**一、BEAWeblogic的历史**

BEA WebLogic是用于开发、集成、部署和管理大型分布式Web应用、 网络应用和数据库应

用的Java应用服务器。将Java的动态功能和Java Enterprise标准的安全性引入大型网络应用的

开发、集成、部署和管理之中。

BEA WebLogic Server拥有处理关键Web应用系统问题所需的性能 、可扩展性和高可用性。

与BEA WebLogic Commerce ServerTM配合使用， BEA WebLogicServer可为部署适应性个性化

电子商务应用系统提供完善的解决方案。

BEA WebLogic Server具有开发和部署关键任务电子商务Web应用系统所需的多种特色和优

势，包括：

1)      领先的标准

对业内多种标准的全面支持，包括EJB、JSB、JMS、JDBC、XML和WML，使Web应用系统的实

施更为简单，并且保护了投资，同时也使基于标准的解决方案的开发更加简便。

2)      无限的可扩展性

BEA WebLogicServer以其高扩展的架构体系闻名于业内，包括客户机连接的共享、资源

pooling以及动态网页和EJB组件群集。

3)      快速开发

凭借对EJB和JSP的支持，以及BEAWebLogic Server 的Servlet组件架构体系，可加速投

放市场速度。这些开放性标准与WebGainStudio配合时，可简化开发，并可发挥已有的技能，

迅速部署应用系统。

4)      部署更趋灵活

BEA WebLogicServer的特点是与领先数据库、操作系统和Web服务器紧密集成。

5)      关键任务可靠性

其容错、系统管理和安全性能已经在全球数以千记的关键任务环境中得以验证。

6)      体系结构

BEA WebLogicServer是专门为企业电子商务应用系统开发的。企业电子商务应用系统需

要快速开发，并要求服务器端组件具有良好的灵活性和安全性，同时还要支持关键任务所必需

的扩展、性能、和高可用性。BEAWebLogic Server简化了可移植及可扩展的应用系统的开发，

并为其它应用 系统和系统提供了丰富的互操作性。

凭借其出色的群集技术，BEAWebLogic Server拥有最高水平的可扩展性和可用性。BEA

WebLogic Server既实现了网页群集，也实现了EJB组件 群集，而且不需要任何专门的硬件或

操作系统支持。网页群集可以实现透明的复制、负载平衡以及表示内容容错，如Web购物车；

组件群集则处理复杂的复制、负载平衡和EJB组件容错，以及状态对象（如EJB实体）的恢复。

无论是网页群集，还是组件群集，对于电子商务解决方案所要求的可扩展性和可用性都是至关

重要的。共享的客户机/服务器和数据库连接以及数据缓存和EJB都增强了性能表现。这是其它

Web应用系统所不具备的。

**二、Weblogic对各jdk版本的支持**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **标准** | **WLS 7.0** | **WLS 8.1** | **WLS 9.0** | **WLS 10.0** | **WLS 10.3** | **WLS 12.0** |
| **java** | 1.3 | 1.4 | 5 | 5 | 6 | 7 |
| **JAVA EE** | 1.3 | 1.3 | 1.4 | 5 | 6 | 6 |

**这个对jdk版本的支持一定要牢牢记住：**

比如说你手上有一个bea weblogic10.0，你拿它布署了只有jdk1.6才能解释的东西，就不要怪weblogic啦，**而且没有为什么的，到时不支持就是不支持。**

**反过来：**

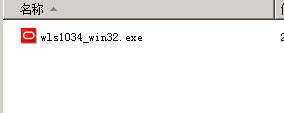
说你机器上装的是jdk1.6，你装个weblogic 8.x，嘿嘿，一样不支持。

Weblogic从8.x后被Oracle收购了，成了和OracleDatabase一样的可以在Oracle官网上免费下载的软件，而且不是那种trial version或者是development version，是真正的企业版，只要你拿它做实验、个人研究、玩不用在任何商业收费行为，是没有任何问题的，因此Weblogic从9.0后各版本可以从Oracle官网上免费下载，而且可以无限期使用没有license.jar这个东西再困扰你了([下载Oracle Weblogic](http://www.oracle.com/technetwork/middleware/ias/downloads/wls-main-097127.html))。

**三、Weblogic的安装**

**3.1 32位的weblogic**

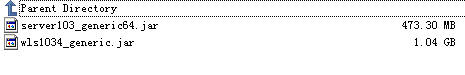
我们在Windows环境下一般安装32位的weblogic：



它一般是一个exe文件，双击即可安装。

**3.2 64位的weblogic**

如果你的机器是64位的操作系统或者是Linux/Unix操作系统，一般下载下来后都是一个.jar文件：



你安装时需要键入java –d64 –jar wls1034\_generic.jar。

如果是32位可以不需要这个-d64的参数，如果是64位机器一定一定一定要记得加这个-d64，要不然。。。。。。嘿嘿，它在安装界面出来后会抛一个out of memory的错。。。。。。偶晕！

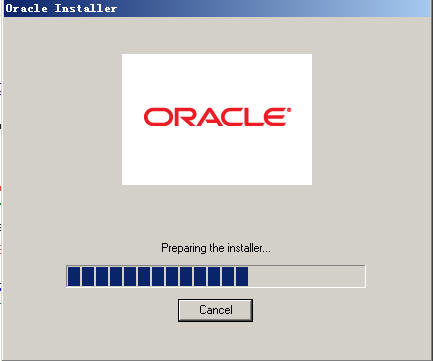
**3.3 安装**

确保你的JAVA\_HOME变量与path里包含有%JAVA\_HOME%\bin;%JAVA\_HOME%\jre\bin。

如果是Linux环境，你的JAVA\_HOME与path的设置如下：

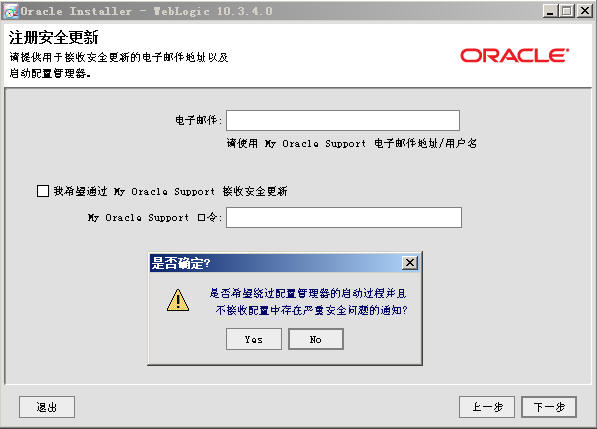
|  |
| --- |
| export $JAVA\_HOME=/opt/jdk/java6  export $path=$path:$JAVA\_HOME/bin:$JAVA\_HOME/jre/bin |

环境设置好后在weblogic安装时会找到你系统的JDK，我在此建议大家在安装Weblogic时尽量用标准的Oracle Sun的JDK，当然Weblogic也自带jdk，由于都是Oracle的产品，因此用Oracle Sun的JDK最标准。









**将“我希望通过My Oracle Support接收安全更新”项前的勾disable掉，然后在弹出对话框中选”Yes”。**



**我们选定制安装**

****

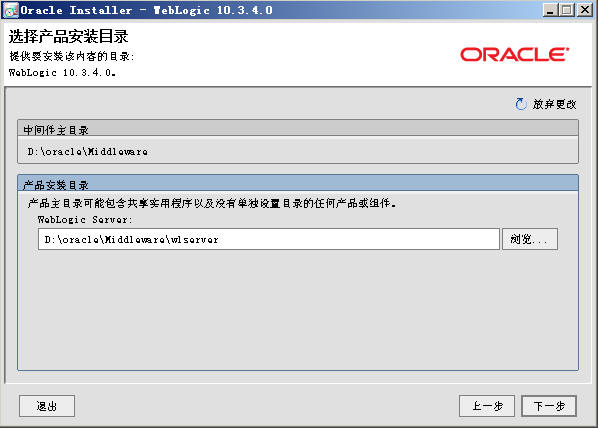
**我们只安装Weblogic Server，暂不安装OracleCoherence，在以后的网格与云里会讲这个Oracle Coherence。**



我们不要使用Weblogic自带的jdk也不用它自带的jrockit，我们点下面的[浏览]按钮，选中本地的jdk的主目录（即OracleSun的标准JDK）。

有人说用JRockit要吧获得更好的性能，还有人建议在32位机下使用JRockit可以突破jvm在32位机下只能使用到2GB内存的限制。。。。。。

如果真要用大内存还不如使用64位机呢，真折腾，而且JRockit里的jvm优化参数调优和标准的SUN的JDK是不同的。



注意这边的路径名，确保不要带有空格，下划线等特殊字符，java对特殊字符很忌讳的，有时一个编译错误不知道为什么，就是因为java home带有空格或者下划线一类的，你的目录名干净点，比如说：

有的人安装java就用默认安装路径，什么c:\programfiles\java\jdk1.6.3.2什么什么之类的。

你直接一个c:\jdk6这样一个名字，多干净？

这边再多啰嗦几句，tomcat的安装路径也是这样的，你直接d:\tomcat1, d:\tomcat2多干净，哪要这么多什么d:\apache\_jakarta\_tomcat6.0.12这样的东西啊，对吧！



选否。

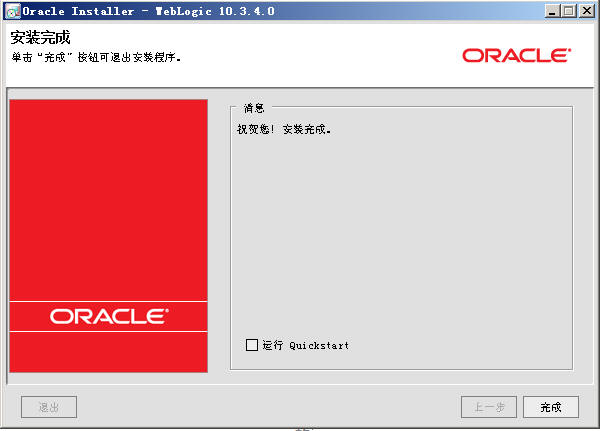


这里随便。

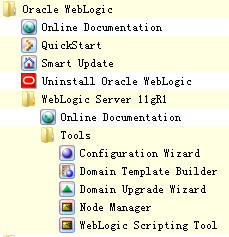


最后review一下，开始安装了。



****

装好了，把“运行Quickstart”选项前的勾disable掉，我们可以看到windows菜单里有了我们的weblogic的菜单了，如下图所示：



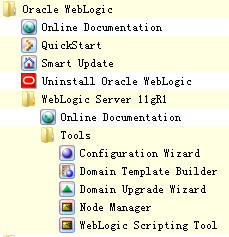
**四、使用Weblogic**

还记得我们的Tomcat如何布署我们的WAR应用程序吗？把WAR包或者是WAR目录结构扔到tomcat的webapps目录下，对吧！

Weblogic的webapps目录被称为domain，因此我们在装完weblogic后如果需要建立自己的webapp，就需要建一个domain。

**4.1 建立domain**

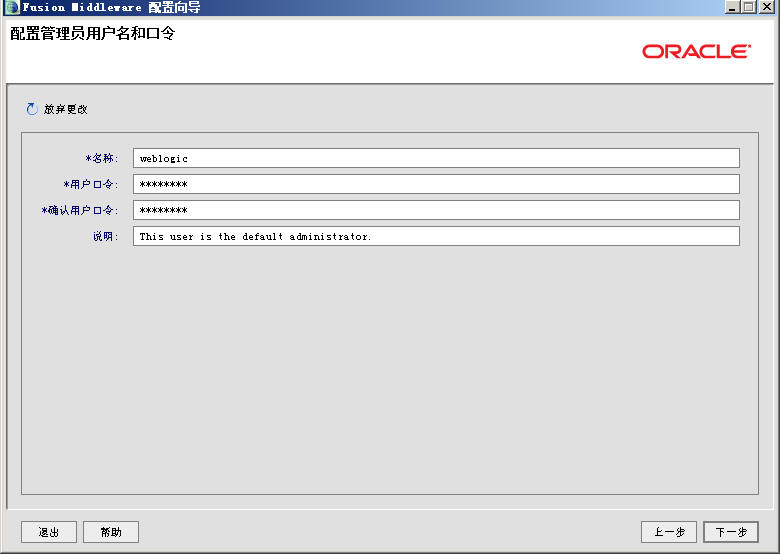
在我们的Windows菜单点选择Oracle Weblogic->Weblogic Server11gR1->Tools->Configuration Wizard



****

****

****

****

这个是Weblogic的admin console的登录信息，即weblogic装完后有一个网页版的主控制台，用于配置和布署我们的web app的应用的，它一般以:<http://ip:7001/console> 这样的形式登录，登录时需要用户名和密码，就如同tomcat的<http://localhost:8080/manager/html>这个console一样，我们的用户名叫”weblogic”，我们的密码就叫”password\_1”吧。

我们将我们这个用于实验的domain名取成mydomain。



这边来了，还记得我们在“（第四天）之Tomcat性能调优-让小猫飞奔”说到过的jvm的参数之：**java –server**与**java –client**的区别吗？

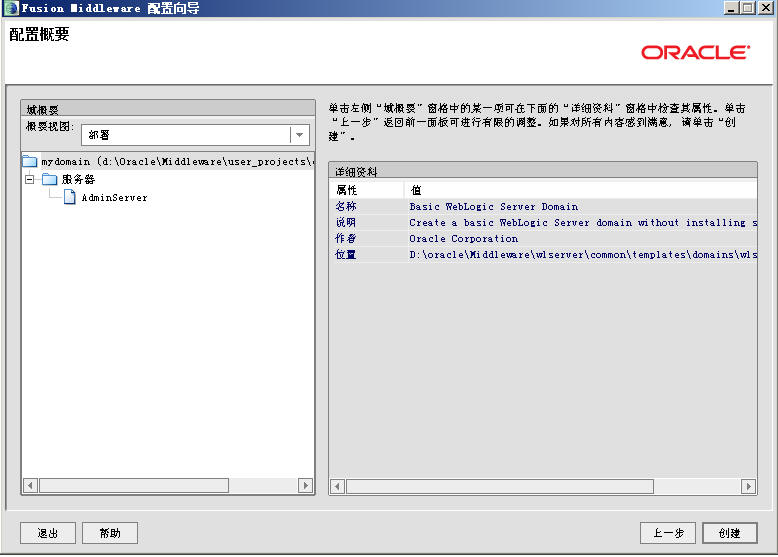
l   这边的”**开发模式**”就是你的weblogic将以java –client模式启动；

l   如果你选的是”**生产模式**” 那么你的weblogic将以java –server模式启。

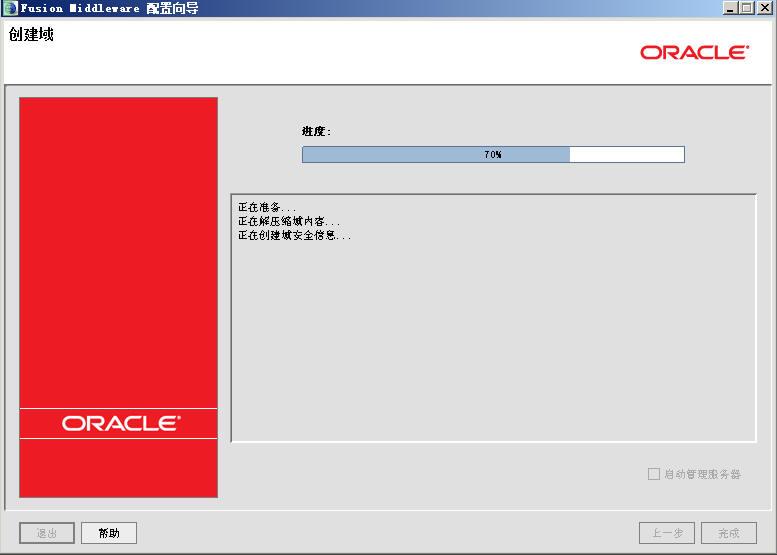
选”**生产模式**”点”下一步”。



这边我们一个不选，因为我们先不做weblogic的集群配置。点”下一步”。

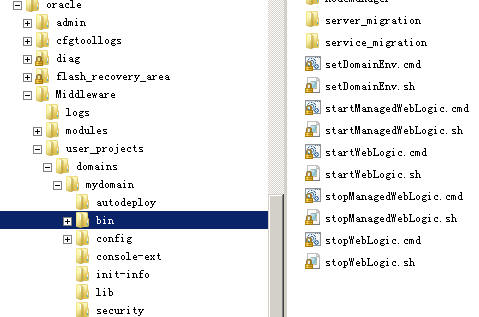


点[创建]按钮。





创建成功。点[完成]结束domain的配置。



Look，这个就是我们刚才创建的domain，现在开始让我脱离windows菜单来用命令行启动、停止weblogic吧，因为在大型应用中一般weblogic或者是IBMWebsphere都是运行在Unix小型机上或者是64位的Linux机器上的，因此大家要开始熟悉命令操作了。

**4.2.1 启动weblogic前的准备工作**

Weblogic的启动一般是调用”**weblogic安装目录**\**yourdomain**\”目录下的**startWebLogic.cmd**或者是**startWebLogic.sh**（UNX/LNX下）来启动weblogic的。

而startWebLogic会选调用”**weblogic安装目录**\**yourdomain**\**bin**\”目录下的**setDomainEnv.cmd**或者是**setDomainEnv.sh**(UNX/LNX下)来初始化环境变量的。

**先别急着启动，启动前我们来做一些优化**

我们使用文件编辑工具打开setDomainEnv.cmd文件，搜索：MEM\_ARGS这个关键字，可以找到下面这么一大陀（这是Weblogic11里的内容，在10里不一样的）

|  |
| --- |
| set WLS\_HOME=%WL\_HOME%\server  if "%JAVA\_VENDOR%"=="Sun" (           set WLS\_MEM\_ARGS\_64BIT=-Xms256m -Xmx512m           set WLS\_MEM\_ARGS\_32BIT=-Xms256m -Xmx512m  ) else (           set WLS\_MEM\_ARGS\_64BIT=-Xms512m -Xmx512m           set WLS\_MEM\_ARGS\_32BIT=-Xms512m -Xmx512m  )  set MEM\_ARGS\_64BIT=%WLS\_MEM\_ARGS\_64BIT%  set MEM\_ARGS\_32BIT=%WLS\_MEM\_ARGS\_32BIT%  if "%JAVA\_USE\_64BIT%"=="true" (           set MEM\_ARGS=%MEM\_ARGS\_64BIT%  ) else (           set MEM\_ARGS=%MEM\_ARGS\_32BIT%  )  set MEM\_PERM\_SIZE\_64BIT=-XX:PermSize=128m  set MEM\_PERM\_SIZE\_32BIT=-XX:PermSize=48m  if "%JAVA\_USE\_64BIT%"=="true" (           set MEM\_PERM\_SIZE=%MEM\_PERM\_SIZE\_64BIT%  ) else (           set MEM\_PERM\_SIZE=%MEM\_PERM\_SIZE\_32BIT%  )  set MEM\_MAX\_PERM\_SIZE\_64BIT=-XX:MaxPermSize=256m  set MEM\_MAX\_PERM\_SIZE\_32BIT=-XX:MaxPermSize=128m |

不难看懂，很简单，我们是32位的机器，因此我们就在32位所对应的jvm的设置里做一些调整。

1)      我们先把32位下的PermSize从原来的48m改成128m，把32位下的MaxPermSize从128m改成256m

|  |
| --- |
| set MEM\_PERM\_SIZE\_32BIT=-XX:PermSize=**128m**  if "%JAVA\_USE\_64BIT%"=="true" (           set MEM\_PERM\_SIZE=%MEM\_PERM\_SIZE\_64BIT%  ) else (           set MEM\_PERM\_SIZE=%MEM\_PERM\_SIZE\_32BIT%  )  set MEM\_MAX\_PERM\_SIZE\_64BIT=-XX:MaxPermSize=256m  set MEM\_MAX\_PERM\_SIZE\_32BIT=-XX:MaxPermSize=**256m** |

2)      更改jvm 的heap size

|  |
| --- |
| if "%JAVA\_VENDOR%"=="Sun" (  set WLS\_MEM\_ARGS\_64BIT=-Xms256m -Xmx512m  set WLS\_MEM\_ARGS\_32BIT=-Xms256m -Xmx512m  ) else (  set WLS\_MEM\_ARGS\_64BIT=-Xms512m -Xmx512m  set WLS\_MEM\_ARGS\_32BIT=-Xms512m -Xmx512m  ) |

上面这段根据你们自己的实际情况将WLS\_MEM\_ARGS\_32BIT=后的值进行修改。

**注意：**

l   Xms与Xmx设成一样大小

l   修改Xmx值前请先用java –Xmx你想要的sizem –version进行测试

我们这个例子里就暂都改成512m就够了。**一定不要在数值后面漏打”m”啊！**

|  |
| --- |
| if "%JAVA\_VENDOR%"=="Sun" (           set WLS\_MEM\_ARGS\_64BIT=-**Xms512m** -Xmx512m           set WLS\_MEM\_ARGS\_32BIT=-**Xms512m** -Xmx512m  ) else (           set WLS\_MEM\_ARGS\_64BIT=-Xms512m -Xmx512m           set WLS\_MEM\_ARGS\_32BIT=-Xms512m -Xmx512m  ) |

找到下面这一段：

|  |
| --- |
| set MEM\_ARGS\_64BIT=%WLS\_MEM\_ARGS\_64BIT%  set MEM\_ARGS\_32BIT=%WLS\_MEM\_ARGS\_32BIT% |

现在可以加入我们的jvm调优参数了，如下：

|  |
| --- |
| set MEM\_ARGS\_64BIT=%WLS\_MEM\_ARGS\_64BIT% **-Dweblogic.threadpool.MinPoolSize=25 -Dweblogic.threadpool.MaxPoolSize=1000** -Xss512k -XX:+AggressiveOpts -XX:+UseBiasedLocking -XX:PermSize=128M -XX:MaxPermSize=256M -XX:+DisableExplicitGC  -XX:MaxTenuringThreshold=31 -XX:+UseConcMarkSweepGC -XX:+UseParNewGC  -XX:+CMSParallelRemarkEnabled  -XX:+UseCMSCompactAtFullCollection -XX:LargePageSizeInBytes=128m  -XX:+UseFastAccessorMethods  -XX:+UseCMSInitiatingOccupancyOnly -Djava.awt.headless=true    set MEM\_ARGS\_32BIT=%WLS\_MEM\_ARGS\_32BIT% **-Dweblogic.threadpool.MinPoolSize=25 -Dweblogic.threadpool.MaxPoolSize=1000**  -Xss512k -XX:+AggressiveOpts -XX:+UseBiasedLocking -XX:PermSize=128M -XX:MaxPermSize=256M -XX:+DisableExplicitGC  -XX:MaxTenuringThreshold=31 -XX:+UseConcMarkSweepGC -XX:+UseParNewGC  -XX:+CMSParallelRemarkEnabled  -XX:+UseCMSCompactAtFullCollection -XX:LargePageSizeInBytes=128m  -XX:+UseFastAccessorMethods  -XX:+UseCMSInitiatingOccupancyOnly -Djava.awt.headless=true |

除了：

l   **-Dweblogic.threadpool.MinPoolSize=25**

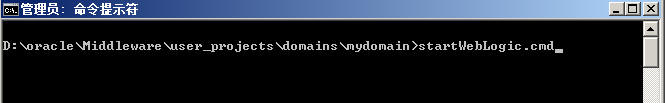
l   **-Dweblogic.threadpool.MaxPoolSize=1000**

其它的参数均已经在“（第四天）之Tomcat性能调优-让小猫飞奔->**2.2 Tomcat启动行参数的优化**”中详细擅诉过了。

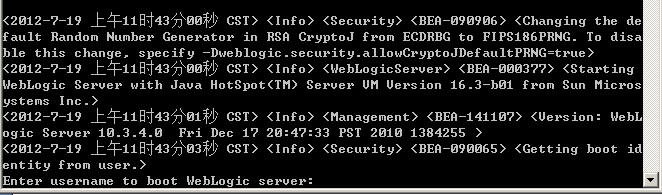
那么上面这两个值是什么意思？

这两个值就是weblogic的“应用程序线程池”，它的大小决定了你的应用程序服务器的“吞吐量”或者说是“容量”，以前在Weblogic中可以通过登录weblogic 的<http://yourip:7001/console>后显视的去做调整，而在weblogic9.x后这两个值变成了隐视的了，因此很多人不知道如何去调Weblogic里的线程池的大小，是因为根据在console界面中找不到这两个值来让你修改的，所以我们在weblogic的启动参数中，可以手工的加上，如果你不加也行，weblogic有一个默认的线程池的大小的。一般为最小25最大200。依据你的服务器的性能可以加大这个值。

**4.2.2 启动weblogic**

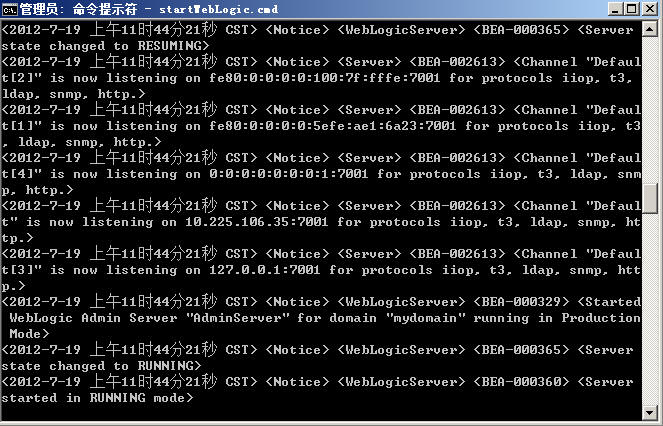


键入上述的命令后回车



Weblogic启动控制台会提示要你输入用户名和密码，即我们在“创建domain”时输入的weblogic/password\_1。

我们输入相应的用户名和密码后，一直看到控制台输出这样一行信息后，即代表我们的weblogic启动正常无误了。



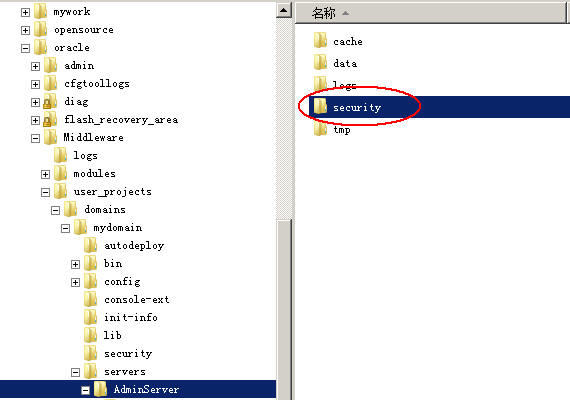
**4.2.3 如何让Weblogic每次启动时不用再输入用户名和密码**

每次启动Weblogic时都要输入用户名和密码，假设我的Weblogic是自动随着我的服务器的启动而启动，那么如果此时因为是无人值守，鬼才来输这个用户名和密码啊，是不是？

所以，我们先文本编辑工具先来制作一个文件，叫boot.properties，内容很简单，如下所示：

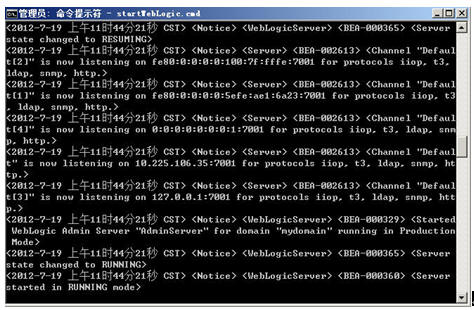
|  |
| --- |
| username=weblogic  password=password\_1 |

然后我们来到我们的domain的目录下



如：”D:\oracle\Middleware\user\_projects\domains\mydomain\servers\AdminServer”目录下，手工建一个security的目录，将刚才新建的boot.properties文件，拷入此目录后，重启weblogic。

此时你会发觉，weblogic再也不会提示要求你输入weblogic管理台的用户名和密码了，等到启动成功后，即看到这样的输出行：



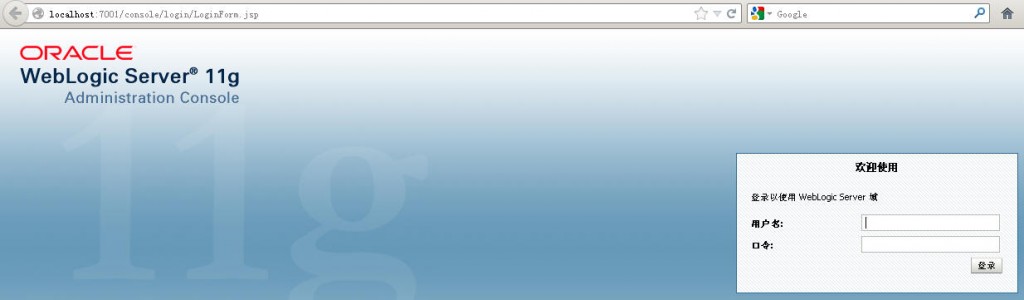
我们来到” D:\oracle\Middleware\user\_projects\domains\mydomain\servers\AdminServer\security”目录，打开刚才拷入的boot.properties，它的内容已经自动变成了下面这样了：

|  |
| --- |
| #Thu Jul 19 11:54:14 CST 2012  password={AES}anBOrQK1z+ErQKBJSKxlN1t2qv+hxmhyzhQW6S4WnLQ\=  username={AES}D2z8M4xrI3E981Q6u4cfbgAMlDcd23LVgLxvTxARc6k\= |

Weblogic把它给加密了，因此只有装Weblogic的那个人即SystemAdmin才真正知道Weblogic控制台的登录信息，这样就很安全了。

**4.2.4 配置Weblogic的数据源**

在Weblogic运行的状态下我们用ie进入Weblogic的adminconsole，输入<http://localhost:7001/console>



输入我们在创建domain时的用户名与密码



记得要把“锁定并编辑”变成“释放配置”的状态，要不然，你们看，右边的数据源的编辑按钮全部是灰色的。



点[新建]按钮选“一般数据源”。

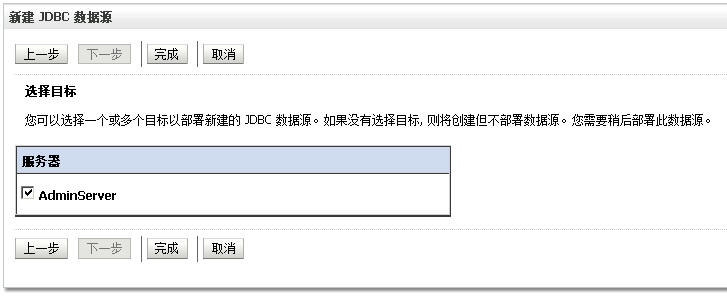


我们使用oracle来配置我们的jdbc数据源。





填入相应的数据库的连接信息下一步下一步到如下这个界面：



把数据源需要起作用的服务器即“target”要选中，因为在Weblogic里可以用一个控制台控制多个Server，那么你的数据源是针对哪台server起作用呢？你需要告诉Weblogic。

点击[完成]



**4.2.5 调优Weblogic的数据源**



记得点击控制台左上角的“激活并保存编辑修改”，要不然刚才所做的操作全部作废，保存后你会看到如上的信息。

点击我们刚才新建的数据源，”jdbc\_cbbsds”



进入此界面后点击左上角的“锁定并编辑”按钮，右边就是我们需要设置的值了。我们对“连接池”进行调整：



ü   初始容量

即最小数据连接数

ü   最大容量

最大数据连接数

ü   容量增长

每次增加的连接数

ü   语句高速缓冲类

为缓存选择 least-recently-used (LRU) 算法；这将从缓存中删除很少使用的语句。

还有一个值为Fixed。

LRU算法在有新的语句到来时，将最不经常被用得语句调整出缓存。FIXED算法为先进先出的算法。

一般应用我们使用LRU。

ü   语句高速缓冲大小

增加Statement Cache Size对于大量使用PreparedStatement对象的应用程序很有帮助,WebLogic能够为每一个连接缓存这些对象,此值默认为10。在保证数据库游标大小足够的前提下,可以根据需要提高Statement Cache Size。比如当你设置连接数为25,Cache Size为10时,数据库可能需要打开25\*10=250个游标。

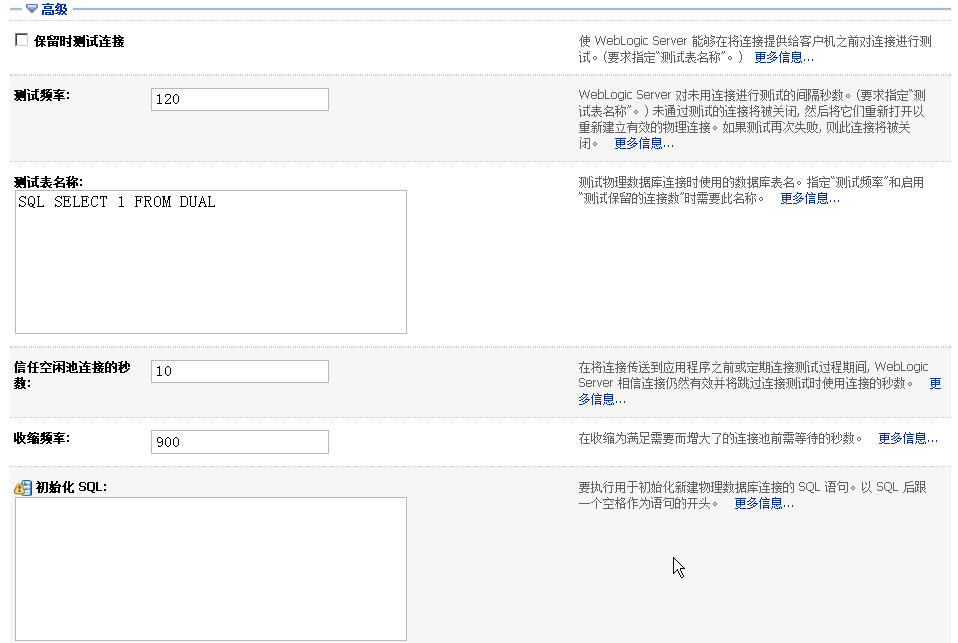
那么我们的oracle数据库默认最大游标假设为1000，我们最大连接数设了25，那么这个statement cache的值就应该为：40，因为25\*40=1000。

这个值不能过大，如果是开发环境上可以把这个值设为0，在生产环境上需要根据你oracle的max cursor数量进行实际设定，**不是说越大越好的，千万不要拍脑袋啦！！！**

**注意：**

和jvm的heap size一样，将“初始容量”与“最大容量”设成相等，根据oracle默认最大支持150个processor，那么oracle默认最大的连接数为max processor\*1.1即165，那么我们就给最大25最小25就够了，必竟我们只是一个测试。

**配完上面的参数后点一下保存，然后我们点“高级”这个链接。**



高级有很多，我们一项项来过。

ü   保留时测试连接

请把此项打上勾。

我们的数据库连接池会不断的扫描我们的jdbc的连接使用情况，当发觉不用的jdbc连接时会“回收”掉它，因此你必须把这个选项的勾打上，要不然，weblogic的数据库连接池不会去扫描你的不用的连接或者长时间不活动的连接，此选项将有助于你系统的稳定性。比如说：有时我们碰到一些忘关的连接或者废弃的连接，那么weblogic将会自动为我们“回收”这些连接，以使得数据库连接池被充分利用。

ü   测试频率

这个值与上面的“保留时测试连接”是关联的，这个值默认为120秒，即2分钟，weblogic会产生一条探测语句“SQL SELECT 1 FROM DUAL”来探测你的oracle数据库中的连接，这个值绝对不能太小比如说：60秒，或者更有甚者设了个5秒，那么你的weblogic会不停的向oracle发送“SQL SELECT 1 FROM DUAL”语句，并且造成大量的inactive状态的数据库连接，最后甚至把你的oracle数据库给活活搞死，搞死搞死。。。。。。而且是“挺尸”就是数据库服务器动也动不了了，真的，不信你试试！J

一般在production环境上，如果相对稳定，可以把这个值，设大点，比如说：15分钟，或者20分钟探测一次。

ü   测试表名称

就是“探测”数据库的语句，我们就用它的默认的，挺好，不变

ü   初始化 SQL

保持默认，我们这边没有特殊的初始化的步骤。

ü   重试创建连接的频率

尝试建立数据库连接的间隔秒数。

如果不设置此值，则在数据库不可用的情况下，创建数据源的操作将失败。如果已设置此值且在创建数据源时数据库不可用，则在您指定的秒数之后，WebLogic Server 将重新尝试在池中创建连接，并会不断尝试创建连接，直到创建成功。

如果设置为 0，则将禁用连接重试。

         我们此处就保持为0，没有特殊需求好failout的相关的需求（这个要根据客户的需求来定的）。

ü   登录延迟

         创建每个物理数据库连接前的延迟秒数。此延迟支持不能快速连续处理多个连接请求的数据库服务器。

在初始数据源创建及数据源生命周期内，每当创建物理数据库连接时都会有此延迟。

我们保持它为0。

ü   非活动连接超时

保留连接处于不活动状态的秒数，该秒数过后WebLogic Server 将收回该连接并将其释放回连接池。

可以使用“非活动连接超时”功能收回泄漏的连接 - 未由应用程序显式关闭的连接。请注意，不能使用此功能来代替正常关闭连接。

如果设置为 0，则将禁用此功能。

如果上面的“保留时测试连接”、“测试频率”，“测试表名称”没有设置或者是“保留时测试连接”没有勾选，那么这个选项是没有用处的，因为它就是我们上面提到的，Weblogic会自动为我们释放一个超时的连接，这个超时要和oracle数据库中的sql\_time\_out一致，比如说：60秒，我们回收一个超时了的inactive的连接。

这个值的设定对我们数据库的稳定性和性能也有着很大的影响。

ü   最大等待连接数

等待保留数据源的连接池中的连接时会阻塞线程的最大并发连接请求数。

来看它的默认值“2147483647”，这边可千万不要用默认值啊，要不然你的数据库也一样会在出现阻塞或者是死锁时活活被搞死，搞死搞死。。。。。。而且是“挺尸”，为啥，一个阻塞了没事，这下好，一个阻塞了，然后它允许后面的连接继续进，进啊进啊进，你人又不可能天天时时分好盯着个oracle的监控端看着它，好了，进到2147483647个连接时，数据库不死了才怪呢。

一般太平点的，给个合理的数量就可以了，比如说100或者是200，不要超过oracle数据库提供的最大max session数就可以了。

ü   连接保留超时

保留连接池中的连接的调用超时之前经过的秒数。

如果设置为 0，则调用永远不会超时。

如果设置为 -1，则调用会立即超时（对数据库来说，这个值是最安全的，SQL语句一被调用就超时，然后经过上面的“非活动连接超时”后，会被weblogic的数据库连接池给回收。

ü   语句超时

此时间后当前正在执行的语句将超时。

StatementTimeout依赖于底层 JDBC 驱动程序支持。WebLogic Server 使用 java.sql.Statement.setQueryTimeout() 方法将指定的时间传递给 JDBC 驱动程序。如果 JDBC 驱动程序不支持此方法，可能会引发异常错误并导致忽略超时值。

值为 -1 时禁用此功能。

值为 0 表示语句不会超时。

最小值：-1

最大值：2147483647

**注：**

**下面三个选项保持默认选择，没有特殊应用时不需要设置。**

ü   忽略正在使用的连接

即便仍在使用从池获取的连接，也能够将数据源关闭。

ü   固定到线程

PinnedToThread 是一个能够改善性能的选项，它启用执行线程以保持已入池的数据库连接，即使在应用程序关闭逻辑连接之后。

启用 PinnedToThread 后，WebLogicServer 将在应用程序首次使用某个执行线程保留连接时将连接池中的一个数据库连接固定到此线程。当应用程序使用完此连接并调用 connection.close() (此方法在其他情况下会将连接返回到连接池) 时，WebLogic Server 会将该连接保留给执行线程，而不会将其返回连接池。当此后某个应用程序使用相同的执行线程请求连接时，WebLogic Server 将提供此线程已保留的这个连接。

使用 PinnedToThread，当多个线程尝试同时保留一个连接时，连接池上不会发生锁定冲突，对于试图在有限个数的数据库连接中保留同一个连接的线程，也没有冲突。

如果应用程序使用相同的执行线程同时保留连接池中的多个连接，WebLogic Server 将创建附加的数据库连接并同样将它们固定到此线程。

ü   已启用删除受影响的连接

指定在应用程序使用底层供应商连接对象之后是否从连接池中删除连接。

如果禁止删除已占用的连接，必须确保该数据库连接适合由其他应用程序重用。

设置为真 (默认设置) 时，应用程序关闭逻辑连接后不会将物理连接返回给连接池。将关闭并重新创建此物理连接。

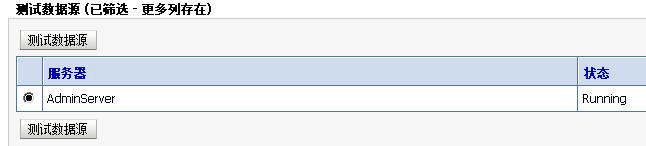
设置为假时，应用程序关闭逻辑连接后，物理连接将返回到连接池并可由该应用程序或其他应用程序重用。

全部设置完了，不要忘了点页面最下端的[保存]按钮，并且点左边菜单上方的[激活并更改]按钮，要不然你刚才的一切设置均作废。

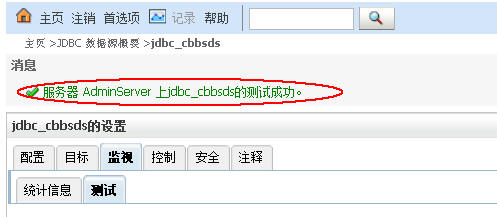


在保存后我们来点“监视”这个tab





点“测试”



Look, 测试成功。重启Weblogic（只有重启Weblogic才会使得“高级”配置里的参数生效）。

**怎么重启：**

ü   Windows下

直接ctrl+c，然后重新启动

ü   Linux下

尽量使用stopWebLogic.sh或者使用ps –ef |grep java，查到weblogic所属的进程，然后使用：

kill -9 进程号

来杀，这个太暴力了！！！

**4.2.6 部署web应用**

Weblogic是真正的J2EE App Server，它不仅可以布署WAR应用，还可以布署EAR应用包括EJB，而Tomcat是做不到的，它只能布署WAR应用，如果要布署EAR或者EJB需要安装和配置额外的插件。

确保你的web应用中web.xml里绑定的数据源的jndi名与weblogic里设定的数据库连接池的jndi名一致

|  |
| --- |
| <resource-ref>                     <description>oracle db</description>                     <res-ref-name>jdbc/cbbsds</res-ref-name>                     <res-type>javax.sql.DataSource</res-type>                     <res-auth>Container</res-auth>                     <res-sharing-scope>Shareable</res-sharing-scope>  </resource-ref> |

你布署的可以是一个“打碎了的符合war格式的目录”也可以是一个war包。



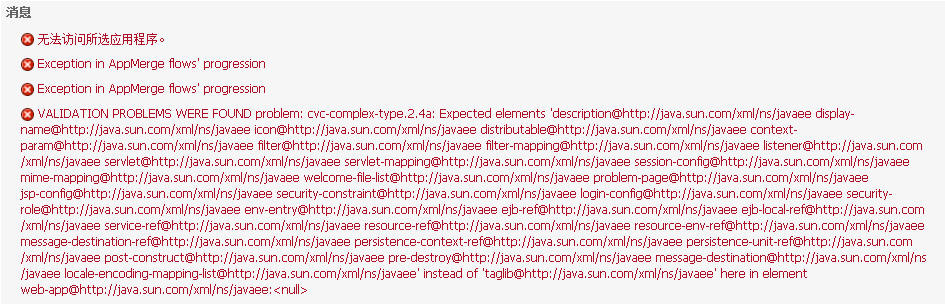
点左边菜单的“布署”，然后点“锁定并编辑”，然后右边的按钮就可以用了。

此时，点[安装]，定位到你的要布署的WAR



点下一步。

然后出错了，嘿嘿！



打开WAR里的WEB-INF\下的web.xml看啊看，看到这一行：

|  |
| --- |
| <taglib>                              <taglib-uri>/myPageTag</taglib-uri>                              <taglib-location>/WEB-INF/tld/myPageTag.tld</taglib-location>                     </taglib> |

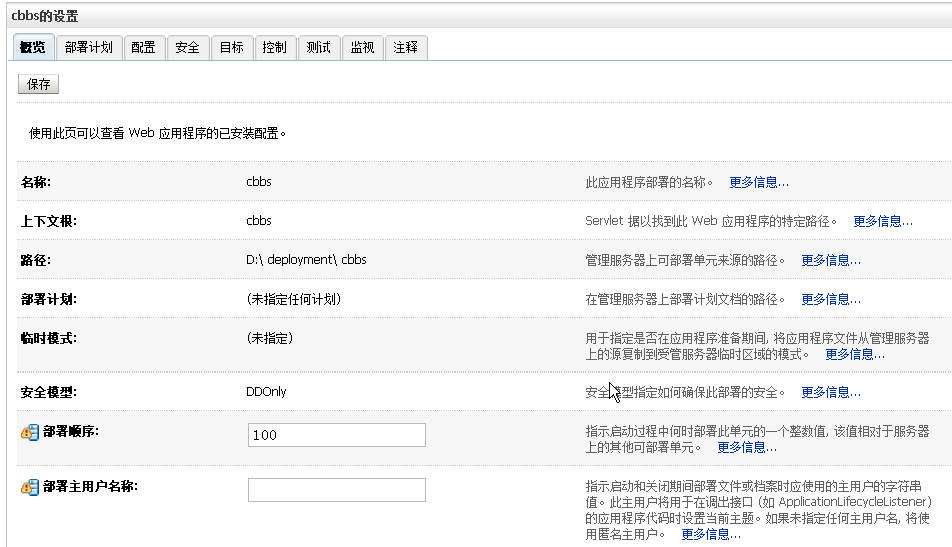
哦，知道了，weblogic的jstl校验很严格，原来在tomcat里工程跑的那叫一个欢呀，跑到weblogic里了，把这段改成下面这样的形式吧：

|  |
| --- |
| **<jsp-config>**                     <taglib>                              <taglib-uri>/myPageTag</taglib-uri>                              <taglib-location>/WEB-INF/tld/myPageTag.tld</taglib-location>                     </taglib>  **</jsp-config>** |

看到区别没有？在<taglib>外要多一对<jsp-config>，嘿嘿，仔细，仔细！

改完后保存web.xml，重新回weblogic布署。

然后下一步下一步，下啊下啊下。。。再往下，到了这一步：



点[保存]。



点“激活更改”。**在“产品模式（production mode）”下，每次作过什么动作都必须“激活更改”一下，才能生效**，一定记住，下次就再也不提了。

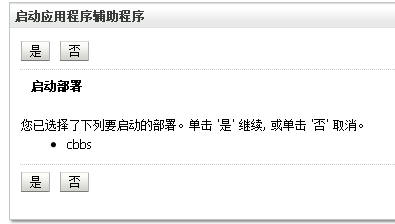


布署好后，看这个状态，这边的状态是“准备就绪”，准备就绪不代表此应用可以用，要使它的状态变成激活状。

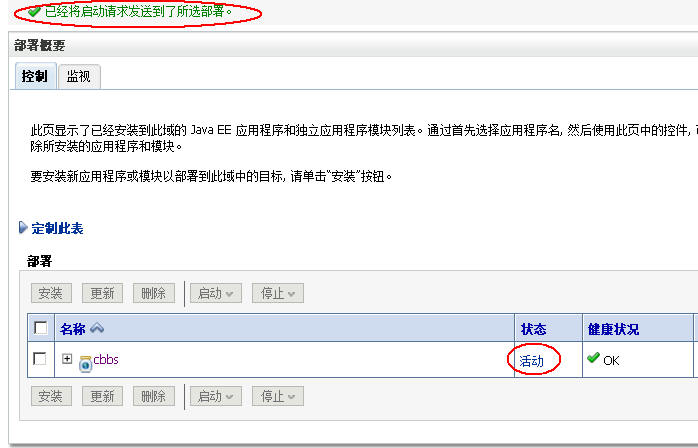
选中此应用前的checkbox



选“为所有请求提供服务”。



选“是”。



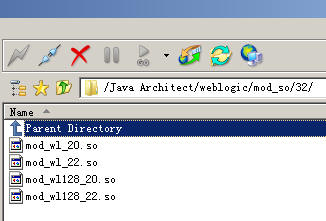
就代表部署成功了，你此时可以打开一个IE输入：<http://localhost:7001/cbbs>，登录后一切正常，Weblogic下WAR应用布署成功。



**五、整合Weblogic与Apache**

还记得我在第一天的“之Apache整合Tomcat”中讲到过，apache和tomcat是通过mod\_jk来派发java请求给tomcat的吗？那么一样，weblogic是通过mod\_wl\_apache的版本号.so来做同样的事。

从我们的FTP中可以得到这些.so文件（也可以直接上oracle官网上下这些.so文件，免费公开的），我们来看。



1)      拷贝mod\_wl\_22.so文件至apache的安装目录下的modules目录，如我的是：D:\tools\httpd\modules。

2)      打开httpd.conf文件，还记得我们在第一天中的httpd.conf文件吗？找到<VirtualHost>，在它的上面，敲一个回车，加入下面这一陀东西

|  |
| --- |
| LoadModule weblogic\_module modules/mod\_wl\_22.so  <IfModule mod\_weblogic.c>     WebLogicHost localhost     WebLogicPort 7001  **MatchExpression** /cbbs/WEB-INF  **MatchExpression** /cbbs/\*WEB-INF  **MatchExpression** /cbbs/\*.action  **MatchExpression** /cbbs/servlet/\*  **MatchExpression** /cbbs/\*.jsp  **MatchExpression** /cbbs/\*fckeditor/editor/filemanager/connectors/\*.\*  **MatchExpression** /cbbs/fckeditor/editor/filemanager/connectors/\*     WLLogFile logs/wlproxy.log  </IfModule> |

看这个**MatchExpression**，还记得mod\_jk里的JKMount吗？到了Weblogic和apache映射就成了MatchExpression了。

**另：**

**在production环境上应该WLLogFile logs/wlproxy.log这一句注释掉，避免较多的log影响服务器的性能。**

**注意在两个地方要把index.jsp放在首要位置：**

**1)       Web.xml文件中**

|  |
| --- |
| <welcome-file-list>  **<welcome-file>index.jsp</welcome-file>**  </welcome-file-list> |

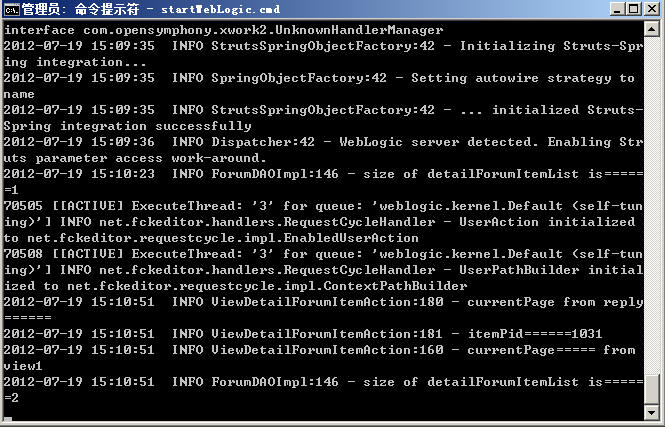
**2)       Httpd.conf文件中**

|  |
| --- |
| DirectoryIndex **index.jsp**index.html index.htm |

启动ie，输入：<http://shnlap93:7001/cbbs/>



再来看weblogic控制台：



Html等静态内容被apache解释了，而java的内容被weblogic解释了，这就是apache+weblogic。

后一天，我们将讲述利用apache+weblogic来实现集群，见识一下商业级App Server的强大吧！