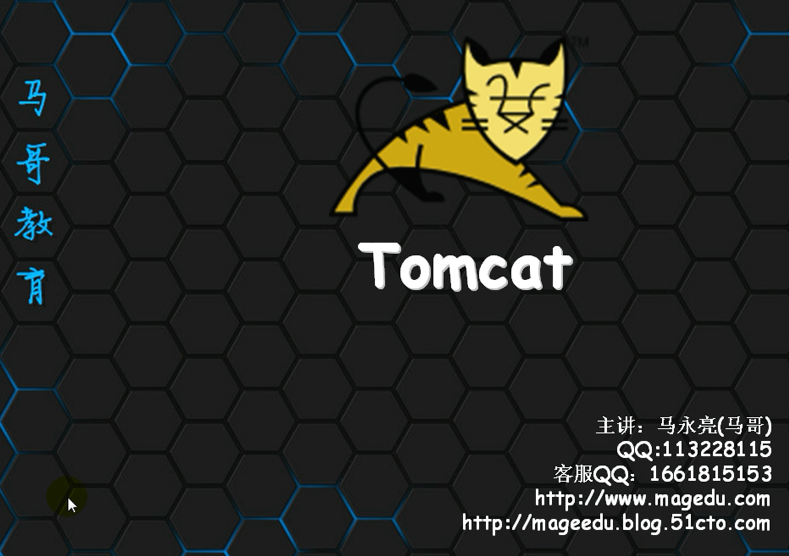
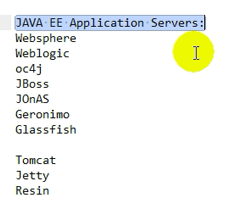
因为2013版的tomcat 马哥的录像没有讲完 所以 我只能找到2015年的 找到需要补上的位置 重听





JVM的组成结构 Java的类加载器 一个是Java执行引擎

53:45之后开始

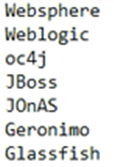


Websphere --- IBM 电信 证券 部署jsp通常用websphere部署 ----- WebSphere几乎实现了所有的j2ee 规范 ----最著名的商业实现 ---- 40%以上

WebLogic ---- BEA公司提供的 30% ---- 又被Oracle收购了 ---- Oracle也有一个叫oc4j只能称得上5% ----可以和上面抗衡

但是很多人 发现上面 两款笨重

JBoss ---- 被Redhat收购 已开源 但是服务要收费

 ----- 商业软件

Sun早些年给了j2ee一种实现 比较差 ----- 开源

因此sun依靠企业级应用赚钱

同时Apache也开始着手研发能够运行jsp的web容器 也不咋好

过了一段时间 sun直接把早期把这个捐献给apache之后 apche和自己的做了整合 就是tomcat

之前 叫catalina

迄今为止 装上tomcat之后 内部很多地方叫catalina

Tomcat某种意义是Apache融合了sun的参考实现的一种简化的不完整的j2ee (servlet jsp jmx)

JBoss自己实现了自己的j2ee平台 但是 核心是tomcat 现在jboss的核心可以换 换成jetty

JBoss是开源的 真正在开源界流行的是tomcat和jetty jetty比tomcat更加轻量级

Resin半开源 学习可以 但是 商业化要授权



JSP自身也是Java程序 所以 也要编译

---- Jetty可以嵌入

---- **jetty本身就是jar文件**



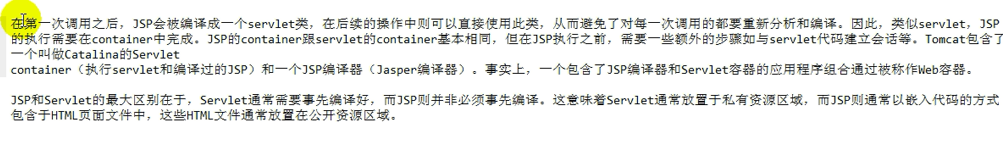
 优化之后 比tomcat更好



Resin比tomcat和jetty重量一点

在JVM中通过Quercus可以运行PHP

Resin不支持开源的商业应用



JSP先通过Japser转换成Java 再编译成class

 ====>Java开发的页面程序运行比较快 ----- 总要比PHP的源代码边解释边执行要快

Jsp Container仅仅多了一个Jasper

Tomcat和Jetty都是使用Java开发 所以 启动tomcat 必须提供jdk ---- jetty也是这样的

要部署tomcat

----- 要部署tomcat 就要提供jdk

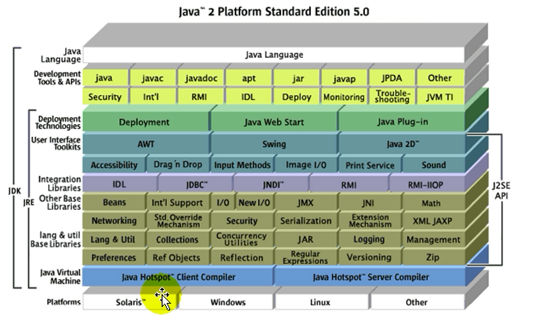
Jdk程序有多种实现

JDK仅仅是一个规范 有多个实现

有Oracle的实现 叫JRocket

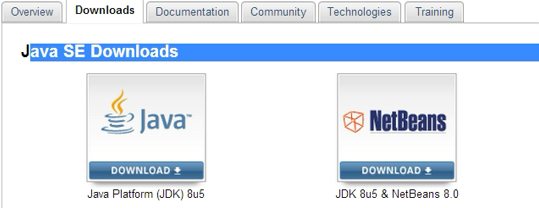
有Sun的实现 就叫JDK ----- 现在被Oracle收购 ----免费但是不开源

有开源实现 叫Open JDK ----- OpenJDK不全是开源的 有一段代码借用了别人的商业代码---- 编译好的加密的程序



淘宝内部都是OpenJDK 已经优化 并且在淘宝内部的改进已经在淘宝的站点开源

我们使用的最多开始官方的JDK



用的广发的仍然的是6

8引入了很多新特性

JDK很多升级版都有bug

**Hadoop就是运行在指定版本的jdk 不是所有的jdk都适合**

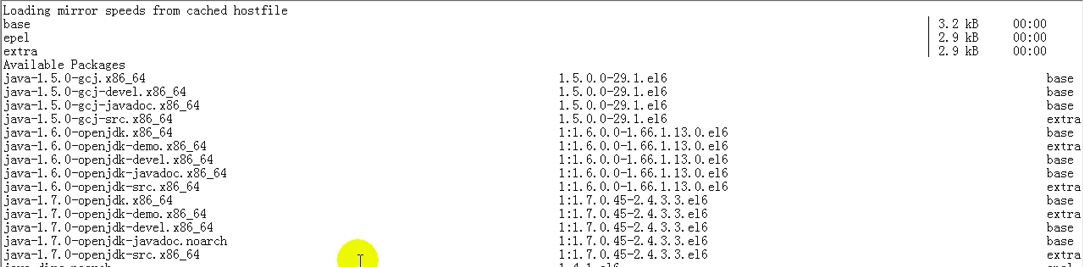
Tomcat jetty都需要编译自己的jsp页面 所以 需要的是jdk 而不是jre



下载对应的版本即可

ftp上下载好了 直接使用 并且编译好了

jdk由于是一个非常常用的程序 所以在linux的发行版都有





我们到官方站点下载就很简单了

演示如何安装官方提供的jdk

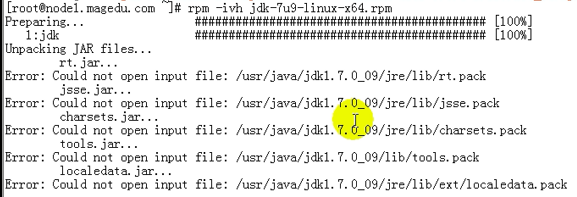


一个是jdk6 第31次升级版 另一个是jdk7 第9次升级版



使用rpm包进行安装



这些错误没关系

按下装载到哪里去了





切换到/usr/java下



latest表示最新版



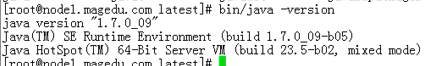




Jar就是打包的类库



Java以来的头文件



虚拟机是HotSpot ---- Server VM

---- 混合模式

JVM运行有两种模式：客观模式和服务器模式 混合模式都支持

现在进行环境变量的配置

编辑/etc/profile  






------- 安装配置tomcat

这个时候配置好了Jdk环境

使用jps 查看系统中的Java进程



编译 运行就可以了

Sun早期提供了java ee参考性的实现 ---- 叫Java Web Server ---- JWS

Apache也创建了这样的一个java ee的实现 ---- 叫 JServ

Sun在1999年把jws捐献给了asf

上面的Jws和Jserv合并成了现在tomcat

因此Tomcat的第一个版本是从Tomcat3.x开始【jws和jserv看做前面的两个版本】

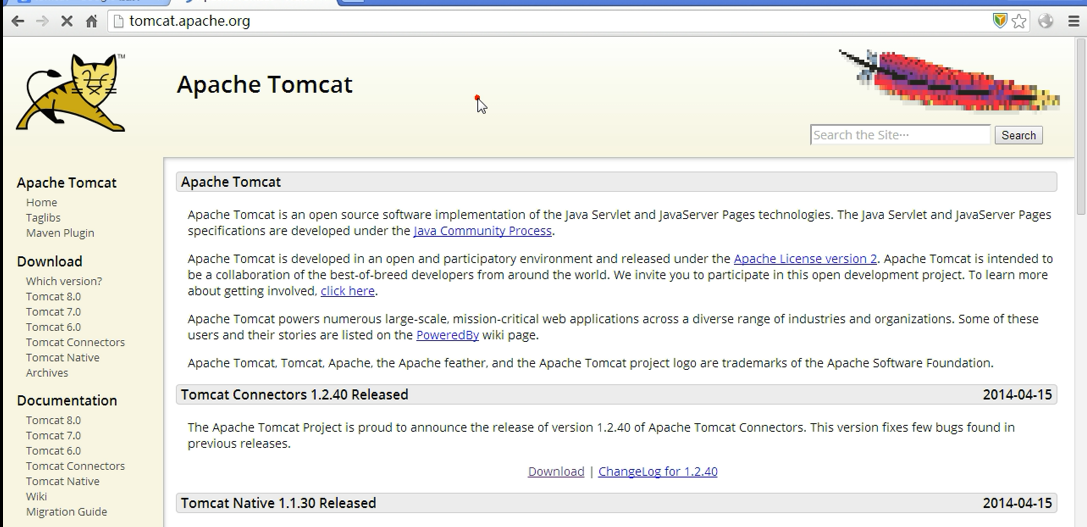
2001年发布了tomcat4 ---- 在tomcat3 jws jserv融合而成 ----项目名称是Catalina  
tomcat仅仅提供了jsp和Servlet两个api ---- ejb jms javamail 都没实现

所以tomcat就实现了web企业级应用 web container

Tomcat是能够执行jsp代码的容器 ----- 要能把jsp代码转换成jvm可执行的程序

转换成Servlet代码 最后 编译成字节码 最后 由类加载器加载到jvm运行

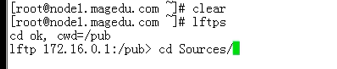
看一下tomcat的官方站点

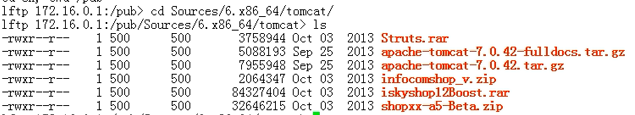


目前的版本是7.0.53

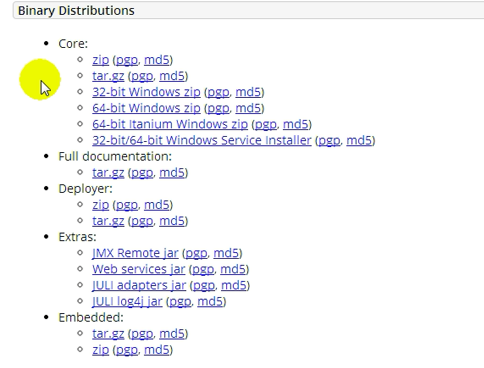
8是公测版

我们可以使用tomcat7和6



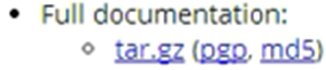


官方提供的下载方式有很多种





---- zip压缩的 tar.gz压缩的 core包含了tomcat最基本的代码



Full Documentation ---- 完整的文档的版本

 ---部署工具

 ---- 额外的提供了jmx的jar和ws的jar

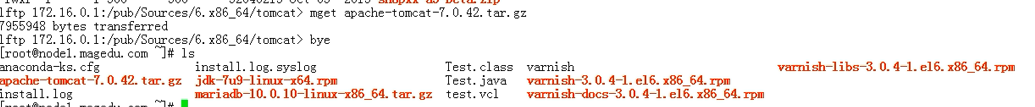
这些版本都是binary distributions 可以直接使用的

【因为tomcat都是Java的中间件 所以 可以直接下载使用这个二进制 因为Java是编译一次 到处运行】

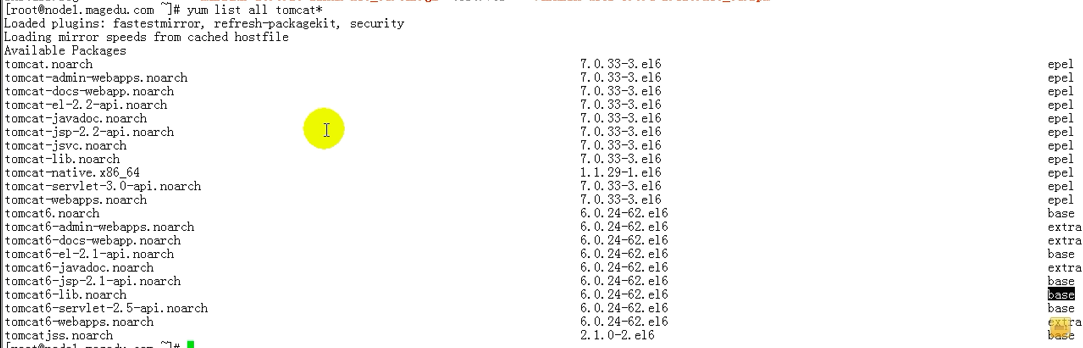
还有源代码版本



我们wget tomcat



使用yum list

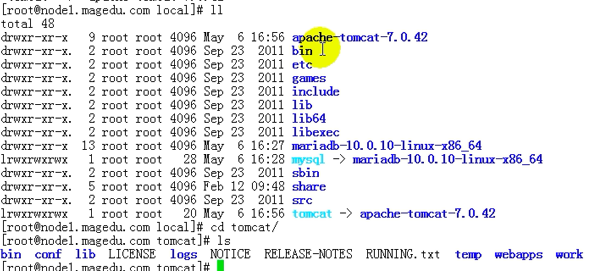


系统也提供了tomcat版本 但是好像是tomcat6的版本 也是官方站点的

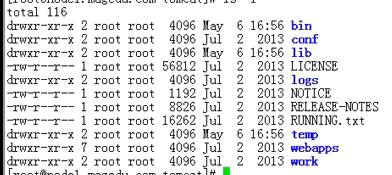


解压到目标文件

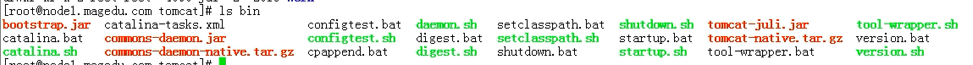




下面简要说明对应的目录



Bin目录是bat和sh脚本



还包含几个能够用于提高tomcat启动速度 + 加速jsp页面对第一次相应的jar文件 



Tomcat配置文件目录 核心配置文件是server.xml

对于tomcat很多配置的修改 都是配置这些文档

Httpd可以实现用户认证 tomcat也支持 ---- tomcat-user.xml



Tomcat各个组件之间共享的类库 通常表现为jar

Tomcat有一个组件是logging -----

Temp是临时文件 运行jsp文件 要先转换成Java源代码 编译的类 文件有可能在tmp源代码

webapps用于存放web应用的目录



ROOT是自己部署文件页面的目录

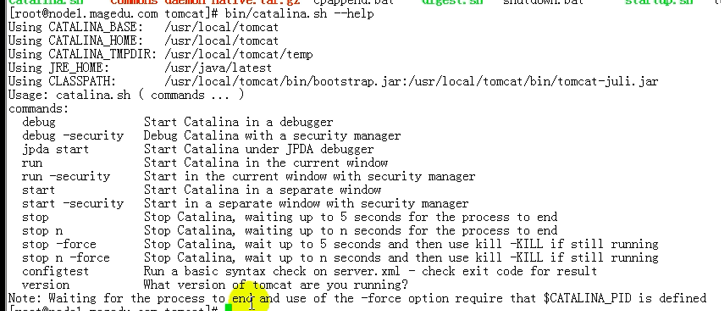
Work是tomcat的工作目录

Jsp编译的结果放在work下面 编译中间过程放在tmp文件夹下

再看bin目录



所有的脚本都可以使用catalina.sh进行执行



很多子命令

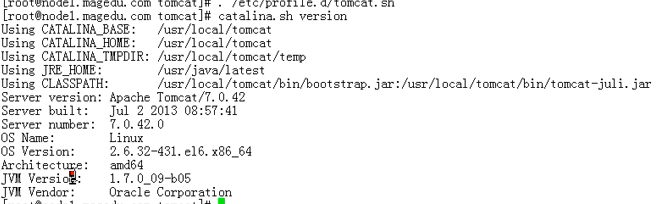
事实上 默认运行catalina脚本 依赖环境变量

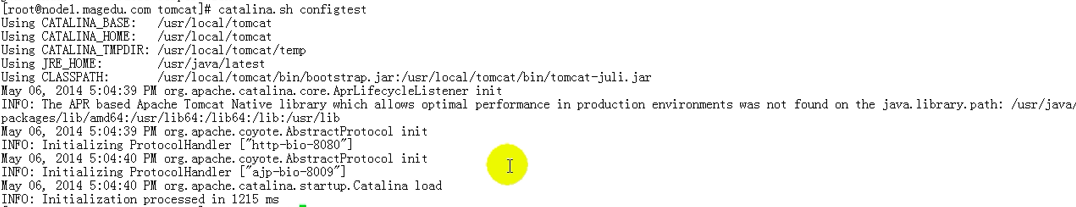
如果不配置  ---- 这些是默认的

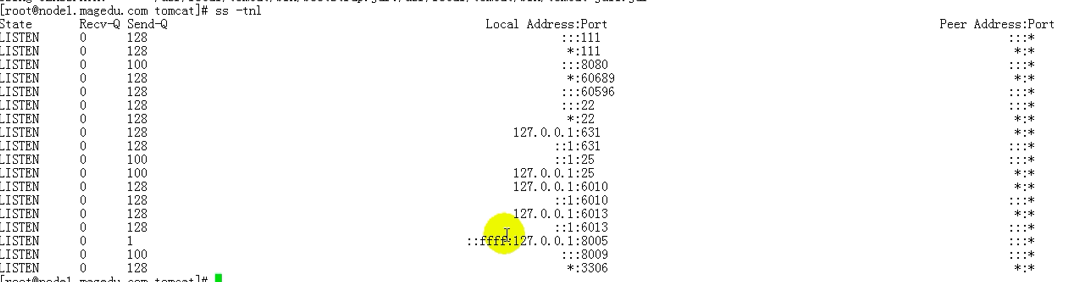
要编写/etc/rc.d/init.d/



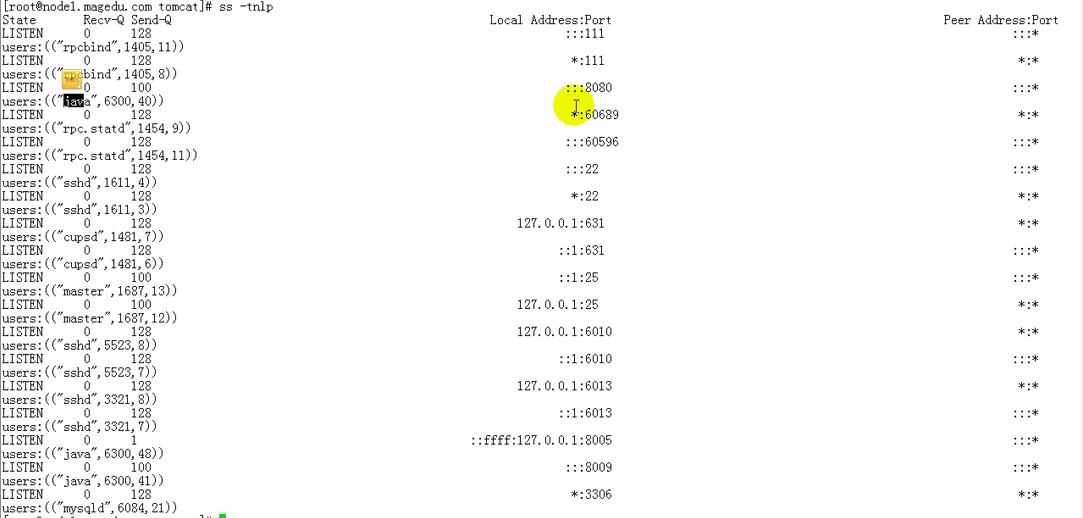








使用tnlp

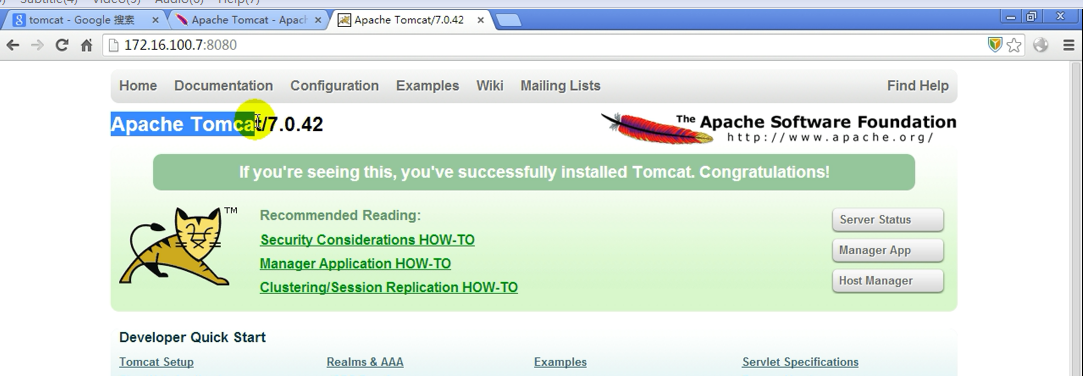






Bin下有一个bootstrap.jar

访问一下



页面上给了提示 告诉我们 内部部署了几个应用程序

查看服务器状态

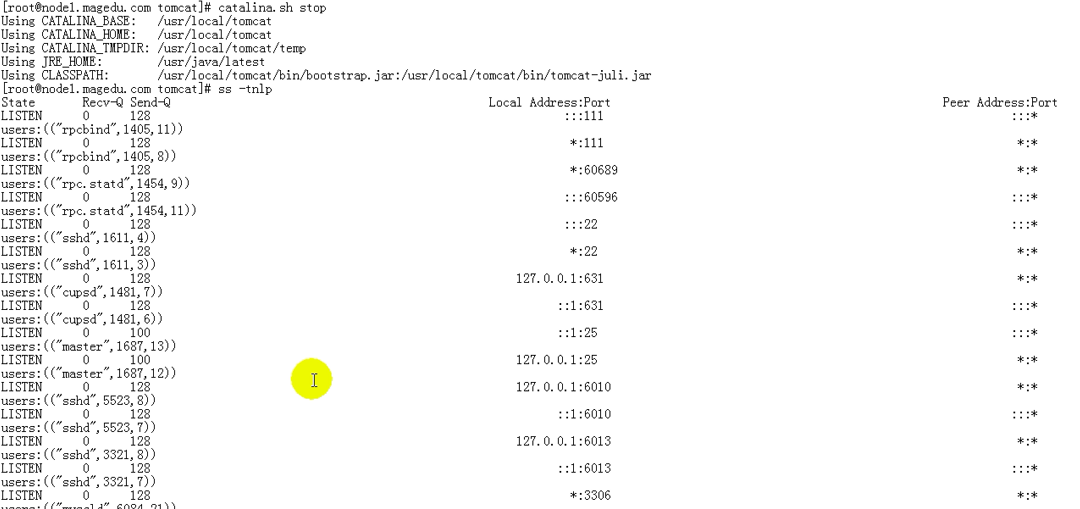
管理应用程序的

管理虚拟主机的

这个需要授权才能进入



停止tomcat --- catalina.sh stop



打开server.xml

 ---- Server是一个组件

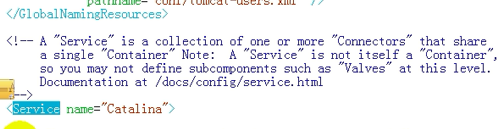
Tomcat内部许多组件组成 实际上就是类

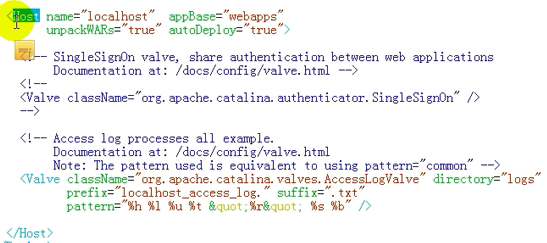
--- 额外配置

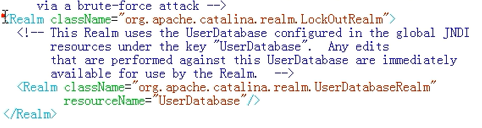
Server是一个顶级组件

里面很多子组件





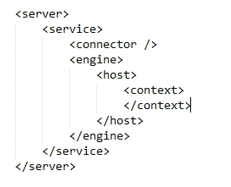




Realm可以嵌套

很多组件组成 组件之间是嵌套关系的 组件内部可以嵌套组件

----- 组件的组成关系

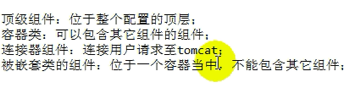
 --- tomcat的配置层次图

总体上 tomcat提供的组件 都是由tomcat自带的类组成的 有的是提供服务器的 有的是提供虚拟主机的

Server 顶级组件 ----整个配置的最外层

Service ----- 容器类组件 但是不是最外层

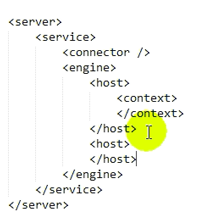
Connector ----- 连接器组件 ---- 链接用户请求到tomcat的



说明一下容器类组件

Engine ----- 是tomcat的核心容器 一个引擎是tomcat真正提供jsp解析 服务的很新组建 ----- 这个名字通常叫Catalina引擎---- 可以负责通过Connector接受用户请求 并响应请求 ---- 根本上的工作模式：检查http的首部信息 来决定到底转交给哪一个虚拟主机来处理

所以 Engine内部有多个Host ---- 多个虚拟主机 –每一个host是一个虚拟主机



Host ---- 类似于Httpd中的虚拟主机

Tomcat中仅仅支持主机名的虚拟主机 + 基于FQON的虚拟主机

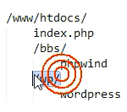
Context ---- 类似于Httpd中的alias

在httpd中有这样一个特性

访问/目录下面的bbs的目录

在php中部署：

 ---- 这个phpwind和上面的index.php没有关系 ---这个phpwind目录是一个独立的程序

 --- wordpress又是一个独立的程序

不管怎么讲 访问如下

 ---- 这里面index.php这个可以调用/test/test.php这个

所以 php部署应用程序 只需要建立子目录 直接放置就可以了

但是Java是不行的

Tomcat每一个应用程序要运行起来 要调用jvm中的其他类才能运行起来 --- 借助于类加载器装载这个项目的额外的其他类才能运行起来

所以 每一个应用程序可以不是类似于php程序那样 放进去 就可以运行 ---- 每一个tomcat程序要部署

部署：就是使用类加载器准备好所有的其依赖的所有类

 ---- 部署可以理解为准备运行环境

部署加载程序需要的所有的类 ---- 对于一个应用程序 就是 说 什么时候部署 部署完通过哪个路径来访问的

---- 要告诉tomcat 哪个项目部署在哪个路径下面 然后说明 通过什么样的uri进行访问---- 这个就是context容器类组件的作用

\*\*\*\*一个context通常代表一个web app

\*\*配置context主要是给出这个项目对外访问的根目录

Context类似于httpd的alias --- 也就是可以不配置在这个目录下面 还可以配置额外的属性 比如部署方式等



容器类组件的外层 有一个Service --- -Service是什么？

Connector类似于httpd中的listen

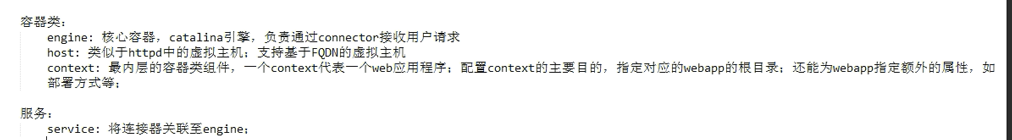
Httpd能不能监听多个套接字 ---- 就是多个端口 ---tomcat可以

Service就是用来把Connector关联到引擎的

用户的请求来了 可以被连接器接受 Connector不是Engine的子组件 是Service的子组件

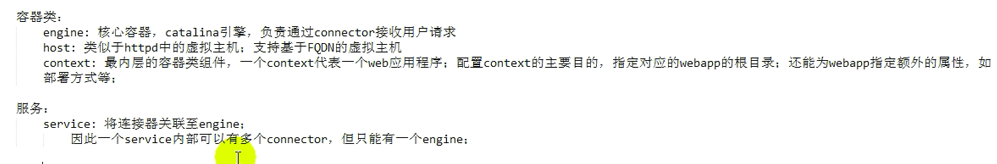
比如一个请求是8080

Service负责连接器和引擎建立关联关系



一个Service内部能关联多个引擎么？

---- 一个Service内部只能有一个引擎 但是可以有多个Connector



一个Server中可以多个Service 一个Service有多个Connector和一个Engine

Server顶级组件 ------ 

Tomcat的请求是由内部的线程来响应的

嵌套类组件 ---- 阀门 Valve ----拦截请求 并将其专职对应的web app之前 进行某种处理操作

Valve有很多种实现方式

Access log valve ----访问之前先记录日志

Remote address filter valve --- 基于IP的访问控制

可以用在任何容器中（容器类组件 Engine Host Context这三个）

Logger 日志记录器 ---- 定义用于记录组件内部的状态信息的

**用于除了Context之外的任何容器中**（容器类组件 Engine Host Context这三个）

Realm

主要功能是用来关联用户密码的数据库 ---- 实现认证和授权 ---- 关联的认证库 有两种

UserDatabaseRealm ----- 通过jndi自定义的用户认证库

MemoryRealm ---- 开启tomcat之后 这个conf/tomcat-users.xml会加载到内存中

JDBCRealm ---- 基于JDBC链接数据库查找用户

可以用于任何容器的组件中（容器类组件 Engine Host Context这三个）

关系如下：

虚拟主机内部可以使用Context

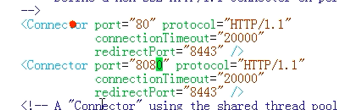
Server是顶级组件

Engine Host && Context 都是容器类组件

如果想监听在80端口 应该配置Http Connector的port为80



现在 我们定义多个Http连接器 一个监听在80 另一个监听在8080



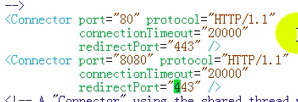
Connector --- port是连接器监听的地址

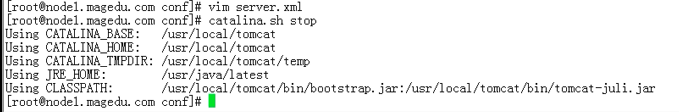
protocol是连接器监听的协议

connectionTimeout是连接超时的时间 单位是ms

redirectPort 就是 如果访问的时候 基于ssl的话 可以重定向至8443端口

----- 实际上是443端口







这样 加一个连接器 就能在多个端口上进行监听

8009

Tomcat有两类连接器 一类是基于http 另一类是基于ajp ---二进制协议 通常使用http反向代理用户请求至tomcat的时候 在httpd在tomcat之间的使用

如果仅仅监听ajp---- 那么必须使用apache做tomcat的前置机 做反向代理才可以 ----

如果使用nginx 那么 用户浏览器的http 就无法转换成二进制协议

----- 如何基于nginx 或者httpd反代

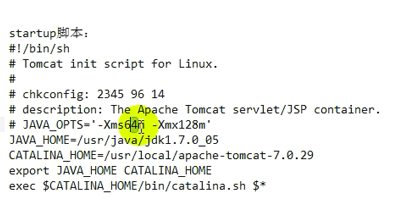
 T是tomcat N nginx --- A apache

php访问mysql用php的底层的驱动 链接到的mysql

Java访问mysql用的是JDBC ---- 底层是Java的驱动 连接到MySQL

------- 如何使用service tomcat进行启动

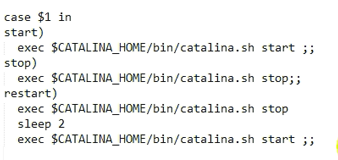
在服务器上



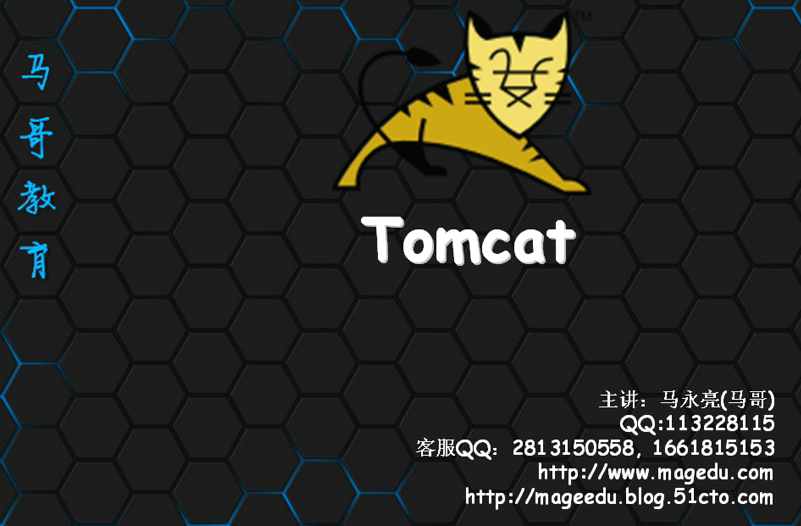
JAVA\_HOME没哟定义 可以写到这个脚本



写对应的脚本



59:22



Java是一种技术体系

包含四个重要组成部分

Java程序设计语言

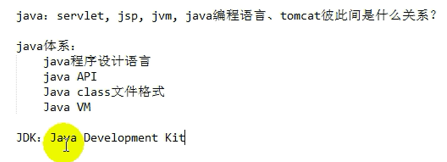
Java API ---- Java编程类库 ----- 为编程语言提供的各种既定的编程模块 不用自己写代码 ---提供好的可以直接搭积木的组成部分 ----有了类库 再+编程语言 就可以很快开发出来 所以API至关重要

Java VM ---- Java虚拟机 ---- Java独特 运行在Java虚拟机中 一个是API 一个是ABI

Java Class ----- Java类文件格式----- 所有利用API开发的源代码 都要转换成Java Class 转换之后 称为类文件 ------ 很多厂商生产自己的VM ---- 这样sun公司定出来了Java Class规范

这四者的关系

Java编程工具 + API + JVM +一些工具 就构成 jdk ----- java开发工具箱



JDK 只是包含了基础类库 核心类库 和工具类库

为了高级功能 需要额外的类库

JDK还有额外的两个类库 通常是面向桌面级别应用的 ---- java se

面向企业级应用 要添加更多企业级类库 --- java ee ----其中最著名的是 jsp servlet

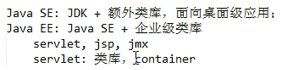
所以 Servlet和jsp就是Java ee中的两个类库

利用Servlet柔和在了监听在某个套接字上面的应用

提到Java的类内部 可以实现各式各样的应用

尽管如此 可以实现监听在某个套接字上的服务

Servlet是一个标准 本身是一个类库--- 如果实现这个标准 就是一个容器 Container



叫servlet容器

Servlet 容器可以让用户基于Servlet开发的程序 可以在这个环境中予以编译 和 执行 **可以理解为Java面向CGI的实现**

Servlet开发web应用的缺陷：就是所有包括html都要硬编码输出

介于asp php 将程序语言 --- 又为Javaee 引入一个新的类库 ---- 这个可以让Java的代码直接嵌入到html文档中

当这个文档提交之后 ----- 有一个Jasper jasper拿到这个jsp页面 或者柔和了Java和html的页面转化为早期Servlet的页面 最终由Servlet编译和执行

**Tomcat就是借助于jdk 补充了Servlet 和 Jsp的类库的开源实现**

---- 这个是java ee的不完整实现（还有JMS Javamail EJB）



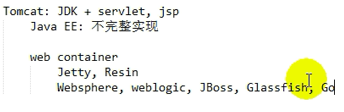
所以 运行web app没有问题

Tomcat就是web container (能够提供web app运行的环境 就叫web container)

Web container

开源 Jetty resoin

商业 Web Sphere --- Web logic --- Jboss ----- Glassfish ----- GonAs



WebSphere和Weblogic是Java ee的完整实现

Tomcat无非提供了servlet jsp 和 web app运行需要的组件

就是让自己的JVM启动 并让Tomcat在这个JVM中跑起来

回顾一下 tomcat安装完成之中 核心都在servlet.xml中