[Linux笔记](#_Toc516583993)

[Git笔记](#_Toc516583994)

[Mysql笔记](#_Toc516583995)

[MongoDB笔记](#_Toc516583996)

[Tomcat笔记](#_Toc516583997)

[Nginx笔记](#_Toc516583998)

[NPM笔记](#_Toc516583999)

[Bower笔记](#_Toc516584000)

[Gulp笔记](#_Toc516584001)

[Gradle笔记](#_Toc516584002)

[JSON笔记](#_Toc516584003)

# Linux笔记

1. 指令

|  |  |  |
| --- | --- | --- |
| sudo | 以系统管理者的身份执行指令 |  |
| systemctl poweroff | 系统关机 |  |
| systemctl reboot | 重新启动 |  |
| systemctl suspend | 进入暂停模式 |  |
| systemctl hibernate | 进入休眠模式 |  |
| systemctl rescue | 强制进入救援模式 |  |
| systemctl emergency | 强制进入紧急救援模式 |  |
| systemctl set-default graphical.target |  |  |
| ps –l | 仅观察自己的 bash 相关进程： |  |
| ps aux | 观察系统所有进程： |  |
| ps -ef | grep java | 查看Tomcat是否以关闭 |  |
| cd .. | 返回上级目录 |  |
| yum search vim | 查找源中的VIM包 |  |
| yum -y install vim-enhanced | 安装vim-enhanced这个包 | -y，就会自动选择y, 跳过询问你Is this OK[y/d/N] |
| which vim | 已安装VIM的系统中查看VIM属于哪个软件包 |  |
| whereis java | 查看JDK安装路径 |  |
| rpm -q < package name> | 查询一个包是否被安装 |  |
| rpm -qa | sort | 查询系统安装的所有软件包 |  |
| rpm -ql< package name> | 列出软件包安装的文件**和位置** |  |
| rm -f <filename> | 直接删除某个文件 | 如果要删除目录要加-r |
| rm –rf /home/test | 删除test目录下所有文件 | 包括test目录 |
| rm -rf /home/test/\* | 删除test目录下所有文件 | **保留**test目录  -f：强制删除;-r：递归的删除目录下面的文件以及子目录下文件 |
| rm –rf !(a|b) | 删除除a和b外所有文件，必须在当前目录下执行 | 如果提示“-bash: !: event not found ” shopt -s extglob 来解决 |
| find a/\* [-not] –name b|xargs rm –rf | 可选参数[-not]存在时，删除a文件下所有除b外的所有其他文件 | b可为文件夹名，若b为文件夹时，b底下的文件也会被删除 |
| find /usr/sam/\* -path  /usr/sam/dir1 –prune -o –print | 在/usr/sam目录下查找不在dir1子目录之内的所有文件 |  |
| find / -name "\*tomcat\*" | 查询系统中所有名字中含有tomcat的文件 |  |
| mkdir 文件夹名称 | 新建文件夹 |  |
| vi文件名称 | 修改此文件 | 按i键进入编辑模式，按esc退出编辑模式；  在浏览模式中：:qw 保存退出，:q直接退出，u撤销上一步的操作 |
| vi 新文件名 | 创建文本文件 |  |

2. 功能提示

* cd .. 返回上级目录

.表示当前目录；..表示上级目录；./aaa/ 表明的就是当前目录下面的aaa目录

* vncserver启动vncserver服务

vncpasswd修改vnc远程登录密码

systemctllist-unit-files | grep vnc查看vnc服务

Systemctlenablevncserver@serivce启用服务

* **Centos 安装vim**
* vim编辑器需安装三个包：

   vim-enhanced-7.0.109-7.el5

  vim-minimal-7.0.109-7.el5

   vim-common-7.0.109-7.el5

* rpm -qa|grep vim，查看本机已经存在的包，确认VIM是否已经安装
* 如果缺少某个，比如说：vim-enhanced，执行：yum -y install vim-enhanced 命令，它会自动下载安装。

-y，就会自动选择y, 跳过询问你Is this OK[y/d/N]

* **vi 和vim 的区别**
* vim对vi的完全兼容,可视为vi的升级版本
* vi有3个模式：插入模式、命令模式、低行模式。

插入模式：在此模式下可以输入字符，按ESC将回到命令模式。

命令模式：可以移动光标、删除字符等。

低行模式：可以保存文件、退出vi、设置vi、查找等功能(低行模式也可以看作是命令模式里的)。

* vim的配置文件/etc/vimrc

|  |  |
| --- | --- |
| **以下命令在命令模式下使用** | |
| vi filename | 打开filename文件 |
| :w | 保存文件 |
| :w vpser.net | 保存至vpser.net文件 |
| :q | 退出编辑器，如果文件已修改请使用下面的命令 |
| :q! | 退出编辑器，且不保存 |
| :wq | 退出编辑器，且保存文件 |
| **以下命令在命令模式下使用，执行下面命令后将进入插入模式，按ESC键可退出插入模式** | |
| a | 当前光标位置的右边添加文本 |
| i | 在当前光标位置的左边添加文本 |
| A | 在当前行的末尾位置添加文本 |
| I | 在当前行的开始处添加文本(非空字符的行首) |
| O | 在当前行的上面新建一行 |
| o | 在当前行的下面新建一行 |
| R | 替换(覆盖)当前光标位置及后面的若干文本 |
| J | 合并光标所在行及下一行为一行(依然在命令模式) |

* screen 离线运行程序
* 安装命令是：yum install screen
* 新建一个huobi会话：screen -S huobi(huobi为会话名，可自己定义)
* 离开会话并让程序断续运行：ctrl +a d (按住ctrl不放，分别按 a 和 d)
* 恢复后台运行的会话：screen -r huobi（huobi为自己定义的会话名）
* 显示所有screen创建的会话：screen -ls
* 在会话里执行exit命令会话是结束运行并退到shell中
* 开机启动screen并在后台运行其他程序
* 首先我们需要制作一个脚本。在终端下输入并回车新建脚本文件startHuobiScreen.sh

touch startHuobiScreen.sh

chmod 777 startHuobiScreen.sh

* 编辑startHuobiScreen.sh

screen\_name="huobi"

screen -mS $screen\_name

cmd="java -jar /usr/local/huobi/h\*";

screen -S $screen\_name -p 0 -h 2 -X stuff "$cmd\n"

* 然后向rc.local文件添加启动代码

chmod +x /etc/rc.d/rc.local (授予/etc/rc.d/rc.local执行权限)

vim /etc/rc.d/rc.local

/usr/local/huobi/startHuobiScreen.sh

3. 常用系统文件

|  |  |  |
| --- | --- | --- |
| /etc/yum.repos.d/ | Yum仓库配置文件夹 | 可在下面添加软件安装源配置文件 |
| ~/ | 用户目录 | windows下也是用户目录 |

* bash环境下通配符：



* bash环境下特殊符号：



* 管线命令





* 管线命令仅会处理 standard output，对于 standard error output 会予以忽略
* 管线命令必须要能够接受来自前一个指令的数据成为 standard input 继续处理才行。

# Git笔记

1. 指令

|  |  |  |
| --- | --- | --- |
| git config --global http.sslVerify false | 设置全局变量http.sslVerify为false |  |
| git pull | = git fetch + git merge |  |
| git push --force origin | 强制推送 |  |
| git reset --hard 8618818（版本号） | 回退到某一版本 |  |

1. 功能提示



* webstorm删除github上的文件
* 先commit到本地仓库，再push到GitHub上
* 强制覆盖本地文件（本地文件有更改）
* 本地先revert到上一个版本，然后git pull

# Mysql笔记

1. 命令

|  |  |
| --- | --- |
| mysql -uroot -p | 以root身份启动mysql，在命令行cmd中输入 |
| mysqldump -u 用户名 -p 数据库名 > 导出的文件名 | 导出整个数据库 |
| mysqldump -u 用户名 -p 数据库名 表名> 导出的文件名 | 导出一个表 |
| show databases; | 显示所有数据库，**以下所有命令都在进入mysql后使用，命令后的必须有分号** |
| show tables； | 显示所有表 |
| use 数据库名； | 进入某个数据库 |
| source sql文件的路径； | 导入外部的sql文件 |

1. windows下重置root密码

1. 以系统管理员身份登陆系统。

2. 打开cmd-----net start 查看mysql是否启动。启动的话就停止net stop mysql.

3. 我的mysql安装在d:\usr\local\mysql4\bin下。

4. 跳过权限检查启动mysql.

d:\usr\local\mysql\bin\mysqld -nt --skip-grant-tables

5. 重新打开cmd，进到d:\usr\local\mysql4\bin下：

d:\usr\local\mysql\bin\mysqladmin -u root flush-privileges password "newpassword"

d:\usr\local\mysql\bin\mysqladmin -u root -p shutdown  这句提示你重新输密码。

6. 在cmd里net start mysql

# MongoDB笔记

下载地址参考：<https://www.mongodb.org/dl/win32/i386>

1. 安装 MongoDB服务
2. 新建数据库根文件夹mkdir c:\data
3. 新建存放数据和日志文件的目录mkdir c:\data\dbmkdir c:\data\log
4. 创建一个配置文件位于 C:\mongodb\mongod.cfg，内容如下

systemLog:

destination: file

path: c:\data\log\mongod.log

storage:

dbPath: c:\data\db

1. 通过执行mongod.exe，使用--install选项来安装服务，使用--config选项来指定之前创建的配置文件。

C:\mongodb\bin\mongod.exe --config "C:\mongodb\mongod.cfg" –install

1. 启动MongoDB服务net start MongoDB

关闭MongoDB服务net stop MongoDB

移除 MongoDB服务C:\mongodb\bin\mongod.exe --remove

1. Centos上安装 MongoDB

1. centos官方yum源，其中是没有mongodb包的，所以要额外增加些配置，编辑Mongodb安装源如下：

* vim /etc/yum.repos.d/mongodb-org-3.6.repo
* 编辑内容：

[mongodb-org-3.6]

name=MongoDB Repository

#baseurl=https://repo.mongodb.org/yum/redhat/$releasever/mongodb-org/3.6/x86\_64/

#阿里云上的镜像

baseurl=http://mirrors.aliyun.com/mongodb/yum/redhat/6/mongodb-org/3.2/x86\_64/

gpgcheck=1

enabled=1

gpgkey=https://www.mongodb.org/static/pgp/server-3.6.asc

2. 安装

yum install -y mongodb-org

3. 启动前修改配置

* （推荐）修改ulimit，改高一点就好了，方法参考其他博客
* （推荐）关闭transparent\_hugepage

vim /etc/rc.local，在最后添加以下代码，重启系统生效

if test -f /sys/kernel/mm/transparent\_hugepage/enabled; then

echo never > /sys/kernel/mm/transparent\_hugepage/enabled

fi

if test -f /sys/kernel/mm/transparent\_hugepage/defrag; then

echo never > /sys/kernel/mm/transparent\_hugepage/defrag

fi

4. 修改mongodb配置文件

* vim /etc/mongod.conf, 修改内容为如下配置:

其中端口被修改为21111（随意），并且允许远程访问，不允许则修改bandIp为：127.0.0.1

已经启用了用户身份验证，并关闭了javascript

systemLog:

destination: file

logAppend: true

path: /var/log/mongodb/mongod.log

storage:

dbPath: /var/lib/mongo

journal:

enabled: true

processManagement:

fork: true

pidFilePath: /var/run/mongodb/mongod.pid

net:

port: 21111

bindIp: 0.0.0.0

security:

authorization: enabled

javascriptEnabled: false

* 保存后重启系统reboot

5. 修改mongodb配置文件

* 启动，以下方式会自动使用/etc/mongod.conf配置文件

service mongod start

* 新建管理员用户

mongo --port 21111

> use admin

>db.createUser({user:"userAdmin",pwd:"123456",roles:[{role:"userAdminAnyDatabase",db:"admin"}]})

> db.auth("userAdmin","123456")

* 新建普通用户

> use test

> db.createUser({user:"test",pwd:"123456",roles:[{role:"readWrite",db:"test"}]})

* 退出mongo命令行exit
* 停止mongodb

mongod -f /etc/mongod.conf --shutdown

3. 指令

|  |  |  |
| --- | --- | --- |
| mongod –dbpath数据库存放目录 | 启动数据库 | 在系统Shell下 |
| mongodb://[username:password@]host1[:port1] [/[database][?options]] | 连接数据库 | 在系统Shell下 |
| mongo | 连接数据库进入后台管理Shell | 在系统Shell下 |
| mongodb://[username:password@]host1[:port1] [/[database][?options]] | 连接数据库 | 以下指令在后台管理Shell下 |
| db.auth("name","pwd") | 验证账户，密码 |  |
| use DATABASE\_NAME | DATABASE\_NAME不存在新建数据库；存在则使用此数据库 |  |
| show dbs | 查看所有数据库 |  |
| db.createCollection(name) | 新建集合 |  |
| db.dropDatabase() | 删除数据库 | 需先use切换入要删的数据库 |
| db.collection.drop() | 删除集合 |  |
| db.原集合名.renameCollection(新集合名) | 修改集合名称 |  |
| show collections | 查看所有集合 |  |
| db.auth(‘name’,’pwd’) | 进行用户授权 | 运行此指令必须要先用use admin切换到admin数据库里进行验证；之下的创建用户，需先授权管理员账户 |
| db.system.users.find() | 查看数据库的用户表和权限 | 需先use admin |
| db.createUser({user:'test',pwd:'test',roles:[{role:'readWrite',db:'testDB'}]}) | 创建新用户 | 需要有管理员权限的授权下 |
| db.dropUser('用户名') | 删除某一用户对某一数据库的权限 | 需先use admin，在db.auth验证后进行 |

3. 功能提示

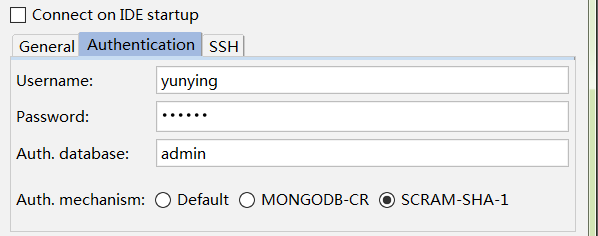
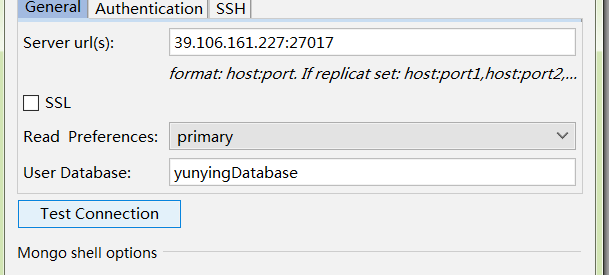
* mongodb 修改用户密码
* 修改密码不能直接更新表数据，这样的话，存到表里的密码是明文的
* 利用db.changeUserPassword

> db.changeUserPassword('tank2','test');

* Intellij使用mongdb插件连接远程服务器时General填写要连接的实际数据库

Authentication必须选择SCRAM-SHA-1，Auth.database填写mongDB的用户库

另外，远程服务器的响应端口必须要打开



* mongo shell里所有的db.auth命令必须要在use admin后，在这个用户库里进行（因为所有的用户数据都保存在这个库里），完成验证后，即可转到相应的其他库进行工作
* mongo 设置权限的一些示例
* 授予角色：db.grantRolesToUser( "userName" , [ { role: "<role>", db: "<database>" } ])
* 取消角色：db.grantRolesToUser( "userName" , [ { role: "<role>", db: "<database>" } ])
* 创建角色db.createRole
* 删除角色db.dropRole("testRole")
* 更新角色

db.updateRole("testRole",{ roles:[{ role: "readWrite",db: "mydb"}]},{ w:"majority" })

* 添加Privileges给角色

db.grantPrivilegesToRole("testRole",

[{ resource: { db: "mydb", collection: "" },actions: [ "update", "insert", "remove" ]}

])

|  |  |  |
| --- | --- | --- |
| **角色分类** | **角色** | **权限及角色**  **（本文大小写可能有些变化，使用时请参考官方文档）** |
| Database User Roles | read | CollStats,dbHash,dbStats,find,killCursors,listIndexes,listCollections |
| readWrite | CollStats,ConvertToCapped,CreateCollection,DbHash,DbStats,  DropCollection,CreateIndex,DropIndex,Emptycapped,Find,  Insert,KillCursors,ListIndexes,ListCollections,Remove,  RenameCollectionSameDB,update |
| Database Administration Roles | dbAdmin | collStats,dbHash,dbStats,find,killCursors,listIndexes,listCollections,  dropCollection 和 createCollection 在 system.profile |
| dbOwner | 角色：readWrite, dbAdmin,userAdmin |
| userAdmin | ChangeCustomData,ChangePassword,CreateRole,CreateUser,  DropRole,DropUser,GrantRole,RevokeRole,ViewRole,viewUser |
| Cluster Administration Roles | clusterAdmin | 角色：clusterManager, clusterMonitor, hostManager |
| clusterManager | AddShard,ApplicationMessage,CleanupOrphaned,FlushRouterConfig,  ListShards,RemoveShard,ReplSetConfigure,ReplSetGetStatus,  ReplSetStateChange,Resync,    EnableSharding,MoveChunk,SplitChunk,splitVector |
| clusterMonitor | connPoolStats,cursorInfo,getCmdLineOpts,getLog,getParameter,  getShardMap,hostInfo,inprog,listDatabases,listShards,netstat,  replSetGetStatus,serverStatus,shardingState,top  collStats,dbStats,getShardVersion |
| hostManager | applicationMessage,closeAllDatabases,connPoolSync,cpuProfiler,  diagLogging,flushRouterConfig,fsync,invalidateUserCache,killop,  logRotate,resync,setParameter,shutdown,touch,unlock |
| Backup and Restoration Roles | backup | 提供在admin数据库mms.backup文档中insert,update权限  列出所有数据库：listDatabases  列出所有集合索引：listIndexes    对以下提供查询操作：find  \*非系统集合  \*系统集合：system.indexes, system.namespaces, system.js  \*集合：admin.system.users 和 admin.system.roles |
| restore | 非系统集合、system.js，admin.system.users 和 admin.system.roles 及2.6 版本的system.users提供以下权限：  collMod,createCollection,createIndex,dropCollection,insert    列出所有数据库：listDatabases  system.users ：find,remove,update |
| All-Database Roles | readAnyDatabase | 提供所有数据库中只读权限：read  列出集群所有数据库：listDatabases |
| readWriteAnyDatabase | 提供所有数据库读写权限：readWrite  列出集群所有数据库：listDatabases |
| userAdminAnyDatabase | 提供所有用户数据管理权限：userAdmin  Cluster：authSchemaUpgrade,invalidateUserCache,listDatabases  admin.system.users和admin.system.roles：  collStats,dbHash,dbStats,find,killCursors,planCacheRead  createIndex,dropIndex |
| dbAdminAnyDatabase | 提供所有数据库管理员权限：dbAdmin  列出集群所有数据库：listDatabases |
| Superuser Roles | root | 角色：dbOwner，userAdmin，userAdminAnyDatabase  readWriteAnyDatabase, dbAdminAnyDatabase,  userAdminAnyDatabase，clusterAdmin |
| Internal Role | \_\_system | 集群中对任何数据库采取任何操作 |

* Mongo shell

当一行以{或[或(结束时，可进入多行模式；连续回车可退出多行模式

# Tomcat笔记

1. Tomcat开机自动启动

* 在/etc/init.d/目录下创建tomcat服务脚本文件。

vim /etc/init.d/tomcat

* 脚本文件内容如下

#!/bin/bash

# tomcat startup script for the Tomcat server

# chkconfig: 345 80 20

# description: start the tomcat deamon

# Source function library

. /etc/rc.d/init.d/functions

prog=tomcat8

JAVA\_HOME=/usr/java/jdk1.8.0\_151/ # 根据自己的路径改写JAVA\_HOME

export JAVA\_HOME

CATALANA\_HOME=/usr/local/tomcat/ # 根据自己的路径改写CATALANA\_HOME

export CATALINA\_HOME

case "$1" in

start)

echo "Starting Tomcat..."

$CATALANA\_HOME/bin/startup.sh

;;

stop)

echo "Stopping Tomcat..."

$CATALANA\_HOME/bin/shutdown.sh

;;

restart)

echo "Stopping Tomcat..."

$CATALANA\_HOME/bin/shutdown.sh

sleep 2

echo

echo "Starting Tomcat..."

$CATALANA\_HOME/bin/startup.sh

;;

\*)

echo "Usage: $prog {start|stop|restart}"

;;

esac

exit 0

* 赋权限，测试启动脚本

cd /etc/init.d/

chmod 755 tomcat #赋予权限

service tomcat start #启动服务

service tomcat stop #停止服务

service tomcat restart #重启服务

* 将服务脚本加入到系统启动队列

chkconfig tomcat on #服务脚本加入到系统启动队列

chkconfig --list tomcat #检查服务是否已经生效

注意：该输出结果只显示 SysV 服务，并不包含原生 systemd 服务。SysV 配置数据可能被原生 systemd 配置覆盖。

* 如果您想列出 systemd 服务,请执行 'systemctl list-unit-files'。
* 欲查看对特定 target 启用的服务请执行 'systemctl list-dependencies [target]'。
* 这里多说点儿，因为CentOS7的自启动服务开始由systemctl逐渐替代了早期版本中的chkconfig和service形式。这里我尝试了一下用指令：systemctl start tomcat8启动服务，系统提示systemctl daemon-reload命令加载服务，执行后，发现可以通过systemctl命令进行简单的控制，如查询状态，启动服务，终止服务，重启服务等操作。但是关于开机启动的设置是不可以的，还需要通过老命令chkconfig实现。

2. Tomcat的目录结构及常用配置

* 基本介绍

|  |  |
| --- | --- |
| $CATALINA\_HOME | Tomcat 安装的根目录 |
| $CATALINA\_BASE | 当配置多个 Tomcat 实例，需为每一个实例都定义一个$CATALINA\_BASE，如果没有配置多个实例，那么 $CATALINA\_BASE 其实就相当于  $CATALINA\_HOME |

* Tomcat的目录层次结构

bin: 存放启动和关闭Tomcat的脚本文件

conf：存放Tomcat服务器的各种配置文件

lib: 存放tomcat服务器支撑的jar包

logs: 存放Tomcat的日志文件

temp: 存放Tomcat运行时产生的临时文件

webapps:web应用虽在目录，即供外界访问的web资源的存放目录

work: Tomcat的工作目录

* JavaWeb应用的组成结构

appName---------------------------Web应用所在目录

|----html、jsp、css、js等文件，根目录下的文件外界可以直接访问

|----WEB-INF目录

|---------classes目录(java类)

|---------lib目录(java类运行所需的jar包)

|---------web.xml(web应用的配置文件)

WEB-INF 这个目录下的文件外界无法直接访问，由web服务器负责调用

* Tomcat的组成结构
* Tomcat本身由一系列可配置的组件构成，其中核心组件是Servlet容器组件，它是所有其他Tomcat组件的顶层容器。
* 每个组件都可以在Tomcat安装目录/conf/server.xml文件中进行配置，每个Tomcat组件在server.xml文件中对应一种配置元素。下面用XML的形式展示各种Tomcat组件之间的关系

<Server>代表整个Servlet容器组件，是最顶层元素，可以包含一个或多个<Service>元素

<Service>包含一个<Engine>元素以及一个或多个<Connector>元素，这些<Connector>共享一个<Engine>

<Connector/>代表和客户程序实际交互的组件，负责接收客户请求，以及向客户返回响应

<Engine>每个<Service>元素只能包含一个<Engine>元素，它处理在同一个<Service>中所有

<Connector>接收到的客户请求

<Host>在一个<Engine>中可以包含多个<Host>,它代表一个虚拟主机(即一个服务器程序

可以部署在多个有不同IP的服务器主机上)，它可以包含一个或多个应用

<Context>使用最频繁的元素，代表了运行在虚拟主机上的单个web应用

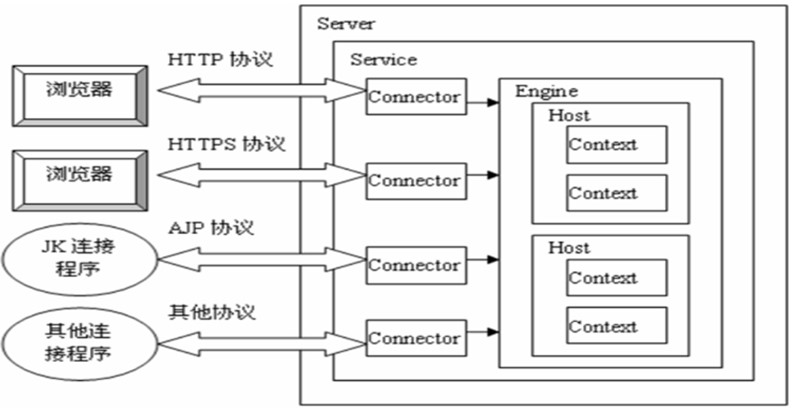
</Host>

</Engine>

</Service>

</Server>

* Tomcat体系结构



* 虚拟目录的映射方式
* 从Tomcat6开始，Tomcat支持自动映射，即tomcat服务器会自动管理webapps目录下的所有web应用，并把它映射成虚拟目录，换句话说，只需把web应用放在webapps目录下，不需要配置Context，外界可以直接访问
* 但是如果你不想讲项目放在webapps下，而是放在其他地方，还是需要配置Context，比如我的tomcat在C盘，我的项目在D盘,则需要在Host下增加如下虚拟目录映射语句
* Context元素配置

tomcat在加载一个web应用时，会一次按照以下五种方式查找web应用中的<Context>元素，优先级一次降低，直到找到为止

1. 到Tomcat安装目录/conf/context.xml文件中查找<Context>元素。

2. 到Tomcat安装目录/conf/[enginename]/[hostname]/context.xml.default文件中查找<Context>元素。

[enginename]:表示<Engine>的name属性

[hostname]:表示<Hostd的name属性

3. 到Tomcat安装目录/conf/[enginename]/[hostname]/[contextpath].xml文件中查找<Context>元素

[contextpath]:表示单个Web应用的URL入口

4. 到Web应用的META-INF/context.xml文件中查找<Context>元素

5. 到Tomcat安装目录/conf/server.xml文件中查找<Context>元素。只适用于单个Web应用

* 将项目部署为Tomcat默认应用

即访问 http://localhost:8080 时出来的是tomcat自带的欢迎页面，改为登录到自己的项目主页

* 在Tomcat默认安装后，tomcat的主目录是webapps/root目录，所以如果想改变tomcat的主目录的话可以如下所做：

在/conf/server.xml文件的<Host></Host>之间加入代码

<Contextpath=""docBase="C:\tomcat7\webapps\myapp"reloadable="true"debug="0"></Context>

docBase改为自己需要的项目路径；

* 如果建立了Apache和tomcat集群，Apache server 的默认端口是80 ，IE访问的方法只需输入：http://localhost,就可以自动定位到xx工程下面去
* Context元素的属性

|  |  |
| --- | --- |
| path | 指定访问该Web应用的URL入口（浏览器上输入的, 不要/），指定该值后，会在appBase文件夹下自动新建一个以path为名称的文件夹，同时把docBase的文件自动copy进此文件件  Note：appBase文件夹在<host>元素的属性上指定 |
| docBase | 指定Web应用的文件路径，可写绝对路径，也可写相对appBase属性的相对路径（**相对路径时不要加/**） |
| className | 指定实现Context组件的Java类的名字，这个类必须实现org.apache.catalina.Contex接口，该属性默认值为org.apache.catalina.core.StandardContext（不建议大家改动） |
| reloadable | 如为true，Tomcat服务器在运行状态下会监视WEB-INF/classes和WEB-INF/lib目录下的class文件的改动，还监视Web应用的WEB-INF/web.xml的改动，如果检测到更新，服务器会自动更新加载web应用。默认值为false。在开发和调试阶段，将其改为true，一般像Eclipse等开发环境都会默认改为true。在正式发布阶段，应将其该为false，可以降低Tomcat的运行负荷，提高Tomcat的运行性能 |

在一般情况下，<Context>元素都会使用默认的标准Context组件，即className属性采用默认值org.apache.catalina.core.StandardContext，它除了拥有上面介绍到的属性外，还有自身专有的属性：

|  |  |
| --- | --- |
| cachingAllowed | 是否允许启用静态资源(HTML、图片、声音等)的缓存。默认值为true。 |
| cacheMaxSize | 设置静态资源缓存的最大值，单位为K |
| workDir | 指定Web应用的工作目录 |
| uppackWAR | 如果为true，会把war文件展开为开放目录后再运行。为false，直接运行war文件。默认值为true。 |

* web.xml文件

该文件必须放在/WEB-INF目录下，但是从7.0版本开始，Tomcat可以不使用web.xml文件，而是使用注

解方式。

3. Tomcat的优化

* 内存使用配置
* Tomcat是依赖于JVM的，所以Tomcat的使用内存配置实质上是JVM的内存配置
* 在/bin目录下的catalina.bat可以直接通过Tomcat设置JVM内存参数,windows下,打开catalina.bat文件，在大概中间的位置，找到:

set "JAVA\_OPTS=%JAVA\_OPT% -server -Xms2048m -Xmx2048m -XX:PermSize=256m

-XX:MaxPermSize=512m -Djava.awt.headless=true"

* 通过内存设置充分利用服务器内存

-server：模式启动应用慢，但是极大程度提高运行性能

-Xms2048m -Xmx2048m：设置的是堆内存

* java8开始，PermSize被MetaspaceSize代替，MetaspaceSize共享heap，不会再有

java.lang.OutOfMemoryError：PermGen space，可以不设置

* Headless=true：适用于Linux系统，与图形操作有关，如生成验证码含义是当前的是无显示器的服务器，应用中如果获取系统显示有关的参数会抛出异常，windows系统可不用设置
* 可通过jmap -heap process\_id查看设置是否成功
* 最大连接数配置（并发能力）
* 在/conf/server.xml里进行配置

在<Connector>属性中将protocol属性改为 "org.apache.coyote.http11.Http11NioProtocol",NIO模型的效率更高，加入

maxThreads="500" //当前可以同时处理的最大用户访问数

minSpareThreads="100" //最小空闲线程连接数，用于优化线程池

maxSpareThreads="200"

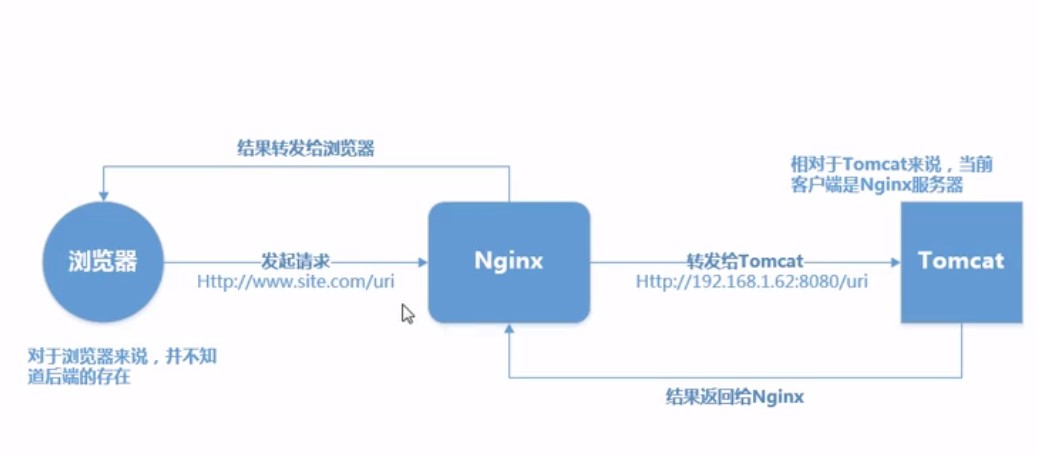
acceptCount="200" //当所有的线程以分配，仍然允许连接进来，但是出于等待状态的用户数。等待线程数+工作线程数=总的可最大连接数,如果超过此数，新的连接将不会被接受，就会产生一个http错误

enableLookups="false" //是否允许DNS反查，如果为true，request.getRemoteHost会执行DNS查找，反向解析ip对应域名或主机名，当没有这样需要的时候，可以将这个功能关闭，在一定程度上提高了Tomcat服务器的性能

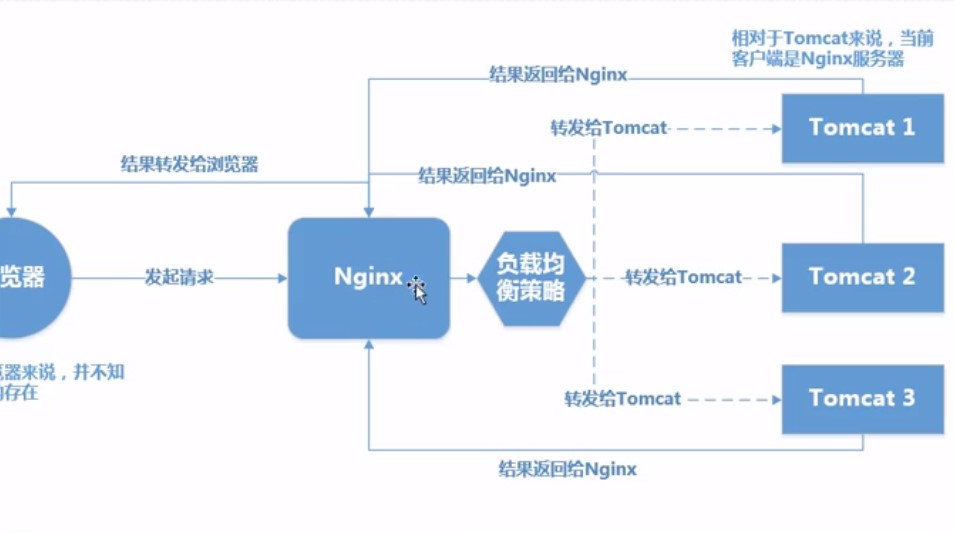
4. 负载均衡配置

* 负载均衡是提高应用负载量和容错的有效手段，可以结合Nginxs与Tomcat实现，Nginx是一款性能优异的反向代理服务器
* 负载均衡配置实现

**反向代理原理示意图：**



**均衡示意图：**



* 负载均衡策略：

Nginx 提供轮询（round robin）、用户IP哈希（client IP）和指定权重 3 种方式

* 负载均衡时Session的处理策略
* Session是Tomcat服务器上的内存空间，如果一个用户发出多个请求，却发到了多个tomcat服务器中，那么就会出现Session不同步的问题
* 解决方案1：将一个用户的请求锁定到某一台服务器上，简单，但是缺乏容错性，一旦某个服务器发生故障，Session可能丢失，（但是服务器发生故障是一个低概率事件，如果一个服务器经常挂掉，要么是硬件有问题，要么是应用有问题），可以使用用户IP哈希实现
* 解决方案2：Session复制策略，基于网络的广播策略，一个节点Session变化，其他节点同步复制，具有容错性，但节点多或复制量大时对网络负荷大，使网络效率低下，甚至阻塞

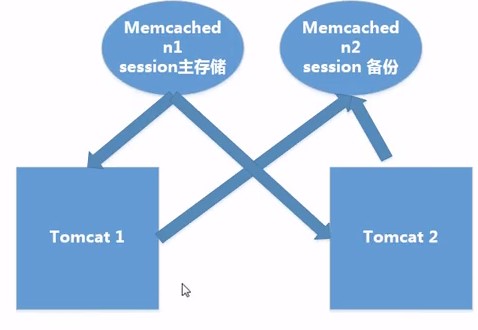
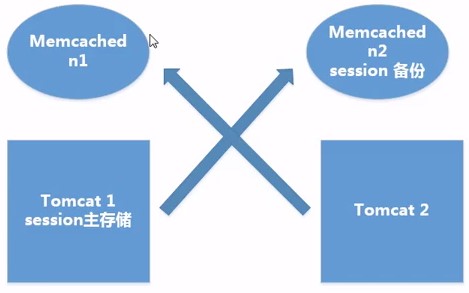
①在/conf/server.xml文件中开启Session复制的选项，将<Cluster>注释去掉，7.0默认值以配置好，需要接收器<Receiver>绑定内网(一般服务器都有两块网卡)网卡地址,修改端口>。

②在应用中指定应用是在分布式部署之下，在web.xml中添加<distributable/>选项

* 解决方案3：额外创建一个共享的空间用来存放Session，所有服务器共享一个Session

memcached缓存共享方案基本原理

Session与非粘性Session方式：



需要一些工具jar包，官网有目录和下载连接，在/conf/context.xml中进行配置，全局有效

* 集群环境中应用代码应注意的问题
* 传递Session需要实体类序列化支持，实现可序列化接口，设置版本号
* 获取用户IP地址方法的变化，获取真实客户端ip而不是Nginx代理地址
* 动静分离结构的预规划

一般高并发的网站上，资源类的文件，如js,css,图片通常是由静态服务器处理，Nginx处理静态文件效率就非常高，而tomcat处理静态文件是它很大的一个弱势，tomcat只负责动态请求的处理。那么编码的时候就要考虑静态资源最后可能要被拿出去，地址会有变化，所以在开发的时候应该规划访问地址，保持最大灵活性。

# Nginx笔记

1. CentOS 7上安装Nginx

1.添加CentOS 7 EPEL仓库

sudo yum install epel-release

2.安装Nginx

sudo yum install nginx

3.启动Nginx

sudo systemctl start nginx

4.如果您正在运行防火墙，请运行以下命令以允许HTTP和HTTPS通信：

sudo firewall-cmd --permanent --zone=public --add-service=http

sudo firewall-cmd --permanent --zone=public --add-service=https

sudo firewall-cmd --reload

5.如果想在系统启动时启用Nginx。请输入以下命令：

sudo systemctl enable nginx

2. 指令

|  |  |  |
| --- | --- | --- |
| nginx –s reload | 重启命令 |  |
| nginx –t | 检查配置文件是否正确 |  |

3.功能提示

* nginx 更改配置文件后需要重启，重启命令：nginx –s reload

# NPM笔记

1. 指令

|  |  |  |
| --- | --- | --- |
| npm config get prefix | 查看npm全局仓库路径 |  |
| npm config set prefix | 设置npm全局仓库路径 |  |
| npm view jquery versions | 查看jquery模块全部版本 |  |
| npm install gulp-sourcemaps --save | 安装gulp-sourcemaps模块并写入package.json |  |

2.功能提示

* package.json版本(^和~区别)
* 波浪符号（~）：他会更新到当前minor version（也就是中间的那位数字）中最新的版本
* 插入符号（^）：将会把当前库的版本更新到当前major version（也就是第一位数字）中最新的版本

# Bower笔记

1. 指令

|  |  |  |
| --- | --- | --- |
| bower list | 列出当前安装的版本和最新版本，以及各个安装包的依赖 |  |
| bower info jquery | 查询jQuery有哪些版本 |  |
| bower install bootstrap --save | 安装bootstrap并更新至bower.json中的Dependencies |  |
| bower install bootstrap –save-dev | 安装bootstrap并更新至bower.json中的devDependencies |  |
| bower install --force | 强制安装 |  |

2.功能提示

* bower.json文件解释

{

"name":"", //必须，如果需要注册包，则该包名唯一。

"description":"", //可选，包描述

"main":[], //可选，入口文件，bower本身不使用，供第三方构建工具会使用,每种文件类型只能有一个。

"ignore":[], //可选，文件或目录列表。bower安装的时候将忽略该列表中的文件,bower是从git仓库或压缩包下载一个包，里面的文件并不一定全部需要。

"dependencies":[], //依赖包，name:value，value可以是包的semverrange(版本号范围)，也可以直接是一个包的git地址或压缩包地址。

"devDependencies":[], //开发依赖包，仅仅在开发过程中测试或者编译文档用，部署生产环境是不需要。

"resolutions":[], //包引用冲突自动使用该模块指定的包版本,格式和dependencies 相同

"overrides" :[//可以覆盖一个包中的默认设置，比如main里面设定的入口文件

"package-name":{ //这样可以根据需要，让第三方工具只打包需要的文件。

"main":[]

}

],

"moduleType":"", //可选，指定包采用那种模块化方式(globals,amd,node,es6,yui)

"private":Boolean, //是否公开发布当前包,如果只是使用bower来管理项目的包，设置为true.

"license":"", //授权方式(GPL-3.0,CC-BY-4.0.....)

"keywords":[], //可选，方便注册后容易被其他人搜索到。

"authors":[], //作者列表

"homepage":[], //主页，包介绍页

"repository":{ //包所在仓库。

"type": "git",

"url": "git://github.com/foo/bar.git"

},

｝

# Gulp笔记

1. 语法

* gulp.src(globs[, options])
* gulp.src (['./src/\*\*', '**!**\*\*/node\_modules/\*\*'])**→**感叹号表示不包括第二个参数代表的内容
* gulp.src ('./src/\*\*/\*.js', {base:'./src/'})**→**base作用：'./src/\*\*/\*.js'路径去除'./src/'得到的路径加到gulp.dest里的路径后面得出最终的目的路径

# Gradle笔记

1. 目录文件解释

|  |  |
| --- | --- |
| **目录文件** | **作用** |
| .gradle | gradle项目产生文件（自动编译工具产生的文件） |
| .idea | IDEA项目文件（开发工具产生的文件） |
| app | 其中一个module，复用父项目的设置，可与父项目拥有相同的配置文件  （里面也能包含build.gradle、gradle.properties、setting.gradle 等相关gradle文件，其实每一层都是一个module，整个项目是一个大的 module 而已） |
| build | 自动构建时生成文件的地方 |
| gradle | 自动完成gradle环境支持文件夹 |
| .gitignore | git源码管理文件 |
| build.gradle | gradle 项目自动编译的配置文件 |
| gradle.properties | gradle 运行环境配置文件  比如配置gradle运行模式，运行时jvm虚拟机的大小 |
| gradlew | 自动完成 gradle 环境的linux mac 脚本，配合gradle 文件夹使用 |
| gradlew.bat | 自动完成 gradle 环境的windows 脚本，配合gradle 文件夹使用 |
| local.properties | Android SDK NDK 环境路径配置 |
| \*.iml | IDEA 项目文件 |
| setting.gradle | gradle 项目的子项目包含文件 |

2. 提示

* 指令

|  |  |
| --- | --- |
| gradle init --type pom | 将maven的pom.xml转换成gradle的build.gradle（在pom文件所在的文件夹） |
|  |  |

* 在项目构建时选择use default gradle warpper
* IDEA会调用项目根目录 gradlew 或者 gradlew.bat （根据linux，windows，osx自动选型）代替原生的 gradle 方法做自动构建
* 解析 gradle/wrapper/gradle-wrapper.properties 文件，获取项目需要的gradle版本下载地址
* 判断本地用户目录下的 ./gradle 目录下是否存在该版本，不存在该版本，下载，存在跳过下载
* 下载 gradle-wrapper.properties 指定版本，并解压到用户目录的下 ./gradle 文件下
* 利用 ./gradle 目录下对应的版本的 gradle 进行相应自动编译操作
* gradle-wrapper.
* gradle-wrapper.jar是Gradle Wrapper的主体功能包。在Intellij安装过程中产生gradle-wrapper.jar，然后每次新建项目，会将gradle-wrapper.jar拷贝到你的项目的gradle/wrapper目录中
* gradle-wrapper.properties文件主要指定了该项目需要什么版本的Gradle，从哪里下载该版本的Gradle；此文件中的GRADLE\_USER\_HOME一般指~/.gradle
* Gradle对应版本下载完成之后，Gradle Wrapper的使命基本完成了
* gradle各版本官网下载地址

<http://services.gradle.org/distributions/>

* gradle官方教程

<https://docs.gradle.org/current/userguide/userguide.html>

* gradle缩写

<<表示doLast，5.0版本将移除，不推荐使用

task hello << {

println 'Hello world!'

}

* gradle路径

例子:krill:hello

首个冒号表示根项目，后面的为项目结构（以冒号分隔）；以上例子表示根项目/krill项目的hello task

* Gradle SSH Plugin Document

https://gradle-ssh-plugin.github.io/docs/

# JSON笔记

1.注意事项

* 基本规则
* JSON官网规定，不管是键或值都用**双引号**引起来
* json文件里的换行符\r\n
* json文件里不允许加注释
* 要描述的一个对象, (里面是这个对象的属性/函数!!) 那么用大括号, 表示对象; 如果你要描述的东西是多个并列的值, 那么用数组用中括号的方式!!!