 14 ,'2003', 3 ,'广东');

### 2.将“CC牛奶厂”的厂址（address）改为“内蒙古”，并且将价格改为3.2

mysql> UPDATE food SET address='内蒙古',price=3.2 WHERE name='CC牛奶';

### 3.将厂址在北京的公司保质期（validity\_time）都改为5年

mysql> UPDATE food SET validity\_time=5 WHERE address='北京';

### 4.删除过期食品的记录。若当前时间-生产年份（produce\_time）>保质期（validity\_time），则视为过期食品

mysql> DELETE FROM food WHERE validity\_time<(2013-produce\_time);

### 5.删除厂址为北京的食品的记录

mysql> DELETE FROM food WHERE address='北京';

### Teacher表的定义

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **字段名** | **字段描述** | **数据类型** | **主键** | **外键** | **非空** | **唯一** | **自增** |
| Id | 编号 | INT(4) | 是 | 否 | 是 | 是 | 是 |
| Num | 教工号 | INT(10) | 否 | 否 | 是 | 是 | 否 |
| Name | 姓名 | VARCHAR(20) | 否 | 否 | 是 | 否 | 否 |
| Sex | 性别 | VARCHAR(4) | 否 | 否 | 是 | 否 | 否 |
| Birthday | 出生日期 | DATETIME | 否 | 否 | 否 | 否 | 否 |
| address | 家庭住址 | VARCHAR(50) | 否 | 否 | 否 | 否 | 否 |

1.向teacher表中插入数据

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Id | Num | Name | Sex | Birthday | Address |
| 1 | 1001 | 张三 | 男 | 1984-11-08 | 北京市昌平区 |
| 2 | 1002 | 李四 | 女 | 1970-01-21 | 北京市海淀区 |
| 3 | 1003 | 王五 | 男 | 1976-10-30 | 北京市昌平区 |
| 4 | 1004 | 赵六 | 男 | 1980-06-05 | 北京市顺义区 |

INSERT  INTO  teacher VALUES(1, 1001, '张三','男' ,'1984-11-08' ,'北京市昌平区'),

(NULL, 1002, '李四','女' ,'1970-01-21' ,'北京市海淀区') ,

(NULL, 1003, '王五','男' ,'1976-10-30' ,'北京市昌平区') ,

(NULL, 1004, '赵六','男' ,'1980-06-05' ,'北京市顺义区') ;

### 2.更新id为1的记录，将生日（birthday）改为“1982-11-08”

mysql> UPDATE teacher SET birthday='1982-11-08' WHERE id=1;

### 3.将性别（sex）为“男”的记录的家庭住址（address）都变为“北京市朝阳区”

mysql> UPDATE teacher SET address='北京市朝阳区' WHERE sex='男';

### 4.删除教工号（num）为1002的记录删除

mysql> DELETE FROM teacher WHERE num=1002;

# PHP

* Php--personal home page：Hypertext Preprocessor 超文本预处理器，一种开源脚本语言，在服务器上执行，结果以纯文本返回浏览器。文件后缀.php 可嵌入html
* 运行：安装web服务器；安装php（zip包）；安装mysql
* Php脚本在服务器上执行，然后向浏览器发送回纯html结果。
* Php脚本可以放置文档中的任何位置，以<?php 开头，以 ?>结尾。
* PHP 语句以分号结尾（;）。PHP 代码块的关闭标签也会自动表明分号（因此在 PHP 代码块的最后一行不必使用分号）。
* 在 PHP 中，所有用户定义的函数、类和关键词（例如 if、else、echo 等等）都对大小写不敏感。不过在 PHP 中，所有变量都对**大小写敏感**。
* **注释**：// , # , /\* \*/

# Java内容

## Java通识

1. Hashtable与HashMap的区别：线程安全，HashMap为非安全；HashMap的key value可以为Null，Hashtable不可以；产生的时间不同Hashtable时间更早
2. I/O流：File类的使用，File file=new File("文件路径名称")  file.creatNewFile();  File.separator, File.pathSeparator, file.exists(), file.delete(), file.mkdir(), file.isDirectory(), file.listFiles()
3. 常用流：字节输入流 InputStream 方法read()；字节输出流 OutputStream 方法write(), close(), flush()；文件输入FileInputStream ,FileOutputStream；字符流Reader, Writer；InputStreamReader和OutputStreamWriter字节流转为字符流的转换器。逐行读写：BufferedReader/BufferedWriter, readLine()返回Null表示已经读取完成，newLine()写一个换行符。管道流PipedInputStream。数据流：DataInputStream ；对象流：ObjectInputStream实现对象序列化，实现Serializeble接口，网络中传递对象必须实现序列化。
4. java集合框架List set map
5. List ArrayList(非线程安全，异步处理，执行效率高，便于查找), Vector(线程安全，同步处理), LinkedList(便于删除)
6. Hashcode()确定数组的下标，要重写此方法
7. Map ,HashMap<,>
8. 集合框架是用来存储复杂数据，他的元素为对象，最基本的为Collection接口。List是有序可重复的集合，他的继承类有LinkedList链表，查询效率低删除效率高，ArrayList查询效率高，删除效率低，相当于大小可变可自动增长的数组，加载大量数据时，手动扩容可提高效率。Set是无序的不可重复的集合。HashSet,Object类中的hashCode是所有类都会继承的方法，通过hashCode判断要加入的对象是否和数组中的元素相同，不同时才会被添加。
9. Map是和Collection同级的接口，Hashtable是java一开始就有的，线程安全同步，HashMap是后加的，非线程安全，异步处理，以键值对的形式存储数据，key值一般是8种基本类型的封装类或者String类，key不可重复

## Java常见异常

**NO.1　java.lang.NullPointerException**

　　调用了未经初始化的对象或者是不存在的对象，这个错误经常出现在创建图片，调用数组这些操作中，比如图片未经初始化，或者图片创建时的路径错误等等。对数组操作中出现空指针，很多情况下是一些刚开始学习编程的朋友常犯的错误，即把数组的初始化和数组元素的初始化混淆起来了。数组的初始化是对数组分配需要的空间，而初始化后的数组，其中的元素并没有实例化，依然是空的，所以还需要对每个元素都进行初始化（如果要调用的话）。

　　在jsp编程中经常出现：if (request.getParameter(“username”).equals(“xxx”))、out.println(session.getAttribute(“record”))等。解决这个问题的方法是在使用前进行判空比较：

　　 if (request.getParameter(“username”)!=null)

　　{if if (request.getParameter(“username”).

　　equals(“xxx”))…}

**NO.2　java.lang.ClassNotFoundException**

　　这个异常是很多原本在JB等开发环境中开发的程序员，把JB（*JBuilder是Borland公司开发的针对java的开发工具，使用JBuilder将可以快速，有效的开发各类java应用，它使用的JDK与sun公司标准的JDK不同，它经过了较多的修改，以便开发人员能够像开发Delphi应用那样开发java应用。*）下的程序包放在WTk（*Wireless Toolkit是Sun的无线开发工具包，这一工具包的设计目的是为了帮助开发人员简化j2me的开发过程。*）下编译经常出现的问题，异常的解释是 “指定的类不存在 “，这里主要考虑一下类的名称和路径是否正确即可，如果是在JB下做的程序包，一般都是默认加上Package的，所以转到WTK下后要注意把Package的路径加上。

**NO.3　java.lang.ArithmeticException**

　　这个异常的解释是 “数学运算异常 “，比如程序中出现了除以零这样的运算就会出这样的异常，对这种异常，大家就要好好检查一下自己程序中涉及到数学运算的地方，公式是不是有不妥了。

**NO.4　java.lang.ArrayIndexOutOfBoundsException**

　　这个异常相信很多朋友也经常遇到过，异常的解释是 “数组下标越界 “，现在程序中大多都有对数组的操作，因此在调用数组的时候一定要认真检查，看自己调用的下标是不是超出了数组的范围，一般来说，显示（即直接用常数当下标）调用不太容易出这样的错，但隐式（即用变量表示下标）调用就经常出错了，还有一种情况，是程序中定义的数组的长度是通过某些特定方法决定的，不是事先声明的，这个时候，最好先查看一下数组的length，以免出现这个异常。

**NO.5　java.lang.IllegalArgumentException**

　　这个异常的解释是 “方法的参数错误 “，很多J2ME的类库中的方法在一些情况下都会引发这样的错误，比如音量调节方法中的音量参数如果写成负数就会出现这个异常，再比如g.setColor(int red,int green,int blue)这个方法中的三个值，如果有超过２５５的也会出现这个异常，因此一旦发现这个异常，我们要做的，就是赶紧去检查一下方法调用中的参数传递是不是出现了错误。

**NO.6　java.lang.IllegalAccessException**

　　这个异常的解释是 “没有访问权限 “，当应用程序要调用一个类，但当前的方法即没有对该类的访问权限便会出现这个异常。对程序中用了Package的情

况下要注意这个异常。

**NO.7　java.lang.IncompatibleClassChangeError**

不兼容的类变化错误。当正在执行的方法所依赖的类定义发生了不兼容的改变时，抛出该异常。一般在修改了应用中的某些类的声明定义而没有对整个应用重

新编译而直接运行的情况下，容易引发该错误。

**NO.8　java.lang.InstantiationError**

实例化错误。当一个应用试图通过Java的new操作符构造一个抽象类或者接口时抛出该异常.

**NO.9　java.lang.LinkageError**

链接错误。该错误及其所有子类指示某个类依赖于另外一些类，在该类编译之后，被依赖的类改变了其类定义而没有重新编译所有的类，进而引发错误的情况

。

**NO.10　java.lang.StackOverflowError**

堆栈溢出错误。当一个应用递归调用的层次太深而导致堆栈溢出时抛出该错误。

## Foreach循环

for(part1:part2){part3};

part2中是一个数组对象,或者是带有泛性的集合

part1定义了一个局部变量,这个局部变量的类型与part2中的对象元素的类型是一致的

public void test1() {

//定义并初始化一个数组

int arr[] = {2, 3, 1};

System.out.println("----1----排序前的一维数组");

for (int x : arr) {

System.out.println(x); //逐个输出数组元素的值

}

public void listToArray() {

//创建List并添加元素

List<String> list = new ArrayList<String>();

list.add("1");

list.add("3");

list.add("4");

//利用froeach语句输出集合元素

System.out.println("----2----froeach语句输出集合元素");

for (String x : list) {

System.out.println(x);

}

# Linux命令

## 系统介绍

1. Linux发行版本可以大体分为两类，一类是商业公司维护的发行版本，一类是社区组织维护的发行版本，前者以著名的Redhat(RHEL)为代表，后者以Debian为代表。
2. **Redhat系列**，包括RHEL、Fedora、CentOS(RHEL的社区克隆版本，免费)。Ubuntu严格来说不能算一个独立的发行版本，Ubuntu是基于Debian加强而来，一个拥有Debian所有的优点，以及自己所加强的优点的近乎完美的Linux桌面系统。Ubuntu社区认为实用性和创新性高于一切。如果你需要一个**比较稳定的服务器系统**而已，那么最好的选择就是CentOS了。
3. Linux有内核**kernel和user**两种模式：**内核、shell、文件系统**一起形成了基本的操作系统结构。

Linux 内核由如下几部分组成：内存管理、进程管理、设备驱动程序、文件系统和网络管理等。

## 文件系统

Linux系统能支持多种目前流行的文件系统，推荐用EXT4。

|  |  |
| --- | --- |
| 目录名 | 简要说明 |
| /bin | 二进制可执行命令 |
| /dev | 设备特殊文件 |
| /etc | 系统管理和配置文件 |
| /etc/rc.d | 启动的配置文件和脚本 |
| /home | 用户主目录的基点，比如用户abc的主目录就是/home/user，可以用~abc表示 |
| /lib | 标准程序设计库 |
| /sbin | 系统管理命令，这里存放的是系统管理员使用的管理程序 |
| /tmp | 公用的临时文件存储点 |
| /root | 系统管理员的主目录 |
| /mnt | 系统提供这个目录是让用户临时挂载其他的文件系统。 |
| /lost+found | 这个目录平时是空的，系统非正常关机而留下“无家可归”的文件 |
| /proc | 虚拟的目录，是系统内存的映射。可直接访问这个目录来获取系统信息。 |
| /var | 某些大文件的溢出区，比方说各种服务的日志文件 |
| /usr | 最庞大的目录，要用到的应用程序和文件几乎都在这个目录。其中包含： /usr/X11R6 存放X window的目录 /usr/bin 众多的应用程序 /usr/sbin 超级用户的一些管理程序 /usr/doc linux文档 /usr/include linux下开发和编译应用程序所需要的头文件 /usr/lib 常用的动态链接库和软件包的配置文件 /usr/man 帮助文档 /usr/src 源代码，linux内核的源代码就放在/usr/src/linux里 /usr/local/bin 本地增加的命令 /usr/local/lib 本地增加的库 |

* Wget 下载链接

## 目录介绍

/：存放系统程序，也就是At&t开发的Unix程序。

　　/usr：存放Unix系统商（比如IBM和HP）开发的程序。

　　/usr/local：存放用户自己安装的程序。

/opt：在某些系统，用于存放第三方厂商开发的程序，所以取名为option，意为"选装"。

## Sudo

* 当自身没有权限访问文件或执行命令时需临时切换到root用户完成相关操作，哪些用户可以临时获得root 权限需要在 /etc/sudoers 文件中进行配置：name ALL=(ALL) ALL name—授权的用户名；ALL—允许从任何中断使用sudo；(ALL)—允许以任何用户执行sudo命令；ALL—允许sudo权限执行任何命令
* 用户 test 只能在本主机（主机名为liudongjie-pc）以 root 账户执行/bin/chown、/bin/chmod 两条命令，那么就应该这样配置：test liudongjie-pc=(root) bin/chown,/bin/chmod
* 当vim编辑后发现忘记使用sudo，可以在 vim 的普通模式下，按 :w !sudo tee % ，这样就可以 root 权限来保存文件了。
* 执行 root 命令忘记加 sudo；只要输入 sudo !! 即可，这里的 !! 代表上一条命令。
* sudo 后面不能跟 shell 的内置命令

## 基本命令



# Maven

## 介绍

[完整介绍](http://www.oracle.com/technetwork/cn/community/java/apache-maven-getting-started-1-406235-zhs.html)

**maven的基本概念**

* **项目构建**
  + 完整的项目构建过程包括【清理项目】→【编译项目】→【测试项目】→【生成测试报告】→【打包项目】→【部署项目】。
  + 理想的项目构建是高度自动化，跨平台，可重用的组件，标准化的，使用maven就可以帮我们完成上述所说的项目构建过程。
* **依赖管理**
  + 依赖指的是jar包之间的相互依赖，比如我们搭建一个Struts2的开发框架时，光光有struts2-core-2.3.16.3.jar这个jar包是不行的，struts2-core-2.3.16.3.jar还依赖其它的jar包，依赖管理指的就是使用Maven来管理项目中使用到的jar包，Maven管理的方式就是“自动下载项目所需要的jar包，统一管理jar包之间的依赖关系”。
* **使用maven的好处**
  + Maven中使用约定，约定java源代码代码必须放在哪个目录下，编译好的java代码又必须放到哪个目录下，这些目录都有明确的约定
  + Maven的每一个动作都拥有一个生命周期，例如执行 mvn install 就可以自动执行编译，测试，打包等构建过程
  + 只需要定义一个pom.xml,然后把源码放到默认的目录，Maven帮我们处理其他事情
  + 使用Maven可以进行项目高度自动化构建，依赖管理(这是使用Maven最大的好处)，仓库管理。

## 安装

[下载bin版本](https://maven.apache.org/download.cgi) 解压到指定目录下

设置path路径：vim ~/.bash\_profile M2\_HOME= /Users/ralphliu/app/apache-maven-3.5.0

PATH=$PATH:$M2\_HOME/bin

查看maven版本是否安装成功：mvn –v

## 构建项目

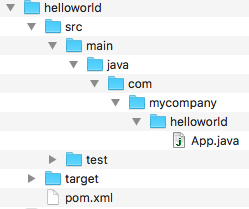
在命令行中执行下面的命令来建立我们的 hello world 项目：

  ~$mvn archetype:generate -DgroupId=com.mycompany.helloworld -DartifactId=helloworld -Dpackage=com.mycompany.helloworld -Dversion=1.0-SNAPSHOT

archetype:generate 目标会列出一系列的 archetype 让你选择。 Archetype 可以理解成项目的模型。 Maven 为我们提供了很多种的项目模型，包括从简单的 Swing 到复杂的 Web 应用。我们选择默认的 maven-archetype-quickstart ，是编号 #106 （使用默认编号）

我们在命令行中用 -D 选项指定了项目的属性。该选项使用 -Dname=value 的格式。回车确认，就完成了项目的建立。可以在当前目录下看到建立的helloworld项目，过程需要联网.

maven 的 archetype 插件建立了一个 helloworld 目录，这个名字来自 artifactId 。在这个目录下面，有一个 Project Object Model(POM) 文件 pom.xml 。这个文件用于描述项目，配置插件和管理依赖关系。源代码和资源文件放在 src/main 下面，而测试代码和资源放在 src/test 下面。



**构建项目：**

*~$cd helloworld*

*~$mvn package*

当你第一次运行 maven 的时候，它会从网上的 maven 库 (repository) 下载需要的程序，存放在你电脑的本地库 (local repository) 中，所以这个时候你需要有 Internet 连接。Maven 默认的本地库是 ~/.m2/repository/ ，在 Windows 下是 %USER\_HOME%\.m2\repository\

运行构建后的程序验证构建成功：自动生成target目录下的jar文件

*~$java -cp target/helloworld-1.0-SNAPSHOT.jar com.mycompany.helloworld.App*

* maven是采用纯java编写的开源项目管理工具。
* maven采用project object model（POM）的概念来管理项目，所有的配置信息都被定义在一个叫POM.XML的文件中（管理项目整个生命周期，编译构建、测试、发布、报告等）
* Apache下绝大多数项目都已经采用maven进行管理。

## 作用

指导开发、自动编译、依赖管理、无限扩展、持续集成、开发协作

## POM文件

一个项目所有的配置都放置在 POM 文件中：定义项目的类型、名字，管理依赖关系，定制插件的行为等等。比如说，你可以配置 compiler 插件让它使用 java 1.5 来编译。

在 POM 中，groupId, artifactId, packaging, version 叫作 maven 坐标，它能唯一的确定一个项目。有了 maven 坐标，我们就可以用它来指定我们的项目所依赖的其他项目，插件，或者父项目。一般 maven 坐标写成如下的格式：

     groupId:artifactId:packaging:version

像我们的例子就会写成：

     com.mycompany.helloworld: helloworld: jar: 1.0-SNAPSHOT

我们的 helloworld 示例很简单，但是大项目一般会分成几个子项目。在这种情况下，每个子项目就会有自己的 POM 文件，然后它们会有一个共同的父项目。这样只要构建父项目就能够构建所有的子项目了。子项目的 POM 会继承父项目的 POM。另外，所有的 POM都继承了一个 Super-POM。Super-POM 设置了一些默认值，比如在[第一篇文章](http://www.oracle.com/technetwork/cn/community/java/apache-maven-getting-started-1-406235-zhs.html)中提到的默认的目录结构，默认的插件等等，它遵循了惯例优于配置的原则。所以尽管我们的这个 POM 很简单，但是这只是你看得见的一部分。运行时候的 POM 要复杂的多。 如果你想看到运行时候的 POM 的全部内容的话，可以运行下面的命令：

$mvn help:effective-pom

## 生命周期

mvn package。这里的 package 是一个maven的生命周期阶段 (lifecycle phase )。生命周期指项目的构建过程，它包含了一系列的有序的阶段 (phase)，而一个阶段就是构建过程中的一个步骤。

那么生命周期阶段和上面说的插件目标之间是什么关系呢？插件目标可以绑定到生命周期阶段上。一个生命周期阶段可以绑定多个插件目标。当 maven 在构建过程中逐步的通过每个阶段时，会执行该阶段所有的插件目标。

maven 能支持不同的生命周期，但是最常用的是默认的Maven生命周期 (default Maven lifecycle )。如果你没有对它进行任何的插件配置或者定制的话，那么上面的命令 mvn package 会依次执行默认生命周期中直到包括 package 阶段前的所有阶段的插件目标：

1. process-resources 阶段：resources:resources
2. compile 阶段：compiler:compile
3. process-classes 阶段：(默认无目标)
4. process-test-resources 阶段：resources:testResources
5. test-compile 阶段：compiler:testCompile
6. test 阶段：surefire:test
7. prepare-package 阶段：(默认无目标)
8. package 阶段：jar:jar

## maven库

但是不是所有的 jar 文件都是可以从默认的远程库下载的，比如说我们自己开发的项目。这个时候，有两个选择：要么在公司内部设置定制库，要么手动下载和安装所需的jar文件到本地库。

本地库是指 maven 下载了插件或者 jar 文件后存放在本地机器上的拷贝。在 Linux 上，它的位置在 ~/.m2/repository，在 Windows XP 上，在 C:\Documents and Settings\username\.m2\repository ，在 Windows 7 上，在 C:\Users\username\.m2\repository。当 maven 查找需要的 jar 文件时，它会先在本地库中寻找，只有在找不到的情况下，才会去远程库中找。

运行下面的命令能把我们的 helloworld 项目安装到本地库：

     $mvn install

一旦一个项目被安装到了本地库后，你别的项目就可以通过 maven 坐标和这个项目建立依赖关系。比如如果我现在有一个新项目需要用到 helloworld，那么在运行了上面的 mvn install 命令后，我就可以如下所示来建立依赖关系：

Xml 代码

    <dependency>

     <groupId>com.mycompany.helloworld</groupId>

     <artifactId>helloworld</artifactId>

     <version>1.0-SNAPSHOT</version>

    </dependency>

## 项目目录约定



# selenium2Python

## webdriver

* 定位一组对象：find\_elements() 应用场景：批量操作对象，比如全选页面上的checkbox；先获取一组对象，再在这组对象中过滤出需要具体定位的对象，如定位所有的checkbox，再选出最后一个。

层级定位：点击两个不同菜单下的元素相同。二次定位，driver.find\_element\_by\_id('xx').find\_element\_by\_link\_text('xx').click()

移动鼠标到子元素上ActionChains(driver).move\_to\_element(menu).perform()

* 定位frame中的元素：switch\_to\_frame
* 浏览器多窗口：获取当前窗口，current\_window\_handle；获取所有窗口，window\_handles；切换到某窗口，switch\_to\_window
* alert/comfirm/prompt处理：switch\_to\_alert() text/accept/dismiss/send\_keys
* 下拉框处理：二次定位，根据不同的下拉框形式（move\_to\_element()）
* 分页处理：获取所有的分页option标签，使用for循环对每个分页执行操作。
* 上传文件：直接定位上传输入框，send\_keys(filepath)
* 下载文件：配置浏览器，FirefoxProfile() , set\_preference("browser.download.dir", os.getcwd()) , browser = webdriver.Firefox(firefox\_profile=fp)
* 调用javascript：1.在页面上直接执行js；2.在某个已经定位的元素上执行js ； execute\_script(script, \*args)---在当前窗口/框架 同步执行 javaScript script:JavaScript 的执行。 \*args:适用任何 JavaScript 脚本。
* 浏览器滚动条处理：#将页面滚动条拖到底部；js="var q=document.documentElement.scrollTop=10000" driver.execute\_script(js)
* cookie处理：get\_cookies() 获得所有cookie信息； get\_cookie(name) 返回特定name有cookie信息；add\_cookie(cookie\_dict) 添加cookie，必须有name和value值；delete\_cookie(name) 删除特定(部分)的 cookie 信息；delete\_all\_cookies() 删除所有cookie信息。
* 获取对象属性：get\_attribute()
* 验证码问题：1）注释掉验证码部分代码；2）设置万能码；3）验证码识别OCR；4）登录前添加cookie，跳过登录验证。
* 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)，来转化浏览器的 native 调用。

## 自动化测试框架

* 线性测试：每个脚本独立，一条用例对应一个脚本，缺点：不便于维护
* 模块化类库，将相同操作、代码写成公共模块调用。
* 数据驱动：测试数据与脚本分离。
* 关键字驱动：

# 前端技术

## 浏览器内核

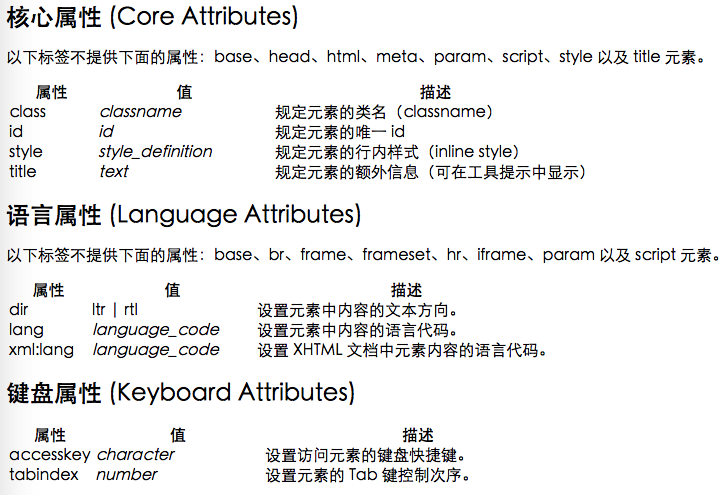
* Trident：IE浏览器使用的内核，该内核IE4沿用到目前的IE10
* Gecko：Netscape6开始采用的内核，后来的Mozilla FireFox也采用该内核
* Presto：目前Opera采用的内核，该内核在2003年的Opera7中首次被使用。
* Webkit：苹果公司自己的内核，也是苹果的Safari浏览器使用的内核，Google的chrome、及市面上号称极速浏览器使用的都是Webkit内核。Webkit内核在手机上的应用十分广泛，目前Android和IOS系统浏览器使用的都是该内核。

## Javascript引擎

* Mozilla：spiderMonkey用于Firefox1.0~3.0；TraceMonkey用于3.5~3.6；JaegerMonkey用于4.0以上版本。
* Google：V8，开放源代码
* 微软：Chakra查克拉，用于IE9的32位版本

## 结构化标准语言

* 网页由三部分组成：结构（structure）、表现（presentation）、行为（behavior）
* HTML（Hypertext Markup Language）--超文本标记语言，用于描述网页文档的一种标记语言。
* Head头部信息里的内容主要是被浏览器所用，不会显示在网页的正文内容里。Title在head中，显示浏览器窗口的标题栏里。
* Meta标记：提供关于网页的信息
* 基本的HTML标签：h1-h6 //标题；<p> //段落；<a> //链接；<img> //图片
* HTML元素指的是从开始标签到结束标签的所有代码<p>正文</p>；大多数HTML元素可拥有属性、可嵌套（包含其他元素）；标签元素使用小写。
* 元素的属性总是在HTML元素的开始标签中规定。
* 创建链接使用<a href=””>标签，href属性中输入#+id可以跳到指定的锚点位置。
* <img>图像标签



* HTML表格：<table>标签 tr—table row ; td—table data
* 列表：ol—有序列表；ul—无序列表；li---列表项；dl—定义列表；dd—定义的描述；
* HTML块元素，块级元素—block level element ；内联元素—inline element；块级元素在浏览器显示时，通常会以新行来开始结束，内联元素在显示时通常不会以新行开始。<div> <span>
* 可以使用<div> 或<table>元素来创建多列。CSS用于对元素进行定位，或者为页面创建背景以及色彩丰富的外观。
* 表单：是一个包含表单元素的区域，表单元素是允许用户在表单中（如文本域、下拉列表、单选框、复选框等等）输入信息的元素。<form>--定义供用户输入的表单；<input>--定义输入域；textarea—文本域（一个多行的输入控件）；label—定义一个控制的标签；select—选择列表；option—下拉列表中的选项；button—定义一个按钮；
* 文本域—text fields ；单选按钮—radio buttons ；复选框—checkboxes ；表单的动作属性action和确认按钮下拉框select。
* HTTP—hypertext transport protocol 一种详细规定了浏览器和万维网服务器中间互相通信的规则，通过因特网传送万维网文档的数据传送协议。
* 请求方法常用的有：get head post 传输文件类型由Content-Type加以标记。
* 无连接：每次连接只处理一个请求，完成后断开；无状态：协议对于事务处理没有无记忆，

# Mac OS

* 安装homebrew（相当于UNIX 中apt-get 在线安装软件命令）：获取最新下载地址，<https://brew.sh/index_zh-cn.html>
* 安装应用：brew install git / 卸载应用：brew uninstall git ;查找要安装的应用：brew search /wge\*/[正则表达式]；列出已安装的软件：brew list ；更新brew：brew update ；brew info；brew deps ，brew home

# git学习

## 问题

### Git pull origin 需要密码？

git和密码根本没什么关系。git基本不涉及账号管理，你使用什么协议访问git就是用哪个协议来认证

http ssh都可以，你git pull的时候 肯定是你远程访问协议的密码。

## 集中式（CVS、SVN）vs分布式（Git）

集中式版本控制：有一个中央服务器，每个人都从服务器取得最新版本，修改完成后再推送给服务器。且必须联网工作。

分布式版本控制：没有中央服务器，每个人都有一个完整的版本库，工作不需要联网，所谓的“中央服务器”作用只是方便大家交换自己的修改。还有极其强大的分支管理。

## 安装

**Linux系统安装git**：通过一行命令，sudo apt-get install git

**源码安装**：./config，make，sudo make install

**MacOs安装**：从AppStore安装Xcode，Xcode集成了Git，不过默认没有安装，你需要运行Xcode，选择菜单“Xcode”->“Preferences”，在弹出窗口中找到“Downloads”，选择“Command Line Tools”，点“Install”就可以完成安装了。

**Windows系统安装**：msysgit <https://git-for-windows.github.io/>

因为Git是分布式版本控制系统，所以，每个机器都必须自报家门：你的名字和Email地址。

$ git config --global user.name "Your Name"

$ git config --global user.email "email@example.com"

## 创建版本库repository

需要明确的是，所有的版本控制系统，其实只能跟踪文本文件的改动，比如TXT文件，网页，所有的程序代码等等，Git也不例外。版本控制系统可以告诉你每次的改动，但图片、视频这些二进制文件，虽然也能由版本控制系统管理，但没法跟踪文件的变化，只能把二进制文件每次改动串起来，也就是只知道图片从100KB改成了120KB，但到底改了啥，版本控制系统不知道，也没法知道。

Windows自带的记事本在每个文件开头添加了0xefbbbf（十六进制）的字符，会出现很多奇怪问题。建议使用notepad++ ，记得把Notepad++的默认编码设置为UTF-8 without BOM即可。

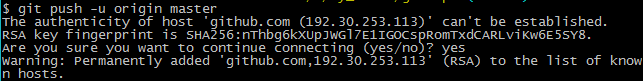
1. 新建一个空目录：mkdir repository
2. 在创建的目录下初始化使目录变成git可以管理的仓库：git init
3. 添加文件：git add <filename> //可一次添加多个文件；git commit -m “comments” //提交到仓库中。
4. 查看文件状态：git status ,要随时掌握工作区的状态，使用git status命令
5. 查看修改内容：git diff <filename> / git diff HEAD -- <filename> //查看工作区与版本库文件的区别
6. 提交修改和提交新文件是一样的两步:git add <filename> / git commit -m “comments”
7. 查看版本修改的历史记录：git log / git log --pretty=oneline //精简输出
8. 分布式版本管理，在多人操作时，很容易出现冲突，因此git会生成一个SHA1计算出来的16进制的“版本号”。
9. 在Git中，用HEAD表示当前版本HEAD^ 表示上个版本，HEAD~100表示上100个版本，回退到某个版本：git reset --hard HEAD^ / git reset --hard 3628164 //版本可以只写前几位。
10. Git记录每一次命令：git reflog //可通过显示的commit id回到当前版本。
11. **Git暂存区**：git有一个称为stage（或者叫index）的暂存区，还有Git为我们自动创建的第一个也是唯一一个分支master，以及指向master的一个指针叫HEAD。工作区即是实际目录。
12. 每次修改后，都需要add到暂存区，不然不会commit到版本库。
13. 使文件回到最近一次git commit或git add时的状态：git checkout -- filename ,一种是readme.txt自修改后还没有被放到暂存区，现在，撤销修改就回到和版本库一模一样的状态；

一种是readme.txt已经添加到暂存区后，又作了修改，现在，撤销修改就回到添加到暂存区后的状态。

1. 命令git reset HEAD file可以把暂存区的修改撤销掉（unstage），重新放回工作区；再丢弃工作区的修改：git checkout – filename
2. git rm fielname //从版本库删除文件。

## 远程仓库

在用户住目录下有一个.ssh目录，这个目录下有两个文件：id\_rsa , id\_rsa.pub 为SSH Key的秘钥对，id\_rsa是私钥，不能泄露出去，id\_rsa.pub是公钥，可以放心地告诉任何人。

* 创建SSH Key：ssh-keygen –t rsa –C [ldjwyyx@163.com](mailto:ldjwyyx@163.com)
* 第2步：登陆GitHub，打开“Account settings”—>“SSH Keys”🡪“Add SSH Key”，填上任意Title，在Key文本框里粘贴id\_rsa.pub文件的内容。
* 为什么GitHub需要SSH Key呢？因为GitHub需要识别出你推送的提交确实是你推送的，而不是别人冒充的，而Git支持SSH协议，所以，GitHub只要知道了你的公钥，就可以确认只有你自己才能推送。当然，GitHub允许你添加多个Key。假定你有若干电脑，你一会儿在公司提交，一会儿在家里提交，只要把每台电脑的Key都添加到GitHub，就可以在每台电脑上往GitHub推送了。最后友情提示，在GitHub上免费托管的Git仓库，任何人都可以看到喔（但只有你自己才能改）。所以，不要把敏感信息放进去。
* 关联远程仓库（使用SSH）：git remote add origin [git@github.com:gitgogo/hello-world.git](mailto:git@github.com:gitgogo/hello-world.git) Git支持多种协议，包括https，但通过ssh支持的原生git协议速度最快。
* 本地内容全部推送到远程库：git push -u origin master //第一次push需加-u ，若执行失败git pull origin master //将远程仓库的文件与本地合并后再推送至远程仓库。
* 克隆版本库：git clone [git@github.com:gitgogo/gitskills.git](mailto:git@github.com:gitgogo/gitskills.git)

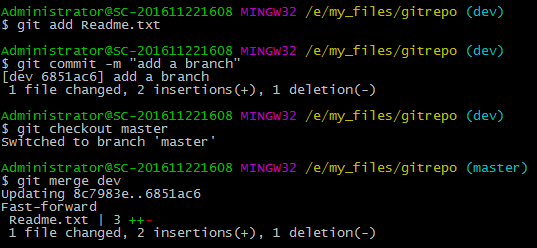
## 分支管理

一开始的时候，master分支是一条线，Git用master指向最新的提交，再用HEAD指向master，就能确定当前分支，以及当前分支的提交点；当我们创建新的分支，例如dev时，Git新建了一个指针叫dev，指向master相同的提交，再把HEAD指向dev，就表示当前分支在dev上。

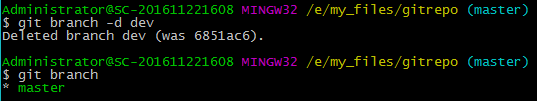
* **创建分支dev**：git checkout –b dev //相当于两条命令：git branch dev git checkout dev

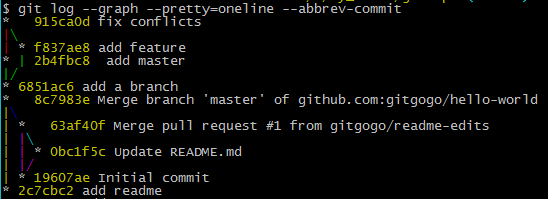


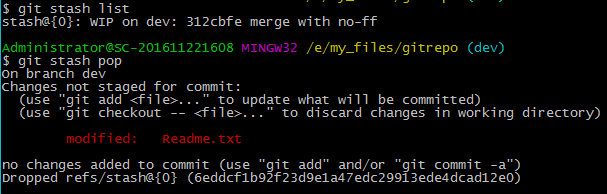
* 编辑本地文件，提交，切换至master分支，合并分支：git merge dev



* **删除分支**：git branch –d dev

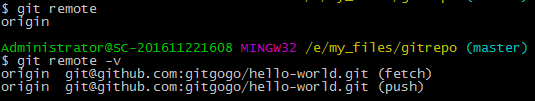


* **解决冲突**：当Git无法自动合并分支时，就必须首先解决冲突。解决冲突后（直接查看被修改文件的内容，git自动标识冲突的地方），再提交，合并完成。可查看合并log：git log --graph --pretty=oneline --abbrev-commit
* 禁用fast-forward模式：通常，合并分支时，如果可能，Git会用Fast forward模式，但这种模式下，删除分支后，会丢掉分支信息。git merge –no-ff –m “merge with no-ff” dev
* **分支策略**：master分支应该是非常稳定的，也就是仅用来发布新版本，平时不能在上面干活；干活都在dev分支上，也就是说，dev分支是不稳定的，到某个时候，比如1.0版本发布时，再把dev分支合并到master上，在master分支发布1.0版本；你和你的小伙伴们每个人都在dev分支上干活，每个人都有自己的分支，时不时地往dev分支上合并就可以了。
* **Bug分支：**当你正在自己的分支dev上开发东西时，需要临时去解决主线上的bug，首先git stash //把dev的现场保留，便于bug fix后继续开发；切换到master，git checkout master🡪git checkout –b issu-112 //新建bug分支；修复完成后提交至主线；删除issu-112分支；切换到dev分支🡪git checkout dev；查看stash列表🡪git stash list；还原dev🡪git stash pop；继续开发。 可多次stash，恢复指定的stash：git stash apply stash@{0}



* 用git stash apply恢复，需要git stash drop删除
* 开发新feature最好新建分支feature-xxx，当feature开发停止时，git branch –D feature-xxx

## 多人协作：

* 查看远程版本库：git remote -v
* **推送分支**：git push origin master //推送master分支到远程版本库；git push origin dev //推送dev分支；**master分支**是主分支，因此要时刻与远程同步；

**dev分支**是开发分支，团队所有成员都需要在上面工作，所以也需要与远程同步；

**bug分支**只用于在本地修复bug，就没必要推到远程了，除非老板要看看你每周到底修复了几个bug；

**feature分支**是否推到远程，取决于你是否和你的小伙伴合作在上面开发。

* **多人协作工作模式**：1）首先，可以试图用git push origin branch-name推送自己的修改；2）如果推送失败，则因为远程分支比你的本地更新，需要先用git pull试图合并；3）如果合并有冲突，则解决冲突，并在本地提交；4）冲突解决掉后，再用git push origin branch-name推送就能成功！5）如果git pull提示“no tracking information”，则说明本地分支和远程分支的链接关系没有创建，用命令git branch --set-upstream branch-name origin/branch-name

## 标签管理

tag就是一个让人容易记住的有意义的名字，它跟某个commit绑在一起。

* 切换到需要打标签的分支；git tag <tag\_name>，git tag //查看所有tag
* 默认标签是打在最新提交的commit上的，在之前的commit上打标签：git tag v1.0 commit\_id
* Git show <tag\_name> //查看标签信息；$ git tag -a v0.1 -m "version 0.1 released" 3628164 //-a指定标签名，-m指定说明文字
* 还可以通过-s用私钥签名一个标签：$ git tag -s v0.2 -m "signed version 0.2 released" fec145a，签名采用PGP签名，因此，必须首先安装gpg（GnuPG），如果没有找到gpg，或者没有gpg密钥对，就会报错。
* 删除标签：git tag –d v1.0
* 推送某个标签到远程，使用命令git push origin <tagname> ，一次性推送全部尚未推送到远程的本地标签：git push origin –tags
* 要删除远程标签：先删除本地标签 git tag –d v1.0 ；再删除远程标签：git push origin :refs/tags/v1.0
* 在GitHub上，可以任意Fork开源仓库； 自己拥有Fork后的仓库的读写权限； 可以推送pull request给官方仓库来贡献代码。

## 自定义git

* 显示颜色：git config –global color.ui true
* 忽略特殊文件：新建 .gitignore 文件，配置需要忽略的文件格式。

忽略文件的原则是：

* 忽略操作系统自动生成的文件，比如缩略图等；
* 忽略编译生成的中间文件、可执行文件等，也就是如果一个文件是通过另一个文件自动生成的，那自动生成的文件就没必要放进版本库，比如Java编译产生的.class文件；
* 忽略你自己的带有敏感信息的配置文件，比如存放口令的配置文件。

*# Windows:*

*Thumbs.db*

*ehthumbs.db*

*Desktop.ini*

*# Python:*

*\*.py[cod]*

*\*.so*

*\*.egg*

*\*.egg-info*

*dist*

*build*

*# My configurations:*

*db.ini*

*deploy\_key\_rsa*

* 可以对忽略的文件强制添加：git add –f file\_name ；检查.gitignore文件的配置详情：git check-ignore –v file\_name
* 配置别名：git config –global alias co checkout；常用别名：ci—commit；co—checkout；br—branch；unstage—‘reset HEAD’ ；**--global**参数是全局参数，也就是这些命令在这台电脑的所有Git仓库下都有用。git config --global alias.last 'log -1' //显示最近一次的提交
* git config --global alias.lg "log --color --graph --pretty=format:'%Cred%h%Creset -%C(yellow)%d%Creset %s %Cgreen(%cr) %C(bold blue)<%an>%Creset' --abbrev-commit"

## 常用命令

### 新建/克隆代码库

$ git init #当前目录新建一个Git代码库

$ git init [project-name] #新建一个目录，将其初始化为Git代码库

$ git clone [url] #下载一个项目和它的整个代码历史

$ git fetch [url] #下载/同步项目到

### 添加/删除文件

$ git add [file1] [file2] ... # 添加指定文件到暂存区

$ git add [dir] # 添加指定目录到暂存区，包括子目录

$ git add . # 添加当前目录的所有文件到暂存区

$ git rm [file1] [file2] ... # 删除工作区文件，并且将这次删除放入暂存区

$ git rm --cached [file] # 停止追踪指定文件，但该文件会保留在工作区

$ git mv [file-original] [file-renamed]# 改名文件，并且将这个改名放入暂存区

### 代码提交

$ git commit -m [message] # 提交暂存区所有文件到仓库区，并指定提交说明

$ git commit [file1] [file2] ... -m [message] # 提交暂存区的指定文件到仓库区，并指定提交说明

$ git commit -a # 提交工作区自上次commit之后的变化，直接到仓库区。是git add 和 git commit的组合操作

$ git commit -v # 提交时显示所有diff信息

$ git commit --amend -m [message] # 使用一次新的commit，替代上一次提交

### 分支

$ git branch # 列出所有本地分支

$ git branch -r # 列出所有远程分支

$ git branch -a # 列出所有本地分支和远程分支

$ git branch [branch-name] # 新建一个分支，但依然停留在当前分支

$ git checkout -b [branch] # 新建一个分支，并切换到该分支

$ git branch [branch] [commit] # 新建一个分支，指向指定commit

$ git checkout [branch-name] # 切换到指定分支

$ git merge [branch] # 合并指定分支到当前分支

$ git branch -d [branch-name] # 删除本地分支

$ git push origin --delete [branch-name] # 方法一：删除远程分支

$ git branch -dr [remote/branch] # 方法二：删除远程分支

### 撤销

$ git checkout [file] # 恢复暂存区的指定文件到工作区（注意区别分支操作中得checkout命令）

$ git checkout [commit] [file] # 恢复某个commit的指定文件到暂存区和工作区

$ git checkout . # 恢复暂存区的所有文件到工作区

$ git reset [file] # 重置暂存区的指定文件，与最新的commit保持一致，但工作区不变

$ git reset --hard # 重置暂存区与工作区，与最新的commit保持一致

$ git reset [commit] # 重置当前分支的指针为指定commit，同时重置暂存区，但工作区不变

$ git reset --hard [commit] # 重置当前分支的HEAD为指定commit，同时重置暂存区和工作区，与指定commit一致

$ git reset --keep [commit]# 重置当前HEAD为指定commit，但保持暂存区和工作区不变

$ git revert [commit] # 新建一个commit，用来撤销指定commit

### 标签

$ git tag # 列出所有tag

$ git tag [tag] # 在当前commit新建一个tag

$ git tag [tag] [commit] # 在指定commit新建一个tag

$ git tag -d [tag] # 删除本地tag

$ git push origin :refs/tags/[tagName] # 删除远程tag

$ git show [tag] # 查看tag信息

$ git push [remote] [tag] # 提交指定tag

$ git push [remote] --tags # 提交所有tag

$ git checkout -b [branch] [tag] # 新建一个分支，指向某个tag

### 查看日志

$ git status # 显示所有变更文件

$ git log # 显示当前分支的版本历史

$ git log --stat # 显示当前分支的版本历史，以及发生变更的文件

$ git blame [file] # 显示指定文件是什么人在什么时间修改过

$ git log -p [file] # 显示指定文件相关的每一次diff

$ git diff # 显示暂存区和工作区的差异

$ git diff --cached [commit] # 显示暂存区和某个commit的差异

$ git diff HEAD # 显示工作区与当前分支最新commit之间的差异

$ git show [commit] # 显示某次提交的元数据和内容变化

$ git show --name-only [commit] # 显示某次提交发生变化的文件

$ git show [commit]:[filename] # 显示某次提交时，某个文件的内容

$ git reflog # 显示当前分支的最近几次提交

### 远程同步

$ git fetch [remote] # 下载远程仓库的所有变动到暂存区

$ git remote -v # 显示所有远程仓库

$ git remote show [remote] # 显示某个远程仓库的信息

$ git remote add [shortname] [url] # 增加一个新的远程仓库，并命名

$ git remote set-url [shortname] [url] # 设置远程仓库的url地址

$ git pull [remote] [branch] # 取回远程仓库的变化，并与本地分支合并

$ git push [remote] [branch] # 上传本地指定分支到远程仓库

$ git push [remote] --force # 即使有冲突，强行推送当前分支到远程仓库

$ git push [remote] --all # 推送所有分支到远程仓库

### 设置

git的配置文件是.gitconfig，支持全局配置和项目配置，全部配置对所有项目有效，用 **--global**选择指定。

$ git config --list #显示配置

$ git config -e [--global] #编辑(全局)配置文件

$ git config [--global] user.name "xx" #设置 commit 的用户

$ git config [--global] user.email "xx@xx.com" #设置 commit 的邮箱

# Jenkins入门

## 1.1概述

Jenkins是一个可扩展的持续集成引擎。用java开发的，但不仅限于构建基于java的软件，它在整个开发过程中的主要作用是控制。主要用于：

* 持续、自动的构建/测试软件项目
* 监控一些定时执行的任务

## 1.2特性

* 易于安装：只要把jenkins.war部署到servlet容器，不需要数据库支持
* 易于配置：所有配置通过web界面实现
* 集成RSS/E-mail通过RSS发布构建结果或构建完成时邮件通知
* 可生成JUnit/Testng测试报告
* 支持分布式构建/测试
* 可跟踪每次构建时的jar文件
* 支持扩展插件

## 1.3使用场景

* 软件构建自动化
* 构建可持续的自动化检查：检查代码
* 构建可持续的自动化测试
* 生成后续过程的自动化
* 部署一个持续集成系统的需要的最低要求是，一个可获取的源代码的仓库、一个包含构建脚本的项目。

## 2安装

* 下载新版本（一个war文件）<http://Jenkins-ci.org/>
* 运行java –jar jenkins.war

另一种安装方式：将下载的war包文件部署到servlet容器，启动容器，在浏览器中输入类似<http://localhost:8080/jenkins/> 即可（Linux+tomcat+java环境）

*用tomcat打开，解压tomcat到某个目录,如/usr/local，进入tomcat下的/bin目录，启动tomcat，将jenkins.war文件放入tomcat下的webapps目录下，启动jenkins时，会自动在webapps目录下建立jenkins目录，所以在地址栏上需要输入的地址于上一种方法有点不一样。*

## 3.配置

### 3.1系统管理：

* 1.utf-8编码—jenkins建议在tomcat中使用utf-8编码【tomcat下conf目录的server.xml文件，若Job的控制台中文输出乱码，请将URIEncoding=”utf-8”更改为useBodyEncodingForURI="true"】
* 提示更新—下载更新版本
* 安全设置
* 系统设置：（web界面）；jdk、maven、ant配置（新增实例）；邮件通知配配置【jenkins URL、system admin e-mailaddress两项都要配置】；subversion配置；
* configure global security（安全配置）【管理用户权限】
* 管理用户设置【注册账号、登录】
* 管理插件设置

### 3.2项目构建设置

* 在web页面新建job
* 构建任务配置：源码管理配置（subversion）；构建触发器【构建间隔时间】；ant构建配置【invoke ant，ant脚本实现编译打包】
* 构建maven风格的job：新建maven构建任务

# 接口测试

接口分类：get 、post、put、delete

get与post的区别：1.get提交信息有大小限制，以参数形式传值；2.get的信息在URL中，相对不安全；post请求的信息在form表单中。

## 功能测试

功能实现，参数正确

## 逻辑测试

某些操作中的逻辑依赖【如我的订单—依赖用户登录】

## 异常测试用例：

* 参数异常：关键字参数、参数为空、多/少参数、错误参数
* 数据异常：关键字数据、数据位空、数据长度限制、错误数据

## 安全测试用例

cookie【验证登陆状态】、header、唯一识别码 【】

## 接口测试工具

接口抓取、接口测试工具

### fiddler

* 查看接口请求方式、查看请求数据、接口返回状态
* 设置代理（抓取移动端数据tools-option-connection—allow remote—端口8888）
* 设置断点：rules—automatic breakpoints—before requests
* 添加筛选：filter【右侧框】
* 查看header、cookie
* 进行接口测试：composer【右侧框】

## postman

可自动反复测试一个需要携带header的post请求，并且针对线上、测试环境同一个数据进行测试，并对实际结果做比较以作为测试结果。可学习代码code

## Python实现

### #get请求

#coding=urf-8

import urllib ,urllib2

url=’http://xxxxxx’

data={}

data[‘loginname’]=’liudongjie’

data[‘password’]=’ 670b14728ad9902aecba32e22fa4f6bd’ #000000🡪md5加密后的结果

data=urllib.urlencode(data)

request=url+’?’+data

response=urlib2.urlopen(request)

rsp\_str=respose.read().decode(‘unicode\_escape’) #若未打印中文字符，可做转码处理。

print rsp\_str

### #post请求

import urllib,urllib2

url=’http://xxxx’

data={}

headers={}

data[‘xx’]=’xx’

req=urllib2.Request(url,data=data,header=headers)

response=urllib2.urlopen(req)

rsp\_str=response.read().decode(‘unicode\_escape’)

print rsp

# loadrunner

## 组件

* Virtual User Generator: 录制最终用户业务流程并创建自动化性能测试脚本，即 Vuser 脚本
* Controller 组织、驱动、管理并监控负载测试，设置并发数量
* Load Generator 通过运行 Vuser 产生负载
* Analysis 用于查看、剖析和比较性能结果
* Launcher 使您可以从单个访问点访问所有 LoadRunner 组件

## 术语

* **场景**文件根据性能要求定义每次测试期间发生的事件。
* 在场景中， LoadRunner 用虚拟用户 （或称 Vuser）代替真实用户。 **Vuser** 模仿真实用户的操作来使用应用系统。一个场景可以包含数十、数百乃至数千个 Vuser。
* Vuser 脚本描述 Vuser 在场景中执行的操作。
* 要评测服务器性能，需要定义事务。**事务**代表要评测的终端用户业务流程。

# Apache

[Apache原理完整介绍](http://blog.csdn.net/hguisu/article/details/7395181)

# Jmeter性能测试

## [下载链接：](http://jmeter.apache.org/download_jmeter.cgi?Preferred=http%3A%2F%2Fmirrors.tuna.tsinghua.edu.cn%2Fapache%2F) （下载source版）

## [详细完整教程](https://wuyan.gitbooks.io/jmeter/content/index.html)

下载完成后bin目录下双击文件ApacheJMeter.jar ，启动jmeter图形界面。

## 建立测试计划

打开JMeter时，它已经建立一个默认的测试计划,一个JMeter应用的实例只能建立或者打开一个测试计划。

## 增加负载信息

* QPS : Query Per Second 每秒查询率。是一台查询服务器每秒能够处理的查询次数。在因特网上，作为域名系统服务器的机器的性能经常用每秒查询率来衡量。
* TPS：Transaction Per Second每秒事物数，性能指标的一个重要因素。单位时间内完成的事物的数量，也叫吞吐量；TPS的计算一般是通过的事物除以时间。
* 吞吐量和并发用户数之前存在着一定的联系。在没有性能瓶颈的时候，吞吐量随着虚拟用户数的增加而增加(计算公式为 吞吐量 = (VU个数 \* 每个VU发出请求数) / 单位时间)。如果性能遇到瓶颈，吞吐量与VU数理之间就不再符合这个关系。
* 响应时间主要包括网络时间，服务器处理时间，网络延迟；对于交互式应用，用户直接的体验就是“响应时间”，通过“并发用户数”和“响应时间”可以确定系统的性能规划;对于交互式应用，响应时间出现拐点系统就可能出现瓶颈
* 随着用户数量的增加，在未出现瓶颈前响应时间保持稳定，TPS值和并发用户数成线性关系，出现瓶颈后响应时间变长，TPS基本保持不变或开始下降。

建模就是建立模型，就是为了理解事物而对事物做出的一种抽象，是对事物的一种无歧义的书面描述，用模型描述系统的因果关系或相互关系的过程。

### 线程组（属性）

jmeter的所有任务都必须由线程处理，所有任务都必须在线程组下面创建。

jmeter中 每个测试计划至少需要包含一个线程组，当然也可以在一个计划中创建多个线程组，在测试计划下面多个线程是并行执行的，也就是说这些线程组是同时被初始化并同时执行线程组下的Sampler的。

测试计划右键添加—Threads（users）--线程组

线程数：发送请求的用户数目，虚拟用户数，一个虚拟用户占用一个线程或者进程

ramp-up period：准备时长，设置虚拟用户数需多长时间全部启动，如果线程数20，准备时长10，那么需要10s启动20个线程，也就是每秒钟启动2个线程。

循环次数：每个线程发送请求的次数。如果线程数20，循环次数100，则每个线程发送100次请求，总请求数20\*100

启动时间：测试计划的启动时间【如果你设置了开始测试时间，就算是你在Jmeter上运行了测试，如果没有到你设置的开始时间，jmeter是不会执行测试的，直到到了你设置的开始时间，才会开始执行测试】

结束时间：测试计划的结束时间【如果你设置了结束时间，一旦到了结束时间，不管当前执行的循环有没有结束，jmeter都会停止执行】

### 取样器

取样器sampler是与服务器进行交互的单元，一个取样器通常进行三部分的工作：向服务器发送请求、记录服务器的响应数据、记录相应时间信息。

**协议**：默认http ，可以使https

**服务器名称或ip**：http请求发送的目标服务器名称或ip地址

**端口号**：目标服务器的端口号，默认80

**方法**：发送http请求的方法，可用方法GET POST HEAD PUT DELETE OPTIONS TRACE

**content encoding**：内容编码方式，默认iso8859

**路径**：目标URL路径（不包括服务器地址和端口）

**自动重定向**：如果选中，当发送http请求后的响应是301/302 ，自动重定向到新的页面。

**Use keep alive：**选中时，jmeter和目标服务器之间使用keep-alive方式进行通信，默认选中

**同请求一起发送参数**：参数化，表中每一行是一个参数值对

**Save response as MD5 hash?** ：选中该项，在执行时仅记录服务端响应数据的MD5值，而不记录完整的响应数据。在需要进行数据量非常大的测试时，建议选中该项以减少取样器记录响应数据的开销。

**从HTML文件获取所有有内含的资源**：当该选项被选中时，jmeter在发出HTTP请求并获得响应的HTML文件内容后，还对该HTML进行Parse 并获取HTML中包含的所有资源（图片、flash等），默认不选中，如果用户只希望获取页面中的特定资源，可以在下方的Embedded URLs must match 文本框中填入需要下载的特定资源表达式，这样，只有能匹配指定正则表达式的URL指向资源会被下载。

### 监听器：

监听范围：当前节点层以及当前节点的子节点（直至叶子节点），即监听器可以监听当前层以及比它更低的层。

常用监听器：察看结果树、聚合报告、图形结果、用表格查看结果

**查看结果树：**

显示取样器请求和响应的细节，包括消息头、请求的数据、响应的数据（如上图）

* 取样器结果：显示的是取样器相关参数（客户端参数与响应参数）
* 请求：发送请求的具体值
* 响应数据：服务器返回的相应参数

**聚合报告：**

* Label： 取样器名称
* Samples ：运行过程中一共发出了多少个请求，一般来说是其值 = 线程数 \* 循环次数（前提：测试按计划跑完，其中没有异常结束的线程）。总体samples是各请求samples之和。
* Average：平均相应时间
* Median：响应时间中间值
* 90%Line:响应时间90%线 （即90%的用户处于这个值）
* Min/Max：最小/最大的响应时间
* Error%：出错率（出错的Request数/所有的Request数）
* Throughput：吞吐量，每秒/每分钟（具体看“/”后面的单位）处理的Request数
* KB/sec：每秒从服务器接收到的数据

**图形报告：**

* 样本数目：发出去的请求数
* 最新样本：最近一个Request的响应时间
* 平均：所有Request响应时间的平均值
* 偏离：所有Request响应时间的偏差
* 吞吐量;单位时间内（一般是1分钟）处理Request的个数
* 中值：所有Request响应时间的中间值

### 正则表达式提取器

* 提取器作用：从请求的响应结果中取到需要的内容，从而实现关联
* 要检查的响应字段：正则表达式提取内容的范围
* 引用名称：其他地方引用提取值得变量名称，具体的引用方式是${变量名}，eg:${memberID}
* 正则表示式：提取内容的正则表达式，（）表示提取，对于你要提取的内容需要用小括号括起来
* 模板：用$$引用起来，如果在正则表达式中有多个提取式（多个括号括起来的东西），则可以是$1$、$2$等等，表示解析到的第几个值给变量，正则表达式的提取模式，值从1开始，值0对应的是整个匹配的表达式。如对于表达式s(.*)值0对应 s(.*)，值1对应(.\*)
* 匹配数字：0代表随机，-1代表所有，其余正整数代表将在已提取的内容中，第几个匹配的内容。
* 缺省值：正则匹配失败时取的值。

### 测试树作用域

jmeter测试树包含元件总是分等级和顺序的。在测试树中的一些元件是严格分级（监听器、配置元素、后置处理器、前置处理器、断言、定时器），一些主要是顺序的（控制器，取样器）。当你创建测试计划时，你将创建一个描述被执行的步骤集的取样请求有序列表。那些请求常组织在有序的控制器中。

### 调整QPS值

一种可行的方法是逐步调整测试计划中的线程计算的数量以及为取样器（Sampler）添加定时器（Timer），以使HTTP取样器发出的请求的QPS保持在20个左右。但这种方法耗时耗力，需要经过多次尝试才能达到；另一方法，完全通过设置定时器来控制QPS，一旦取样器的响应时间发生改变（网络环境发生改变），就需要重新调整定时器的等待时间。

**定时器**：Constant Throughput Timer 常数吞吐量定时器，可以方便地控制给定的取样器发送请求的吞吐量。

Target throughput（in samples per minute）：目标吞吐量。注意这里是每分钟发送的请求数，因此，对应测试需求中所要求的20 QPS ，这里的值应该是1200 。

Constant Throughput Timer只有在线程组中的线程产生足够多的request 的情况下才有意义，因此，即使设置了Constant Throughput Timer的值，也可能由于线程组中的线程数量不够，或是定时器设置不合理等原因导致总体的QPS不能达到预期目标。

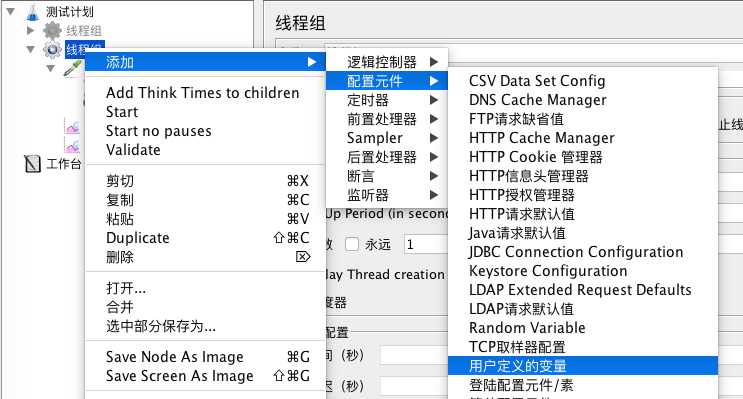
### 添加监听器

Jmeter 中使用监听器元件收集取样器记录的数据并以可视化的方式来呈现。

线程组—添加—监听器—聚合报告

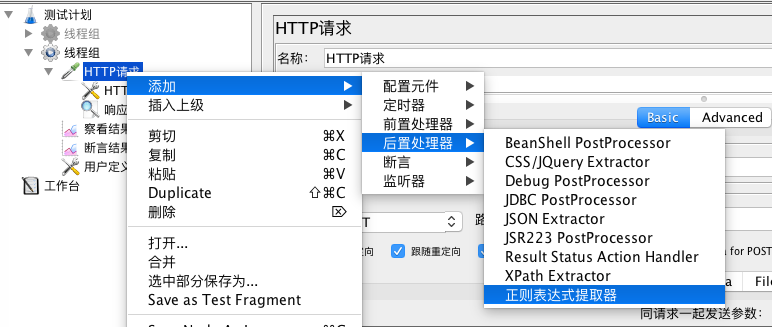
### http请求

* 添加线程组—添加—sampler—http请求
* http请求—添加—配置元件—http信息头管理器
* http请求—添加—断言—响应断言
* 线程组—添加—监听器—断言结果；断言会影响作用域的所有采样器。如果要让断言只影响某个采样器，需要将断言作为该采样器的子项
* 线程组—添加—监听器—查看结果树
* 添加用户自定义变量：在http请求中使用 path:/weather2d/**${city}**.shtml 或者/search?cityname=${city}



### 关联

关联， 就是第二个Requst, 使用第一个Request中的数据，使用正则表达式提取结果中的值传给第二个http请求的参数。



## 参数化：

参数化数据，用记事本写参数并保存为.dat格式的文件，编码问题在使用CSV Data Set Config参数化时要求比较严格，记事本另存为修改编码为UTF-8，注意参数要一一对应，中间用逗号隔开（,）

### 两种形式

借助函数助手：

* 点击菜单栏“选项”---->函数助手对话框，如下图（CSV文件列号是从0开始的，第一列0，第二列1，第三列2...）



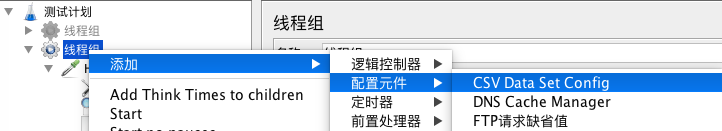
* 复制生成的参数化函数，参数化我们需要参数化的字段，这里是对id和pageIndex参数化，id的列号为0，pageIndex的列号为1

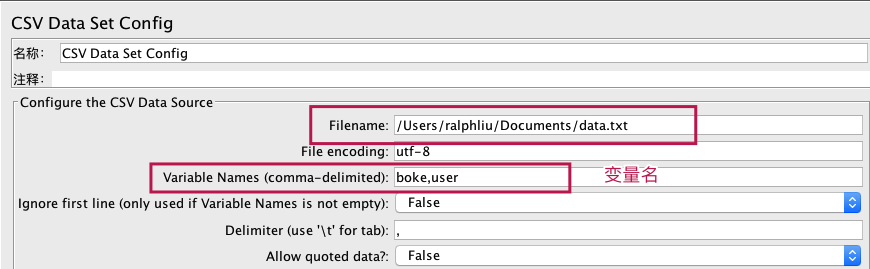


借助jmeter中配置元件CSV Set Config：

## 压测数据准备

添加CSV Set Config





设置线程组，模拟用户数。

* Filename --- 参数项文件
* File Encoding --- 文件的编码，设置为UTF-8
* Vaiable Names --- 文件中各列所表示的参数项；各参数项之间利用逗号分隔；参数项的名称应该与HTTP Request中的参数项一致。
* Delimiter --- 如文件中使用的是逗号分隔，则填写逗号；如使用的是TAB，则填写\t；(如果此文本文件为CSV格式的，默认用英文逗号分隔)
* Recycle on EOF? --- True=当读取文件到结尾时，再重头读取文件
* False=当读取文件到结尾时，停止读取文件
* Stop thread on EOF? --- 当Recycle on EOF为False时，当读取文件到结尾时，停止进程，当Recycle on EOF为True时，此项无意义

## Jmeter录制脚本

方法一：[badboy](http://www.badboy.com.au/download)本身是一个针对Web的测试工具，它录制的脚本可以直接供Jmeter使用

方法二：把Jmeter配置成一个Web代理，用Jmter自己来录制脚本。

第一步： 创建一个Thread Group (邮件点击： Test Plan -> Add -> Thread Group)

第二步： 创建http 代理服务器 （邮件点击"工作台"（WorkBench）, Add-> Non-Test Elements -> HTTP(S) Test Script Recorder）

## 接口测试

### 功能测试

* 所有的接口参数都必须经过加密以及压缩后再传给服务器，加密和压缩的过程怎么处理？
  + 所有参数经过加密和压缩之后再传给jmeter
  + 直接改写jmeter源码，传参的加密压缩解密都交给jmeter处理，我们只需传正常参数即可

这里我们采取第二种（需要导入jmeter源码，添加加密压缩请求，重新打包）

### 自动化用例分层

* CGI:单次接口请求，是最小粒度的操作
* Function：是一个有逻辑意义的功能，可能需要多个CGI共同完成，相当于做了一层封装，为后面构建Testcase打下基础
* TestCase：包含了本系统和其他系统的一些Fucnction步骤



## MySQL性能测试

添加线程组==》添加JDBC Connection Configuration==》添加JDBC Request==》添加监听器【结果树】==》运行

将mysql-connector-java-3.1.14-bin.jar的jar包加入C:\Program Files\Java\jre1.8.0\_02\lib\ext文件夹下



# 提高网站性能

## 硬件方面

更多的服务器、高性能的CPU，内存，带宽

## web服务器-负载均衡

负载平衡（Load balancing）是一种计算机网络技术，用来在多个计算机（计算机集群）、网络连接、CPU、磁盘驱动器或其他资源中分配负载，以达到最优化资源使用、最大化吞吐率、最小化响应时间、同时避免过载的目的。

负载平衡最重要的一个应用是利用多台服务器提供单一服务，这种方案有时也称之为服务器农场。通常，负载平衡主要应用于Web网站，大型的Internet Relay Chat网络，高流量的文件下载网站，NNTP（Network News Transfer Protocol）服务和DNS服务。现在负载平衡器也开始支持数据库服务，称之为数据库负载平衡器。

算法：当前，负载均衡器有各种各样的工作调度算法（用于决定将前端用户请求发送到哪一个后台服务器），最简单的是随机选择和轮询。更为高级的负载均衡器会考虑其它更多的相关因素，如后台服务器的负载，响应时间，运行状态，活动连接数，地理位置，处理能力，或最近分配的流量。

高性能系统通常会使用多层负载均衡。另外专用的硬件负载均衡器以及纯软件负载均衡器，包括开源的，例如Apache Web服务器的mod proxy balancer扩展，nginx，Varnish和Pound反向代理和负载均衡器。使用Gearman将合适的计算任务分发给多台计算机，如此大量的任务就可以更快的完成了。

## Web服务器-CDN

内容分发网络（Content delivery network或Content distribution network，缩写：CDN）是指一种通过互联网互相连接的电脑网络系统，利用最靠近每位用户的服务器，更快、更可靠地将音乐、图片、视频、应用程序及其他文件发送给用户，来提供高性能、可扩展性及低成本的网络内容传递给用户.

内容分发网络另外一个好处在于有**异地备援**。当某个服务器故障时，系统将会调用其他邻近地区的服务器服务，进而提供接近100%的可靠度。除此之外，内容分发网络提供给服务提供者更多的控制权。提供服务的人可以针对客户、地区，或是其他因子调整。**服务器的工作方式一般是基于nginx的模式**，也就是**仅仅缓存网站的静态内容**，不过，随着今年来世界各地CDN服务商纷纷推出HTTPS加速功能，工作方式也变得略有不同，变成了nginx+SNI（一个CDN节点上可以借此技术绑定N个域名），同时，BGP anycast技术也逐渐引入了CDN领域中。

以 Yahoo! 为例，他们把静态内容从应用程序中移到 CDN（上面提到 CDN 服务提供商，以及他们自己的 CDN）上，提高了最终用户 20％ 以上的响应时间。使用 CDN 是一个只需要相对简单地修改代码，显著改善站点速度的方法。

## Web服务器-独立的图片服务器

将图片与页面进行分离，这是基本上大型网站都会采用的策略，他们都有独立的、甚至很多台的图片服务器。这样的架构可以降低提供页面访问请求的服务器系统压力，并且可以保证系统不会因为图片问题而崩溃。

## Web服务器-Gzip压缩

在HTTP协议中，http压缩其实是内容编码的一种。

HTTP压缩是指: Web服务器和浏览器之间压缩传输的”文本内容“的方法。 HTTP采用通用的压缩算法，比如gzip来压缩HTML,Javascript, CSS文件。 能大大减少网络传输的数据量，提高了用户显示网页的速度。当然，同时会增加一点点服务器的开销。

浏览器发送请求，携带信息Accept-Encoding:Gzip,告诉服务器可以支持Gzip压缩格式，==》服务器接收请求后，生成原始的响应数据，并对其Gzip压缩，返回给浏览器==》浏览器接收到服务器的响应后，对原始数据解码decode==》展示内容。编解码在浏览器和服务器中处理，传输的数据始终是编码（压缩）后的数据，从而可提高响应速度。

## Web服务器-缓存

### 直接本地缓存

直接使用浏览器端的Cache，不会发出真正的request，直到Cache过期；最省流量和服务器资源，但可能存在数据不一致，适合实时性不高的场景。

Header信息：Cache-Control: public, max-age=31536000

* public : 用户浏览器和中间proxy都会cache
* private : 只有用户端会cache
* max-age=xxx : 设置cache的最大存活时间，单位s
* no-store: 不要缓存
* no-cache: 同上，但是浏览器的支持可能不一致，最好两个一起用
* Expires: Mon, 19 Nov 2012 08:40:01 GMT   
  指定cache的*绝对*过期时间，和Cache-Control一起使用时以后者为准

### 304校验缓存

将浏览器端的Cache信息发送到服务器校验，如果服务器认为依然有效，则返回304，浏览器使用Cache；会发出一次请求且需要服务器校验，没有第一种方式那么有效，但可以保证数据实时性。

校验的两种方式：

* 基于内容ETag，ETag是Entity Tag的缩写，它由服务器端指定，通常是一个资源的内容的hash。

服务器返回给浏览器：Etag: 427fe7b6442f2096dff4f92339305444

浏览器请求传回给服务器：If-None-Match: 427fe7b6442f2096dff4f92339305444

服务器收到请求将两个值对比，若不一致，返回新数据，否则使用缓存

* 基于最后修改时间：Last-Modified

Request将本地缓存的最后修改时间通过 If-Modified-Since 发送到服务器

## 数据库

使用存储过程

使用数据库缓存

尽早的关闭数据库连接

## 优化图片

图片格式，压缩，不同的算法处理图片的显示

# 关于URL编码

RFC 1738规定只有字母和数字[0-9a-zA-Z]、一些特殊符号"$-\_.+!\*'(),"[不包括双引号]、以及某些保留字，才可以不经过编码直接用于URL。RFC 1738没有规定具体的编码方法，而是交给应用程序（浏览器）自己决定。

因此URL中包含汉字的需要编码后才能使用：

## 不同的编码格式

1. 网址路径中包含汉字，http://zh.wikipedia.org/wiki/春节
   1. 统一使用utf-8编码
2. 查询字符串包含汉字，<http://www.baidu.com/s?wd=春节>
   1. 使用操作系统的默认编码格式（Windows中文版使用gb2312）
3. get、post方法的编码使用网页的编码。<meta http-equiv="Content-Type" content="text/html;charset=xxxx">
4. 在Ajax调用中，IE总是采用GB2312编码（操作系统的默认编码），而Firefox总是采用utf-8编码。

## 统一编码

可使用JavaScript的方法统一编码后再发送到服务器处理。

1. escape函数
2. encodeURI函数
3. encodeURIComponent函数 / 对应decodeURIComponent