# 1 JAVA Core

# 2常用概念

集群：同一个业务，部署在多个服务器上(不同的服务器运行同样的代码，干同一件事)

分布式：一个业务分拆多个子业务，部署在不同的服务器上(不同的服务器，运行不同的代码，为了同一个目的)



1. keepalived + ngnix
2. 集群情况下session 共享问题

Session存储于Redis或memcached同步session. 把session抽取出来，放到内存级数据库里面，解决了session共享问题，同时读取速度也是非常之快。

1. Zookeeper
2. Redis (缓存数据库/非关系型数据库/NOSQL)

Mongodb

Redis与Memcached区别：

Redis 支持持久化（保存到硬盘） Memcached 不支持持久化

Redis比Memcached支持更丰富的数据类型

应用场景：

减轻数据库压力 解决集群情况下的session共享问题

5种数据类型 string list set sortedset hash

1. vmare 安装虚拟机

具体安装Centos （linux系统），

安装后查询网络，ifconfig, 网卡eth0 只有ipv6，没有ipv4，那是因为

原因是Linux 系统未开启上网功能. **在命令行中,输入命令ifup eth0 来打开网络**

**输入ifconfig eth0 up 可启用eth0网卡, ifconfig eth0 down  可停用eth0网卡**

具体参见<https://blog.csdn.net/qq_33229669/article/details/79138556>

有了ipv4，就可以使用winscp xschell crt工具连接虚拟，便于操作

Linux 安装redis 从后台线程方式启动redis 客户端连接redis服务器，连接超时。 这是因为linux系统防火墙开通原因， 直接service iptables stop 关闭防火墙。 防火墙这块还是有些问题？

防火墙知识：

Linux中有两种防火墙软件，ConterOS7.0以上使用的是firewall，ConterOS7.0以下使用的是iptables

在自己配置jdk 环境变量时，修改了/etc/profile, 重新启动后， 现象就是在用户输入密码时，不显示密码框，切换不了用户，登录不进去。 centos6 修改profile后，无法登录系统。解决方案详情：<https://blog.csdn.net/dreary001/article/details/84484599>

使用单用户模式

1、以grub 系统引导管理器，单用户登录方法；

1）在centos 系统启动时，按空格键进入grub系统引导界面。

2）在grub 启动后，移动键盘到Linux的启动项；按e键；然后再移动键盘到类似下面的一行，也就是kernel的那行：

kernel /boot/vmlinuz-2.6.11-1.1369\_FC4 ro root=LABEL=/1 rhgb quiet

把光标移动这行后，再按一下e键，进入编辑这行；在行尾条一个空格 ，然后输入 linux single，也就是类似如下的：

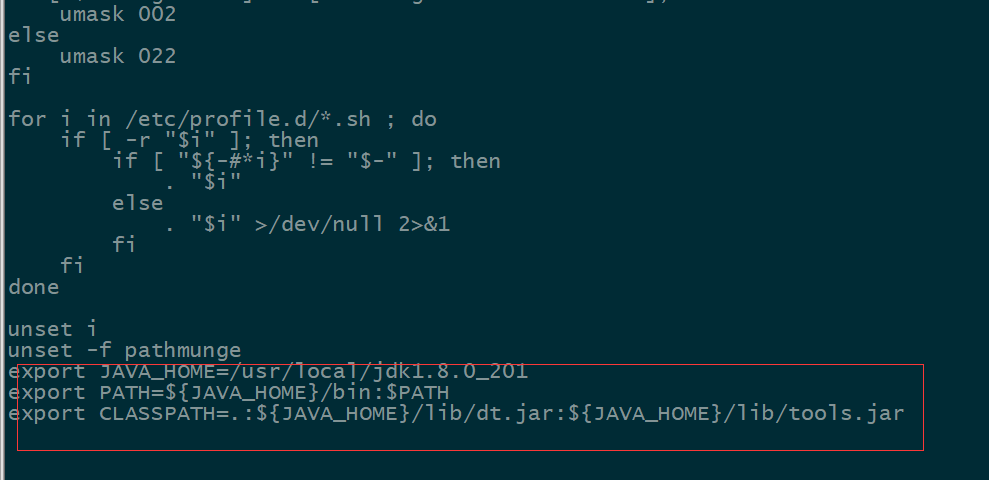
kernel /boot/vmlinuz-2.6.11-1.1369\_FC4 ro root=LABEL=/1 rhgb quiet linux single

结束编辑，按回车返回；

接着我们要启动系统，按一下b键启动；

3）重启后直接登录了，进入/etc/profile 把修改的删除掉， reboot重启就可以了。

Jdk配置 /etc/profile 该方法时对所有用户生效



export JAVA\_HOME=/usr/local/jdk1.8.0\_201

export PATH=${JAVA\_HOME}/bin:$PATH

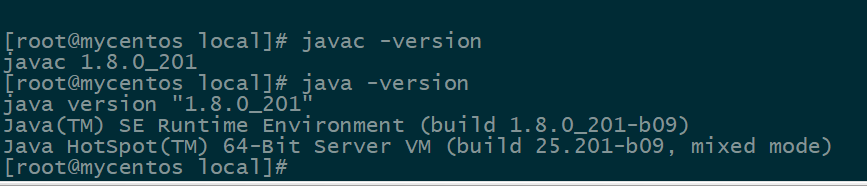
export CLASSPATH=.:${JAVA\_HOME}/lib/dt.jar:${JAVA\_HOME}/lib/tools.jar

保存退出，然后输入下面的命令来使之生效

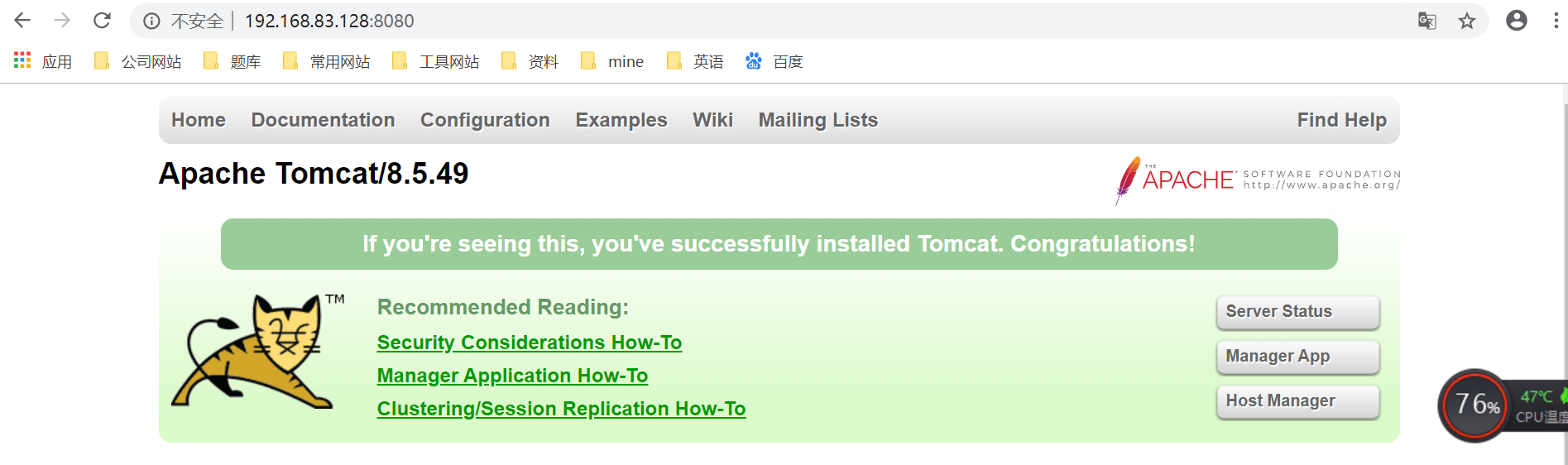
source /etc/profile

查看jdk是否配置OK

javac -version



Tomcat 直接启动就可以使用了



部署一个web项目

<http://192.168.83.128:8080/centos-webdemo/>



详情见 virtualBox安装centos，并搭建tomcat：

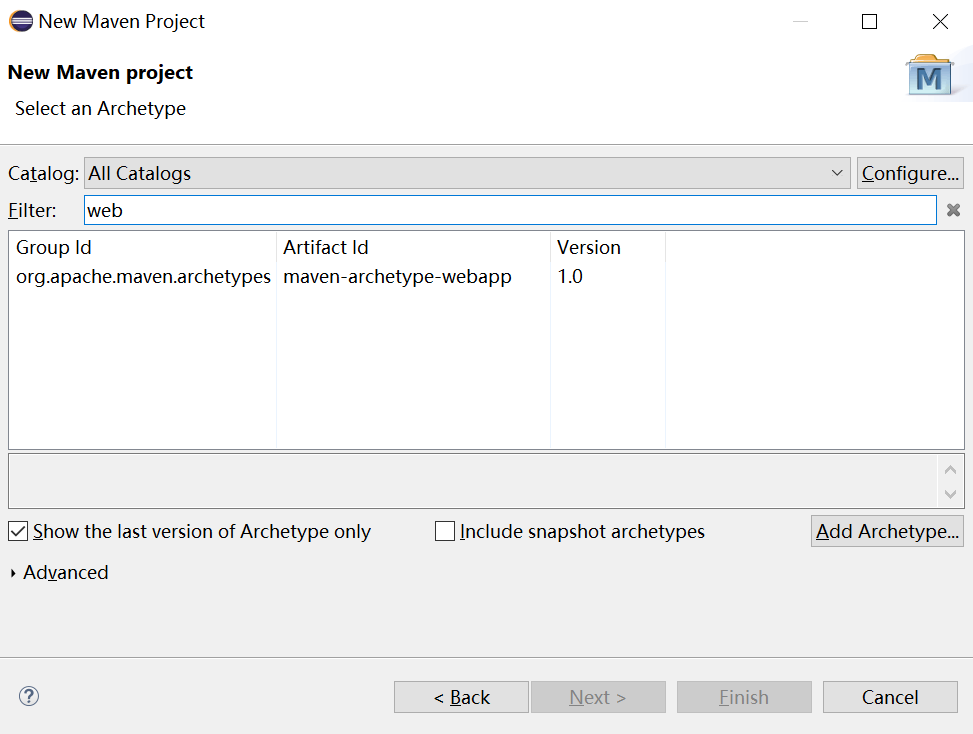
<https://www.cnblogs.com/youzhibing/p/5031080.html>

6项目搭建：

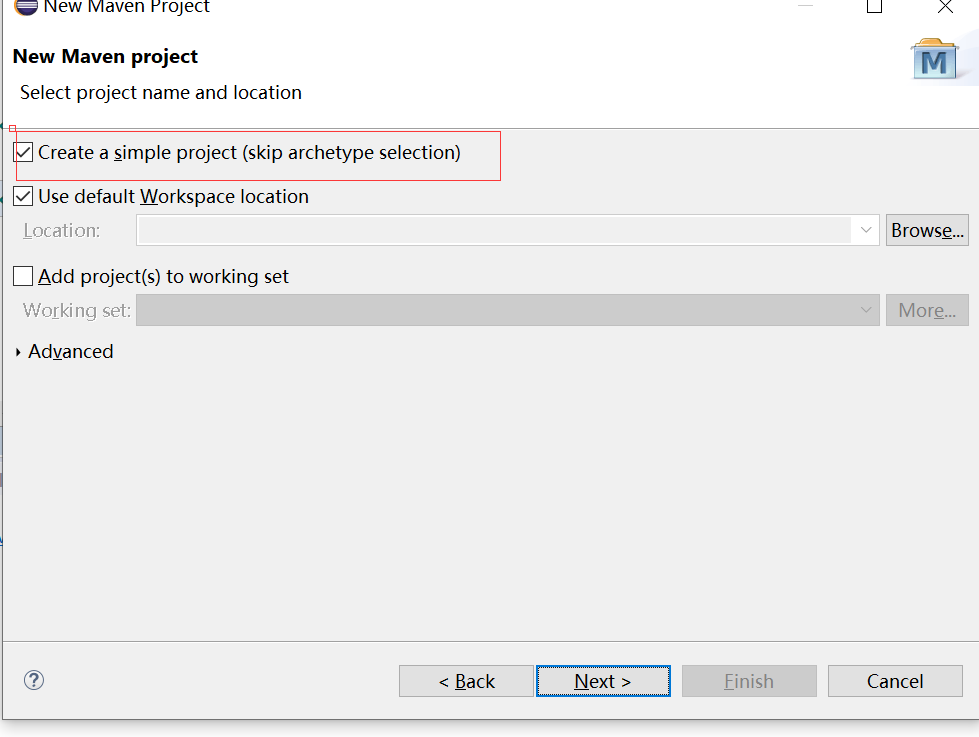
6.1 Maven 工程搭建

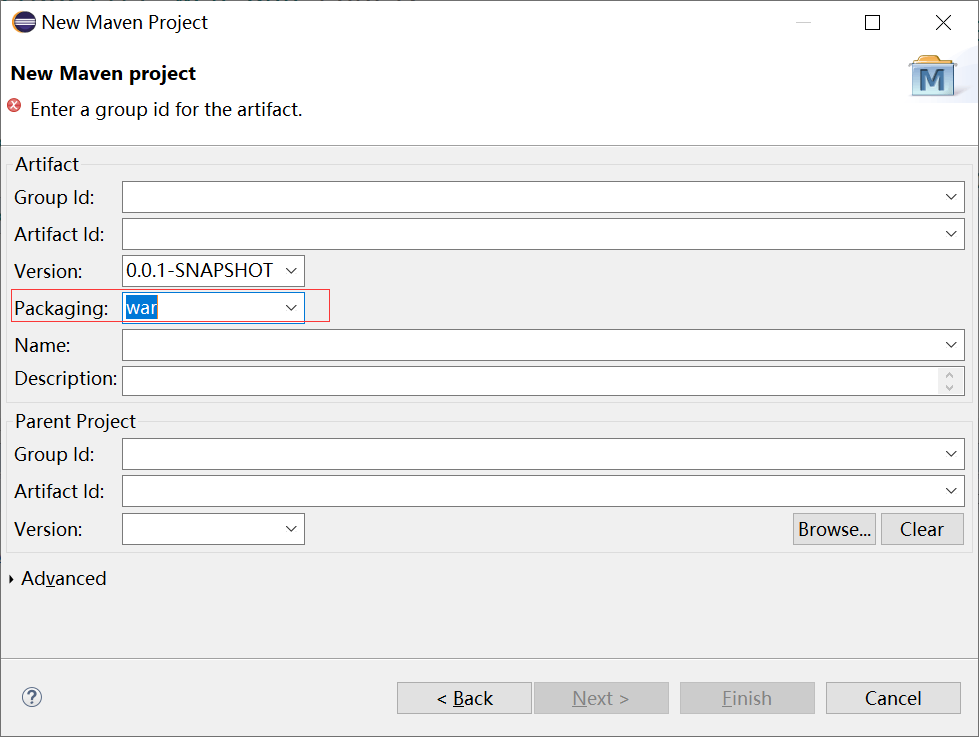
6.2 Maven web工程搭建

第一种



第二种

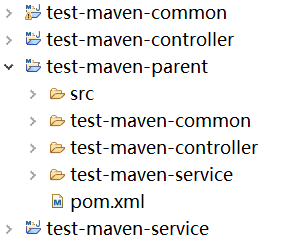




6.3 Maven 聚合项目搭建

6.3.1 maven 工程 （pom类型） 为父工程 一个

6.3.2 maven module模块， （jar/war类型）,为父工程的子模块 多个



Redis基础

数据类型 ： string list set sortedset hash 5种

Java 操作redis

Springboot 集成redis

Redis 高级 （redis支持主从同步，实现高可用，只是简单对redis.conf文件配置）

1、主从同步:从数据库启动时直接给主数据库发sync命令，拷贝主数据库rdb文件，载入数据库中，实现主从同步。具体过程如下图。

配置方式：

修改从redis中的 redis.conf文件

slaveof 192.168.33.130 6379

masterauth 123456--- 主redis服务器配置了密码,则需要配置

自己搭建主从同步环境（一主多从）：1、克隆已经安装好的虚拟机 2、克隆完后需要把mac地址重新生成 （新虚拟机ip一般都是ip+1,克隆好后可以查看）



**过程：**

1：当一个从数据库启动时，会向主数据库发送sync命令，

2：主数据库接收到sync命令后会开始在后台保存快照（执行rdb操作），并将保存期间接收到的命令缓存起来

3：当快照完成后，redis会将快照文件和所有缓存的命令发送给从数据库。

4：从数据库收到后，会载入快照文件并执行收到的缓存的命令。

2、Redis 哨兵机制

哨兵机制本质就是对redis环境的容灾机制。 就是在其中一个从数据库sentinel.conf 中配置信息，让此从数据库同时兼任哨兵的角色，当然也可以单独配置在一个环境上。是以心跳的方式检验主数据库是否挂掉，当主数据库挂掉时，通过算法选取其中一个从数据库为新主数据库，从只能读变为能够读写了，此时其他从数据库为新主数据库的从数据库，但是当原始主数据库恢复时，此时无法恢复主数据库身份，依然是新主数据库的从数据库，只能读不能写。

配置步骤:

1.拷贝到etc目录

cp sentinel.conf /usr/local/redis/etc

2.修改sentinel.conf配置文件

sentinel monitor mymast 192.168.110.133 6379 1 #主节点 名称 IP 端口号 选举次数

3. 修改心跳检测 5000毫秒

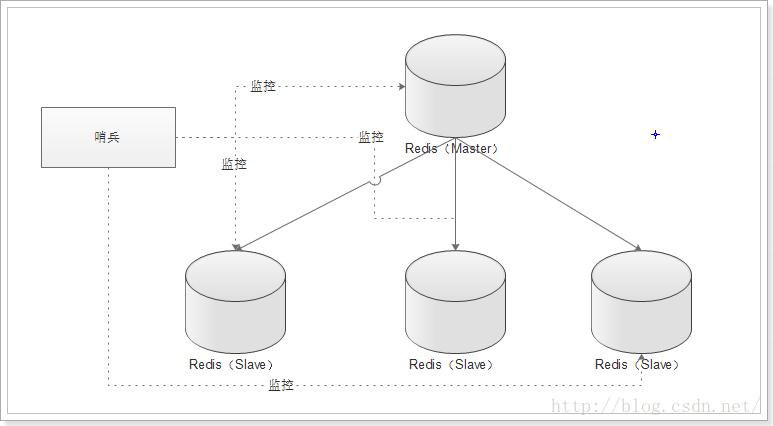
sentinel down-after-milliseconds mymaster 5000

4.sentinel parallel-syncs mymaster 2 --- 做多多少合格节点

5. 启动哨兵模式

./redis-server /usr/local/redis/etc/sentinel.conf --sentinel &

6. 停止哨兵模式



1. redis事务

确定：redis中是存在事务的

以 MULTI 开始一个事务， 然后将多个命令入队到事务中， 最后由 EXEC 命令触发事务

1. redis持久化

rdb模式：默认生成rdb文件，生成条件可以在配置文件中修改。生成dump.rdb文件存储数据。 默认启动数据库服务时加载dump.rdb,

aof模式：要在配置文件中修改配置，appendonly no ,修改为appendonly yes,这种方式主要是以日志的方式在日志文件尾部追加新增数据的命令，查询等命令不保存，数据库服务启动时，会加载aof文件。

两中方式的区别：aof方式比较粒度小（一般是将增加数据命令追加aof文件后，才进行实际的数据变更,基本上是零数据丢失），数据安全性高； rdb方式粒度大(生成rdb文件前，服务器挂掉，丢失数据)，数据安全性低。

1. redis 发布与订阅

redis 支持发布与订阅，subscribe XXX（channel名称） 订阅指定频道后，就可以publish XXX（channel名称）yang(发布内容) 发布信息，并且转发到订阅方

1. 遗留一个知识点，redis集群？？

ngnix

7 消息中间件

为了保证消息接受方能够实时接受到消息，保证生产者、消费者消息一致性。 通过数据库等其他方式定时发送数据，不能够达到接受消息实时性。

采用异步通讯方式。

JMS: java 消息服务；实际上是指JMS API，JMS是由Sun公司早期提出的消息标准，旨在为java应用提供统一的消息操作，包括create、send、receive等，类似于JDBC的角色。

消息模型：

1. 点对点（p2p）
2. 发布与订阅(pub/sub)
   1. ActiveMQ
      1. java使用mq
      2. springboot 集成mq
   2. RabbitMQ
   3. Kafaka

如何保证消息的幂等性？（大多数是问如何保证消息消费时的幂等性？或者说如何保证消息不被重复消费？ 当然生产者也会由于mq没有响应ack信息，恢复时，生产者也会生产重复消息？））

消息幂等性其实本质是问面试者使用消息队列如何保证幂等性。消息重复消费的问题是不可避免的，消息中间件 RabbitMQ、RocketMQ、Kafka，都存在这种问题， 这种问题考验面试者是否对这方面考虑以及怎样解决这种问题？

1. 手动签收方式
2. 消息id 消息id是唯一的，比如消费一条消息在数据库中保存消息id, 当消费消息时，根据id判断是否已经消费过，从而确保消息唯一性，
3. 日志记录 消息都有记录，查看日志是否有重复消息，手动删除

详情：<https://www.jianshu.com/p/86d3d72bd3fb>

<https://www.jianshu.com/p/8d1c242872a4>

消息中间件的关键概念：

**Broker** 消息服务器，作为server提供消息核心服务

**Producer** 消息生产者，业务的发起方，负责生产消息传输给broker

**Consumer** 消息消费者，业务的处理方，负责从broker获取消息并进行业务逻辑处理

**Topic** 主题，发布订阅模式下的消息统一汇集地，不同生产者向topic发送消息，由MQ服务器分发到不同的订阅者，实现消息的 广播

**Queue** 队列，PTP模式下，特定生产者向特定queue发送消息，消费者订阅特定的queue完成指定消息的接收

**Message** 消息体，根据不同通信协议定义的固定格式进行编码的数据包，来封装业务数据，实现消息的传输

rabbitMQ：erlang语言开发，依靠活跃的开源社区支持。

数据一致性、稳定性和可靠性高。

rocketMQ：阿里出品。

kafka：业界标准.。Kafka 实际上有个 offset 的概念，就是每个消息写进去，都有一个 offset，代表消息的序号，然后 consumer 消费了数据之后，每隔一段时间（定时定期），会把自己消费过的消息的 offset 提交一下

可扩展，高性能，多partition可容错。适合大数据领域等，可改造可靠性。

1. Ngnix

Nginx("engine x")是一款是由俄罗斯的程序设计师Igor Sysoev所开发高性能的 Web和 反向代理 服务器，也是一个 IMAP(交互式邮件访问协议)/POP3(邮局协议的第3个版本)/SMTP(简单邮件传输协议) 代理服务器。

在高连接并发的情况下，Nginx是Apache服务器不错的替代品。

Windows cmd

查询进程：netstat -ano | findstr “fsd”

查询端口：tasklist | findstr’ “端口号”

Linux 命令

查看之前系统安装jdk版本 rpm -qa | grep jdk

卸载程序 rpm -e --nodeps 程序名

解压压缩包 tar -zxvf 压缩包名

查找字符串搜索： find / | xargs grep "字符串"

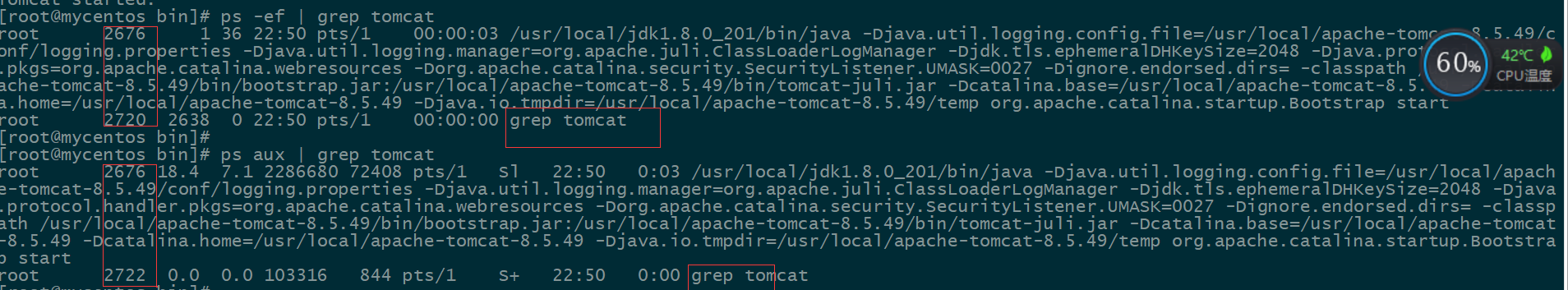
查找文件 find . -name "\*config.jar"

查询指定进程 ps -ef |grep java

ps aux | grep 'ngnix'

杀死进程 kill -9 560(进程id)

Linux 查询进程，出现两个进程：



[root@mycentos bin]# ps -ef | grep tomcat

root 2676 1 36 22:50 pts/1 00:00:03 /usr/local/jdk1.8.0\_201/bin/java -Djava.util.logging.config.file=/usr/local/apache-tomcat-8.5.49/conf/logging.properties -Djava.util.logging.manager=org.apache.juli.ClassLoaderLogManager -Djdk.tls.ephemeralDHKeySize=2048 -Djava.protocol.handler.pkgs=org.apache.catalina.webresources -Dorg.apache.catalina.security.SecurityListener.UMASK=0027 -Dignore.endorsed.dirs= -classpath /usr/local/apache-tomcat-8.5.49/bin/bootstrap.jar:/usr/local/apache-tomcat-8.5.49/bin/tomcat-juli.jar -Dcatalina.base=/usr/local/apache-tomcat-8.5.49 -Dcatalina.home=/usr/local/apache-tomcat-8.5.49 -Djava.io.tmpdir=/usr/local/apache-tomcat-8.5.49/temp org.apache.catalina.startup.Bootstrap start

root 2720 2638 0 22:50 pts/1 00:00:00 grep tomcat

原因： ps -ef|grep tomcat命令其实是分两步完成的。第一步执行ps -ef查询所有进程，第二步执行grep tomcat过滤出进程中带有tomcat关键字的进程。 其实grep tomcat这个命令本身也是个进程，所以会显示出来。 注意别把进程看错

如果不想显示grep进程怎么办，可以使用下面的命令：  
# ps -ef|grep tomcat|grep -v grep grep的-v参数是取反，带grep关键字进程，即把grep tomcat这个进程过滤掉。

<https://blog.csdn.net/chengdongyuan/article/details/79043870>

EJB

EJB 是运行在独立服务器上的组件，客户端是通过网络对EJB 对象进行调用的。在Java

中，能够实现远程对象调用的技术是RMI，而EJB 技术基础正是RMI。通过RMI 技术，J2EE

将EJB 组件创建为远程对象，客户端就可以通过网络调用EJB 对象了。

<https://blog.csdn.net/jojo52013145/article/details/5783677>

# 1 SpringBoot

集成第三方框架， 快速开发的脚手架。

特点：

独立的应用程序

搭建springboot工程

启动方式

基本操作：

1. 访问静态资源

文件

@RestController = @Controller + @ResponseBody (返回json串)

@ EnableAutoConfiguration

1. Mybatis 学习

具体参见：<https://www.w3cschool.cn/mybatis/7zy61ilv.html>

2.1 #{}和${}的区别。 主要就是#{}取值安全,防止sql注入

1)、#{}是预编译处理，MyBatis在处理#{ }时，它会将sql中的#{ }替换为？，然后调用PreparedStatement的set方法来赋值，传入字符串后，会在值两边加上单引号，如上面的值 “4,44,514”就会变成“ ‘4,44,514’ ”；

2)、 ${}是字符串替换，在处理是字符串替换，MyBatis在处理时,它会将sql中的{}是字符串替换，在处理{ }是字符串替换， MyBatis在处理{ }时,它会将sql中的是字符串替换，在处理是字符串替换，MyBatis在处理时,它会将sql中的{ }替换为变量的值，传入的数据不会加两边加上单引号。适用于在SQL 语句中插入一个不改变的字符串，比如order by ${columnName}

注意：使用场景：

一般情况首选#{}，因为这样能避免sql注入；如果需要传参 动态表名、动态字段名时，需要使用${}

比如：select \* from ${tableName} where id > #{id};

什么是SQL注入问题：

举个例子，如果使用${}出现的注入问题：

select \* from ${tableName};

如果传参 t\_user;delete from t\_user，则预编译后的sql如下，将会导致系统不可用：

2 网站搭建

三 JAVA web

1. servlet

创建servlet有三种方式：

1.1 实现servlet接口 implements Servlet

1.2 继承 GenericServlet 类

1.3 继承 HttpServlet 方法 （最常用的一种方法）

2、servlet 生命周期

1) 实例化（使用构造方法创建对象）

2）初始化 执行init方法

3）执行服务 执行service方法

4）销毁 执行destroy方法

3、@WebServlet注解的使用

在servlet3.0以后，我们可以不用再web.xml里面配置servlet，只需要加上@WebServlet注解就可以修改该servlet的属性了。web.xml可以配置的servlet属性，在@WebServlet中都可以配置。

4、过滤器 cookie session会话