**通向八段的道路**

**第十五天**

lifetragedy

目录

[一、 IBM Websphere 3](#_Toc332707073)

[二、 基本概念 3](#_Toc332707074)

[三、 WAS安装 4](#_Toc332707075)

[四、 WAS的启动与停止 19](#_Toc332707076)

[4.1 启动WAS 19](#_Toc332707077)

[4.2 停止WAS 20](#_Toc332707078)

[五、 管理WAS 21](#_Toc332707079)

[5.1 优化WebSphere 21](#_Toc332707080)

[5.1.1 修改系统打开文件数（windows系统忽略） 21](#_Toc332707081)

[5.1.2优化数据库连接池 21](#_Toc332707082)

[5.1.3更改Web容器线程池大小 22](#_Toc332707083)

[5.1.4 更改会话超时和启用servlet高速缓存 23](#_Toc332707084)

[5.1.5更改JVM参 数 23](#_Toc332707085)

[IBM JDK的JVM常用参数 25](#_Toc332707086)

[5.1.6 更改ORB服务 26](#_Toc332707087)

[5.1.7 去除应用程序的类重新装入选项 26](#_Toc332707088)

[5.2 创建数据源 26](#_Toc332707089)

[5.3 在WAS内布署应用 35](#_Toc332707090)

[5.3.1 application.xml文件 35](#_Toc332707091)

[5.3.2 使用Ant打包 36](#_Toc332707092)

[build.properties 36](#_Toc332707093)

[build.xml 36](#_Toc332707094)

[使用IBM WAS自带的JDK对工程进行编译 40](#_Toc332707095)

[5.3.3 布署EAR 40](#_Toc332707096)

# IBM Websphere

02年开始接触EJB1.x时,当时有一本巨肥厚无比的书叫作”ejb从入门到精通(master ejb2.0)”，红皮的，wrox公司出版的。该书带有1张光盘，光盘里有3个App Server。

1. Jboss2.2.1
2. Weblogic6.1
3. IBM Websphere Application Server5.x

从那时起开始知道，哦，原来这3大厂商是做j2ee容器的。果然，时至今日这三大容器还是处于世界上无可争议的地位。

因此，我们前面讲了tomcat+apache（集群）, weblogic+apache（集群），Websphere的基本应用与概念也是逃不了的。

Websphere现在已经在IBM的任何一个产品中占据了主要的地位，几乎各个IBM的企业级应用产品如：IBM iLog Jrulz, IBM Lombardi, IBM Websphere Portal Server, IBM TAM等等等，这些产品在安装时第一个就是将IBM Websphere Application Server或者是IBM Websphere Application Server Network Deployment(WASND)先安装在你的服务器上。

因此我们很有必要去好好学习和掌握IBM WAS的安装，与IBM HttpServer(被IBM改进过的商业级Apache Server)的集成以及WASND（用作集群的IBM WAS安装版）。

# 基本概念

IBM系列产品使用的是IBM自身带的JDK，这个JDK也是被IBM进行过了改进的，被称为IBM JDK。

该JDK和我们传统的JDK即ORACLE-SUN的不太一样，不要试图去用前面所讲的一系列的Oracle Sun的JDK里的优化参数去应用在IBM JDK上，这是完全不对的。

而且，IBM JDK与IBM Websphere Application Server(WAS)之间的整合有严格要求：

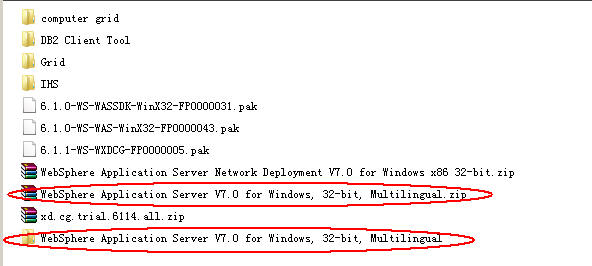
IBM WAS6.X可以支持JDK1.5，如果你的工程里有用到JDK1.6的特性，那么在IBM WAS6.X上不要想跑起来。

IBM WAS7.x即后续版本可以支持JDK1.6，但是，你不要认为在IBM WAS6.x上能够运行的跑到IBM WAS7.x上一样可以运行，因为有些工程用到了IBM JDK的特性，比如说有些工程用到了IBM JDK中的webservice，结果在6.x上跑得挺好，到了7.x上死活跑不起来，没办法，这就是IBM JDK，它公司大，所以修改了一些东西为它自己的利益所服务。而不像我们的Oracle Sun的JDK目前还是遵照着标准，高版本一定兼容低版本里编译出来的东西。

说这个问题就是因为以前曾经发生过，有项目组用ssh2(struts2+spring3+hibernate3)，结果客户的环境是IBM WAS6.X，而项目开发用的是Tomcat6.x与jdk1.6，结果在实际production环境上布署失败，虽然后来通过补丁等一系列手段可以在IBM WAS6.X上布署起工程来，但是前面已经给客户带来了极不好的影响。

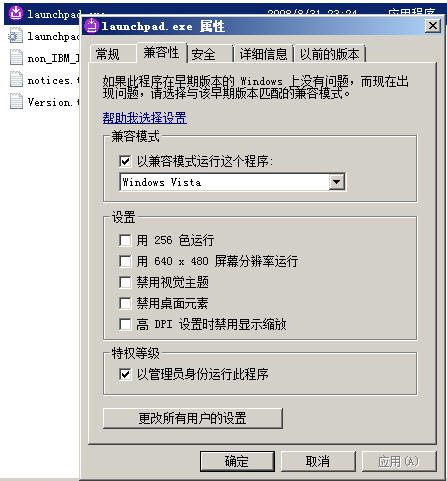
为此，IBM公司还出了一套绑定其WAS的开发工具即：IBM RAD，IBMRAD7.0只支持WAS6.X，而IBMRAD7.5即后续版本可以支持到WAS7.0并支持JDK1.6的相关特性，因此如果你的工程环境全部是IBM的WAS相关环境（WAS很贵，三大产品中最贵的要属这个WAS了，买得起这个的客户一般非银行即大型保险公司），那么也请你最好使用IBM RAD去开发相关的工程吧，让客户掏钱就是了！

# WAS安装

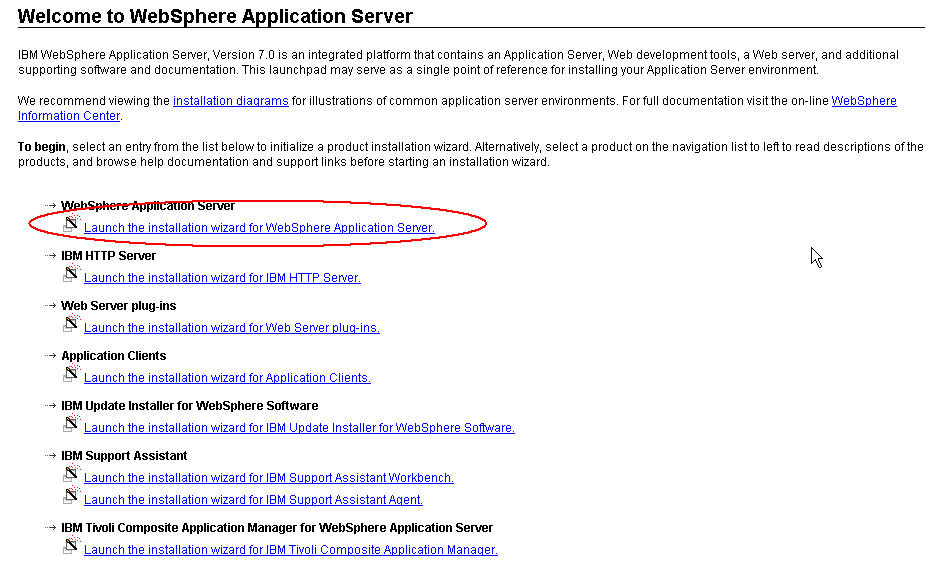


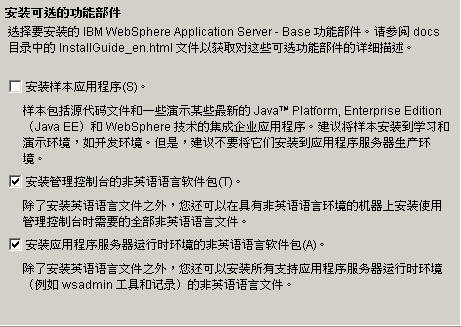
WAS的安装包里一般有一个launchpad.exe或者是在下载后的WAS安装包.zip解压后会有一个was目录，该目录中有一个install.exe文件，目前的IBM WAS6.x系列不能在Win7下正常运行安装程序，因此如果你的机器是win7，请安装下面这个设置截图来修改WAS安装程序的相关属性，即可让WAS6.X系列的安装程序也能够在win7下顺利运行并安装了。

如果你是WAS7.X ，那么完全可以在win7下顺利安装，不需要修改任何属性。



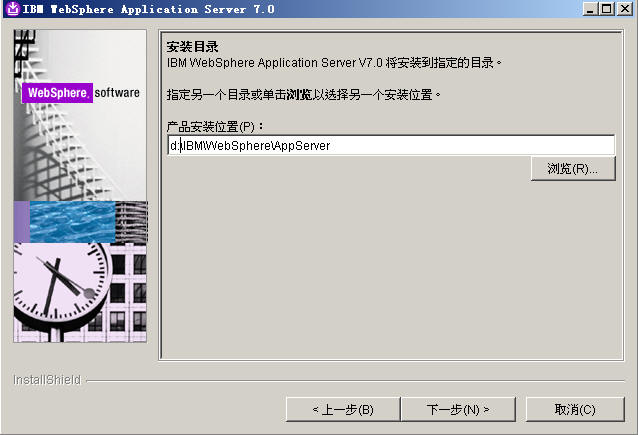
**Launchpad运行后的主界面**



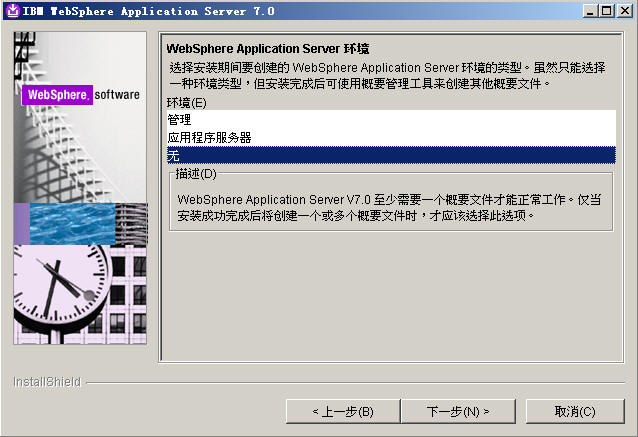


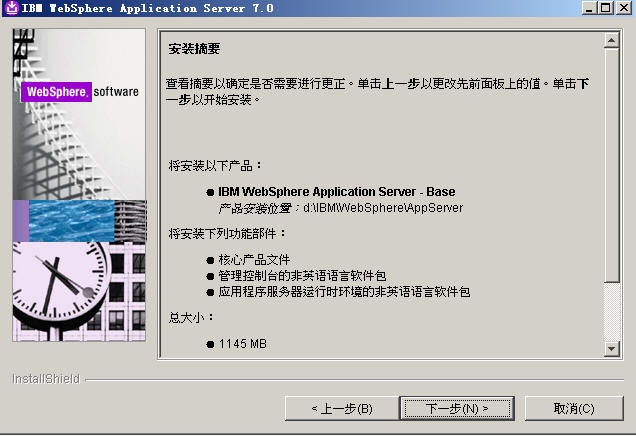
**下一步**

设置你的WAS的安装目录，这边这个”d:\IBM\WebSphere\AppServer”又被称为“WAS.HOME”。

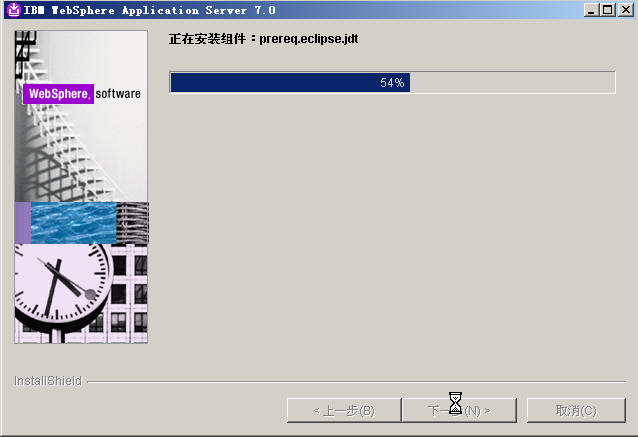


请在这一步时选择“无”，一定记住，要不然WAS会建一个默认的server（类似于weblogic里的domain）并且将它注册为我们的windows的service，这是我们不需要的，我们将会在稍后的教程中使用命令行来启动和停止我们的WAS，必竟以后我们在production环境中一般都使用64位的UNX或者是LNX，而非Windows。

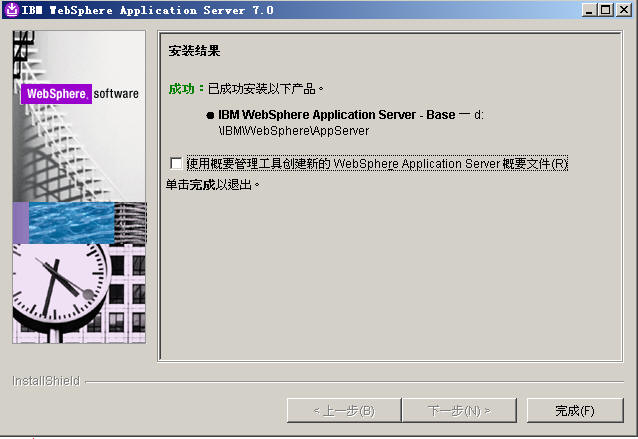




**下一步，下一步一路到底，然后开始安装**



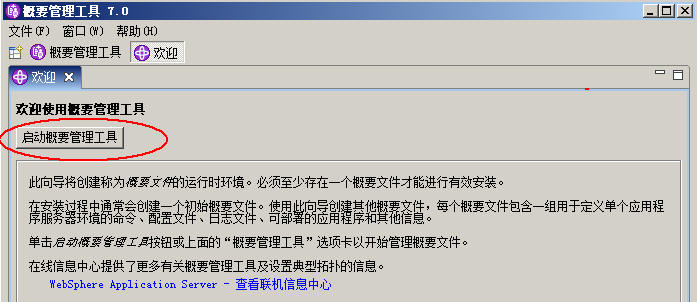
安装完了，请把“**使用概要管理工具创建新的WebSphere Application Server概要文件(R)**”前的“勾”去掉，我们将使用菜单来创建WAS的概要文件（即相当于weblogic里的domain）。点[完成]结束安装。



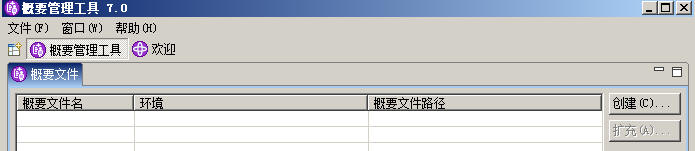
安装完后使用菜单启动我们的概要管理工具，概要管理工具会为你创建WAS的概要文件，即profile，WAS的运行都是基于一个个profile的。



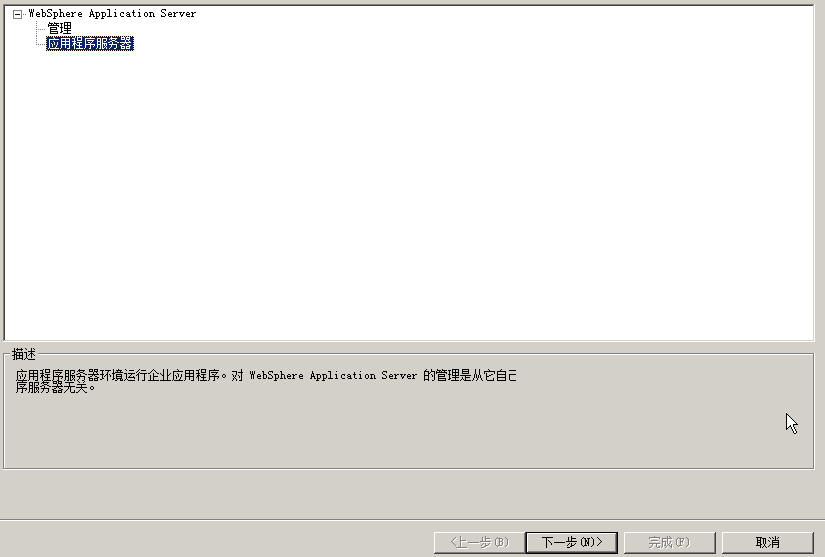
**在概要管理工具的主界面点[启动概要管理工具]这个按钮。**



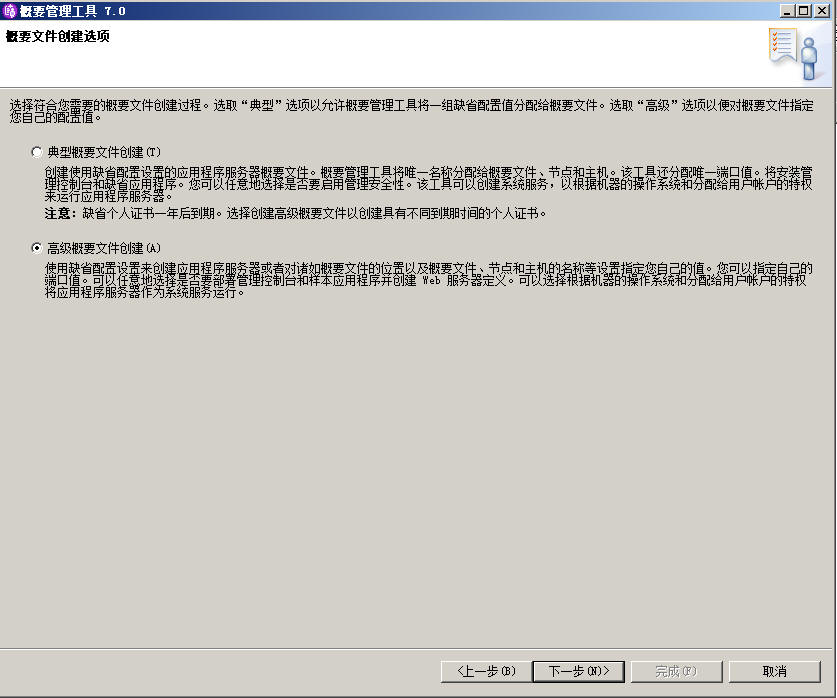
**点[创建]按钮**



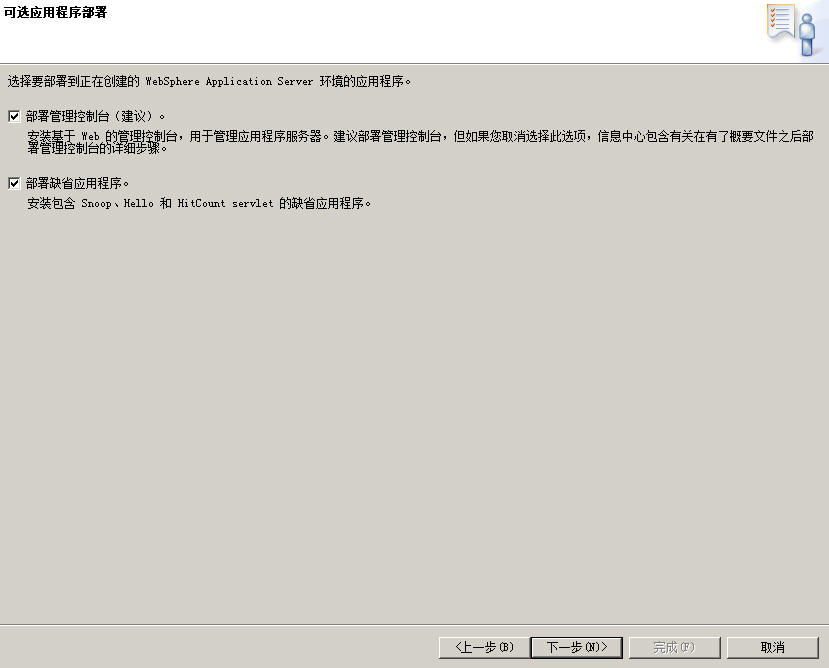
**选中“应用程序服务器”，下一步**



**我们使用“高级概要文件创建”。**



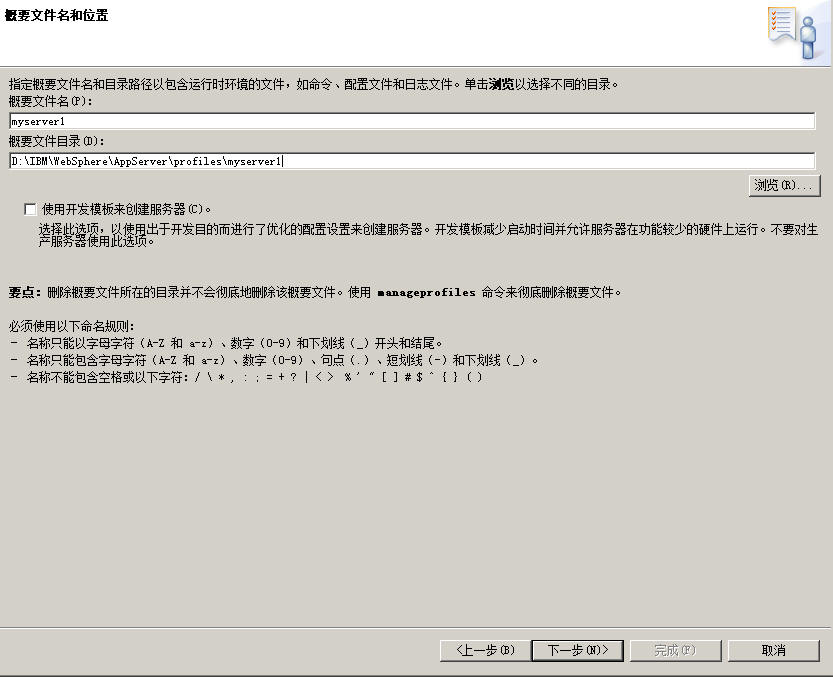
**下一步**



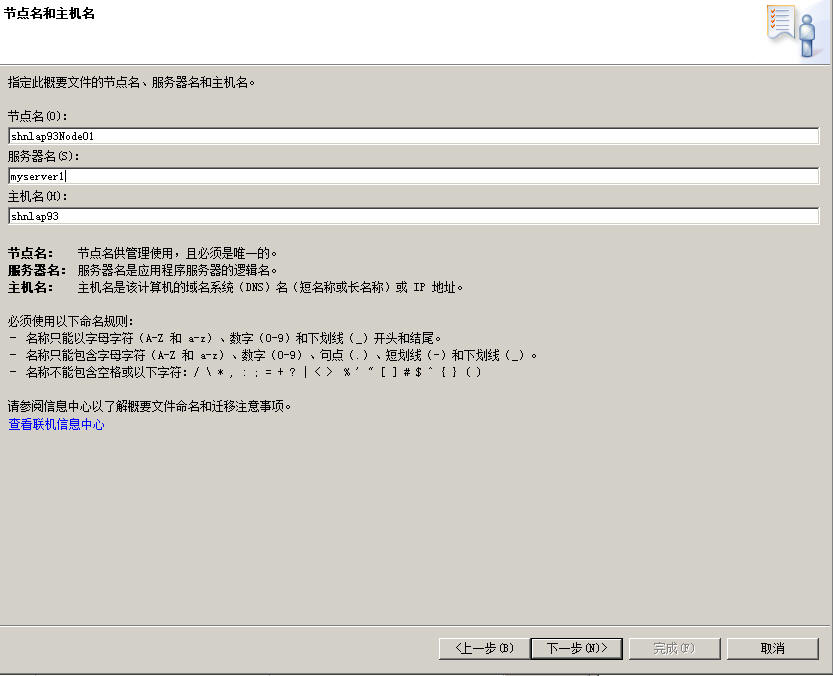
**下一步**

概要文件名，因为前面我们说过了概要文件就是WAS内的一个server，一个WAS可以包含多个概要文件。

因此这边我们的概要文件名为myserver1，而我们的物理位置也叫myserver1。



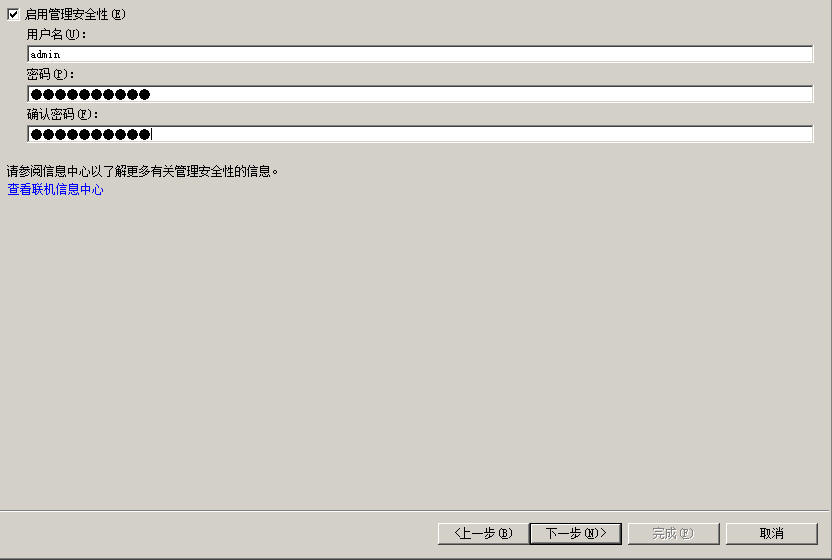
而该概要文件指向的server name也为myserver1。



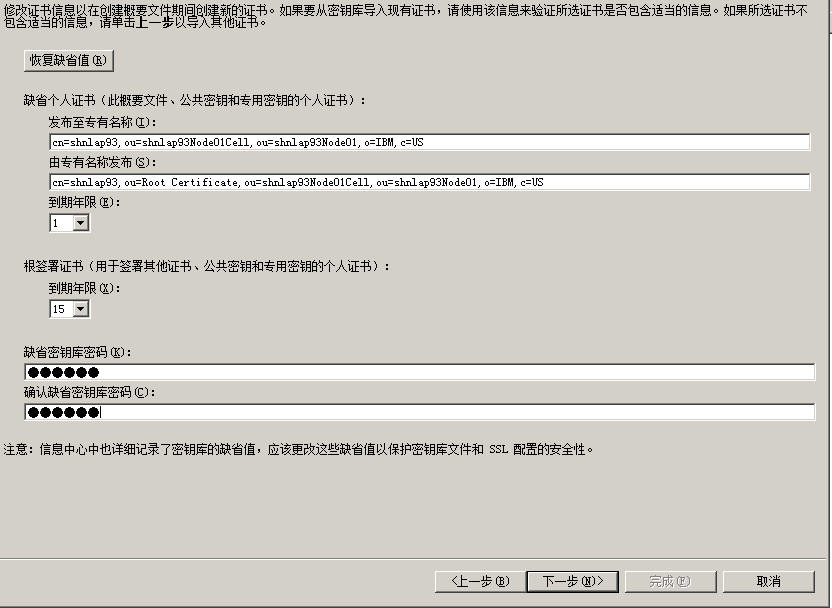
这里用于配置我们的WAS的管理界面登录时所需的用户名与密码，我们使用：

loginId: **admin**

password: **password\_1**



**此处的“缺省密钥库密码”我们全部设成“aaaaaa”，即六个a，然后下一步**



这边是设的是WAS的端口，常用的主要为：

9060（was console的端口）

9080（was应用程序布署完后的端口，类似于tomcat的8080端口）

9043（was console的https端口）

9443（即was应用程序布署完后的https端口，类似于tomcat的8443端口）

如果你的WAS已经事先创建过一个profile，你再创建一个profile，这些端口的数字都会自动在最未尾加一，比如说我们这个profile叫myserver1，那么我们再创建一个myserver2，那么myserver2在此处的端口自动会显示成：

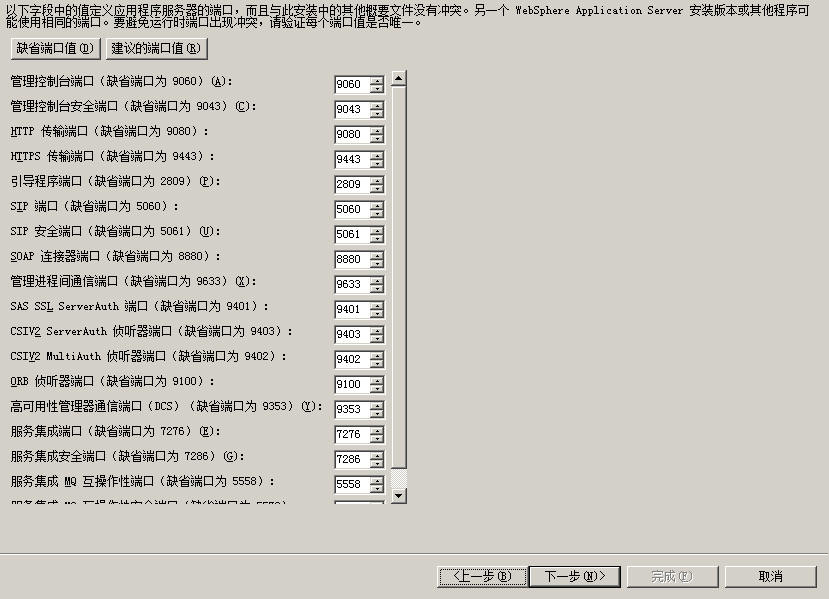
9061

9081

9044

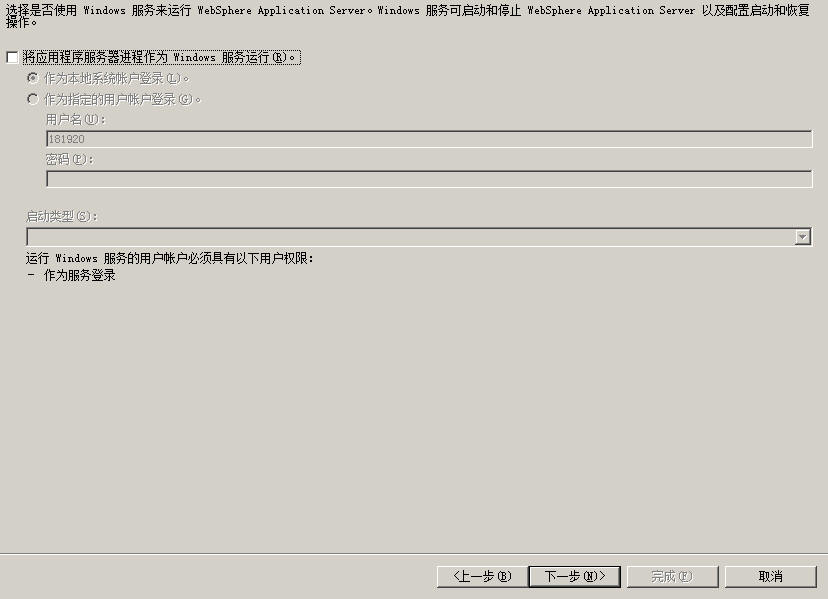
9444

等等等。

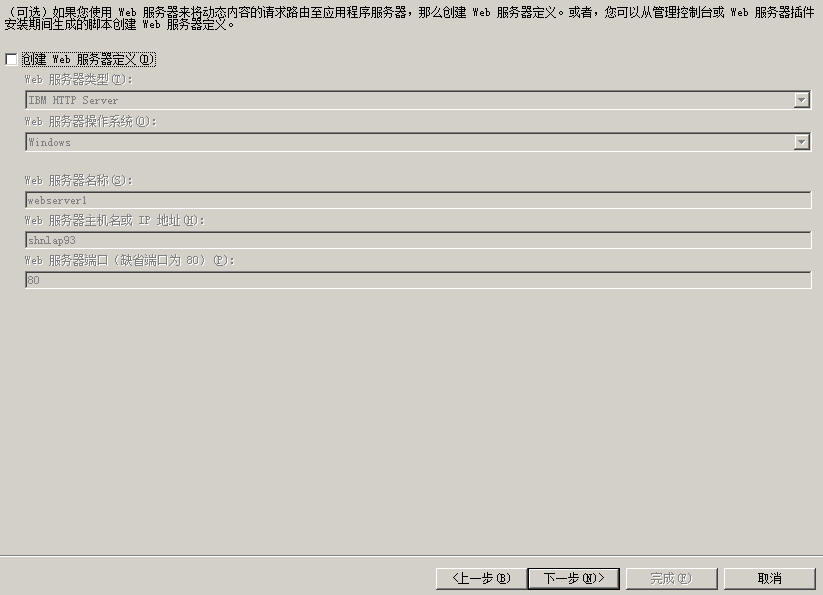


**下一步**

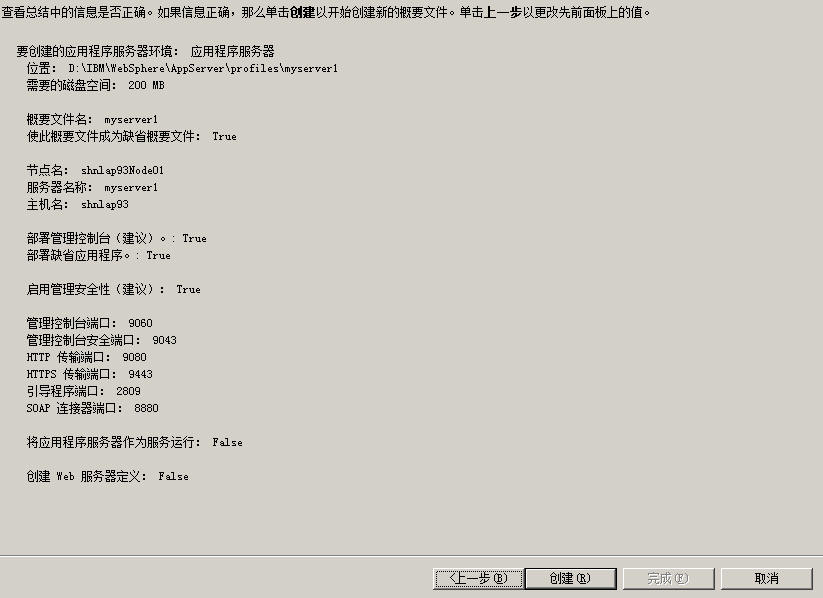
**请将“将应用程序服务进程作为Windows服务运行（R）”**选择前的**“勾”**去掉。



下一步



下一步，最后review一下，点[创建]按钮

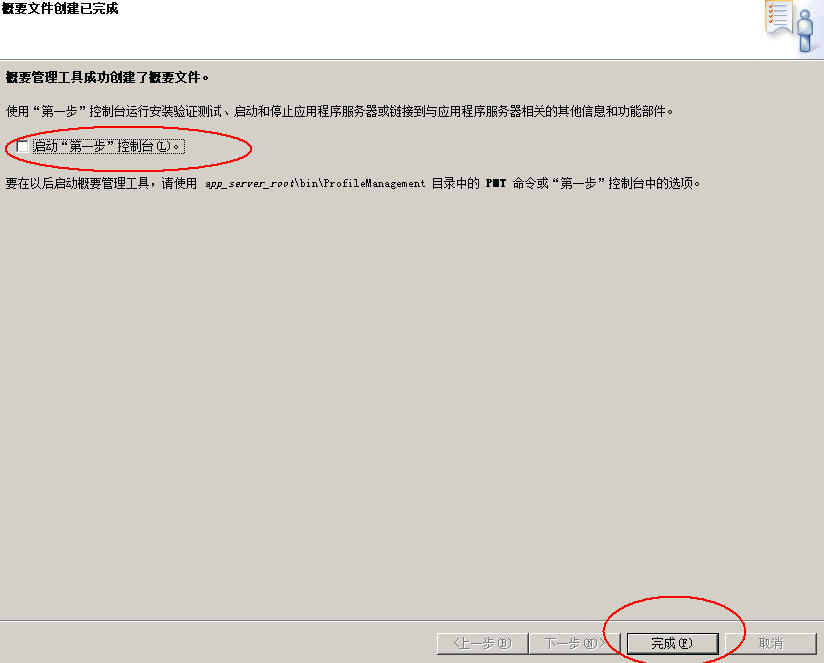


开始创建profile



创建完后

1. 把启动第一步控制台这个选项前的勾去掉。
2. 点击[完成]按钮



该profile创建完成后我们的目录结构如下截图所示：

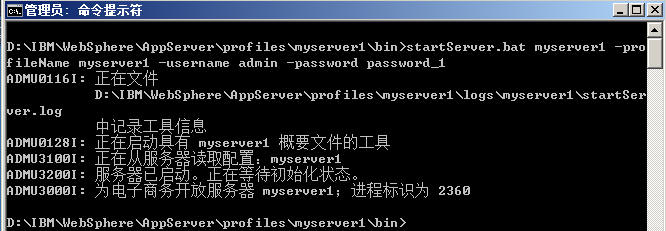


# WAS的启动与停止

## 4.1 启动WAS

命令格式为：

startServer serverName –profileName yourProfileName –username loginId –password password。

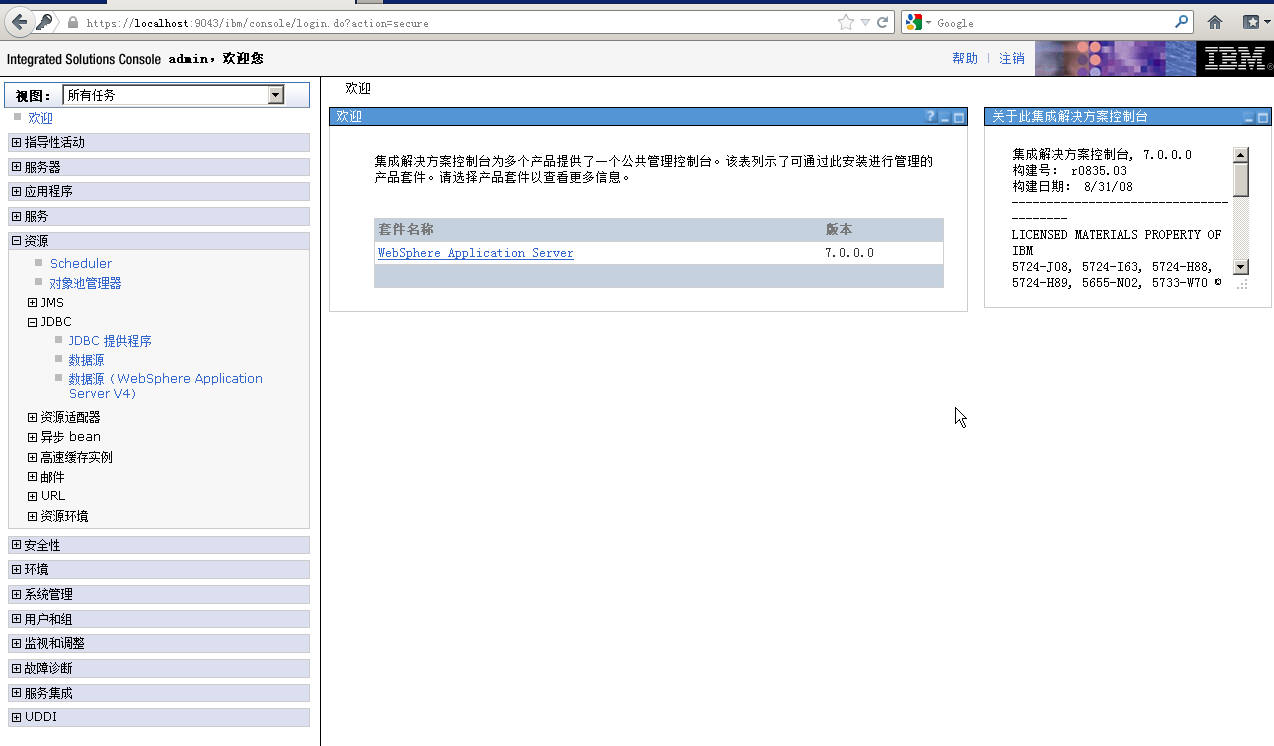


看到“进程标识为2360”，即代表我们的WAS启动成功了。

启动成功后可以通过<http://localhost:9060/ibm/console> 这个地址来访问我们的“管理控制台”

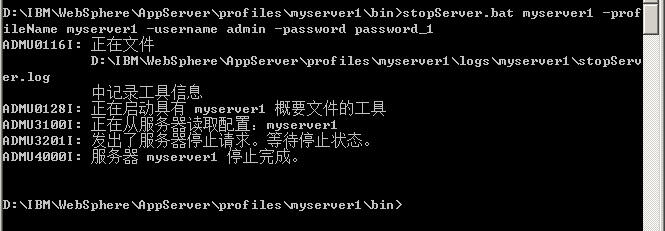


该管理控制台会自动将地址变为：https://localhost:9043 这样的格式。



## 4.2 停止WAS

把startServer换成stopServer即可，后面参数一样！



# 管理WAS

## 5.1 优化WebSphere

WebSphere里的profile刚配完，一般默认的heap size即Xms与Xmx值只有256mb，而IBM WAS是几个J2EE服务器中最吃内存的机器，在布署一些EAR应用时，如果你的EAR中使用的lib即jar files较多，加载时往往会超出256mb的限制，如果你的WAS在安装完后不进行适当的优化就用来布署应用，很快就会成死机状，然后在相应的profile的目录中会留下一堆的heapdump即内存out of memory并造成了was档机后留下的dump文件。

因此在装完WAS配完profile后，请先进行适当的优化。

### 5.1.1 修改系统打开文件数（windows系统忽略）

如果你的机器为Linux/Unix，请：

调整Linux/Unix系统允许打开的最大文件数，系统默认一般为1024。

我们可以执行ulimit -n可查看这个数值。

通过vi /etc/security/limits.conf加入以下两行：

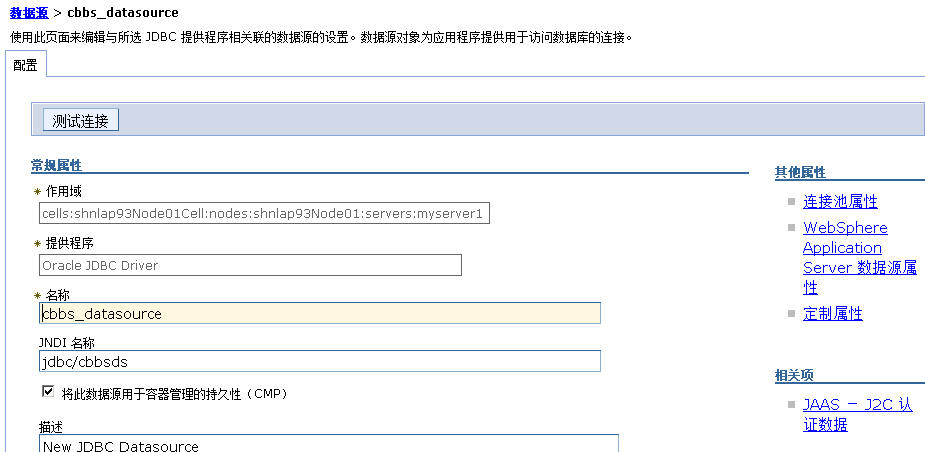
\* soft nofile 300000

\* hard nofile 300000

重新系统后通过ulimit -a可以查看结果。

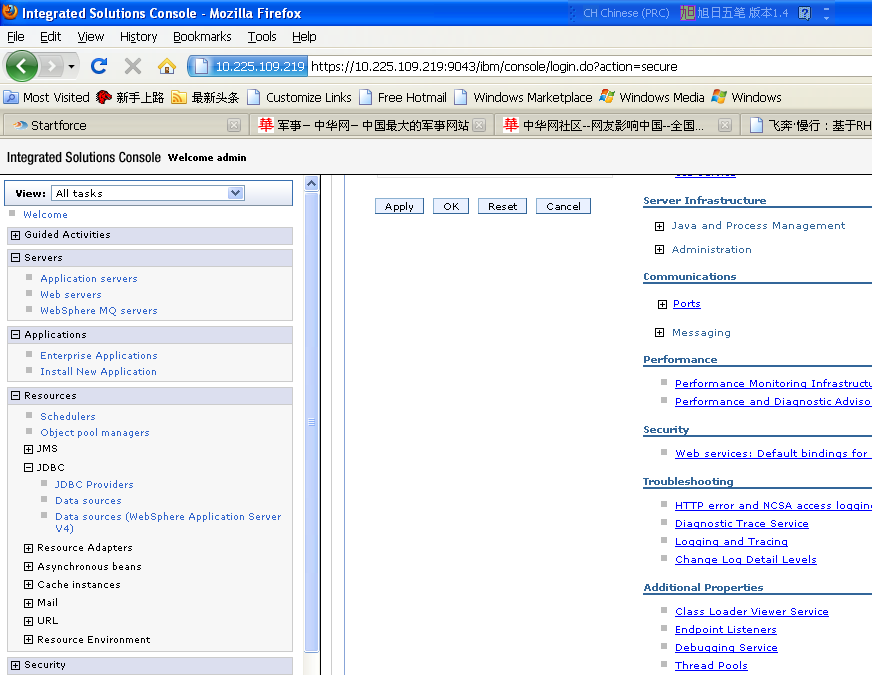
### 5.1.2优化数据库连接池

优化数据库连接池，通过点击相应的datasource进入属性页后，点击“连接池属性”可以进行连接池的优化（相关数据库连接池优化见第八天之weblogic与apache的整合与调优中的4.2.5 小节调优Weblogic的数据源）。

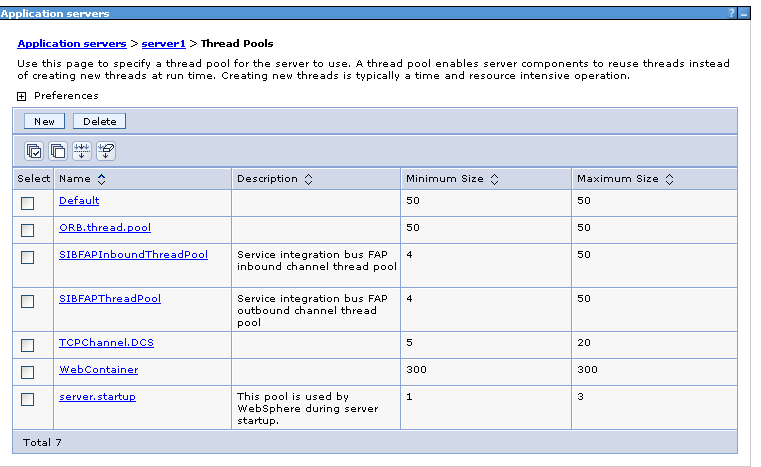


### 5.1.3更改Web容器线程池大小

该参数在管理控制台里的“服务器→应用程序服务器→server1→线程池”的“WebContainer”中进行设置，将“最大 大小”的默认值50改成40，“最小大小”的默认值10改成40。逐一对每个成员做相应的修改。

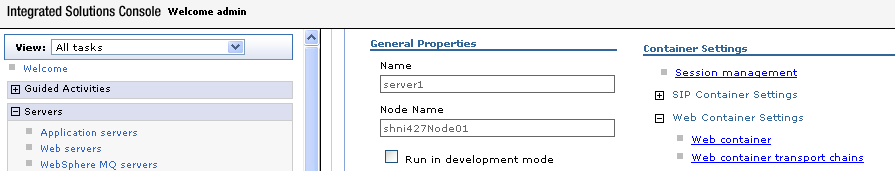


**不要勾选“允许线程分配超过最大线程大小”。**

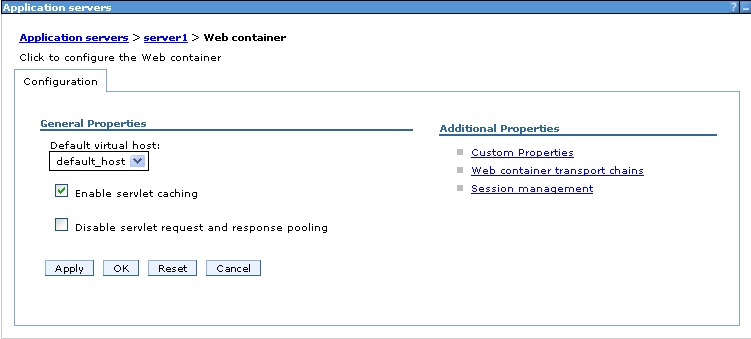


### 5.1.4 更改会话超时和启用servlet高速缓存

该参数在管理控 制台里的“服务器→应用程序服务器→server1→会话管理”的“会话超时”中进行设置，将默认的30改成15；在“服务器→应用程序服务器 →server1→Web容器设置→Web容器”。逐一对每个成员做相应的修改。



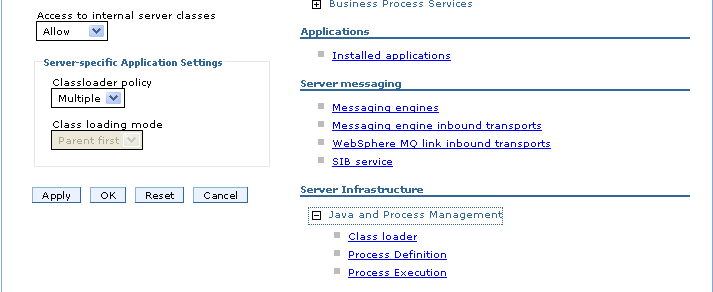
**记得Web容器里勾选“启用servlet高速缓存”。**



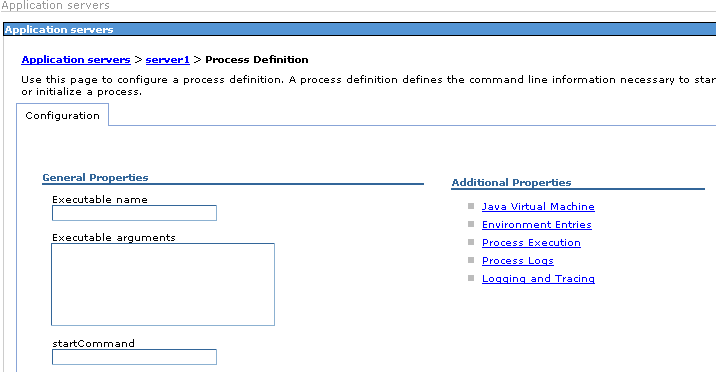
### 5.1.5更改JVM参 数

即修改相应的heap size与添加一些JVM调优参数

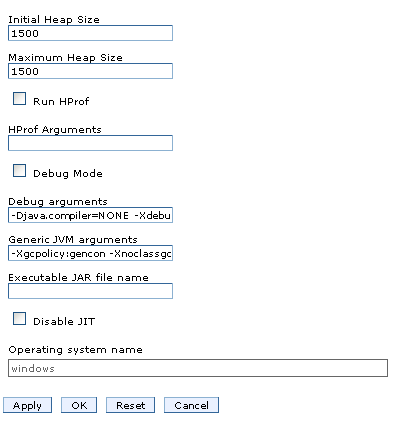
该参数在管理控制台里的“服务器→应用程序服务器→server1→进程定义→Java虚拟机”里定义



点这个”Process Definition”，然后在下面这个界面中点“Java Virtual Machine”



显示下面这个界面



**注意：**

这边的Generic JVM arguments就是IBM的jvm优化参数了，但是此处的jvm优化参数不能直接套用我们在前面tomcat和weblogic中所提到过的相关jvm优化参数，**因为这个是IBM JDK**，和ORACLE-SUN的JDK不一样的。

### IBM JDK的JVM常用参数

* -Xquickstart

此设置影响 IBM JVM 使用较低优化级别来编译类方法的方式，这将提高服务器启动速度，但会使运行时性能下降。缺省情况下，如果未指定此参数，IBM JVM 最初将使用较高的初始优化级别来执行编译。此设置能够提高运行时性能，但会减慢服务器启动速度。

|  |  |
| --- | --- |
| 缺省值： | 高初始编译器优化级别 |
| 建议值： | 高初始编译器优化级别 |
| 用法： | -Xquickstart 可以加快服务器启动速度。 |

* -server

基于 Sun 的 Hotspot 技术的 JVM 最初使用低优化级别来编译类方法。这些 JVM 使用简单编译器和能够进行优化的 JIT 编译器。通常情况下，使用简单 JIT 编译器。然而，可以通过设置此选项来使用能够执行优化的编译器。此更改将显著提高服务器的性能，但使用能够执行优化的编译器时，服务器的预备时间将会较长。

|  |  |
| --- | --- |
| 缺省值： | 简单编译器 |
| 建议值： | 能够执行优化的编译器 |
| 用法： | -server 启用能够执行优化的编译器。 |

* -Xms与-Xmx

这个就不用多说了吧，前面的tomcat与weblogic的优化中提到过多次了。

**注 意：**

**32位系统的JVM最大值不超过2048m，如果有充足的内存，可以多建立几个节点集群成员。**

* -Xgcpolicy

将 gcpolicy 设置为 optthruput 会禁用并发标记。如果没有暂停时间问题（表现为应用程序响应时间不规律），那么应该使用此选项来实现最大吞吐量。将 gcpolicy 设置为 optavgpause 会使用缺省值来启用并发标记。此设置将减少由正常垃圾回收所引起的应用程序响应时间不规律情况。然而，此选项可能会降低整体吞吐量。

|  |  |
| --- | --- |
| 缺省值： | optthruput |
| 建议值： | optthruput |
| 用法： | Xgcpolicy:optthruput |

* -Xnoclassgc

缺省情况下，当一个类没有任何活动实例时，JVM 就会从内存中卸装该类，但是这样会使性能下降。如果关闭类垃圾回收，就可以消除由于多次装入和卸装同一个类而造成的开销。

如果不再需要某个类，那么该类在堆中所占用的空间通常将用于创建新对象。但是，如果应用程序通过创建类的新实例来处理请求，并且该应用程序的请求是 随机出现的，那么可能会发生以下情况：先前请求者完成后，正常的类垃圾回收将通过释放这个类占用的堆空间来清除这个类，但当下一个请求出现时，又必须将这 个类重新实例化。在这种情况下，您可能想使用此选项来禁用类垃圾回收。

|  |  |
| --- | --- |
| 缺省值： | 启用类垃圾回收 |
| 建议值： | 禁用类垃圾回收 |
| 用法： | Xnoclassgc 禁用类垃圾回收 |

下面是我本机上的一个设置，供参考：

**-Xgcpolicy:gencon –Xnoclassgc**

### 5.1.6 更改ORB服务

进入管理控制台里的“服务器→应用程序服务器→server1→ORB服务”，勾选“按引用传递”； 线程池设置则选择“使用与线程池管理器相关联的 ORB.thread.pool 设置（建议）。”，点击“ORB.thread.pool”，修改“最小大小”的默认值10为40，“最大大小”的默认值50为40，不要勾选“允许线程 分配超过最大线程大小”。逐一对每个成员做相应的修改。

### 5.1.7 去除应用程序的类重新装入选项

进入管理控制台里的“应用程序→企业应用程序”，点击某一应用程序进入，点击“Web模块的 JSP 重新装入选项”，将“JSP启用类重新装入”的勾去掉。

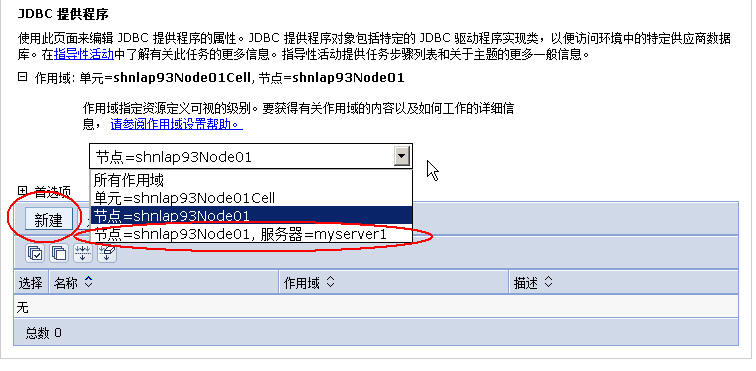
## 5.2 创建数据源

我们拿WAS来布署一个web应用，那么就需要先创建数据源，请按照下面的步骤。

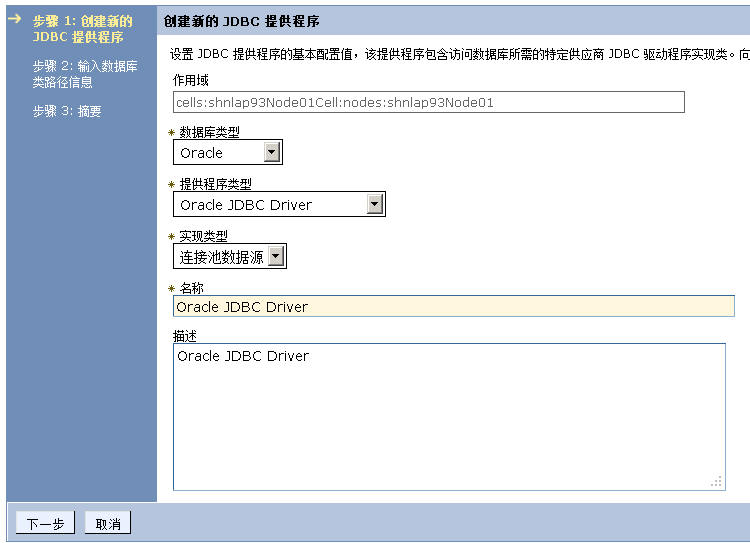
先创建JDBC Provider(提供程序)。



注意在下拉框中选择具体的“作用域”，即类似于weblogic里的target，然后点[新建]按钮。



我们使用oracle11g来做我们的数据源。

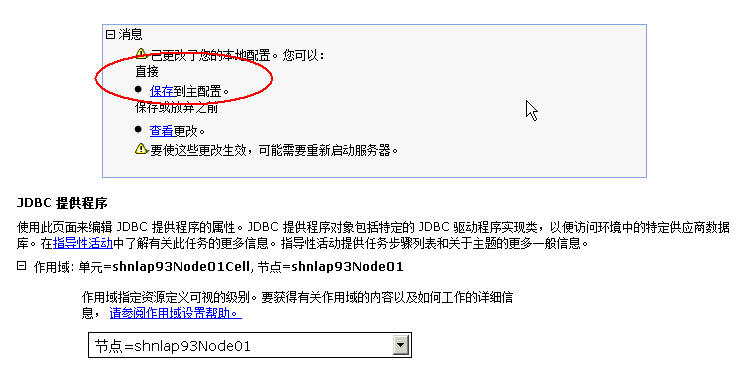


**[下一步]**



**类路径保持默认，我们回头会再给它具体指定。**

点击[完成]后，注意要按这个“保存”连接，在WAS里作过任何东西的改动在点[确定]或者是[完成]后都会**在admin console的右上端有这么一个提示，一定要记得点这个“保存”，要不然前面做的一切操作会被废弃。**

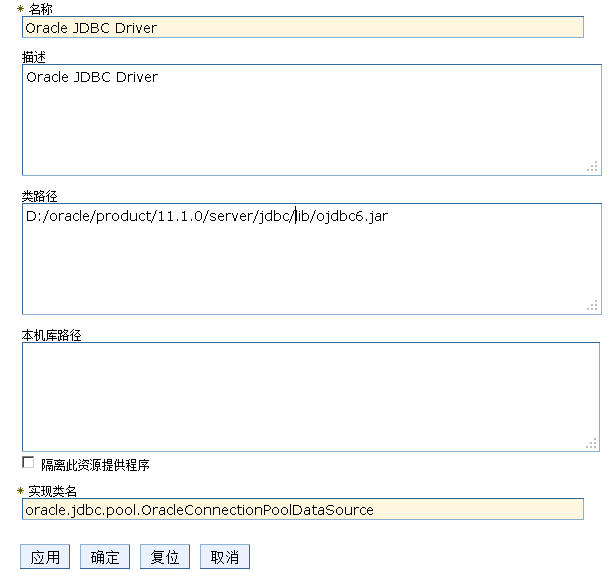


点击“保存”后，我们的JDBC提供程序列表里多了一项。



我们点一下这个“Oracle JDBC Driver”，对它进行设置。

在“类路径”里，我们填入完整的该ojdbc6.jar所在的路径，**注意要用“/”作路径分隔符，而不是“\”。**

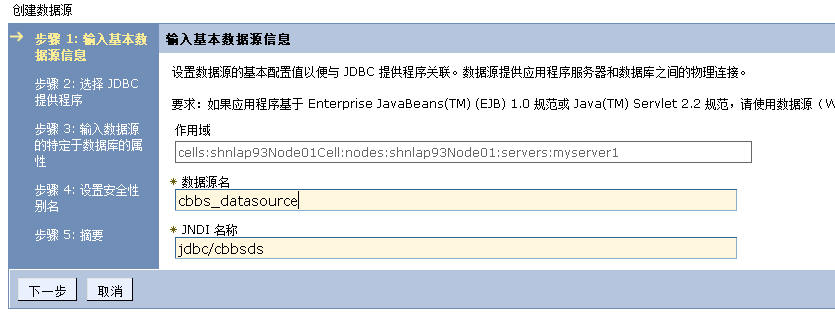


设置完后点[确定]返回，并点“保存”。

我们的JDBC Provider有了，我们要设置我们的DataSource，将DataSource绑定JDBC Provider后给它取个JNDI名，这个JNDI名就是我们工程中连接数据库时需要指定的JNDI名了。



填入和我们的工程中一样的JNDI名称，此处“数据源名”与“JNDI名称”不能同名啊，不要把“数据源名”和“JNDI名”搞错了。



**[下一步]**

为我们的数据源指定前面设置的JDBC Provider。



填入你要连接的数据库的相应的URL。



完成并保存后返回数据源设置列表，我们新设置的数据源已经在列表中了。



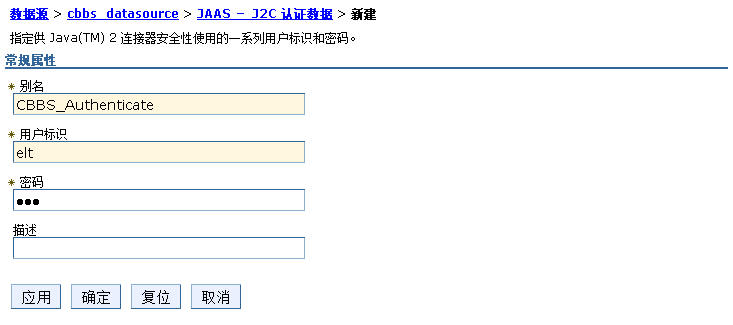
点这个我们刚才新建立的数据源。

我们现在有了jdbc driver，有了jdbc url，为了连接数据库我们还需要什么？用户名 & 密码，对吧！



点右边这个“JAAS-J2C认证数据”。

为我们的认证起个名字，然后输入用户标识（数据库连接用户名）与密码（数据库连接密码）。



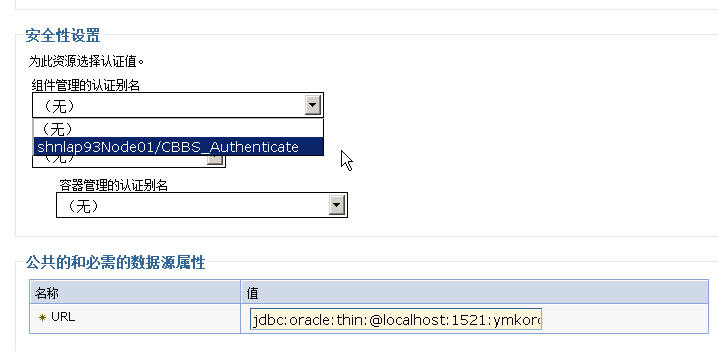
点[确定]后返回并“保存”。



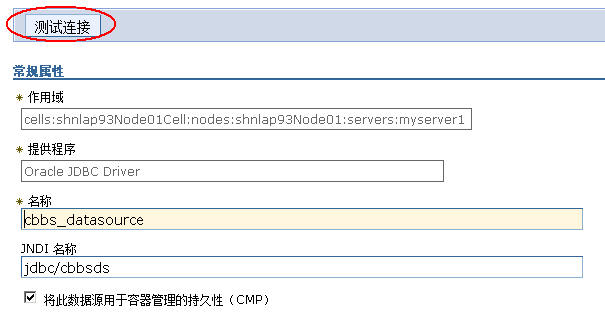
还是选择我们刚才创建的“数据源”，可以直接在“面包屑”中找到我们刚才创建的数据源。



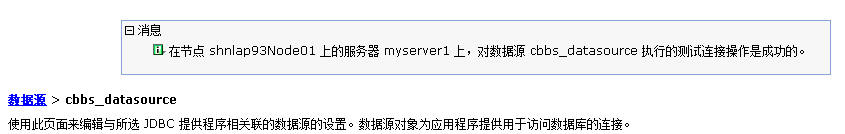
在数据源属性页面中往下拖，到页面底部看到有一个“安全性设置”，OK，将我们的数据源绑定我们刚才创建的“J2C认证”吧。



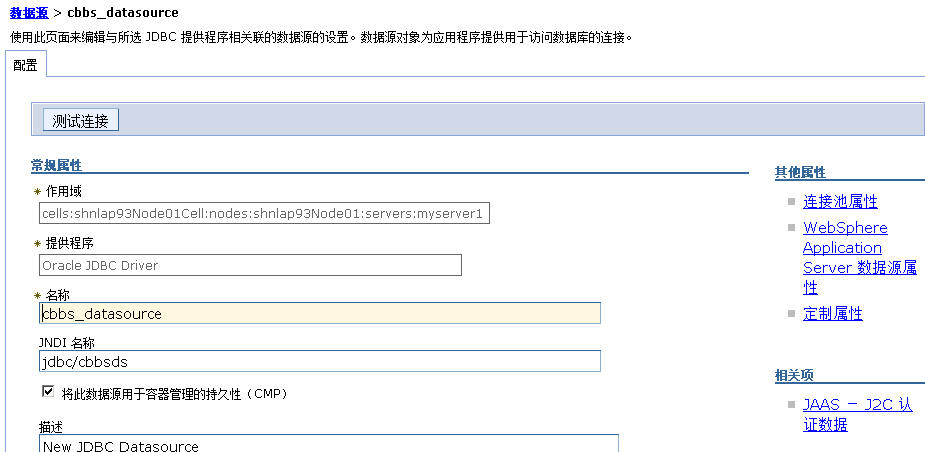
[确定]并“保存”后，可以在数据源的属于页里点这个[测试连接]按钮来进行我们的数据库连接测试。



下面是数据源测试成功的消息提示，代表着我们的数据源已经测试成功。



在我们的数据源的属性页面右边有一个“连接池属性”



点“连接池属性”，此处就是用来配置和优化我们的数据库连接池，如何优化这边就不多讲了，因为已经在weblogic相关教程中详细描述过了，大同小异，一通百通。



## 5.3 在WAS内布署应用

一般我们使用ear格式在WAS内布署我们的web应用，因此此处和weblogic, tomcat稍稍有点不一样。

为此，我们做了一个ant脚本用于打包我们的ear。

一个ear文件的格式应该如下：

myEAR

|\_\_ META-INF

|\_\_application.xml

|\_\_myWAR.war

可以看到，一个ear文件：

包含一个META-INF目录，在该目录下会有一个application.xml文件。

然后和META-INF目录同级的地方会有一个.war文件，这个war文件就是标准的war格式的web应用。

### 5.3.1 application.xml文件

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?>  <application xmlns="http://java.sun.com/xml/ns/javaee"  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="5"  xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/application\_5.xsd">  <display-name>cbbsEAR</display-name>  <module>  <web>  <web-uri>cbbs.war</web-uri>  <context-root>/cbbs</context-root>  </web>  </module>  </application> |

### 5.3.2 使用Ant打包

实际上我们的ant脚本很简单，它作了如下几件事：

1. 用IBM WAS自带的JDK即IBM JDK编译我们的工程，确保我们的工程可以被我们的WAS能够解析。
2. 将我们的工程打成.war文件

|  |
| --- |
| <target name="makeWAR" depends="compile">  <jar destfile="${build.dir}/app/${war.name}" basedir="${build.dir}" />  </target> |

1. 将.war文件与application.xml文件打成.ear文件

|  |
| --- |
| <target name="makeEAR" depends="makeWAR">  <ear destfile="${build.dir}/${ear.name}" appxml="${eardescription.file}">  <fileset dir="${build.dir}/app">  <include name="\*\*/\*.war" />  </fileset>  </ear>  </target> |

下面给出我们完整的Ant脚本，该脚本还使用到了一个外部的build.properties文件用于设置一些常量。

### build.properties

|  |
| --- |
| websphere.v7.home=D:/IBM/WebSphere/AppServer  project.home=D:/wspace  webAppName=cbbs  war.name=${webAppName}.war  ear.name=${webAppName}EAR.ear  eardescription.file=D:/wspace/${webAppName}/application.xml |

### build.xml

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?>  <project default="makeEAR" name="cbbs\_build">  <!-- ####################################### CONFIG PROPERTIES ########################################-->  <property file="build.properties" />  <property name="web.home" VALUE="${project.home}/cbbs" />  <property name="src.dir" VALUE="${project.home}/cbbs/src" />  <property name="webcontent.dir" VALUE="${project.home}/cbbs/WebRoot" />  <property name="lib.dir" VALUE="${webcontent.dir}/WEB-INF/lib" />  <property name="extlib" value="${project.home}/${webAppName}/ext-lib" />  <property name="classes.dir" value="${webcontent.dir}/WEB-INF/classes" />  <property name="tld.dir" value="${webcontent.dir}/WEB-INF/tld" />  <property name="build.dir" value="${project.home}/${webAppName}/build" />  <property name="resource.dir" value="${project.home}/${webAppName}/resource" />  <!-- ####################################### CLASSPATH ########################################-->  <path id="classpath">  <pathelement path="${src.dir}" />  <pathelement path="${classes.dir}" />  <fileset dir="${extlib}">  <include name="\*\*/\*.jar" />  </fileset>  <fileset dir="${lib.dir}">  <include name="\*\*/\*.jar" />  </fileset>  <fileset dir="${websphere.v7.home}\lib">  <include name="\*\*/\*.jar" />  </fileset>  <fileset dir="${websphere.v7.home}\plugins">  <include name="\*\*/\*.jar" />  </fileset>  </path>  <!-- ####################################### CREATE INITIAL DIRECTORIES ##########################-->  <!-- Creating Initial Directoris -->  <target name="init">  <echo message="Creating init directories..." />  <delete dir="${build.dir}" />  <mkdir dir="${build.dir}" />  <mkdir dir="${build.dir}/app" />  <mkdir dir="${build.dir}/WEB-INF" />  <mkdir dir="${build.dir}/WEB-INF/lib" />  <mkdir dir="${build.dir}/WEB-INF/tld" />  <mkdir dir="${build.dir}/WEB-INF/classes" />  <mkdir dir="${build.dir}/css" />  <mkdir dir="${build.dir}/images" />  <mkdir dir="${build.dir}/js" />  <mkdir dir="${build.dir}/jsp" />  <mkdir dir="${build.dir}/fckeditor" />  <mkdir dir="${build.dir}/userfiles" />  <mkdir dir="${build.dir}/userfiles/image" />  </target>  <!-- ####################################### COMPILE ########################################-->  <target name="compile" depends="init">  <echo message="Calling a compile on all the Java source files" />  <javac destdir="${build.dir}/WEB-INF/classes" srcdir="${src.dir}" debug="true" executable="${websphere.v7.home}\java\bin\javac" compiler="javac1.6" fork="true">  <classpath refid="classpath" />  </javac>  <copy todir="${build.dir}" overwrite="true">  <fileset dir="${webcontent.dir}">  <include name="\*.\*" />  </fileset>  </copy>  <copy todir="${build.dir}/WEB-INF/" overwrite="true">  <fileset dir="${webcontent.dir}/WEB-INF/">  <include name="\*.\*" />  </fileset>  </copy>  <copy todir="${build.dir}/jsp" overwrite="true">  <fileset dir="${webcontent.dir}/jsp">  <include name="\*\*/\*.\*" />  </fileset>  </copy>  <copy todir="${build.dir}/WEB-INF/classes" overwrite="true">  <fileset dir="${resource.dir}">  <include name="\*\*/\*.\*" />  </fileset>  </copy>  <copy todir="${build.dir}/WEB-INF/lib" overwrite="true">  <fileset dir="${lib.dir}">  <include name="\*.\*" />  </fileset>  </copy>  <copy todir="${build.dir}/WEB-INF/tld" overwrite="true">  <fileset dir="${tld.dir}">  <include name="\*\*/\*.\*" />  </fileset>  </copy>  <copy todir="${build.dir}/js" overwrite="true">  <fileset dir="${webcontent.dir}/js">  <include name="\*\*/\*.\*" />  </fileset>  </copy>  <copy todir="${build.dir}/css" overwrite="true">  <fileset dir="${webcontent.dir}/css">  <include name="\*\*/\*.\*" />  </fileset>  </copy>  <copy todir="${build.dir}/images" overwrite="true">  <fileset dir="${webcontent.dir}/images">  <include name="\*\*/\*.\*" />  </fileset>  </copy>  <copy todir="${build.dir}/fckeditor" overwrite="true">  <fileset dir="${webcontent.dir}/fckeditor">  <include name="\*\*/\*.\*" />  </fileset>  </copy>  </target>  <target name="makeWAR" depends="compile">  <jar destfile="${build.dir}/app/${war.name}" basedir="${build.dir}" />  </target>  <target name="makeEAR" depends="makeWAR">  <ear destfile="${build.dir}/${ear.name}" appxml="${eardescription.file}">  <fileset dir="${build.dir}/app">  <include name="\*\*/\*.war" />  </fileset>  </ear>  </target>  </project> |

### 使用IBM WAS自带的JDK对工程进行编译

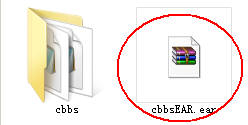
该套Ant脚本其实还可以继续优化，这个就留给大家了，关键是提供给了大家一个思路。

除了上边所说的涉及到如何打war，如何打ear的核心脚本外，关键在于这个“使用WAS自带的IBM JDK”对工程进行编译是很重要的，下面是核心语句，注意**红色加组**的部分。

|  |
| --- |
| **<path id="classpath">**  <pathelement path="${src.dir}" />  <pathelement path="${classes.dir}" />  <fileset dir="${extlib}">  <include name="\*\*/\*.jar" />  </fileset>  <fileset dir="${lib.dir}">  <include name="\*\*/\*.jar" />  </fileset>  <fileset dir="${websphere.v7.home}\lib">  <include name="\*\*/\*.jar" />  </fileset>  <fileset dir="${websphere.v7.home}\plugins">  <include name="\*\*/\*.jar" />  </fileset>  </path>  <javac destdir="${build.dir}/WEB-INF/classes" srcdir="${src.dir}" debug="true"  executable="${websphere.v7.home}\java\bin\javac" compiler="javac1.6" fork="true">  **<classpath refid="classpath" />**  </javac> |

### 5.3.3 布署EAR

编译完后该Ant脚本会产生一个cbbsEAR.ear的文件：



这个文件就是我们用于布署的EAR文件了。

根据下面的步骤一步步来布署我们的ear吧。

点[安装]

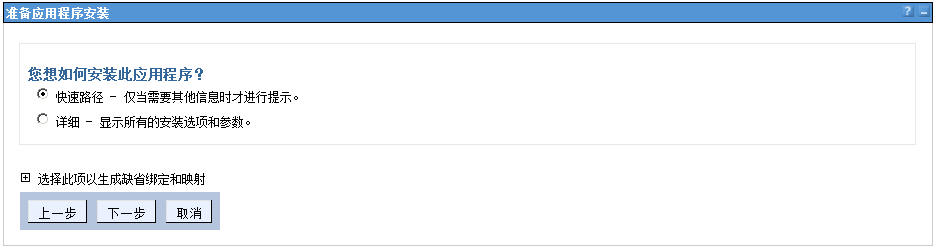


指定我们的ear全路径

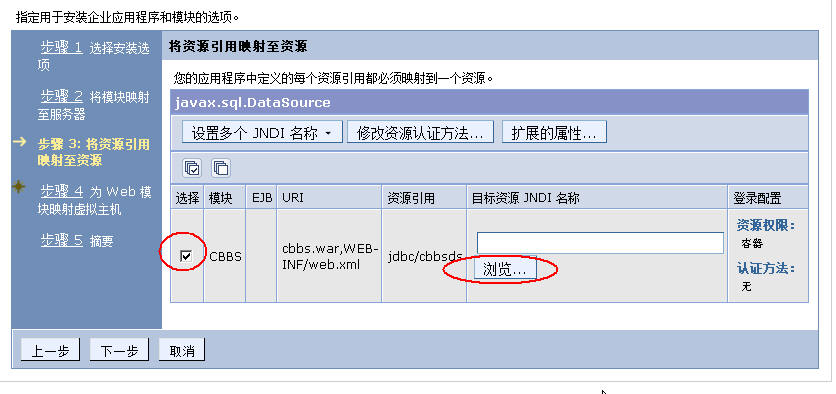


下一步

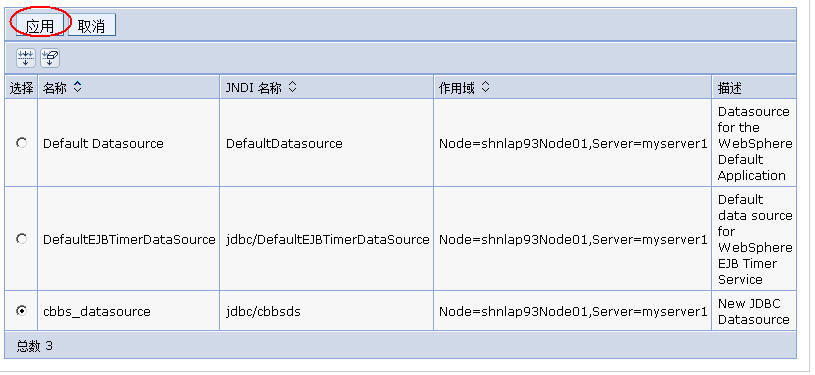
这步会比较慢，需要慢慢等一会。



下一步下一步到一直到下面这个界面，请一定记住要把我们之前设置的“数据源”与工程中的jdbc的JNDI名进行绑定啊，一定不要忘。



把“选择”处的勾打上后，点击[浏览]按钮，选择我们之前建立的“DataSource”。



点[应用]按钮后回到继续布署的界面。

将需要布署的一些选项review一遍并确认无误后点[完成]按钮，开始布署。



WAS布署的进度界面



布署完后不要忘记点**红圈**处的“保存”啊，要不一切白费。

布署完后，我们的应用的应用程序状态为“停止”状态。



选择我们的应用，点[启动]按钮



启动后我们的应用程序应该为如下状态



打开一个ie，输入: <http://localhost:9080/cbbs>



登录后进行一些操作，一切正常，WAS安装设置成功。

## Snap64.jpg

结束今天的教程，明天的教程将会着重在使用IBM的HttpServer即IHS如何去结合IBM WebSphere Application Serve做集成。