Copyright

限制条款

商标或者服务标记

BEA WebLogic 服务器管理指南

文档日期	软件版本
2001年7月30日	BEA WebLogic Server Version 6.1

目录

关于本文档

本文档说明了配置与监控 WebLogic 服务器的管理子系统，按以下内容组织：

第 1 章，“WebLogic 服务器管理概述”描述了 WebLogic 服务器管理子系统的架构。

第 2 章，“启动与终止 WebLogic 服务器”说明了启动与终止 WebLogic 服务器的步骤

第 3 章，“节点管理器”说明了如何设置和使用节点管理器。节点管理器被用来远程启动与终止 WebLogic 服务器

第 4 章，“配置 WebLOgic 服务器与集群”讲述配置在一个 WebLogic 服务器域资源的功能部件

第 5 章，“监控 WebLogic 域”描述了用于监控 WebLogic 服务器域内资源的 WebLogic 功能部件

第 6 章，“用日志消息管理 WebLogic 服务器”描述了如何使用 WebLogic 服务器本地日志与挂历 WebLogic 服务器域的域日志。

第 7 章，“部署应用管理”描述了如何把应用安装到 WebLogic 服务器上，以及如何部署应用组件

第 8 章，“配置 WebLogic 服务器的 Web 组件”说明了如何把 WebLogic 服务器当作 Web 服务器来用。

第 9 章，“代理对另一个 HTTP 服务器的请求”描述了如何使用 WebLogic 服务器代理对其它 Web 服务器的请求

第 10 章，“代理对 WebLogic 集群的请求”描述了如何代理对 WebLogic 服务器集群的 HTTP 请求。

第 11 章，“配置 Apache-WebLogic 服务器插件”说明了如何安装与配置 WebLogic Server Apache 插件

第 12 章，“配置 Microsoft-IIS 插件”说明了如何在 Microsoft Internet Information Server 安装与配置的 WebLogic Server 插件

第 13 章，“配置 Netscape Enterprise Server 中的插件（NSAPI）”说明了如何在 Netscape Enterprise Server 中安装与配置 WebLogic Server 插件

第 14 章，“安全管理”讨论了 Weblogic 服务器的安全资源及其管理

第 15 章，“管理事务”说明了如何在 WebLogic 服务器域中管理 Java 事务子系统

第 16 章，“管理 JDBC 连接”讨论了在 WebLogic 服务器域中的管理 JDBC 资源

第 17 章，“管理 JMS”讨论了如何在 WebLogic 服务器域中管理 Java 消息服务

第 18 章，“管理 JNDI”讨论了如何使用 WebLogic JNDI 名字树，包括查看和编辑 JNDI 命名树上的对象以及如何把对象绑定到 JNDI 树。

第 19 章，“管理 WebLogic J2EE 连接器架构”描述了提供对其它企业信息系统连接的 WebLogic J2EE 平台扩展是如何被管理的。

第 20 章，“管理 WebLogic 服务器许可证”描述了如何更新 BEA 许可证

附录 A，“使用 WebLogic Java 工具”描述了提供给开发人员与管理员的工具集合

附录 B，“WebLogic 服务器的命令行接口参考”描述了管理 WebLogic 域的命令行接口的语法与使用。

附录 C, “Web 服务器插件的参数”讨论了 Web 服务器插件的参数。

致读者

本文档主要是针对管理 WebLogic 服务器应用平台和其各个子系统的系统管理员。

e-docs 网站

BEA 产品文档都可以从 BEA 公司的官方网站获得。在 BEA 主页上，点击产品文档。

如何打印文档

你可以通过流览器打印本文档的副本，每一次一个主题，在浏览器的菜单中选择 File->Print 进行打印。

PDF 格式的文档在 e-docs 站点上的 WebLogic 服务器文档主页上（在文档光盘上也有）。你可以通过 Adobe Acrobat Reader 以书面格式打开这些文档并选择打印整个文档（或者一部分）。要得到这些 PDF 格式的文档，只要打开 WebLogic 服务器文档主页，点击文档下载，并选择你所需要的文档就可以了。

Adobe Acrobat Reader 在 Adobe 的站点http://www.adobe.com上可以免费得到。

联系我们!

您对 WebLogic 服务器文档的反馈对我们非常重要。如果您有什么疑问和解释可以通过电子邮件docsupport@bea.com联系我们。那些建立和更新 WebLogic 服务器文档的 BEA 专业人员会直接参考您的评论。

在你的电子邮件信息里，请指出您在使用 BEA WebLogic 服务器软件格式和版本，以及文档的名称、文档的日期。

如果您对此版本的 WebLogic 服务器有任何疑问，或者在安装或运行 BEA WebLogic 服务器时遇到问题，可以通过 BEA 的在线支持 http://www.bea.com 联系 BEA 的客户支持人员。您也可以通过客户支持卡上的客户支持人员联系方式联系我们，他们都在产品包里。

当联系客户支持人员时，请准备好以下信息：

－您的姓名，电子邮件地址，电话号码和传真号码

－您的公司名称和公司地址

－您的机器型号和认证号码

－您在使用的产品名称和版本

－您遇到的问题的描述和相关的错误信息

第一章 WebLogic 服务器管理概述

本章将介绍 WebLogic 服务器的管理工具，内容如下：

域、管理服务器以及受管理的服务器

启动管理控制台

运行时的对象与配置的对象

对日志消息的集中访问

创建一个新域

BEA WebLogic Server™ 软件的实施过程中包含了许多互相关联的资源。对这些资源的管理包括下列任务：服务器的启动及终止，服务器以及连接池的负载平衡，资源配置的选择和监视，诊断并修改问题，监视并评估系统性能，部署 Web 应用、EJB 以及其它资源。

WebLogic 服务器提供了一个基于 Web 的工具 — 管理控制台，用以执行上述的管理任务。通过管理控制台，你可以访问 WebLogic 管理服务。管理服务实现了 Sun 的 Java 管理扩展（JMX 标准（Jave Management Extension），它提供了一系列的管理手段来管理 WebLogic 的资源。

通过管理控制台来配置资源的属性，部署应用及组件，监视资源的使用情况（如服务器负载，Java 虚拟机的内存使用情况以及数据库连接池的负载），查看日志消息，启动或终止服务器，以及执行其它管理任务。

域、管理服务器与受管服务器

作为一个单元来管理的，并相互关联的一组 WebLogic 服务器资源被称为域。一个域可以包含一个或多个 WebLogic 服务器，还可以包含 WebLogic 服务器集群。

域的配置使用扩展标记语言（XML）定义。在 install_dir/config/domain_name 目录中的 config.xml 文件定义了域的配置，install_dir 是 WebLogic Server 软件的安装目录。

域是一个完备的管理单元。向域里分发应用的时候，该应用的各组件只能分发到域之内的服务器上。如果域中包含集群，那么集群中的所有服务器都必须属于同一个域。一个域可能包含多个集群。

J2EE 应用是一个组件集合，这些组件被组织成一个部署单元（例如 EAR,WAR, 或 JAR 文件）。应用所需要的各种组件 — EJB 或 Web 应用，服务器或集群，JDBC 连接池等等都定义在一个域的配置中。将这些资源组合在一个单一的、完备的域中使我们可以以统一的方式来查看或访问这些相互关联的资源。

运行管理服务的 WebLogic 服务器称为管理服务器。管理服务集中管理并监控域的所有资源。如果要对某个域执行管理操作，该域的管理服务器必须处于运行状态。

一个包含多个 WebLogic 服务器的域只能有一个管理服务器，其它服务器被称为受管服务器。每个 WebLogic 受管服务器都会在启动时从管理服务器得到各自的属性配置。

管理服务器和 WebLogic 受管服务器启动时都运行 weblogic.Server 类。没有作为受管服务器启

动的 WebLogic 服务器就是管理服务器。

在生产环境中，系统的典型配置是这样的：应用及业务逻辑组件被分发在多个受管服务器上，而管理服务器则负责配置及监视受管服务器。管理服务器的作用是配置与监视受管服务器。如果管理服务器宕机了，部署在受管服务器上的应用不会因此而受到影响，可以继续处理客户端发送来的请求；这种情况下，当管理服务器被重启后，可以重新获得对活动域的控制（详细内容，参见“受管服务器运行时重启管理服务器”中的内容）。

把应用或应用组件分发到一组受管服务器上能带来很多好处。将 EJB 以及其它组件部署到一组服务器上可以保证主应用接入点的可用性。如果将完成不同功能的组件（如数据库访问与帐单事务）部署于不同的受管服务器上处理，可以提高系统的性能。象 EJB 这种可以实现各种功能的组件或应用是可以被分隔开的，从而使它的可用性不依赖于其它组件的状态。多个应用可以部署在一个域中。

当管理服务器使用这样的配置启动以后，我们说该域是活动（active）的。在域处于运行期间，只有管理服务器才可以修改配置文件。管理控制台及命令行管理工具提供了访问管理服务器的手段，你可以通过它们来修改域的配置。一个域被激活后，可以通过管理控制台监视或配置整个域的资源。

另外非活动域的配置信息保存在配置存储库（configuration repository）中，你可以通过管理控制台来编辑这些文件。配置存储库由位于/config 目录下的一系列子目录构成。任何域都是由位于一个与该域同名的子目录下的 config.xml 唯一定义的。你可以通过管理控制台在启动时出现的欢迎页面上的 Domain Configuation 链接来访问非活动域的配置文件。

启动管理控制台

管理控制台是一个 Web 应用，它使用 JSP 来访问管理服务器所管理的资源。

管理服务器启动以后（见“启动与终止 WebLogic 服务器”），在浏览器中使用以下 URL 启动管理控制台。

http://*hostname:port*/console

其中 hostname 为管理服务器 DNS 的名字或 IP 地址，而 port 则为管理控制台用来监听请求的端口（缺省为 7001）。如果你是用安全套接层(SSL)来启动管理服务器，那么必须在 http 后面加上 s，如下所示。

https://*hostname:port*/console

如果浏览器被配置为使用代理服务器来发送 HTTP 请求，要将浏览器配置改为不使用代理服务器。如果管理服务器与浏览器位于同一台机器上，那么你要确保发送给 localhost 以及 127.0.0.1 的请求没有被发送给代理服务器。

管理控制台的左窗格包含了一个层次树状结构 — 域配置树，你可以通过这配置树来浏览数据表、配置页面、监控页面，或者是访问日志文件。在树中选择一个项目（即用鼠标左点项目），就可以显示某种类型的资源的相关数据或者显示某个资源的配置页面以及监控页面。域配置树的顶层的节点是一些容器，如果容器里包含有叶节点，那么你可以点击其左边的加号来展开配置树并访问叶节点。

实体表（某特定类型的资源的数据表）是可以定制的，这可以通过减少或增加显示不同属性

的列来实现。你也可以点击表格上部的“Customize this table”链接对它进行定制。表中的每一列都对应于被选中一个属性。

启动管理控制台需要输入口令。第一次可以使用与启动管理服务器相同的用户名和口令来启动管理控制台，然后你可以使用管理控制台来向管理员用户组中添加用户，此后这些用户就可以通过管理控制台来执行管理工作了。管理员用户组中缺省的成员是 system

因为管理服务器只能管理一个活动的域，同一时间你只能使用管理控制台访问一个活动域。如果你有不同的管理服务器运行，那么每个服务器都有他自己的活动域，你可以激活你希望访问的管理服务器上的管理控制台，来切换要管理的域。

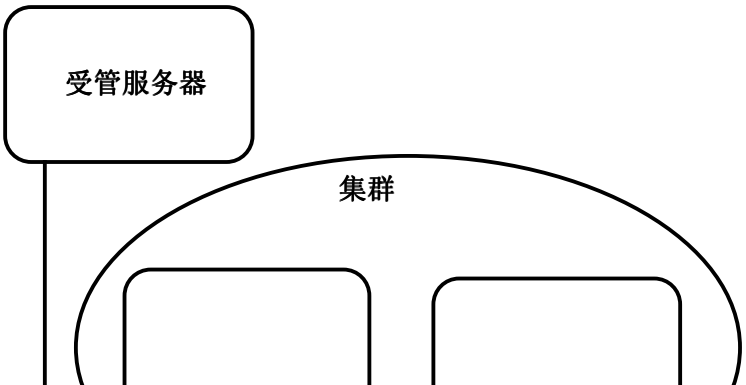
运行时对象与配置对象

管理服务器中有许多类似于 JavaBean 的 Management Beans(MBeans)对象。Mbeans 遵循 Sun 的 Java 管理扩展标准（JMX）。这些对象提供了对域资源的管理访问。

管理服务器包含了 configuration Mbeans 与 run-time Mbeans。管理 Mbeans 提供了配置属性的 SET（写） 与 GET（读）的访问。

Run-time Mbeans 提供了域资源信息的快照，例如当前 HTTP 会话的信息与 JDBC 连接池的负载信息。如果域的某个资源（例如 Web 应用）被实例化，那么服务器会创建一个 Mbeans 的实例来收集这个资源的信息。

当你从管理控制台访问某一资源的监控页面时，管理服务器执行 GET 操作获取当前的属性值。



管理服务允许域资源的属性可以被动态的修改，即 WebLogic 服务器处于运行状态中，也可以修改这些属性。许多属性改变不需要重启服务器就能生效。这时，修改后的属性不仅表示当前属性值，还会被保存到配置文件中。（有关配置 WebLogic 服务器的更多信息，请参见“配置 WebLogic 服务器与集群”中的内容。）

除了基于 Web 的管理控制台外，WebLogic 服务器还提供了命令行工具来访问域资源配置及监视属性。可以用命令行工具创建脚本，可以使系统的管理自动化。（请参见“WebLogic 域管理命令”）

日志消息的集中访问

通过管理服务器提供的域日志，你可以集中地访问所有服务器的关键系统消息。通过 JMX 提供的基本功能，消息可以转发到订阅该消息的实体中。订阅实体通过设置过滤器来选择感兴趣的消息。本地服务器在启动时发向其它网络实体的信息称为一个布告。JMX 布告使域内所有服务器的关键日志消息都被转发给管理服务器。

在 WebLogic 受管服务器启动时，管理服务器会进行注册以便接受关键日志消息。这些消息被存储在域日志中。管理服务器向域里的每一个 WebLogic 服务器注册一个域日志过滤器来选择需要转发的消息。你可以通过管理控制台改变域日志过滤器，查看域日志以及查看本地服务

器日志。（详细内容，请参见“使用日志消息管理 WebLogic 服务器”）

创建一个新域

本节讲述如何创建一个新的域。所有的 WebLogic 管理域的配置信息都在配置存储库中，它位于/config 目录下。每个域在 /config 目录下都有一个单独的子目录。为域建立的子目录的名字必须和域名相同。

当你第一次安装 WebLogic Server 软件的时候，建议你创建一个 zip 文件包含缺省/mydomain 配置目录。你应该保留这个 zip 文件，作为一个备份，你可以创建新的域。这个子目录包含了运行配置需要的组件，例如 filerealm.properties 文件和配置文件。

要创建新域，如下进行：

1 在已经存在的域下（比如 mydomain），启动管理服务器。

2 让浏览器访问http://hostname:port/console 来激动活管理控制台。

这里的 hostname 是启动管理服务器的机器名，端口是管理服务器的监听端口（缺省是 7001

3 选择 mydomain->Create or edit other domains。显示域列表。

4 选择 Default->Create a new Domain

输入新的域名，点击 Create

5 从列表中选择新建的域，使之成为当前域。

6 现在你可能希望为新域创建一个管理服务器项：

a. 选择 Servers->Create a new Server

b. 输入新的管理服务器的名字，单击 Create

7 管理控制台将以你的命名的新域名字创建一个子目录，并在这个子目录下创建一个新配置文件，config.xml。现在你需要创建一个 \application 子目录。在 unix 下通过 shell 创建，Windows 下，可以通过资源管理器创建。

8 缺省的 mydomain 目录下包含启动 WebLogic Server 所需要的脚本。对于 Windows 是 startWebLogic.cmd 和 startManagedWebLogic.cmd 。对于 Unix 是 startWebLogic.sh 和 startManagedWebLogic.sh

9 你可能需要的文本编辑器其中编辑脚本。缺省的，启动脚本设置域名为：
-Dweblogic.Domain=mydomain
用新建的域名替代这里的 mydomain
缺省的管理服务器设置成：
-Dweblogic.Name=MyServer
用新的管理服务器的名字替换 MyServer

10 脚本的最后是 cd 命令：
cd config\mydomain
使用新域名替换里面的子目录 mydomain。同样在启动脚本里面也有一行：
echo startWebLogic.cmd must be run from the config\mydomain directory.
使用新域名替换掉里面的 mydomain

11. 从缺省的 mydomain 目录复制文件 SerializedSystemIni.dat 和文件 fileRealm.properties 到你的新建域的目录。不要在复制完这些文件之前就试图重新启动管理服务器。

一旦你已经完成这个过程，你可以为你的新域启动管理服务器。

第二章 启动与终止 WebLogic 服务器

本章将介绍以下内容

WebLogic 管理服务器与 WebLogic 受管服务器

启动 WebLogic 管理服务器

将 WebLogic 受管服务器加入域

启动 WebLogic 受管服务器

通过管理控制台终止 WebLogic 服务器

受管服务器暂停和恢复

将 WebLogic 服务器设置为 Windows 服务

注册启动类与终止类

在 J2EE1.2 模式下运行 WebLogic 服务器。

WebLogic 管理服务器与 WebLogic 受管服务器

一个 WebLogic 域由多个 WebLogic 服务器组成，WebLogic 服务器可以运行为管理服务器或者受管服务器模式。每个域中必须有一个（不可以多于一个）管理服务器。域中的其它 WebLogic 服务器被称为受管服务器。WebLogic 服务器是管理服务器还是受管服务器取决于启动时的命令行选项。

管理服务器是 WebLogic 服务器的缺省角色。因此如果域中只有一个 WebLogic 服务器，那么该服务器的角色就是管理服务器。在一个多服务器的域中，只有当服务器在启动时被要求从一个运行着的管理服务器获得配置时才会成为受管服务器。

管理服务器控制对 WebLogic 域配置的访问以及提供诸如监控及日志消息浏览等功能。用户通过管理控制台来访问管理服务器所提供的管理服务。

WebLogic 受管服务器在启动时会从管理服务器获得它的配置。因此启动一个多服务器的域需要两个步骤：先启动管理服务器，然后启动受管服务器。

注意：受管服务器的版本必须与管理服务器的版本相同。

启动消息

在 WebLogic 启动时，标准日志子系统还不能用于日志记录。因此，任何在启动时发生的错误都会输出到 stdout，如果使用节点管理器，从管理控制台远程启动受管服务器，这些消息会显示在管理控制台中的右侧窗格中。

启动 WebLogic 管理服务器

启动 WebLogic 管理服务器有以下多种方式：

从命令行启动

启动 WebLogic 服务器的命令可以在命令窗口中手工输入，也可以把启动命令写在一个脚本中，从而避免每次启动服务器时都要重输命令。有关 WebLogic 服务器提供的脚本示例的详细信息，请参见“使用脚本启动 WebLogic 受管服务器”中的内容。

从启动菜单启动 WebLogic 服务器（只用于 Windows

如果你将 WebLogic 服务器安装为一个 Windows 服务，那么在计算机启动时 WebLogic 服务器将自动启动。

WebLogic 服务器启动时的口令使用

安装 WebLogic 的过程中，系统会要求你输入一个用于 WebLogic 启动的口令。如果你是用脚本来启动管理服务器与受管服务器，那么应该在脚本中将口令加入命令参数（请参见“从命令行启动 WebLogic 管理服务器”）。如果启动服务器的脚本没有将口令指定为命令行参数，那么在启动时系统会提示你输入口令。将口令设置为命令行参数可以避免启动时出现口令输入提示，但这样的话，口令是以明文形式保存在脚本文件中。

从 Start 菜单启动 WebLogic 管理服务器

在 Windows 平台下，如果 WebLogic 服务器是通过 BEA 安装程序安装的，那么你可以使用 Windows 启动菜单中的 WebLogic Server 快捷方式启动 WebLogic 管理服务器。选择：

Start-> Programs-> BEA WebLogic E-Business Platform-> Weblogic Server Version -> Start Default Server

其中 version 是指 WebLogic 服务器软件的版本号

使用启动菜单中的 WebLogic Server 快捷方式启动实际上就是调用了 startWebLogic.cmd 脚本（该脚本位于 install_dir/config/domain_name 目录下，其中 domain_name 是指域的名字，install_dir 是指 WebLogic 服务器软件的安装目录）。启动时，系统会提示你输入口令。

启动与终止 Windows 服务形式的 WebLogic 服务器

如果把 WebLogic 安装成 Windows 服务，那么 WebLogic 服务器会在计算机启动时自动启动。WebLogic 服务器通过启动脚本（如 startWebLogic.cmd）运行，执行 startWebLogic.cmd 脚本会将 WebLogic 服务器启动为管理服务器。参见“从命令行启动 WebLogic 管理服务器”中的内容。

要使 WebLogic 服务器作为 Windows 服务运行，需要在安装时设定。有关安装及删除 Windows 服务形式的 WebLogic 服务器，请参见“将 WebLogic 服务器设置为 Windows 服务”。

你可以按以下步骤从服务控制面板启动或终止 WebLogic 服务器：

1．选择 Start->Settings->Control Panel（编者注：对应中文 windows 就是开始->设置->控制面板->管理

工具）

2．双击服务控制面板，这样便打开了服务控制面板

3．在服务控制面板中找到 WebLogic 服务器。如果 WebLogic 已经启动，你可以使用 Stop 按扭来终止 WebLogic 服务器。如果 WebLogic 已经终止，那么 Start 按扭就可以被用来启动 WebLogic 服务器。

Windows 服务有三种模式：自动，手动与禁用。你可以通过 Startup 按钮来选择其中一种模式。

从命令行启动 WebLogic 管理服务器

因为 WebLogic 服务器是一个 Java 类文件，因此与其它 Java 应用一样，你可以使用 Java 命令来启动 WebLogic 服务器。启动 WebLogic 服务器的参数非常长，因此如果要从命令行来启动它，那么你必须输入一长串的参数，这是非常烦人的。为了保证启动命令的正确性，BEA 建议你将命令写入到一个脚本中，然后用这个脚本来启动 WebLogic 服务器。

以下参数是用 Java 命令行启动 WebLogic 管理服务器所必需的：

Java 堆内存的最大与最小值

例如，你想使用缺省的 64M 堆内存来启动 WebLogic 服务器，那么你就应该使用 java –ms 64m 与 –mx 64 选项来启动服务器。

建议最大和最小值使用相同的值，这样 JVM 不需要调整全局堆大小，提供最好的效率。

上述参数值会严重影响 WebLogic 服务器的性能，上面所提供的值只是一个缺省值。在生产环境中，你应该仔细考虑应用及环境所要使用的堆内存的大小。

设置 java –classpath 选项

该选项的最简要的设置可以参见“设置类路径选项”

指定服务器的名字

域的配置通过服务器名字指定。在命令行中通过以下参数来指定服务器的名字：

-Dweblogic.Name=Servername

缺省值为 myserver

提供用户口令

缺省用户为 system，口令为安装时所输入的口令。使用以下参数给出该用户的口令：

-DwebLogic.management.password=password

如果你不是从 WebLogic 根目录启动 WebLogic 服务器，那么需要指定 WebLogic 根目录的位置。

域的安全资源以及配置存储库（缺省为\config 目录）位于 WebLogic 根目录下。你可以用以下参数在命令行中指定 WebLogic 的主目录：

-Dweblogic.RootDirectory=*path*

其中 path 是主目录的路径。如果命令行中没有指定该属性，那么当前目录就被设置为该属性的运行值。

为了使在升级 WebLogic Server 软件的时候，维护域配置信息和应用方便，建议不用让 WebLogic

Server 软件的安装路径和根路径相同。当它不在安装路径下的时候，RootDirectory 属性用来定位你的域配置。

指定 bea.home 路径位置

-Dbea.home = root_install_dir

其中 root_install_dir 是你安装 WebLogic 服务器的根目录。

如果要使用 SSL 协议，那么在启动时需要把私钥密码传递给服务器以便服务器可以对 SSL 私钥文件解密。启动时，在命令行中用以下参数来传递 SSL 私钥口令：

-Dweblogic.pkpassword=*pkpassword*

其中 pkpassword 是 SSL 私钥密码。

在命令行中使用以下参数可以在启动管理服务器时指定域配置的名字：

-Dweblogic.Domain=domain_name

其中 domain_name 是域的名称。用来启动域的配置文件保存在同名子目录下。

配置存储库由/config 目录下的域组成。配置存储库可能包含多个域配置。每个域分别位于一个子目录中，子目录的名字与域的同名。指定 domain_name 时，实际指定的是这个子目录的名字。所指定的子目录包含了一个 XML 配置文件（config.xml）以及对应域的安全资源（见下面的例子）。域的配置由 config.xml 文件指定。

启动管理服务器所使用的域配置使这个域成为活动域。每次只能有一个域是活动的。

在命令行中还可以指定 WebLogic 配置属性的值。所指定的值成为属性的运行时值。而保存在永久配置中的值将被忽略。在命令行中设置 WebLogic 属性的值采用以下格式：

-Dweblogic.*attribut*e=*value*

自动部署功能（缺省值为允许使用），可以探测在活动域的\application 目录下所部署的应用的变更情况。建议你在生产系统环境中，关闭此功能，如果启动管理服务器时希望关闭自动部署功能，在命令行中添加：

-Dweblogic.ProductionModeEnabled = true

设置类路径选项：

以下参数必须包含在 java 命令行的 classpath 选项中

/weblogic/lib/weblogic_sp.jar

/weblogic/lib/weblogic.jar

WebLogic 服务器还包含一个名为 Cloudscape 的试用版数据库系统。Cloudscape 数据库系统是纯 Java 的数据库管理系统。如果你想使用这个 DBMS，那么 CLASSPATH 还应该包含:

/weblogic/samples/eval/cloudscape/lib/cloudscape.jar

如果使用 WebLogic Enterprise Connectivity，那么类路径中还应该包含：

/weblogic/lib/poolorb.jar

其中 weblogic 指 WebLogic 服务器的安装目录。

用脚本启动管理服务器

WebLogic 提供了一个用于启动 WebLogic 服务器的脚本示例。你可以根据环境及应用的需要对该脚本做适当的修改。启动管理服务器与启动受管服务器使用不同的脚本。启动管理服务器的脚本为 startWebLogic.sh(UNIX 环境)与 startWebLogic.cmd(Windows 环境)。这些脚本位于域配置子目录下。

使用 WebLogic 软件所提供的脚本示例时，应：

特别注意类路径的设置与目录名称

将变量 JAVA_HOME 的值改为 JDK 所在的目录

UNIX 用户还要修改示例脚本文件的权限，以使该文件可以被执行。例如

chmod +x startAdminWebLogic.sh

在受管服务器运行时重启管理服务器

在生产环境中，我们建议将包含关键商业逻辑的应用部署在受管服务器中。这种情况下，管理服务器所起的作用只是配置与监控受管服务器。因此，在这种配置下，即使管理服务器不可用，运行在受管服务器中的应用仍然可以继续处理客户端请求。

管理服务器在启动时，会复制一份用来启动活动域的配置文件。所复制的文件被保存为：

install_dir/config/*domain_name*/config.xml.booted

其中 install_dir 指 WebLogic 服务器软件所在的目录，domain_name 是域的名字。只有当管理服务器成功启动并可以处理请求时，它才会创建 config.xml.booted 文件。

你应该对这个文件进行备份，这样你可以获得一个包含工作配置的文件，它可以帮助你将通过管理控制台更改过的活动配置，回退到以前的配置。

如果在受管服务器运行时，管理服务器发生失败，你不需要重启受管服务器来恢复对域的管理。如何恢复对活动域的管理取决于是否可以在同台机器上启动管理服务器。

在同一台机器上重启管理服务器

在受管服务器正在运行的情况下重启管理服务器时，如果让管理服务器执行寻找操作，那么管理服务器会寻找到所有正在运行的受管服务器。要让管理服务器执行寻找受管服务器的操作，需要在启动管理服务器的命令行中使用以下参数：

-Dweblogic.management.discover=true

该属性的缺省值为 true。域配置目录中的 running-managed-servers.xml 文件列出了该管理服务器能识别出的受管服务器。如果管理服务器被指示在启动时执行寻找操作，那么它将使用这个列表来检查处于运行中的受管服务器。

重启管理服务器不会改变受管服务器的运行时配置。因此如果你修改了那些只能静态配置的属性，那么只有重启受管服务器才能使更改生效。发现受管服务器的操作可以使管理服务器的监视受管服务器或者是更改那些可以动态配置属性的运行时值。

在其它机器上重启管理服务器

如果机器崩溃导致你无法在先前运行管理服务器的机器上重启管理服务器，你可以按照以下步骤来恢复对受管服务器的管理。

1．将另一台机器的主机名设为先前管理服务器所在服务器的主机名。

2．在这台将作为管理服务器的新机器上安装 WebLogic 服务器软件（如果该机器上没有安装 WebLogic 软件的话）

3．先前用来启动管理服务器的机器中的/config 目录（the configuation repository）必须可以被新机器使用。/config 目录可以通过备份介质获得也可以通过 NFS mount 获得。该目录下包含用来启动活动域的配置文件(config.xml)以及安装在/applications 目录下的应用与组件。

4．在命令行中加入以下参数来重启新机器中的管理服务器

-Dweblogic.management.discover=true

使用上述参数会强制管理服务器去检测正在运行的受管服务器。

将 WebLogic 受管服务器加入域

在运行管理服务器之前，你必须在域的配置文件中添加该服务器的条目。步骤如下

启动域中的管理服务器

在浏览器中输入http://hostname:port/console 以启动管理控制台。其中 hostname 是运行管理服务器的主机名， port 是管理服务器的监听端口（缺省为 7001

创建一个运行服务器的机器（Machines->Create a new machine (如果和管理服务器的机器不同)

在管理控制台中为服务器创建一个条目（Servers->Create a new server）。将受管服务器中 Machine 选项设置为刚刚创建的机器。

有关服务器配置的更多内容，请参见“配置 WebLogic 服务器集群”中的内容。

启动 WebLogic 受管服务器

WebLogic 受管服务器可以用下面的方式启动：

- 从管理控制台远程启动，此种方式使用要启动的受管服务器所在的目标机器上的节点管理器来进行
- 本地启动，通过在命令 shell 使用 java 命令行启动。

本节讨论如何本地启动 WebLogic 受管服务器。关于建立和使用节点管理器远程启动受管服务器，请参考“节点管理器”。

注意：用鼠标右键点击管理控制台左侧窗格中 Server 名字时，有一个选项，”Start this Server …” 此选项只能在运行受管服务器的机器上安装了节点管理器时才可用，有关信息请参阅“节点管理器”。

在把 WebLogic 受管服务器加入到配置中以后（见“将受管服务器加入到域中”），你可以用 java 命令行启动受管服务器。启动受管服务器的命令可以手工输入，也可以编写成脚本以避免每次重启服务器时重复输入相同的内容。有关 WebLogic 所提供的脚本示例请参见“用脚本启动

WebLogic 受管服务器”中的内容。

受管服务器与管理服务器启动参数的主要区别在受管服务器需要一个用来识别管理服务器位置的参数，受管服务器通过这个参数从管理服务器获取配置。如果命令中没有这个参数，那么 WebLogic 服务器将启动为管理服务器。

启动受管服务器时候，除了要指定启动管理服务器时候指定的参数（参阅“通过命令行启动管理服务器”），还要指定以下启动 WebLogic 受管服务器所必须的参数：

指定服务器的名字

当 WebLogic 受管服务器从管理服务器请求自己的配置信息时，管理服务器通过服务器名来识别该受管服务器，这样管理服务器就可以将合适的配置信息传递给受管服务器。因此，在启动受管服务器时，你必须设置服务器名。你可以在启动 WebLogic 受管服务器的命令中使用以下参数：

-Dweblogic.Name=*servername*

指定管理服务器的主机名与监听端口

在启动受管服务器时，必须指定管理服务器的主机名与监听端口，因为受管服务器需要从管理服务器获得配置信息。你可以在启动受管服务器的命令行中使用以下参数：

-Dweblogic.management.server=*host*:*port*

或

-Dweblogic.management.server=http://*host*:*port*

其中 host 是管理服务器所在机器的名字或 IP 地址，port 是管理服务器的监听端口。缺省情况下，该监听端口为 7001

如果使用 SSL 与管理服务器通信，那么管理服务器必须指定为：

-Dweblogic.management.server=https://*host*:*port*

如果管理服务器与受管服务器的通信采用 SSL 协议，那么你应该在管理服务器中启用 SSL 详细内容请参见“安全管理”。

注： 如果 WebLogic 服务器在启动时没有指定管理服务器的位置，那么该 WebLogic 服务器将启动为管理服务器。

注： 因为受管服务器从管理服务器获得其配置，因此所指定的管理服务器必须与受管服务器在同一个域中。

通过脚本启动 WebLogic 受管服务器

WebLogic 提供了用来启动 WebLogic 服务器的脚本示例。你可以根据实际运行环境与应用的情况修改脚本示例。启动管理服务器与受管服务器的脚本是不一样的。启动受管服务器的脚本为 startMangagedWebLogic.sh　Unix）与 startManagedWebLogic.cmd　Windows）。这些脚本位于域的配置子目录中。你可以对这些提供的模斑加以修改，以编写自己的启动脚本。

使用脚本示例时：

应注意 classpath 路径的设置与目录名称

将变量 JAVA_HOME 的值设置为 JDK 所在的目录

UNIX 用户还应修改示例脚本的权限，使该文件成为可执行文件。例如：

chmod +x startManagedWebLogic.sh

以下是用脚本启动受管服务器的两种方式：

如果已经设置了 SERVER_NAME 与 ADMIN_URL 环境变量，调用启动脚本时就不需要提供这两个参数的值了。SERVER_NAME 变量应该设为要启动的 WebLogic 受管服务器的名字。ADMIN_URL 设置为管理服务器所在机器的主机名及其监听端口（缺省为 7001）。例如

set SERVER_NAME=bigguy

set ADMIN_SERVER=peach:7001

startManagedWebLogic

你可以在调用启动脚本的命令行中传入受管服务器的名字与管理服务器的 URL

startManagedWebLogic *server_name admin:url*

其中 server_name 是要启动的受管服务器的名字，admin_url 可以是 http://*host*:*port* 或者是 https://*host*:*port*，其中 host 是指管理服务器所在机器的主机名或者是 IP 地址， port 是管理服务器的监听端口。

从管理控制台终止 WebLogic 服务器

当你在管理控制台左侧窗格中，用鼠标右键点击一个服务器时，你可以看到两个选择“Kill this Server…”和“Stop this Server…”。当你选择“Kill this Server…”时，管理服务器向运行该受管服务器的机器上的节点管理器发送请求，节点管理器则终止掉 WebLogic 服务器的进程。“Kill this Server…”不可用于关闭管理服务器，而且该选项假设运行 WebLogic 服务器的目标服务器的机器上运行节点管理器，关于节点管理器的安装和设置，参见“节点管理器”。

如果选择“Stop this Server…”，管理服务器向所选的受管服务器发送关闭服务器指令请求，在这种情况下，节点管理器并不被使用，与“Kill this Server…”不同，“Stop this Server…”选项可以终止管理服务器。

因为“Stop this Server…”选项使用受管服务器的管理功能来执行关闭动作，只有该服务器处于运行状态，而且可以响应管理请求才可行。“Kill this Server…”选项通常用于目标受管服务器挂起或不响应管理服务器的管理命令请求的情况下。

从命令行停止服务器

你可以用以下命令停止 WebLogic 服务器：

java weblogic.Admin -url *host*:*port* SHUTDOWN -username *adminname* -password *password*

其中：

host 是运行 WebLogic 服务器的主机名或 IP 地址。

port 是 WebLogic 服务器的监听端口（缺省为 7001

adminname 指的是具有存在于控制台访问控制列表的成员用户（或者是控制台访问控制列表的组成员的成员）。缺省的控制台访问控制列表的成员为 system

password 指的是 adminname 用户的口令。

暂停和恢复受管服务器

你可以通过管理控制台暂停一个 WebLogic 受管服务器，此时，WebLogic 受管服务器只接受来自管理服务器的请求。这种情况的典型应用是将一个 WebLogic 服务器作为另一台服务器的“热”备份运行。该备份服务器将一直保持暂停状态，直到你让它处理请求为止。

注意： 被暂停的 WebLogic 服务器只是不响应 HTTP 请求，而 Java 应用与 RMI 调用没有被暂停。

要暂停一个 WebLogic 受管服务器：

在管理控制台的域树上（位于左边的窗格），选择你要暂停的服务器点击右键。

选择“Suspend this server”链接。

要使受管服务器恢复对客户端请求的处理：

在管理控制台的域树上，选择需要恢复的服务器点击右键。

选择 Resume this server 链接

将 WebLogic 服务器设置为 Windows 服务

WebLogic 服务器可以作为 Windows 服务运行。如果你将 WebLogic 安装为 Windows 服务，那么，在启动计算机时，系统会调用启动脚本 startWeblogic.cmd 而启动 WebLogic 服务器。WebLogic 服务器是启动为管理服务器还是受管服务器取决于调用 WebLogic 服务器的 java 命令中的参数设置。具体内容请参见“通过命令行启动 WebLogic 受管服务器以及 WebLogic 管理服务器”。

要使 WebLogic 服务器以 Windows 服务的形式运行或者不再将其运行为 Windows 服务，你首先要有管理员级权限。要将 WebLogic 服务器作为 Windows 服务启动，需要：

1　找到 weblogic\config\mydomain 目录（其中 weblogic 是安装 WebLogic 服务器的目录，mydomain 是与你所在的域对应的子目录）。

2．执行 installNTService.cmd 脚本。

删除 Windows 服务形式的 WebLogic 服务器

删除 Windows 服务形式的 WebLogic 的步骤如下：

1. 定位到 weblogic\config\mydomain 目录（其中 weblogic 是安装 WebLogic 服务器的目录，而 mydomain 是域配置所在的子目录）。

2．执行 uninstallNTService.cmd 脚本

你也可以从 Windows 的启动菜单中卸载 WebLogic 服务。

更改安装成 Windows 服务的服务器口令

如果你将缺省服务器安装为 Windows 服务，那么创建服务会用到安装 WebLogic 软件时键入的口令。如果要更改这个口令，你应该：

1．使用 uninstallNTService.cmd 脚本来卸载作为 Windows 服务的 WebLogic 服务器（该脚本位于 install_dir/config/domain_name 目录下，其中 install_dir 是安装 WebLogic 产品的目录）。

2　installNTservice.cmd 脚本包含了以下命令：

rem *** 安装服务

“C:\bea\wlserver6.0\bin\beasvc” -install -svcname:myserver

-javahome:”C:\bea\jdk130” -execdir:”C:\bea\wlserver6.0”

-extrapath:”C\bea\wlserver6.0\bin” -cmdline:

%CMDLINE%

在上述命令后加上以下命令：

-password:”your_password”

其中 your_password 是新口令

3．执行更改后的 installNTservice.cmd 脚本。这将使用更新的口令创建一个新服务

注册启动与终止类

你可能想在 WebLogic 启动或正常关闭时执行某些任务，那么 WebLogic 所提供的启动类与终止类就是实现这些任务的一种机制。启动类是在 WebLogic 服务器启动或重启时自动装载并执行的 Java 程序，启动类在所有的服务器初始化任务都完成后才会被装载及执行。

终止类的工作原理与启动类相同。当你通过管理控制台或者是使用 *weblogic.admin shutdown* 命令来终止 WebLogic 服务器时，终止类会自动装载并执行。

要使 WebLogic 服务器使用启动类或终止类，你必须通过管理控制台注册这些类。

注册启动类或终止类的步骤如下：

1．通过管理控制台的 Domain 树（位于左窗格）访问 Startup & Shutdown 表。在这个表中创建启动类与终止类的条目。

2．在 Configuration 标签页中，为所添加的终止及启动类提供名字及其它必要的参数。

详细信息，请参见管理控制台在线帮助以下部分的内容：

启动类

终止类

WebLogic 服务器 Windows 服务程序（beasvc.exe

将 WebLogic 服务器安装为 Windows 服务以及卸载该服务会调用服务程序（beasvc.exe），采用 beasvc.exe 可以安装或卸载多个作为 Windows 服务的 WebLogic 服务器实例。

同时 beasvc.exe 程序也可以将节点管理器安装为 Windows 服务或者对其卸载。关于如何安装几卸载作为 Windows 服务的节点管理器，参见“节点管理器”。

多个服务的配置信息，以不同的服务名称保存于 Windows 注册表中一个的特定服务器路径下：HKEY_LOCAL_MACHINE\SYSTEM\Current\ControlSet\Services

当启动服务时，Windows 注册表寻找 JVM 信息并启动，因为每个服务的安装独立于其他服务，你可以运行多个作为 Windows 服务的 WebLogic 服务器实例，每个服务器指定唯一的服务名称。

以下是 beasvc.exe 命令的有关选项：

–install

安装指定的服务

–remove

卸载指定服务

–svcname: *service_name*

用户指定要安装或卸载的服务名称

–cmdline: *java_cmdline_parameters*

用于启动windows服务的WebLogic服务器的java命令行参数，

–javahome: *java_directory*

Java 安装的根路径，启动命令由 *java_directory*加\bin\java组成。

–execdir: *base_dir*

启动命令所在的根路径。

–extrapath: *additional_env_settings*

其他路径设置，用于添加到可用的路径前，用以执行此命令。

–help

打印出执行 beasvc.exe 的使用帮助

Win32 系统对命令行长度有 2K 的长度限制，如果作为 Windows 服务的 WebLogic 服务器启动命令的 classpath 很长，2K 的限制可以超过。

在 Sun Java 1.2 及其以后的版本，你可以用@选项指定命令行的 classpath，下例显示了在 beasvc.exe 中使用@选项。

```
beasvc –install –svcname:myservice -classpath:@C:\temp\myclasspath.txt
```

在 J2EE1.2 模式下运行 WebLogic 服务器

缺省情况下，WebLogic6.1 支持 J2EE1.3 API 接口，它包括

- J2EE2.0
- JSP1.2
- Servlet 1.2
- J2EE Connector Architecture 1.0

在 WebLogic6.1 版本发布时，这些 API 接口还没有最终定稿，尽管 J2EE2.0 向后兼容 J2EE1.2，但一些客户希望加强对 WebLogic 服务器的限制，让它只使用 J2EE1.2 所支持的 API.

这样做，WebLogic 只运行于 J2EE1.2-only 模式下，J2EE1.2-only 模式不支持上面所列出的这些 J2EE1.3 的 API.

使用 WebLogic 运行于 J2EE1.2-only 模式，在启动 WebLogic 服务器的命令行中添加：

–Dweblogic.J2EE12OnlyModeEnabled=true

WebLogic服务器提供了一个运行于J2EE1.2-only模式下，启动管理服务器的启动脚本，脚本名为startWebLogic_j2ee12.cmd，它位于*install_dir*/config/mydomain目录下，其中*install_dir*是WebLogic 服务器软件安装的根路径。

第三章 节点管理器

本章将介绍以下内容：

节点管理器（Node Manager）概述

配置节点管理器

节点管理器平台支持

从命令行启动节点管理器

使用脚本程序启动脚本管理器

受管服务器的远程启动与终止

将节点设置为 Windows 服务设置

节点管理器概述

节点管理器是一个 Java 应用程序。借助该应用，你可以从管理控制台远程地启动或终止 WebLogic 受管服务器。节点管理器是单独的一个 Java 应用，随同 WebLogic 服务器软件供应。

你可以通过管理控制台来终止受管服务器，另一种方式是用节点管理器终止远程受管服务器。当远程服务器被挂起或没有响应时，就需要终止远程服务器进程。

为了能远程启动受管服务器，首先要在受管服务器所在的机器上配置并运行节点管理器。一个节点管理器进行可以负责一台机器上所有受管服务器的远程启动与终止。为了保证节点管理器的可用性,应该把节点管理器配置为 UNIX 机器的守护程序或 Windows NT 机器的 Windowns NT 服务。这保证了机器上的节点管理器可以用来启动受管服务器。

如果一台机器中运行了节点管理服务器，那么当它获得管理服务器的请求后，可以启动或终止这台机器上所安装配置的任何受管服务器。节点管理器与管理服务器之间的通信采用安全套接字层（Secure Socket Layer

节点管理器日志

WebLogic 服务器在启动时，会将各种启动与错误消息打印到 STDOUT 或 STDERROR 中，这些消息同时会显示在管理控制台的右窗格中。在其他时刻，这些文件是可以追溯的，在管理控制台左侧窗格中，右键点击一个服务器，选择“Get StdOut for this Server…”和“Get StrErr for this Server”来查看。

缺省情况下，节点管理器将这些信息保存在节点管理器日志目录下的文件中，缺省的，这个目录为 NOdeManagerLogs，它在启动节点管理器时建立，若需要改变目录名称时，可以在启动节点管理器时设置，详细信息参见“命令行参数”。

对对节点管理器启动的每一个受管服务器，节点管理器会分别创建一个单独日志文件子目录。存储在目录中的日志包括：

servername.pid

以受管服务器 servername 名保存的 processID，当管理服务器发送请求时，节点管理器可以利用它来终止服务器进程。

config

当从管理服务器启动受管服务器时，所有经过节点管理器的配置信息将被保存

servername-output.log

当节点管理器试图启动名为 servername 的受管服务器时，它保存了打印到 stdout 的信息。

如果有新的启动服务器的尝试时，该文件名通过附加-prev 而修改名字。

servername-error.log

当节点管理器试图启动名为 servername 的受管服务器时，它保存了打印到 stderr 的信息。

如果有新的启动服务器的尝试时，该文件名通过附加-prev 而修改名字。

节点管理器日志也保存在运行管理服务器的机器上的一个名为/config/NodeManagerClientLogs 的临时目录下。当你通过节点管理器去启动一个受管服务器时，它会为每一个受管服务器建立一个相应的子目录。这些目录中的每个日志对应了相应的动作，例如：启动或终止服务器，日志文件名包含了时间戳，用以表示动作的执行时间。建议你定期删除累积的这些由于节点管理器而产生的客户日志文件。

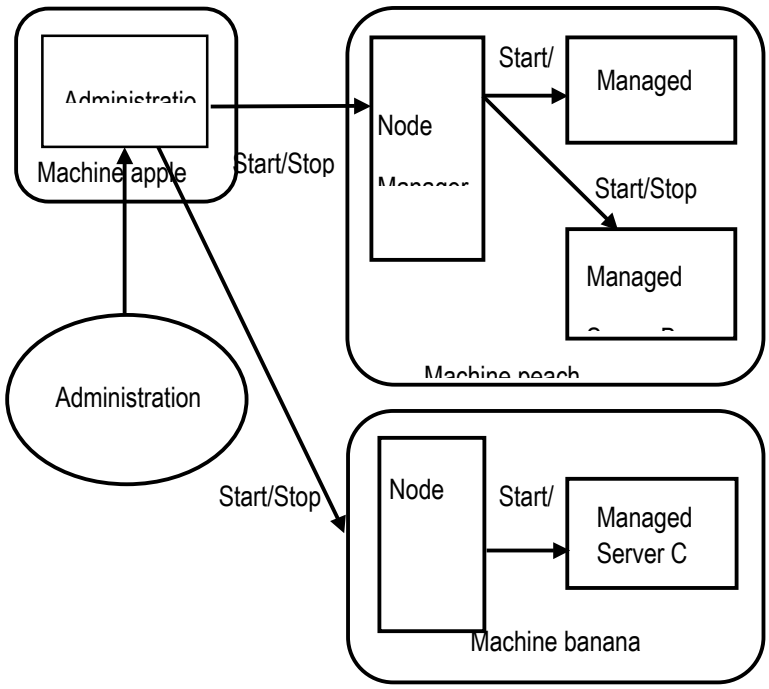


图 3—1 节点管理器结构图

配置节点管理器

节点管理器与管理服务器之间的所有通信都使用安全套接层（SSL）以提供身份认证与加密。同时还用客户端验证保证节点管理器与管理服务器之间的所有通信都使用双向认证。为了进一步提高安全性，节点管理器还保存了一个可靠主机的列表。节点管理器只接受这些主机上的管理服务器所发出的命令。因此你需要为节点管理服务器编辑一个可靠主机列表文件，在文件中为每个可以向节点管理器发出命令的管理服务器所在主机加上一行。该主机列表文件的名字为 nodemanager.host，安装在\config 目录中。缺省情况下，该文件包括以下两个条目：

localhost

127.0.0.1

你可以在命令行中修改节点管理器所寻找的可信任主机列表的文件名，详细命令参见“命令行参数”，对可信任主机可以使用 DNS 或者 IP 地址来表示，如果使用 DNS 名，你需要在节点管理器启动时，允许 DNS 反向搜索。通过设置以下参数可以实现：

-Dweblogic.nodemanager.reverseDNSEnabled = true

缺省情况下是不允许 DNS 反向搜索

在典型的生成环境中，节点管理器与管理服务器不会运行在同一主机中。因此，应该编辑该可信任主机列表文件，使它只包含可以启动或终止本机受管服务器的管理服务器所在机器的主机名或 IP 地址。可信任主机列表文件的每个条目由一行构成，列出了管理服务器所在机器的 DNS 主机名或 IP 地址。

节点管理器的 SSL 设置

节点管理器与管理服务器之间的所有通信都使用安全套接层（SSL）。为保证节点管理器和管理服务器之间通讯安全，使用了双向的 SSL 认证。

认证需要采用公用钥密构。它包括了私钥和证书。证书通常包含了用户的公钥，并被证书的发行方加以签名，用以验证用户名称和内置公钥所包含信息的绑定。

节点管理器的证书采用 X.509 格式。而节点管理器使用私钥遵从私钥密码加密标准 PKCS#5/#8 Private Key Cryptography Standard #5/#8 PKCS#5 是基于口令的加密标准，它描述了私钥和口令的加密方法。PKCS#8 是私钥的语法标准，并规定了私钥的字符特征。

节点管理器使用的公钥证书结构不同于遵从早期标准的 WebLogic 服务器的数字证书，它们之间的主要区别包括：

- 节点管理器使用单一证书文件，它包含了私钥以及用户公共身份的证书。
- 节点管理器使用的私钥必须有口令保护，并遵守 PKCS#5/#8 标准

WebLogic 服务器软件提供了一个用于节点管理器的证书范例，它保存为 /config/demo.crt，建议你在生产环境中使用新申请的数字证书。

节点管理器中建立数字证书的步骤如下：

步骤 1：获取数字证书和私钥

有两种方式可以获取用于节点管理器的数字证书：

- 你可以从认证机构获取私钥和 X.509 格式的数字证书，有关如何获取数字证书，见“安全管理”。如果获取的私钥不符合 PKCS#5/#8 格式，你需要使用 WebLogic 的密钥转换工具转换。如果从认证机构获取了 PKCS#5/#8 格式的证书，可以忽略步骤 3 合并证书至单一文件。
- 如果你希望使用由 WebLogic 证书生成器所生成的证书，可以将其转换以供节点管理器使用。

步骤 2：转换一个 WebLogic 风格的私钥

如果你想在节点管理器中使用 WebLogic 风格的证书，你首先需要将私钥转为新的 PKCS#5/#8 格式。WebLogic 提供了一个转换工具。用于将 WebLogic 风格的证书转换供节点管理器使用的工具是 wlkeytool，它位于：

- WebLogic 安装路径下的/bin 目录中　Windows 平台）
- WebLogic 安装路径下的/lib 目录中（UNIX 平台）

wlkeytool 语法为：

wlkeytool *old_key new_key*

你会被提示输入私钥口令以解锁，若没有口令则直接回车。然后你会被提示输入新口令以用于加密。同节点管理器一起使用时需要一个口令。

例如：

wlkeytool demokey.pem demokey_new

步骤 3：合并证书至单一文件

WebLogic 对私钥、公钥和证书认证机构（证书链中一系列的认证机构）使用单独的证书文件（扩展名为.pem）。另外私钥还要遵循 PKCS#5/#8 格式，并有口令保护，节点管理器则将这些证书组件合并为一个证书文件（扩展名为.crt

注：尽管表明用户的 SSL 身份的所有组件合并在一个文件中，私钥信息并不在服务器之间传递。

三个组件简单的合并连接为一个扩展名为.crt 的文件

例如：

cat demokey_new democert.pem ca.pem > demo.crt

在这个例子中，ca.pem 是 WebLogic 服务器证书文件，它类似于缺省的认证文件，trusted.crt 文件是 demokey.pem 运行 wlkeytool 的结果（参见步骤 2 的描述）

有关数字证书和 SSL 的信息，参见“安全管理”。

配置管理服务器以启用节点管理器

为了让管理服务器能够使用节点管理器来启动或结束 WebLogic 受管服务器，需要完成一些事情。你可以使用 WebLogic 管理控制台来完成这些任务。

步骤 1：创建一个机器配置条目

你应该在域配置中每个安装了受管服务器的机器创建一个条目，步骤如下

1．运行管理服务器，启动管理控制台（如果还没有被运行起来）

2．在左窗格中选择 Machines 节点以显示机器表格

3．选择表上部的 Create a new Machine 链接（或 Create a new UNIX Machine

4．填写这个机器的信息并点 Apply 按钮创建一个新的机器条目。

步骤 2：配置每个机器的节点管理器

对于每个要使用节点管理器的机器，应该相应地改变这个机器的配置条目：

在管理控制台中，选择 Machines->*machine_name*->Node Manager。其中 *machine_name* 是运行节点管理器的机器的名字。

填写 Node Manager 标签页的以下字段：

Listen Address 或者是监听管理服务器请求的节点管理器所在机器的 IP 地址或主机名。该监听地址为你在启动节点管理器时所指定的地址。

Listen Port 的值必须与你在启动节点管理器时所指定的端口号一样。

管理服务器用来与该节点管理器会话的证书。默认的证书为/config/demo.crt。在生产环境中，建议你换一个新证书。详细信息参见“安全管理”中有关部分。

证书口令因加密而不可见，若需要更换节点管理器使用的证书，你需要修改口令，并使之与新证书中用于加密私钥的口令相同。

trustedCerts 文件包含了已知的认证机构列表，缺省的为 config/trusted.crt。你所使用的数字证书的颁发机构必须在此文件中。

1　点 Apply 按钮。

步骤 3：配置受管服务器的启动信息

如果要用节点管理器启动 WebLogic 受管服务器，那么应该为它提供启动受管服务器所需要的启动参数与选项。步骤如下：

1. 启动管理控制台，若它还没有启动。

2. 在管理控制台中，选择 *server_name* → Configuration→ start 其中 *server_name* 是受管服务器的名字。

有 5 个配置参数需要填写，以供管理服务器在启动目标受管服务器时使用：

注意：若未指定这些信息，就试图从控制台启动目标服务器，节点管理器会试着使用启动节点管理器时的这些属性值去启动受管服务器。当你在启动节点管理器命令行中提供了这些参数的值时，节点管理器就可以启动受管服务器了。

BEA Home

你需要指定 BEA Home 路径，它是所有安装在目标服务器上的 BEA 产品和证书的根路径。

Root Directory

是 WebLogic 软件安装所在的根路径。

Class Path

启动受管服务器的 classpath

最少情况下，需要对 classpath 做如下设置：

/weblogic/lib/weblogic_sp.jar

/weblogic/lib/weblogic.jar

还需要包含安装 JDK 的根路径，它用于启动受管服务器。

Arguments

在 Arguments 选项中添入你传递给启动命令的参数。

例如，你希望设定 java 使用的最大和最小堆内存。使用 –ms64m 和-mx64m 选项表明分配 64M 堆内存给 WebLogic 服务器。

注：不需要在此处指明服务器名、用户名或口令，同时也不需要指定管理服务器的主机名和端口。

Security Policy File

缺省使用的 JVM 的安全策略文件，WebLogic 也提供了一个安全策略文件，是 weblogic/lib/weblogic.policy

3. 点 Apply 按钮。

节点管理器的平台支持

目前，节点管理器只可用于 Windows 和 Unix 平台。在 Windows、Solaris 和 HP UX 平台运行节点管理器，需要本地库的支持。对非 Solaris 和 HP UX 的 UNIX 操作系统平台，你需要在启动节点管理器时，在 java 命令行中添加：

-Dweblogic.nodemanager.nativeVersionEnabled = false

从命令行启动节点管理器

有两种方式启动节点管理器。可以从 java 命令行启动或者使用启动脚本。有关使用脚本信息，参见“使用启动脚本启动节点管理器”，节点管理器还可以作为 Windows 服务。若作为 Windows 服务，当 Windows 重新启动时，该服务会自动重起。关于将节点管理器设置为 Windows 服务的详细信息，参见“将节点管理器设置为 Windows 服务”。

环境设置

启动节点管理器之间，需要设定一些环境变量，一种方式是通过 WebLogic 提供的脚本来设置环境变量。脚本文件为 setEnv.sh Unix 平台下）和 setEnv.cmd Windows 平台下）。脚本文件位于 install_dir/config/domain_name 目录下，其中 install_dir 是 WebLogic 安装的路径，domain_name 是域的名字（缺省为 mydomain

注：若使用节点管理器启动脚本（startNodeManager.cmd 在 Windows 平台下，startNodeManager.sh 在 Unix 平台下），你不需要设置这些环境变量，因为它们已经在脚本中设置了。详细信息参见“节点管理器启动脚本”

设置 Windows 平台下的环境变量

在你使用节点管理器，确认设定 JAVA_HOME 变量指向安装 JDK 的根路径。例如：

set JAVA_HOME= D:/bea/jdk131

节点管理器和 WebLogic 服务器对 JDK 的版本要求相同。

同时你还需要设定 WL_HOME 环境目录，例如：

set WL_HOME = D:\bea\wlserver6.1

另外，你还需要设定 PATH 环境变量，使之可以访问节点管理器的类和 java 可执行文件，例如：

set PATH=%WL_HOME%\bin;%JAVA_HOME%\bin;%PATH%

设置 Unix 平台下的环境变量

假定你已设定 WL_HOME 环境变量指向安装 WebLogic 的路径，下列例子演示了设定 PATH 环境变量，指向 WebLogic 和 JDK

PATH=$WL_HOME/bin;$JAVA_HOME/jre/bin:$JAVA_HOME/bin:$PATH

上例中假定 JAVA_HOME 环境变量指向 JDK 安装的根路径。

你还需要设定节点管理器使用的本地库的路径。下面是一个 Solaris 上的例子：

LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$WL_HOME/lib/solaris:$WL_HOME/lib/solaris/oci8

16_8

下面是 HPUX 的例子：

SHLIB_PATH=$SHLIB_PATH:$WL_HOME/lib/hpux11:$WL_HOME/lib/hpux11/oci816_8

设置 Classpath

可以在环境变量中或 java 命令行中设定 classpath 环境变量，下面是一个在 Windows 平台中将 classpath 设置为环境变量的例子：

```
set CLASSPATH= .;.\lib\weblogic_sp.jar;\lib\weblogic.jar
```

启动节点管理器

如果你不使用节点管理器的启动脚本，那么需要确认节点管理器在你安装 WebLogic 服务器软件的根目录中。该目录包含了\config 子目录。

启动节点管理器的命令为：

java weblogic.nodemanager.NodeManager

命令行参数

你需要指定节点管理器用于侦听管理服务器请求的主机名和 IP 地址：

-Dweblogic.nodemanager.listenAddress=*address*

节点管理器缺省侦听请求的端口为 5555，你可以在启动时修改该参数：

-Dweblogic.nodemanager.listenPort=*port*

节点管理器为每一个其负责管理的受管服务器建立日志，缺省的这些日志在 NodeManagerLogs 目录下，你可以在启动时通过参数修改：

-Dweblogic.nodemanager.savedLogsDirectory=*path*

节点管理起使用 SSL 和管理服务器之间进行通讯。因此，你必须在启动节点管理器时指明数字证书。在启动时加入如下参数指明证书的位置：

-Dweblogic.nodemanager.certificateFile=*path_to_cert*

节点管理器缺省使用的证书类型为 RSA。如果你想指明不同类型的证书（如：DSA），在命令行中使用如下参数：

–Dweblogic.nodemanager.certificateType=*type*

其中 type 为 RSA 或 DSA.

为了传递私钥的口令以访问加密的私钥，在命令行使用下述参数：

–Dweblogic.nodemanager.certificatePassword=*pkpassword*

其中 *pkpassword* 为私钥的口令。

用于验证用户身份的证书认证机构或者认证机构链，保存于可信任认证机构文件中。缺省的为 config/demo.crt，你可以通过启动参数使用其他可信任认证机构文件：

–Dweblogic.nodemanager.trustedCerts=*path*

其中 *path* 是保存可信任认证机构文件的位置。

你还可以指定 BEA Home 路径，它是所有 BEA 产品安装的根路径。你可以在命令行参数中设定 BEA Home 路径：

-Dbea.home=*directory*

如果在可信任主机列表文件中使用了 DNS 名而非 IP 地址的话，还需要设定如下的启动参数：

-Dweblogic.nodemanager.reverseDnsEnabled=true

缺省的，DNS 反向搜索是禁用的。

还可以在启动时指定保存可信任主机列表文件的文件名：

-Dweblogic.nodemanager.trustedHosts=*path*

其中 *path* 为保存可信任主机列表文件的位置。缺省的该文件位于/config 目录下。

WebLogic 安全策略文件缺省是 weblogic/lib/weblogic.policy。如果想指明使用其他的安全策略文件，在命令行中添加：

-Djava.security.policy==*policy_file*

其中 *policy_file* 指明 WebLogic 安全策略文件的位置。

缺省的节点管理起不做主机名的 SSL 验证。若想打开此选项，在命令行中添加：

–Dweblogic.nodemanager.sslHostNameVerificationEnabled=true

Classpath 选项

节点管理器也需要使用 WebLogic 使用的 java 类，当启动节点管理器时，下列值必须作为 java 命令行中的-classpath 选项的值：

- /weblogic/lib/weblogic_sp.jar
- /weblogic/lib/weblogic.jar

使用启动脚本启动节点管理器

WebLogic 服务器软件提供了启动节点管理器的启动脚本范例。这些脚本文件位于安装 WebLogic 的/config 目录下。Windows 平台下为 startNodeManager.cmd，而 Unix 平台下为 startNodeManager.sh。这些脚本文件在启动节点管理器时已经设定了环境变量。

远程启动或终止受管服务器

如果你不使用节点管理器的启动脚本，那么需要确认节点管理器在你安装 WebLogic 服务器软件的根目录中。该目录包含了\config 子目录。

如果在受管服务器所在的机器中运行了节点管理器，可以按以下步骤启动受管服务器：

1．启动管理控制台（如果它还没有运行起来）

2．在导航树（位于左窗格）中右点服务器的名字

3．选择 Start this server…

在启动管理服务器时，通常打印到 STDOUT 与 STDERROR 的消息会显示在管理控制台的由窗格中。这些消息同时被写到节点管理器的日志文件中。

终止受管服务器的方式也一样：

1．右键点击左窗格中受管服务器的名字

2．选 Kill this server ...

Kill this server ...选项指示运行受管服务器的机器上的节点管理器终止该 WebLogic 服务器的进程。

注：Kill this server ...选项不能用于终止管理服务器

停止和终止受管服务器的区别

若在管理控制台左侧窗格中右键点击受管服务器名，一个选项是 **Stop this Server...**，该选项并不使用节点管理器去停止受管服务器，若选择了 **Stop this Server...** 管理服务器发送关闭命令请求到所选的受管服务器，在这种情况下，节点管理器不被使用。与 Kill this Server...不同，**Stop this Server...**可用于关闭管理服务器。

因为 **Stop this Server...** 选项使用受管服务器的管理功能来关闭服务器，它只能用于该服务器处于活动状态并响应命令请求时，而 **Kill this Server...** 选项通常用于目标服务器已经挂起或者不再响应管理服务器发送的管理命令请求的情况下。

同样的弹出式菜单还可以让你查看受管服务器产生到 StdOut 和 StdErr 的信息。选择 **Get StdOut for this serverc** 查看 StdOut 上的输出，选择 **Get StdErr for this server** 查看 StdErr 上的输出。

启动和终止域和集群

你可以同样在活动域中启动和终止受管服务器：

1. 在左侧窗格中，右键点击活动域的名字
2. 选择 **Kill this domain...** 或 **Start this domain...**

如果你想从管理控制台中启动整个域，右侧窗格中会显示一系列配置在域中的受管服务器的运行结果链接。

类似的，你也可以选择启动或终止一个选定的集群内的所有受管服务器：

1. 在左侧窗格中，右键点击集群的名字
2. 选择 **Kill this cluster…** 或 **Start this cluster…**

注：你不可以通过节点管理器启动或终止管理服务器。

将节点管理器设置为 Windows 服务

目录 *install_dir/config/mydomain*，包含了将 WebLogic 服务器安装为 Windows 服务以及卸载的脚本。脚本 installNtService.cmd 用于将 WebLogic 服务器安装为 Windows 服务，uninstallNtService.cmd 则用于卸载作为 Windows 服务的 WebLogic 服务器。你可以复制并修改这些脚本用于将节点管理器安装为 Windows 服务以及卸载。

将节点管理器安装为 Windows 服务：

1. 从 *install_dir/config/mydomain* 目录复制 installNtService.cmd 脚本（其中 *install_dir* 为 WebLogic 软件安装的根目录），将其重命名为 installNMNtService.cmd
2. 从 *install_dir/config/mydomain* 目录复制 uninstallNtService.cmd 脚本（其中 *install_dir* 为 WebLogic 软件安装的根目录），将其重命名为 uninstallNMNtService.cmd
3. 修改 installNMNtService.cmd 脚本，使之包含启动节点管理器的命令，并确认启动命令中的启动类从 weblogic.Server 该为 weblogic.nodemanager.NodeManager。关于命令行选项的详细信息，参见“从命令行启动节点管理器”。
4. 给服务重新命名，如 nodemanager
5. 修改 uninstallNMNtService.cmd 脚本，将目标服务名改为 installNMNtService.cmd 脚本中启动节点管理器为 Windows 服务的名字。
6. 确认 installNMNtService.cmd 脚本在 WebLogic 安装的根目录启动，如：c:\bea\wlserver6.1
7. 调用 installNMNtService.cmd 脚本，将节点管理器安装为 Windows 服务。
8. 也可以从菜单启动

   Start→Setting→Control Panel → Administrative Tools → Services

卸载作为 Windows 服务的节点管理器

要卸载作为 Windows 服务的节点管理器，调用 uninstallNMNtService.cmd 脚本文件。

第四章 配置 WebLogic 服务器与集群

本章将介绍以下内容：

服务器与集群配置概述

管理服务器的角色

启动管理控制台

动态配置的原理

规划一个集群配置

服务器的配置任务列表

集群的配置任务列表

服务器与集群配置概述

一个域的 WebLogic 服务器与集群的永久配置保存在一个 XML 配置文件中，你可通过以下途径改变这个文件：

通过管理控制台。管理控制台是用来管理与监控域配置的一个图形用户界面。这是改变或监控域配置的主要方式。

使用 WebLogic 服务器所提供的配置应用编程接口，以编程的方式修改配置属性。

通过行 WebLogic 服务器的命令行工具访问域资源的属性。这主要是为那些希望通过脚本自动管理域资源的用户提供的。

管理服务器的角色

无论你使用上述方式的哪一种，在修改域的配置之前首先要运行管理服务器。

管理服务器是运行管理服务的 WebLogic 服务器。管理服务提供了 WebLogic 服务器功能，同时还提供了管理域所需要的功能。

缺省情况下，WebLogic 服务器的实例会成为管理服务器。当管理服务器启动时，它会从 *WEBLOGIC_HOME* 目录的子目录 \config 中装载配置文件。与该管理服务器关联的每一个域在 \config 目录下都有同名子目录。域的配置文件就存放在同名子目录的 config.xml 文件中。缺省情况下，管理服务器在启动时，首先会在缺省子目录下查找配置文件（config.xml），缺省子目录的名字为 mydomain

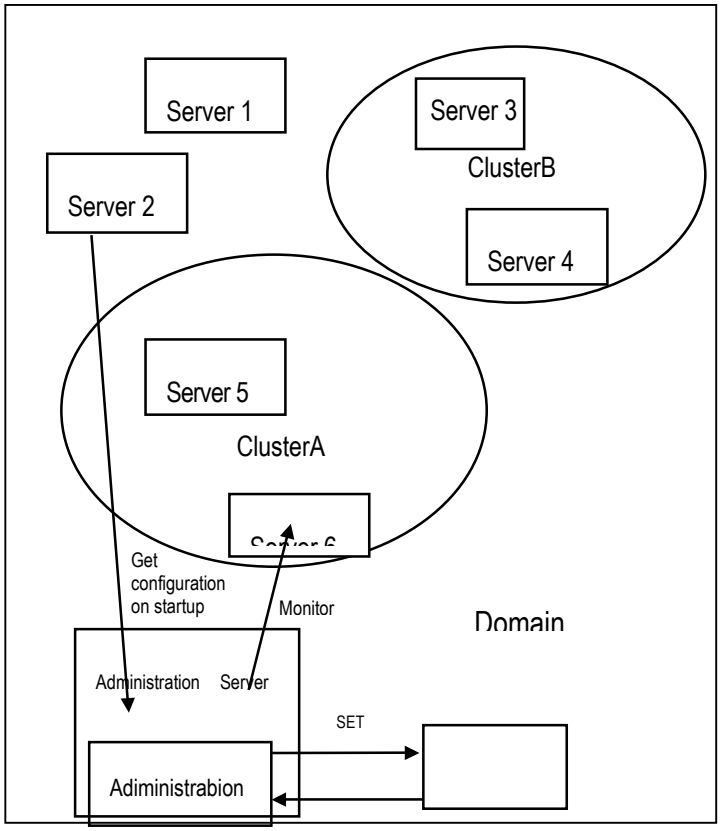
管理服务器启动后，会创建名为 config.xml.booted 的配置文件备份并将它保存在相应的域目录下。如果在服务器的生命周期中，config.xml 文件遭到破坏（当然这种情况极为少见），你就可以将配置恢复到该备份文件中的配置。

一个域可能只有一个 WebLogic 服务器，该服务器就是管理服务器，因为每个域中至少（至多）有一个管理服务器。

图 4-1 是一个典型的生产环境的配置例子，域由一个管理服务器与多个 WebLogic 服务器组成

在启动这个域的 WebLogic 服务器时，首先必须先启动管理服务器，其它服务器启动时，会被命令从管理服务器获得配置信息。这样，管理服务器就成为整个域的配置控制中心。一个域只能有一个活动的管理服务器，只有运行的管理服务器能够修改配置文件。

注：不要采用共享文件系统和在不同机器上安装一个 WebLogic 而运行多个实例。否则所有服务器必须竞争的去读写文件系统（可能产生独立的日志文件），另外一旦共享文件系统出错，就无法启动受管服务器或者集群了。



WebLogic 服务器的配置

启动管理控制台

访问管理服务器的主要途径是管理控制台。打开管理控制台的步骤如下：

1. 输入 URL http://host:port/console

URL 中的 host 是管理服务器所在机器的主机名或 IP 地址。port 是管理服务器监听请求的端口（缺省为 7001

2. 系统提示你输入用户名与口令。输入你的用户名与口令。系统会进行身份验证与权限检查，它依据用户数据库来验证用户号与口令。

如果通过了身份验证并被确认可以使用管理控制台，那么管理控制台将显示为系统管理员分配给你的访问模式：或者是只读模式，或者是读/写模式。

动态配置的工作原理

WebLogic 服务器允许动态（即在运行时）改变域资源的配置属性，大多数情况下，不需要重启 WebLogic 服务器就能使修改生效。属性值的改变会立即反映到当前正在运行的属性值中，也会被作为永久值保存在 XML 配置文件中。

当然也会有例外。例如，如果你改变了 WebLogic 服务器的监听端口。新设置的端口只有等到下次重启时才生效。这种情况下，属性值的改变体现在保存属性值的 XML 文件中，当前的运行时配置值不同于属性文件中的值。

当属性的永久值与运行时的值不同时，管理控制台用图标表示这种情形，指示服务器需要重启才能使该值生效。

管理控制台会对用户修改的属性值进行有效验证。所支持的错误有越界错误与数据类型不匹配错误。在这两种情况下，都会弹出错误信息提示你属性值设置有误。

在管理控制台启动后，如果有其它进程使用了分配给管理服务器的监听端口，那么应该删除这个进程。如果不能删除这个进程的话，那么必须编辑 Config.XML 文件修改管理服务器的监听端口。有关编辑 Config.XML 文件的更多信息，请参见“配置参考”。

集群配置规划

在规划集群配置时，应该牢记以下关于网络环境与集群配置的限制。

1　首先，集群中的 WebLogic 主机必须使用永久的静态 IP 地址。动态 IP 地址分配不能用于集群环境。如果服务器位于防火墙后面，而客户机位于防火墙外面，那么服务器必须有公共的静态 IP 地址，只有这样，客户端才能访问服务器。

2．集群中的所有 WebLogic 服务器必须位于同一个局域网，并且必须是 IP 广播可到达的。

3　集群中的所有 WebLogic 服务器必须使用相同的版本。

配置集群中的服务器，使它们支持所提供的服务。

对于使用了 JDBC 连接的 EJB，所有部署了某 EJB 的服务器必须具有相同的部署与持久化配置。也就是说所有服务器都应该有相同的 JDBC 配置。

所有部署了 servlet 的主机必须维护一组具有相同 ACL 的 servlets

如果客户端应用直接使用 JDBC 连接池，那么你必须为每个 WebLogic 服务器创建相同的连接池（并具有相同的 ACL）。这意味着集群所使用的连接池应该可以在所有的机器上创建。例如，一台运行 WebLogic 的 NT 服务器配置了连接 Microsoft SQL Server 数据库的连接池，那么一个包含非 Windows 机器（即不支持 Microsoft SQL Server 连接的机器）的集群不能使用这个连接池。

其它配置细节可能会因不同的集群成员而不同。例如，一台 Solaris 服务器可以比一台小的 NT 工作站处理更多的登录请求。这种差异是可以接受的。因此，正如这里所给出的例子，对于那些与性能相关的属性，你可以根据每个集群成员的特点来配置不同的值，只要所有成员的服务配置相同即可。因此，集群中的 WebLogic 服务器在所有与 WebLogic 服务、类文件以及外部资源（例如数据库）相关的方面具有相同的配置。

服务器配置任务列表

可以通过管理控制台进行以下服务器配置：

通过管理控制台的 Server 节点配置单独的服务器。可以配置的属性包括名字，监听端口与 IP 地址。

通过管理控制台的 Server 节点克隆一个服务器。克隆的服务器保存了原来服务器的属性值，你可以使用 Server 节点中的 Configuration 配置新服务器的名字。

使用管理控制台的 Server 节点来删除一个服务器。点击要删除的服务器的图标，将弹出一个删除服务器的确认对话框，点击对话框中的 Yes 按钮将删除服务器。

使用管理控制台的 Server 节点查看一个服务器的日志。点击要查看的服务器，点击 Monitoring 标签页。点击 View Server Log 连结，便可以在管理控制台的右窗格查看服务器日志。

使用管理控制台的 Server 节点查看一个服务器的 JNDI 树。点击所要查看的服务器，然后点击 Monitoring 标签页，点击该页面上 View JNDI Tree 连接，该服务器 JNDI 树的信息便显示在管理控制台的右窗格中。

使用管理控制台的 Server 节点查看服务器的执行队列。点击所要查看的服务器，然后点击 Execute Queue 链接，然后查看管理控制台右边窗格里的表格中的内容。

使用管理控制台的 Server 节点查看服务器的执行线程。点击所要查看的服务器，然后点击 Execute Queue 链接，然后查看管理控制台右边窗格里的表格中的内容。

使用管理控制台的 Server 节点查看 server sockets。点击所要查看的服务器,点击 View Sockets 连接，然后查看管理控制台右边窗格里的表格中的内容。

使用管理控制台的 Server 节点查看服务器连接。点击所要查看的服务器，点击 View Connections 连接，然后查看管理控制台右边窗格里的表格中的内容。

使用管理控制台的 Server 节点进行强制垃圾收集。点击要监控的服务器，点击 JVM 标签页，点击页面上的 Force Garbage Collection 连接。将弹出是否要进行垃圾收集的确认对话框。

通过管理控制台的 Server 节点监视服务器的安全。点击要监控的服务器，点击 Monitoring 标签页，点击 Security 标签页。将显示安全信息。

通过管理控制台的 Server 节点查看服务器的版本。点击要查看的服务器，点击 Version 标签页，将显示服务器的版本信息。

通过管理控制台的 Server 节点监控服务器集群。点击要监控的服务器，点击 Cluster 标签页，将显示该服务器的集群数据。

通过管理控制台的 Server 节点来部署 EJB。点击需要部署 EJBs 的服务器，点击需要分发的 EJB 并使用移动控件将它移到被选列中。点击 Apply 来保存你的选择。

通过管理控制台的 Server 节点来监视部署在某一服务器上的所有 EJB。点击需要监视的服务器，点击 Monitor All EJB Deployments 连接来显示 EJB 的部署列表。

通过管理控制台的 Server 节点将 web 应用组件部署在某一服务器上。选择要部署 web 应用的服务器。选择需要部署的 web 应用，然后通过移动控件将它移到被选列中。点击 Apply 来保存你的选择。

通过管理控制台的 Server 节点来监控某一服务器上的所有 web 应用组件。点击 web 应用所

在的服务器，然后点击 Monitor All Web Applications 连接来显示 Web Application 的部署列表。

通过管理控制台的 Server 节点在服务器上部署启动与终止类。点击需要部署启动类的服务器，然后点击需要部署的启动类并将它移到被选列中。点击 Apply 来保存你的选择。使用终止类控件来部署终止类的过程与此相同。

通过管理控制台的 Server 节点为服务器分配 JDBC 连接池。点击 web server 分配表中的一个服务器。在 Available 列中点击一到多个 JDBC 连接池，并通过移动控件将所选择的 JDBC 连接池移到 Chosen 列。点击 Apply 来保存你所做的分配。

通过管理控制台的 Server 节点为一个服务器分配 WLEC 连接池。点击需要分配 WLEC 连接池的服务器。在 Available 列中选择一个或多个要分配的 WLEC 连接池，使用移动控件将所选择的 WLEC 连接池移动到 Chosen 列。

通过管理控制台的 Server 节点监视某一服务器上的所有 WLEC 连接池。选择一个需要监视连接池的服务器。点 Monitor All WLEC Connection Pools on This Server 链接，所有分配给这台服务器的连接池会显示在右窗格中的 WLEC Connection Pools 列表中。

通过管理控制台的 Server 节点为一台服务器分配 XML 注册表。选择要分配 XML 注册表的服务器，从 XML 注册表的下拉列表中选择一个注册表。点 Apply 保存设置。

通过管理控制台的 Server 节点分配邮件会话。选择一个要分配邮件会话的服务器。从 Available 列中选择要分配给服务器的邮件会话，使用移动控件把所选择的移动会话移动到 Chosen 列中。点 Apply 按钮保存设置。

通过管理控制台为服务器分配文件 T3s。选择一个要分配文件 T3 的服务器。从 Available 列中选择要分配给服务器的文件 T3s，使用移动控件把所选择的文件 T3s 移动到 Chosen 列。点 Apply 按钮保存设置。

集群配置列表

通过管理控制台可以进行以下集群配置：

通过管理控制台的 Cluster 节点配置集群服务器。可以配置的属性包括 Cluster Name, Cluster ListenPort 以及集群中的服务器名。

通过管理控制台的 Cluster 节点克隆一个集群。克隆的服务器与原服务器具有相同的属性设置，包含同样的服务器。新集群的名字在 Server 节点中的 Configuration 部分设置。

通过管理控制台的 Cluster 节点监控集群中的服务器。选择需要监控的集群，点 Monitor Server Participating in This Cluster 链接。该集群中的所有服务器显示在右窗格中的服务器表格中。

通过管理控制台的 Cluster 节点为集群分配服务器。选择需要分配服务器的集群，从 Available 列中选择要分配的服务器，使用移动控件把所选的服务器移到 Chosen 列中。点 Apply 按钮保存设置。

通过管理控制台的 Cluster 节点删除集群。选择 Delete 图标，删除一个集群，右窗格中显示一个删除确认对话框，点 Yes 确认你的删除请求。

第五章 监控WebLogic 域

本章主要介绍对 WebLogic 域的监控，包括以下内容：

概述

监控服务器

监控 JDBC 连接池

概述

通过管理控制台，你可以对 WebLogic 域的性能以及运行状况进行监控。在管理控制台中，你可以查看到 WebLogic 资源的状态与统计数字，例如服务器、HTTP、JTA 子系统、JNDI、安全性、CORBA 连接池、EJB、JDBC 与 JMS 等资源。

监控信息显示在管理控制台的右侧窗格中。在左窗格的域层次树上，选择一个容器或子系统或者是容器中的某一实体，相关的监控信息便显示在管理控制台的右窗格中。

管理控制台提供了三种监控信息页面：

某一实体的 Monitoring 标签页（例如 JDBC 连接池实例或服务器的性能）

某一实体类型的数据表（例如 WebLogic 服务器表）

域日志与本地服务器日志视图。有关日志消息的详细内容，请参见“通过日志消息管理 WebLogic 服务器”中的内容。

管理控制台从管理服务器获得有关域资源的信息。管理服务器包含了 Management Beans(MBeans) Mbeans 基于 Sun 的 Java 管理扩展标准（JMX），它提供了对域资源的管理访问策略。

管理服务器包含两种 Mbeans configuration Mbeans 与 run-time Mbeans configuration Mbeans 用于域的配置，run-time Mbeans 则提供了诸如 JVM 内存资源使用状况以及 WebLogic 服务器状态等资源的信息快照。当域中的某一资源实例化时（例如 Web 应用），相应地就会创建一个 Mbeans 实例来收集该资源的信息。

在通过管理控制台访问某一资源的 Monitoring 页面时，管理服务器执行 get 操作来获得该资源的当前属性值。

以下部分的内容描述了部分用来管理 WebLogic 域的 Monitoring 页面。我们以这些页面为例简要说明管理控制台所提供的监控功能。

监控服务器

服务器列表和每一个服务器上的 Monitoring 标签页允许你去监控 WebLogic 服务器。服务器列表提供了域中所有服务器的状态统计。如果只有服务器日志消息的一个子集发送给域日志，那么访问服务器本地日志对于排除问题和事件研究大有益处。

关于日志文件和日志子系统的更多信息，参见“使用日志消息管理 WebLogic 服务器”。

你可以通过服务器上的 Monitoring 标签页来监控 WebLogic 服务器上的数据，而 Logging 标签页可以访问服务器的本地日志（即存放于运行 WebLogic 服务器的机器上的日志）。

Monitoring → General 标签页显示了服务器的当前状态，活动执行队列列表，活动的套接字列表以及连接信息列表等。活动执行队列提供了执行效率的信息，如等待时间最长的请求响应、队列吞吐量信息等。

执行效率

Monitoring → Performance 标签页以图表的方式实时的方式显示了 JVM 堆内存的使用情况，请求的吞吐量信息，请求队列列表。该标签页还允许你对堆内存执行强制垃圾回收。

Java 堆内存用于保存 java 对象（活动的或非活动），通常你不需要手工执行垃圾回收，因为 JVM 会自动进行这项工作，而当 JVM 内存开始不够分配时，它停止其他所有运行，通过垃圾回收算法释放那些 java 应用程序中不再使用的空间。

另外，开发人员在应用调试的过程中，偶尔也需要手动执行垃圾回收，手动进行垃圾回收是很有用处的，例如：在测试快速消耗 JVM 内存的内存泄露情况时。

服务器安全性

Monitoring → Security 标签页可以提供非法登录尝试的统计，以及锁定用户和解锁等功能。

JMS

Monitoring → JMS 标签页提供了 JMS 服务器和连接的统计信息，同时也提供了对活动的 JMS 服务器和连接的链接列表，通过它可以监控诸如：总的并发会话等属性。

JTA

Monitoring → JTA 标签页提供了 java 事务管理子系统的统计信息，如：所有交易事务总数信息和回滚交易数，同时它也以资源和名字的形势提供了事务资源列表和运行中的事务列表。

监控 JDBC 连接池

通过管理控制台，你可以对 Java 数据库连接（JDBC）子系统实行监控。在 JDBC 连接池的 Monitoring 标签页面上，你可以了解到有关连接池中的实例的统计信息。正如管理控制台的其它实体表，你可以定制该统计信息表，选择需要显示的属性。

许多属性提供了有关管理客户端数据库请求的重要信息。

Waiters Hight 字段指明了最多有多少客户等待数据库连接。Waiters 字段告诉你当前有多少客户正在等待连接。Connections Hight 字段给出最大的并发连接数。Wait Seconds Hight 字段显示了客户等待数据库连接的最长时间。通过这些属性，你可以判断当前的配置在响应用户请求方面是否有效。

如果 Connections High 字段的值接近了 Maximum Capacity 字段的值（该值在 Configuration Connection 标签页中设置），那么有必要考虑增加 Maximum Capacity 字段的值（即最大的并发连接数）。如果 Waiters Hight 字段的值显示用户必须经过长时间等待才能获得数据库连接，那么应该增大连接池的容量。

Shrink Period 字段用来指定 JDBC 子系统在连接池从最大值开始进行缩减时所要等待的时间。当 JDBC 子系统要减小连接池的大小时，那么数据库连接将被释放。因为创建数据库连接非常消耗资源和时间，因此如果密集的客户端请求阵歇性地出现，那么比较短的收缩周期将意味数据库连接要不断地被重新创建，从而导致性能的下降。

第六章 用日志消息管理WebLogic 服务器

本章将介绍以下内容：

日志子系统概述

本地服务器日志文件

消息属性

消息目录

消息的严重级别

浏览日志文件

创建域日志的过滤器

日志子系统概述

日志消息是管理系统的一个非常有用的工具。通过日志可以发现问题，跟踪错误来源，及时了解系统性能。WebLogic 服务器产生的日志消息被保存在两个地方：

WebLogic 服务器组件子系统产生的消息保存在本地文件中，即该文件位于运行服务器的机器上。如果一台机器上同时运行了若干个 WebLogic 服务器，那么每一个服务器都有自己的日志文件。部署在 WebLogic 服务器上的应用可以将它所产生的消息保存在服务器的本地日志文件中。

此外，一部分本地日志消息还被存储在由管理服务器维护的域日志文件中。

内置在 WebLogic 服务器中的 JMX Java 管理扩展）工具，会把 WebLogic 服务器产生的日志消息传送到管理服务器。如果本地 WebLogic 服务器主动将消息发送到其它实体，用 JMX 的术语来说，这就是“布告”。

在 WebLogic 服务器启动时，管理控制台的消息处理器会向该服务器注册以便接收来自该服务器的日志消息。在注册过程中，本地服务器将使用一个可以被用户修改的过滤器选择要转发到管理服务器的日志消息。这些消息被收集在域日志中。

一般而言，只有最重要的日志消息（由“Message severity”决定）才会从本地服务器转发到域日志中。域日志提供了域的一个总体视图，同时又能使你聚焦在那些最为关键的消息上。

你可以通过管理控制台来更改日志消息过滤器，用以接收从本地服务器发送来的日志消息的子集，这一过程是动态的，即不需要重启本地服务器，更新即可生效。（参见“创建日志过滤器”中的内容）。

开发人员可以用定制的消息处理器向 WebLogic 服务器注册并通过 JMX 布告接收日志消息。

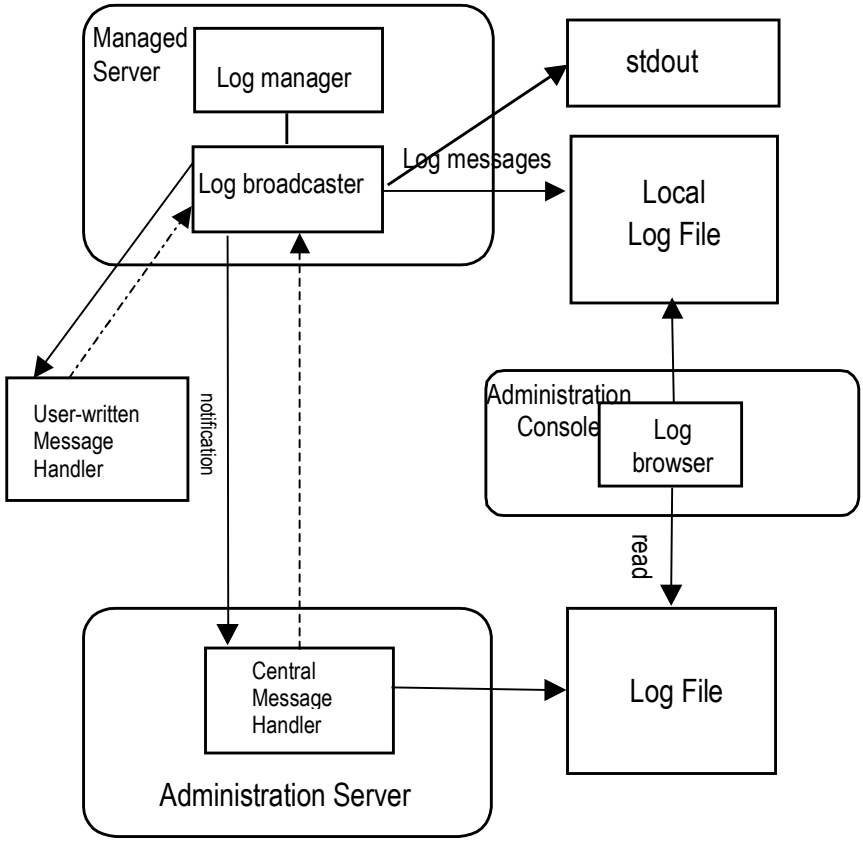


图 6-1 WebLogic Server 日志子系统

本地服务器的日志文件

在 6.0 以前版本的 WebLogic 服务器，在日志文件达到最大长度时会创建一个新的日志文件。这种自动创建日志文件的机制被称为日志回旋（Log Rotation）。在此版本中，你可以选择日志回旋的方式：（1）基于时间，（2）基于文件大小。在管理控制台中按以下步骤配置日志回旋：

1 在管理控制台的左窗格，选择一个服务器。

2 在右窗格，选择 Configuration->Logging

3 在 Rotation Type 字段中，选择时间或长度。

如果该字段的值为 none，那么日志不进行回旋。如果选择基于时间的日志回旋，那么新日志文件按指定的时间间隔（由 File Time Span 指定）被创建。

缺省情况下，本地服务器的日志文件名为 servername.log（其中 servername 指服务器的名字）。日志文件位于 WebLogic 服务器所在的目录。日志文件的名字可以在 Configuration->Logging 页面中设置。

File Count 字段的值指定回旋日志文件的最大数量。如果日志文件的数量数达到这个数目，那么每次发生日志文件回旋时，最老的日志文件将被删除。日志文件按照创建的先后被命名为 filenamennnnn，其中 filename 是在 configuratin->Logging 页面中设置的日志文件名。例如：

webLogic.log00007

本地服务器日志包含所有被记录的日志消息。

在配置本地服务器日志时,你可以指定哪些日志消息将被输出到标准输出中。通过指定所要记录的日志的最低严重性级别，可以把一些不太严重的消息排除在外。也可以设置是否将调试消息输出到标准输出。

注： 请不要手工的修改服务器日志文件。基于时间的日志回旋依据文件的时间戳。如果你修改了文件，将改变文件的时间戳，会造成日志回旋混乱。如果你手工编辑文件，会锁定文件并阻碍服务器更新文件。

客户端日志

使用 WebLogic 日志功能的 Java 客户端也会产生日志消息，但是这些客户端所产生的消息不能被转发到域日志中。要配置客户端的日志属性，可以在命令行中使用以下参数：

-Dweblogic.log.*attribute*=*value*

其中，attribute 可以是任何 LogMBean 属性。缺省情况下，客户端的日志消息不会记录在日志文件中，而是输出到标准输出。你可以在命令行中打开此选项，并指定日志文件名：

-Dweblogic.log.*FileName* = *logfilename*

其中 *logfilename* 为指定的客户端日志文件名

下列命令也可以用于客户端日志：

-Dweblogic.StdoutEnabled=*boolean*

-Dweblogic.StdoutDebugEnabled=*boolean*

-Dweblogic.StdoutSeverityLevel = [64 | 32 | 16 | 8 | 4 | 2 | 1]

其中 *boolean* 可以是 true 或 false

日志文件的格式

在日志文件中，每条消息的第一行都以####开始，紧跟其后是消息头。消息头说明了消息的运行时上下文。每个消息属性都包含在尖括号对中。

如果是异常消息，那么消息体后面是异常的堆栈信息记录。如果消息不在事务上下文中，属性 Transaction ID 的尖括号对仍然会有，只是没有事务号而已。

以下是一个日志消息的例子：

####<Jun 2, 2000 10:23:02 AM PDT> <Info> <SSL> <bigbox> <myServer>

<SSLListenThread> <harry> <> <004500> <Using exportable strength SSL>

这个例子包含以下消息属性：Timestamp, Severity, Subsystem, Machine Name, ThreadID, UserID, Transaction ID, Message ID, 以及Message Text

注：客户端记录的日志消息没有 Server Name 与 Thread ID 属性

注：日志文件中的字符集编码使用的是系统的编码。

消息属性

日志文件中的每条消息都定义了下表所列出的属性。通过消息号，消息还可以关联 Message Catalog 的其它属性 (例如，Probable Cause 与 Recommended Action)

属性	描述
时间戳	消息产生的日期与时间，格式因地区设置而异。
严重级别	指消息事件的重要性或严重程度见“消息的严重级别”
子系统	该属性指出了该消息是 WebLogic 服务器的哪个子系统产生的。例如:EJB, RMI 与 JMS
服务器名 机器名 线程号 事务号	这四个属性定义了该消息的来源。如果消息位于事务上下文中，那么就具有事务号属性。注意：Java 客户端所产生的日志消息没有服务器名以及线程号属性，同时也不保存在客户端日志中。
用户号	在产生该消息时，安全上下文中的用户
消息号	唯一性的六位标识符。小于 499999 的消息号保留给 WebLogic 服务器系统消息
消息文字	对于 WebLogic 服务器消息，该属性的内容是由系统消息目录所定义的短描述（见“消息目录”），对于其它消息，该属性的内容由程序开发者指定。

消息目录

除了保存在日志消息中的那些信息外，WebLogic 服务器系统组件（或者是用户编写的代码）产生的消息，包括了其它预先定义的消息、封装的消息等，这些信息被保存在消息目录中。下表描述了保存在消息目录中的属性：

属性	描述
消息体	对消息产生时的环境的简短的文字性描述。与消息的消息文字属性相同
消息细节	对消息产生时的环境的更详尽的描述
可能原因	解释记录该消息的原因。以及产生该消息的可能原因
建议	管理员对解决或避免引起该消息的问题建议

你可以在管理控制台的日志视图查看上述属性

消息的严重级别

WebLogic 服务器消息具有一个称作“Severity”的属性。该属性反映了消息所报告的事件或系统状态的重要性或者可能对用户的潜在冲击.

下表描述了被定义的各类严重性。下表按严重程度列出了各种严重情形，其中 Energency 的严重级别最高

严重级别	缺省情况下是否发送到域日志	含义
报告 (Informational)	否	用于报告普通操作
警告 (Warning)	否	发生了可疑的操作或配置，但对普通操作没有影响
错误 (Error)	是	发生了用户操作。系统或应用不需要中断服务就可以处理该错误
注意 (Notice)	是	发生了可疑的错误或配置，可能不会影响服务器的正常操作
严重 (Critical)	是	发生了系统或服务错误。系统可以恢复但是可能会造成服务瞬间失去或永久性的服务性能降低
警报 (Alert)	是	某个服务不能使用，系统的其它部分还能正常工作。不可能进行自动恢复；管理员需要立即采取措施解决问题
危急 (Emergency)	是	服务器不能使用，该严重级别表明了发生服务器系统失败，情况十分危机。

消息调试

严重级别为调试的消息是一个特殊的情形。调试消息不会发送到域日志。调试消息包含了有关应用与服务器的详细信息。这些消息只有在应用是以调试模式运行时才会产生。

浏览日志文件

管理控制台提供了以下日志查看功能：

查看任一服务器的本地日志文件

查看域日志文件

无论是查看域日志还是本地服务器日志，你都可以

按照产生时间、用户号、子系统、消息严重级别或者是消息的短描述选择所要查看的消息

查看被记录的消息，或对日志消息进行查找

选择要显示在管理控制台的消息属性及其排列顺序

查看日志

你可以从管理控制台访问域日志与本地服务器日志。有关这些任务的具体操作参见控制台的在线帮助：

查看域日志（viewing the Domain Log

查看本地服务器日志（Viewing the Local Server Log

创建域日志过滤器

缺省情况下，被 WebLogic 服务器转发到域日志的日志消息是本地消息日志的子集。过滤器决

定了要把哪些日志消息发送到域日志，你可以对过滤器进行配置，这样就可以根据消息的严重级别、产生消息的子系统或者是用户号来选择要转发的日志消息。（调试消息属特殊情况，它不能被转发到域日志）你可以创建一个过滤器也可以更改域日志过滤器表中的过滤器。从 Domain Monitoring 标签页可以访问域日志过滤器表。有关创建域日志的详细内容参见管理控制台的在线帮助。

第七章 分发应用

本章将介绍如何在 WebLogic 服务器上部署应用程序以及应用组件，内容如下

所支持的部署格式

通过管理控制台分发应用

自动部署

所支持的部署格式

J2EE 应用可以以企业应用包（Enterprise Application Archive，简称为 EAR 的形式或者是展开目录格式的形式部署到 WebLogic 服务器上。

如果以展开格式的形式部署 J2EE 应用，那么我们建议你在此格式中只包含 Web 应用组件。如果是以包的形式部署 J2EE 应用，我们建议应用的所有组件都应打在该包文件中。

一个组件可以被打包在 EJB 包（JAR）文件中，也可以在 Web 应用包（Web Application Archive，简称为 WAR）文件中，或者是资源适配器包（Resource Adaptor Archive，简称为 RAR）文件中。

有关 Web 应用的更多信息，请参见“配置 WebLogic 服务器 Web 组件”中的内容。

有关资源适配器组件的信息，可以参见“管理 WebLogic J2EE 连接器构架”中的内容。

通过管理控制台部署应用

你可以通过管理控制台安装或部署应用或应用的组件（例如 EJB JAR 文件）并且将应用组件的实例部署到目标 WebLogic 服务器上。步骤如下：

步骤 1：配置与部署应用

1．选择 Deployments->Application 调用 applications 表格.

2．点击 Configure a new Application 链接调用 Create a new Application 页面。

3．填写以下应用的配置条目

应用的名字

应用（EAR 文件）所在的路径

应用是否要部署到 WebLogic 服务器上了

4．点 Create 按钮创建一个新的条目

当你用管理控制台分发应用（或应用组件）时，管理服务器会为被分发的应用或应用组件在域配置文件中（/config/domain_name/config.xml）加上一个相应的条目。同时，管理服务器还会创建用以配置及监控该应用或应用组件的 JMX Management Beans MBeans

步骤 2：分发应用组件

可以在 WebLogic 服务器中部署以下三种类型的应用组件：Web 应用组件，EJBs 与资源连接器组件。

注意：如果把应用组件（如 EJBs 或 WAR 文件）部署到集群中的受管服务器上，那么必须保证集群中的所有成员服务器都部署了同样的应用组件。为此，你可以选择集群作为部署的目标。

部署 Web 应用组件

以下是将 Web 应用部署到受管服务器上的步骤：

1. 选择 Deployments → Web Applications 调用 Web Applications 表
2. 点击 Configure a new Web Application 链接调用 Create a new WebApp Component 配置页面
3. 填写以下相应字段：

   组件的配置条目的名字

   指向该组件的统一资源标识符（URI

   WAR 文件的路径，或者是展开格式的 Web 应用所在目录的路径

   选择部署顺序。该顺序指定了服务器启动时 Web 应用的部署顺序。（见分发顺序）

   指出应用是否已经被部署了
4. 点击 Create 按钮创建一个新的组件条目
5. 选择组件的部署目标是受管服务器还是集群。点 Targets->Servers 将组件部署到目标服务器上。点 Targets->Clusters 将组件部署到目标集群上。
6 Available 字段列出了受管服务器（如果选择 Targets->Clusters 则列出集群）。使用箭头按钮将它们移到 Chosen 字段，这样便选择了要部署该 Web 组件的目标服务器。点 Apply 使你的设置生效。

有关配置 Web 应用的更多信息，请参见“配置 WebLogic 服务器的 Web 组件”中的内容。

部署 EJB 组件

以下是把 EJBs 部署到受管服务器上的步骤：

1. 选择 Deployments → EJB 调用 EJB Deployment 表格
2. 点击 Configure a new EJB 链接调用 Create a new EJB Component 页面
3. 填写以下字段：

   该组件的配置条目的名字

   指向该组件的统一资源标识符（Uniform Resource Identifier，简称为 URI

   JAR 文件的路径

   选择部署顺序。它决定了服务器启动时，该 EJB 被部署的顺序。

   指出是否已经部署了该组件
4. 点 Create 创建一个新的组件条目
5. 选择组件的部署目标是受管服务器还是集群。点 Targets->Servers 将组件部署到目标服务器上。点 Targets->Clusters 将组件部署到目标集群上。
6 Available 列中列出了受管服务器（如果选择 Targets->Clusters 则列出集群）。使用箭头按钮将

它们移到 Chosen 列中，这样便选择了要部署该 EJB 的目标服务器。点 Apply 使你的设置生效。

部署资源适配器组件

以下是把资源适配器组件部署到受管服务器上的步骤：

1．选择 Deployments → Connectors 调用 Resource Connectors 表格

2．点击 Configure a new Connector Component 链接调用 Create a new Connector Component 页面

3．填写以下字段：

该组件的配置条目的名字

指向该组件的统一资源标识符（Uniform Resource Identifier，简称为 URI

RAR 文件的路径

选择部署顺序。它决定了服务器启动时，该资源连接器被部署的顺序。（见“部署顺序”中的内容）

指出是否已经部署了该组件

4．点 Create 创建一个新的组件条目

5. 选择组件的部署目标是受管服务器还是集群。点 Targets→Servers 将组件部署到目标服务器上。点 Targets→Clusters 将组件部署到目标集群上。

6 Available 列中列出了受管服务器（如果选择 Targets->Clusters 则列出集群）。使用箭头按钮将它们移到 Chosen 列中，这样便选择了要部署该 Web 应用的目标服务器。点 Apply 使你的设置生效。

有关资源连接器的更多信息，可以参考“管理 WebLogic J2EE 连接器构架”中的内容。

如果一个应用或应用组件（如一个 EAR 或 WAR 文件，或 EJB JAR 文件）被部署到一个特定的 WebLogic 服务器上，那么文件首先被复制到

/config/*domain_name*/applications/wlnotdelete 目录中，管理控制台将调用一个文件部署 servlet 将文件复制到目标服务器上。

部署顺序

对于同一类型的组件，例如 EJBs，你可以指定 WebLogic 服务器在启动时部署这些组件的顺序。你在部署组件时，为 Deployment Order 字段所指定的整数是该组件相对于其它同类型组件的优先级，例如不同的 EJB 之间的部署顺序。部署顺序为 0 的组件是同类型组件中最先部署的组件。

但是，不同组件类型之间的部署顺序不受用户所定义的同类型部署顺序所影响。WebLogic 在启动时，按以下类级别的方式，顺序部署各类型的组件：

1 JDBC 连接池

2 JDBC 多池

3 JDBC 数据源

4 JDBC Tx 数据源

5 JMS 连接工厂

6 JMS 服务器

7. 连接器组件

8 EJB 组件

9 Web 应用组件

自动部署

自动部署（Auto-deployment）能够快速地在管理服务器上部署应用。但我们只建议你在开发环境中使用这种部署方式来测试应用。不建议你在生产环境或受管服务器上使用使用自动部署。

如果目标 WebLogic 服务器域启用了自动部署，当应用被复制到 WebLogic 管理服务器的 */config/domain_name/applications* 目录下时，管理服务器会自动检测到新应用并自动部署该应用（如果管理服务器处于运行状态）。（子目录 *domain_name* 是启动管理服务器时所使用的 WebLogic 服务器域。 ）如果在你把应用复制到 /applications 目录时，WebLogic 服务器没有被运行，那么当 WebLogic 服务器稀下次启动时将部署这个应用。

如果对一个允许自动部署的应用，在管理控制台中对其配置进行任何修改，这些配置都不会永久保存，即配置的任何修改不会保存于活动域中的 config.xml 文件中。如果对自动部署的应用的配置做了任何修改，下次再重新启动管理服务器时，这些修改就不存在了。

启用与禁用自动部署

缺省情况下，自动部署是启用的。

要确定是否启用了自动部署选项，可以进入管理控制台的域应用设置页面（domain_name → Configuration → Applications），在这个页面上可以设置自动部署是否启用以及设置 WebLogic 服务器检查新应用的时间间隔。缺省情况下，如果启用了自动部署，那么管理服务器每 3 秒钟，会检查 \applications 目录下文件的变化。

展开目录格式的应用自动部署

自动部署的应用或应用组件可以采用展开目录格式或者 EAR 文件、WAR 文件以及 JAR 文件等格式。

以下是展开式应用的自动部署的步骤：

1. 展开式应用的目录名应该与应用的上下文路径相同。

2. 把应用复制到 */config/domain_name/applications* 子目录下，其中。*Domain_name* 是要部署应用的目标域。如果启用了自动部署，那么该应用被自动部署。

卸载或重新部署被自动部署的应用

被自动部署的应用或应用组件可以在服务器运行的情况下动态地重新部署。因此可以不需要停止并重启 WebLogic 管理服务器，就能对应用或应用组件进行更新。如果要动态地重新部署

一个应用或应用组件，只需要用更新后的文件覆盖/applications 目录下的相应文件。

该功能对于开发人员很有用。开发人员只需要将更新后的文件复制到/applications 目录下，服务器就会进行相应的更新。

展开式应用的自动重分发

你可以动态地重新部署展开式应用或应用组件。如果所部署的应用是展开格式，那么管理服务器会周期性地检查展开式应用目录中的 REDEPLOY 文件。如果该文件的时间戳改变了，那么管理服务器会自动地重新部署这个应用。

如果要更新展开式应用目录中的文件，按以下步骤：

1. 第一次部署展开式应用时，在该应用所在的目录下创建一个名字为 *REDEPLOY* 的空文件
2. 用更新后的应用文件覆盖目录下的相应文件。
3. 复制完文件后，改变这个目录下的 REDEPLOY 文件的时间戳

当管理服务器发现该文件的时间戳改变了，它会自动部署展开式应用目录下的内容。

第八章 配置WebLogic 服务器的Web 组件

本章将介绍 Web 组件的配置，主要内容如下：

概述

HTTP 参数

配置监听端口

Web 应用

配置虚拟主机

WebLogic 服务器如何解析 HTTP 请求

设置 HTTP 访问日志

防止通过 POST 方式的“拒绝服务攻击”

设置 WebLogic 服务器的 HTTP 隧道

用本地 I/O 提供静态文件服务

概述

除了可以部署动态的、基于 Java 的分布式应用外，WebLogic 服务器还是一个具有完整功能的 Web 服务器，它可以处理高流量的 Web 站点，并提供象 servlets 以及 JSP 等一样，提供 HTML 文件、图像文件等静态页面的访问。WebLogic 服务器支持 HTTP 1.1 标准。

HTTP 参数

你可以通过管理控制台配置每一个 WebLogic 服务器实例（或每个虚拟主机）的以下 HTTP 参数。

属性	描述	值范围	缺省值
Default Server Name	当 WebLogic 服务器重新定向一个请求，在 HTTP 响应请求头中，以 String 格式将 Default Server Name 设置为主机名。 当使用放火墙或者负载均衡器，以及将原始的请求通过流览器以相同主机名重定向时使用	String	Null
Enable Keepalives	设置是否允许 HTTP keep-alive	Boolean True=允许 False= 不允许	selected
Send Server Header	若设为 False，服务器名并不在 HTTP	Boolean	True

	响应中发送。在只有很少的头资源的无线应用中使用。	True=允许 False= 不允许	
Duration (在 Virtual Host 面板中标记的 Keep Alive Secs)	WebLogic 关闭不活动的 HTTP 连接之前所等待的时间（以秒计算）	Integer	30
HTTPS Duration (在 Virtual Host 面板中标记的 HTTPS Keep Alive Secs)	WebLogic 关闭不活动的 HTTPS 连接之前所等待的时间（以秒计算）	Integer	60
WAP Enabled	当选中，SessionID 中不再包含 JVM 的信息。 当使用 WAP 设备时，它对 URL 有 28 位的字符长度限制。 选择 WAP Enabled，会影响会话在集群中的复制	Enabled Disabled	not Enabled
Post Timeout Secs	WebLogic 服务器在相邻两次接收 HTTP POST 大量数据之间的等待时间，该参数被用来防止采用 POST 方法使服务器超载的“拒绝服务攻击”。	Integer	0
Max Post Time	限制了 WebLogic 服务器用于接收 POST 大量数据的时间	Integer	0
Max Post Size	该参数限制了WebLogic服务器对每个请求所能接收的 POST 数据量（以字节为单位）。	Integer	0
External DNS Address	如果系统中包含了用于地址翻译的放火墙，它位于 WebLogic 集群和 web 服务器的前端插件（如 Netscape 插件）之间，该属性设定为插件与该 web 服务器会话的地址。		

配置监听端口

你可以指定 WebLogic 服务器监听 HTTP 请求的端口。尽管可以设置任何有效的监听端口，但如果把 WebLogic 服务器的监听端口设为 80，那么当你访问 Web 应用的资源时，你就可以省略 HTTP 请求中的端口号。例如，当你使用 80 端口作为监听端口时，你就可以采用以下形式的 URI

http://hostname/myfile.html 而不是

http://hostname:portnumber/myfile.html

普通请求与安全（使用 SSL）请求分别使用不同的监听端口。你可以通过管理控制台的 Servers 节点的 Configuration/Gereral 标签页定义普通的监听端口，在 Configuration/SSL 标签页中定义 SSL 监听端口。

Web 应用

HTTP 以及 Web Services 的部署遵照 Sun MicroSystems 的 Servlet2.2 标准。该标准定义 *Web 应用* 作为各种基于 Web 应用的组件组合在一起的标准。这些组件可以是 JSP 页面，HTTP servlets 以及诸如 HTML 页面与图象文件等静态资源。Web 应用可以访问外部资源，例如 EJBs 与 JSP 标记库。一个服务器可以存放任意多个 Web 应用。当你从 Web 应用请求资源时，一般都要在请求的 URI 中包含该应用的名字。

详细内容，参考“组装和配置 Web 应用”，位于：

http://e-docs.bea.com/wls/docs61/webapp/index.html

Web 应用与集群

Web 应用可以部署到 WebLogic 服务器组成的集群中。当用户请求 Web 应用的资源时，那么请求会被引导到集群中的一个服务器上。如果应用使用了会话对象，那么会话应该复制到集群中的所有节点上。有多种方式复制会话。

详细内容，参见“使用 WebLogic 服务器集群”，位于：

http://e-docs.bea.com/wls/docs60/cluster/index.html

指定缺省的 Web 应用

域中的每个服务器或虚拟主机都有一个特殊的 Web 应用 — 缺省 Web 应用。任何不能被其它 Web 应用所解析的请求将由缺省应用来处理。与一般 Web 应用不同的是，指向缺省 Web 应用的 URI 不需要包含应用的名字。部署在服务器或虚拟主机上的任何 Web 应用都可以声明为缺省 Web 应用。（稍后，我们将讨论“部署 Web 应用”）。有关虚拟主机的更多信息，请参见本章“配置虚拟主机”中的内容。

WebLogic 服务器软件中的缺省域和 examples 域有配置有一个缺省的 Web 应用，在这些域中，缺省应用名为 *DefaultWebApp_servernam*，,并且位于每个域的/applications 目录下。

如果没有正确部署所声明的缺省 Web 应用，那么当请求缺省 Web 应用的资源时，用户将接收到 HTTP 400 错误消息，同时日志中将记录这一错误消息。

例如，如果有一个名字为 shopping 的 Web 应用，那么应该用以下 URI 访问该 Web 应用的 cart.jsp 页面：

http://host:port/shopping/cart.jsp

但是，如果你将该 Web 应用声明为缺省 Web 应用，那么你就可以用以下 URI 访问 cart.jsp

http://host:port/cart.jsp

（其中 host 是运行 WebLogic 服务器的机器的主机名，port 是 WebLogic 服务器监听请求的端口）。

以下是用管理控制台为服务器或虚拟主机指定缺省 Web 应用的步骤：

1．在左窗格中，点击 Web Appliction 节点

2．选择你的 Web 应用

3．在右窗格中，点 Targets 标签页

4．选择 Servers 标签页，将服务器（或虚拟主机）移到 Chosen 列。 （选择 Clusters 标签页面，将集群移动到 Chosen 列中，可以使集群中所有的服务器作为设置的目标）。

5．点击 Apply 按钮

6．点击左窗格中的 Servers（或虚拟主机）节点使之扩展

7．选择合适的服务器或虚拟主机

8．点击右窗格中的 General 标签页面。

9．选择 HTTP 标签页面

10．从 Default Web Application 下拉列表中选择一个 Web 应用

11．点击 Apply 按钮

12．如果要为多个受管服务器指定缺省 Web 应用，那么对每个受管服务器重复上述步骤

配置虚拟主机

通过配置虚拟主机，你可以定义服务器或集群所响应的主机名。虚拟主机就是通过 DNS 将一个 WebLogic 服务器或集群的 IP 地址映射到一个或多个主机名并且指定用哪个虚拟主机来服务哪个 Web 应用。如果在集群中使用虚拟主机，那么负载平衡能够以最有效的方式使用硬件系统，即使是某一 DNS 主机名比其它 DNS 主机名处理更多的请求。

例如，你可以指定一个名字为 books 的 Web 应用响应对虚拟主机名 www.books.com 的请求，这些请求将由 WebLogic 服务器 A、B、C 来处理，而一个名字为 cars 的 Web 应用将响应对虚拟主机名 www.autos.com 的请求，这些请求将由 WebLogic 服务器 D、E 来处理。你可以根据应用以及 Web 服务器的需要来组合使用虚拟主机、WebLogic 服务器、集群与 Web 应用。

每个虚拟主机都可以定义自己的 HTTP 参数与 HTTP 访问日志。为虚拟主机设置的这些参数会覆盖服务器的这些参相应数设置。你可以指定任意数量的虚拟主机。

当把服务器或集群作为虚拟主机的目标时，虚拟主机被激活。以集群为目标的虚拟主机会应用于集群中的所有服务器。

虚拟主机与缺省 Web 应用

每个虚拟主机都可以指定自己的缺省 Web 应用。一个虚拟主机的缺省应用将处理所有不能为该虚拟主机的其它 Web 应用所解析的请求。

与其它 Web 应用不同的是，在访问缺省应用的资源时，URI 不需要包含缺省应用的名字（即上下文路径）。

例如，有一个 www.mystore.com 虚拟主机名，该虚拟主机名以 WebLogic 服务器为目标。如果你在该服务器上部署了一个名为 shopping 的 web 应用，那么你可以要以下 URI 访问该应用的 cart.jsp 页面：

http://www.mystore.com/shopping/cart.jsp

如果，shopping 应用被声明为虚拟主机 www.mystore.com 的缺省 Web 应用，那么访问 cart.jsp

页面的 URI 是：

http://www.mystore.com/cart.jsp

详细信息，请参见本章的“WebLogic 服务器如何解析 HTTP 请求”

设置虚拟主机

可以通过管理控制台来定义虚拟主机：

1．创建一个新的虚拟主机

a.在左窗格中点击 Services 节点，该节点被展开并显示一组服务。

b.点击 Virtural Host 节点。如果定义了虚拟主机，那么该节点将被展开并显示所有的虚拟主机

c.在右窗格中点击 Create a New Virtual Host

d.输入新建虚拟主机的名字

e.输入虚拟主机名，一行一个。只有那些与虚拟主机名匹配的请求才会被虚拟主机的目标 WebLogic 服务器或 WebLogic 集群处理。

f.（可选步骤）为虚拟主机分配一个缺省 Web 应用。

g.点击 Create 按钮

2．定义日志与 HTTP 参数

a.（可选步骤）点击 Logging 标签页，填写 HTTP 访问日志属性（详细内容，请参见本章“设置 HTTP 访问日志”中的内容）

b.选择 HTTP 标签页，填写 HTTP 参数

3．定义虚拟主机的目标服务器

a.选择 Targets 标签页

b.选择 Servers 标签页，将列出可用服务器列表

c.选择 Available 列中的一个服务器并用右箭头按钮将该服务器移到 Chosen 列中。

4．定义虚拟主机的目标集群（可选步骤）。在此之前先要定义一个 WebLogic 集群。详细内容，参见以下页面的“使用 WebLogic 服务器集群”部分：

http://e-docs.bea.com/wls/docs60/cluster/index.html.

a.选择 Targets 标签页

b.选择 Clusters 标签页，该标签页上显示了一组可用的服务器集群。

c.选择 Available 列中的一个集群，用右箭头按钮将该服务器移到 Chosen 列中。集群中的所有服务器都将成为该虚拟主机的目标。

5．将 Web 应用部署到虚拟主机上。

a.点击左窗格中的 Web Applications 节点

b.选择要部署到虚拟主机上的 Web 应用

c.选择右窗格中的 Targets 标签页面

d.选择 Virtual Hosts 标签页面

e.选择 Available 列中的虚拟主机并用右箭头按钮将它移动到 Chosen 列中。

WebLogic 服务器如何解析 HTTP 请求

当 WebLogic 服务器接到一个 HTTP 请求时，它对 URL 的各个部分进行解析，然后根据解析所得到的信息来确定由哪个 Web 应用或服务器处理该请求。以下的例子说明了 WebLogic 服务器如何解析对 Web 应用、虚拟主机、servlets、JSP 以及静态页面的请求及响应。

注意：如果 Web 应用包含在企业应用中，那么 Web 应用可以有另一个名字，可以用这个名字可解析对 Web 应用的请求。有关这方面的更多内容，请参见“部署 Web 应用作为 Enterprise 应用的一部分”的内容，位于：

http://e-docs.bea.com/wls/docs61/webapp/deployment.html#war-ear.

下表给出了一些示例 URL 以及 WebLogic 服务器所服务的文件。Index Directories Checked 列指的是 Index Directories 属性，该属性决定当请求没有指定文件时是否要提供目录列表服务。该属性在管理控制台的 Web Application 节点的 Configuration/Files 标签页设置。

表 8-1 WebLogic 应用解析 URL 示例

URL	Index Directories Checked	响应的文件
http://host:port/apples	No	Apples web 应用所定义的欢迎页面*
http://host:port/apples	Yes	列出 apples web 应用的最顶层目录的子目录
http://host:port/oranges/naval	无关要紧	oranges 应用中映射<url-pattern>为/naval 的 servlet。有关 servlet 映射还有一些需考虑的东西。详细内容，请参见 “配置 Servlets”的内容，位于：http://e-docs.bea.com/wls/docs61/webapp/components.html#configuring-servlets

http://host:port/naval	无关要紧	Oranges 应用中映射到<url-pattern>为/naval 的 servlet 并且 oranges 是缺省的 Web 应用。详细内容，参见以下页面的“Configuring servlets”。 http://e-docs.bea.com/wls/docs61/webapp/components.html#configuring-servlets
http://host:port/apples/pie.jsp	无关要紧	Apples 应用顶层目录中的 pie.jsp 文件
http://host:port	Yes	缺省 Web 应用顶层目录的目录列表
http://host:port	No	缺省 Web 应用的欢迎页面*
http://host:port/apples/myfile.html	无关要紧	Apples 应用顶层目录中的 myfile.html 文件
http://host:port/myfile.html	无关要紧	缺省 Web 应用顶层目录的 myfile.html 文件
http://host:port/apples/images/red.gif	无关要紧	Apples 应用顶层目录的 images 子目录中的 red.gif
http://host:port/myFile.html apples 应用没有定义 myFile.html 并且也没有定义缺省 servlet	无关要紧	Error 404 详细内容，参见 8-20 页的“定制 HTTP 错误响应”
http://www.fruit.com/	No	虚拟主机www.fruit.com的缺省 Web 应用的欢迎页面*
http://www.fruit.com/	Yes	给出虚拟主机www.fruit.com上的缺省应用的最高层目录的子目录列表
http://www.fruit.com/oranges/myfile.html	无关要紧	位于虚拟主机www.fruit.com上的 oranges Web 应用中的 myfile.html 文件

* 表示详细内容参见“配置 Welcome Pages”的部分，位于：

http://e-docs.bea.com/wls/docs61/webapp/components.html#welcom-pages

设置 HTTP 访问日志

WebLogic 服务器可以把所有关于 HTTP 事务的日志保存到一个文本文件，该文本文件可以采用*普通日志格式* common log format）或*扩展日志格式* extended log format），缺省格式为遵循标准约定的普通日志格式。扩展日志格式可以定制要记录的信息。每个服务器或虚拟主机都可以定义自己的 HTTP 访问日志属性。

日志回旋（Log Rotation

可以设定日志的回旋是基于文件大小还是基于时间间隔。当满足其中的一个条件时，当前访问日志被关闭，新的访问日志被创建。如果没有配置日志回旋，那么 HTTP 访问日志会无限增长。访问日志名中的数字部分随日志回旋而增长。每个 Web 服务器都定义了自己的 HTTP 访问日志。

使用管理控制台设置 HTTP 访问日志

有关使用管理控台设置 HTTP 访问日志的内容可以参见以下页面中的信息：

http://e-docs.bea.com/wls/docs60/ConsoleHelp/virtualhost.html

1. 如果设置了虚拟主机：

   a.在左窗格中选择 Services 节点

   b.选择 Virtual hosts 节点，该节点被展开并显示了一组虚拟主机

c.选择一个虚拟主机

如果还没有设置虚拟主机

a.选择左窗格中的 Servers 节点，该节点被展开并显示了一组服务器。

b.选择一个服务器

c.选择 Logging 标签页

d.选择 HTTP 标签页

2．选中 Enable Logging 复选框

3．输入日志文件的名字

4．从 Format 下拉列表中选择 Common 或 Extended

5．选择 Rotation 类型：By Size 类型还是 By Date 类型的

By Size：当日志文件的大小大于 Log Buffer Size 参数的值时发生日志回旋。

By Date：当时间间隔大于 Rotation Period 参数的数值时发生日志回旋。

6．如果日志回旋的类型为 By Size，那么将 the Max Log File Size K Bytes 字段设置为所期望的日志文件的最大容量（以字节数为单位）

7．设置 Flush Every 参数。该参数设置了每隔多长时间访问日志进行日志条目的写操作。

8．如果日志回旋的类型设置为 By Date，那么把 Rotate time 参数设为日志文件进行第一次回旋的时间（该参数只对 By Date 类型的日志回旋有效），日期的格式使用 java.text.SimpleDateFormat，即 MM-dd-yyyy-k:mm:ss。详细内容请参见 java.text.SimpleDateFormat 类的 javadocs 文档。

9．如果日志回旋的类型设为 By Date，那么将 Rotation Period 设置为日志回旋的时间间隔。

普通日志格式

HTTP 日志信息的缺省格式为普通日志格式。普通日志格式由 http://www.w3.org/Daemon/User/Config/Logging.html#common-logfile-format 定义，它遵循以下模式：

host RFC931 auth_user [day/month/year:hour:minute:second UTC_offset] "request" status bytes

其中：

host

指远程客户端的 DNS 名字或 IP 地址

RFC931

任何由 INDENTD 返回给客户端的消息；WebLogic 服务器不支持用户标识

auth_user

如果远程客户发送了一个用于身份验证的用户号，那么就是指用户名，否则就为“-”

Day/month/year:hour:minute:second UTC_offset

日期、月份、年以及时间（使用 24-小时格式）以及当地时间与 GMT 时间的时间差，它们被括在方括号中

“request”

远程 HTTP 请求的第一行，被双引号括起

status

服务器返回的 HTTP 状态代码，如果服务器没有返回 HTTP 状态代码，则为“-”。

bytes

HTTP 头中 content-length 所记录的内容长度（不包括 HTTP 头本身）。如果不可知，则为“-”

使用扩展日志格式

WebLogic 服务器支持由 W3C 定义的 1.0 版本的扩展日志文件格式，这是一个正在形成的标准。目前 WebLogic 服务器遵循该标准的草案规范（www.w3.org/TR/WD-logfile.html）。请参见“W3C Technical Reports and Publications”的内容，位于：www.w3.org/pub/www/TR

当所记录的 HTTP 日志采用扩展日志格式时，可以指定记录哪些类型的信息以及按什么顺序排列信息。在管理控制台的 HTTP 标签页中，将 Format 属性设置为 Extended，这样便启用了扩展日志格式。（请参见本章“用管理控制台设置 HTTP 访问日志”中的步骤 4

日志文件所要记录的信息类型通过日志文件中的指令来指定，它包含在实际的日志文件中。每条指令必须新起一行，以#符号开始指令。如果系统还没有日志文件，那么会新建一个包含缺省指令的日志文件。如果在服务器启动时，已经有日志文件了，那么这个日志文件头中必须包含合法指令。

创建 Fields 指令

日志文件的第一行必须是表明日志文件格式的版本号指令，紧跟其后是 Fields 指令：

#Version: 1.0

#Fields: XXXX XXXX XXXX ...

其中，每个 *XXXX* 描述要记录的数据字段。字段类型要么由简单的标识符来指定，要么采用 W3C 规范中的前缀标识符格式。以下是一个例子：

#Fields: date time cx-method cs-url

该指令要求服务器要记录每次 HTTP 访问的以下信息：事务的日期与时间，客户端请求方式以及请求的 URI。字段之间用空格阁开。每条日志记录都新起一行，并附加在日志文件尾部。

注意：#Fields 指令后面必须有一新行，这样第一条日志信息才不会与#Fields 指令在同一行上

字段标识符

以下是所支持的标识符，这些标识符不需要前缀。

date

事务完成的日期，类型为 W3C 规范定义的<date>类型

time

事务完成的时间，类型为 W3C 规范定义的<time>类型

time-taken

事务花费的时间（以秒为单位），类型为 W3C 规范定义的<fixed>类型

bytes

传送的字节数，类型为<integer>

注意，WebLogic 服务器不支持 W3C 规范中定义的 cached 字段。

以下是需要前缀的标识符，这些标识符不能单独使用。

与 IP 地址相关的字段：

这些字段给出了请求客户端或应答服务器的 IP 地址与监听端口。字段的类型为 W3C 规范定义的<address>类型。所支持的前缀有：

c-ip

客户端的 IP 地址

s-ip

服务器的 IP 地址

与 DNS 相关的字段

这些字段给出了客户端与服务器的域名。字段的类型为 W3C 规范定义的<name>类型。所支持的前缀有：

c-dns

客户端的域名

s-dns

被请求服务器的域名

sc-status

响应的状态代码。例如（404）表示状态为“文件没有找到”。该字段的类型为 W3C 规范所定义的<integer>类型

sc-comment

状态码的注释。例如，“文件没有找到”。该字段的类型为<text>类型

cs-method

请求方式，例如 GET 或 POST。该字段的类型为 W3C 规范所定义的<name>类型

cs-uri

请求的完整 URI。该字段的类型为 W3C 规范所定义的<uri>类型

cs-uri-stem

URI 的主干部分（省略了查询）。该字段的类型为 W3C 规范所定义的<uri>类型

cs-uri-query

URI 的查询部分。该字段的类型为 W3C 规范所定义的<uri>类型

定制字段标识符

扩展日志格式的 HTTP 访问日志可以使用用户定义的字段。要创建一个定制字段，你必须在

扩展日志格式（即 ELF 格式）的日志文件中使用 Fields 指令来标识定制的字段，同时定义与该字段匹配的 Java 类以生成所需要的输出。你可以为每个字段创建一个 Java 类，也可以用一个 Java 类匹配多个字段。本文提供了这种 Java 类的代码示例。详细内容，请参见“Java 类与创建定制 ELF 字段”。

要创建一个定制字段，

1　在 Fields 指令中包含该字段的名字，形式如下：

x-myCustomField

其中 myCustomField 为类的完整名字

有关 Fields 指令的详细信息，请参见“创建 Fields 指令”

2　创建一个与定制字段同名的 Java 类（例如 myCustomField）。这个类定义了要记录自定制字段中的信息。该 Java 类必须实现以下接口：

weblogic.servlet.logging.CustomELFLogger

这个 Java 类必须实现 logField()方法，该方法使用 HttpAccountingInfo 对象与 FormatStringBuffer 对象作为参数。

用 HttpAccountingInfo 对象访问 HTTP 请求与响应的数据，这些数据将被输出到所定制的字段中。它提供了许多 Get 方法来访问这些信息。Get 方法的完整列表请参见 7-15 页的“HttpAcountingInfo 对象的 Get 方法”。

用 FormatStringBuffer 类创建定制字段的内容。他提供了创建定制字段的合适的方法。有关这些方法的详细内容，请参见 FomatStringBuffer 类的 Java 文档。（见以下网页：http://e-docs.bea.com/wls/docs60/javadocs/weblogic/servlet/logging/FormatStringBuffer.html

3　编译 Java 类，并把它加到启动 WebLogic 服务器的 CLASSPATH 语句中。如果用脚本启动 WebLogic 服务器，那么要修改脚本中的 CLASSPATH 语句

注意：不要将这个类放在扩展格式的 Web 应用或企业应用中。

4　对 WebLogic 服务器进行配置，让它使用扩展日志格式。详细内容，参见“使用扩展日志格式”。

注：在编写定制字段的 Java 类时，务必不要在这个类中执行降低系统性能的代码（例如，访问 DBMS，以及执行繁重的 I/O 或网络调用。）记住，对于每一个 HTTP 请求，系统都会相应地创建一条 HTTP 访问日志记录。

注：如果要输出到多个字段，那么这些字段之间应该用制表符分隔开。有关字段分隔以及扩展日志格式的更多内容，请参见“扩展日志格式”的内容，位于：
http://www.w3.org/TR/WD-logfile-960221.html

HttpAccountingInfo 对象的 Get 方法

以下方法返回与 HTTP 请求相关的数据。这些方法与 javax.servlet.ServletRequest, javax.servlet.http.Http.ServletRequest 以及 javax.servlet.http.HttpServletResponse 等类的方法相似。

这些方法在 Java 接口中定义，下表列出了这些方法以及从什么地方可以找到有关这些方法的信息。

表 8-2　HttpAccountingInfo 的 Getter 方法

HttpAccountingInfo 的方法	参考以下类中的相应方法
Object getAttribute	javax.servlet.ServletRequest
Enumeration getAttributeName()	javax.servlet.ServletRequest
String getCharacterEncoding()	javax.servlet.ServletRequest
Int getResponseContentLength()	Javax.servlet.servletResponse.setContentLength() 该方法返回响应的长度，该长度由 setContentLenth()方法设置
String getResponseContentLength()	javax.servlet.ServletRequest
Strng getContentType();	javax.servlet.ServletRequest
Locale getLocale();	javax.servlet.ServletRequest
Enumeration getLocales();	javax.servlet.ServletRequest
String getParameter(String name);	javax.servlet.ServletRequest
Enumeration getParameterNames();	javax.servlet.ServletRequest
String[] getParameterValues(String name);	javax.servlet.ServletRequest
String getProtocol();	javax.servlet.ServletRequest
String getRemoteAddr();	javax.servlet.ServletRequest
String getRemoteHost();	javax.servlet.ServletRequest
String getScheme();	javax.servlet.ServletRequest
String getServerName();	javax.servlet.ServletRequest
int getServerPort();	javax.servlet.ServletRequest
boolean isSecure();	javax.servlet.ServletRequest
String getAuthType();	javax.servlet.http.Http.ServletRequest
HttpAccountingInfo Methods	javax.servlet.http.Http.ServletRequest
String getContextPath();	javax.servlet.http.Http.ServletRequest
Cookie[] getCookies();	javax.servlet.http.Http.ServletRequest
long getDateHeader(String name);	javax.servlet.http.Http.ServletRequest
String getHeader(String name);	javax.servlet.http.Http.ServletRequest
Enumeration getHeaderNames();	javax.servlet.http.Http.ServletRequest
Enumeration getHeaders(String name);	javax.servlet.http.Http.ServletRequest
int getIntHeader(String name);	javax.servlet.http.Http.ServletRequest
String getMethod();	javax.servlet.http.Http.ServletRequest
String getPathInfo();	javax.servlet.http.Http.ServletRequest
String getPathTranslated();	javax.servlet.http.Http.ServletRequest
String getQueryString();	javax.servlet.http.Http.ServletRequest
String getRemoteUser();	javax.servlet.http.Http.ServletRequest
String getRequestURI();	javax.servlet.http.Http.ServletRequest
String getRequestedSessionId();	javax.servlet.http.Http.ServletRequest
String getServletPath();	javax.servlet.http.Http.ServletRequest
Principal getUserPrincipal();	javax.servlet.http.Http.ServletRequest
boolean isRequestedSessionIdFromCookie();	javax.servlet.http.Http.ServletRequest
boolean isRequestedSessionIdFromURL();	javax.servlet.http.Http.ServletRequest
boolean isRequestedSessionIdFromUrl();	javax.servlet.http.Http.ServletRequest
boolean isRequestedSessionIdValid();	javax.servlet.http.Http.ServletRequest
String getFirstLine();	返回 HTTP 请求的第一行。例如 GET/index.html HTTP/1.0

long getInvokeTime();	返回查找 servlet 的 service()方法所花费的方法，这个 servlet 的作用是把数据回写到客户端
int getResponseStatusCode();	javax.servlet.http.HttpServletResponse
String getResponseHeader(String name);	javax.servlet.http.HttpServletResponse

列表 8-1 用于创建 ELF 字段的 Java 类

```
import weblogic.servlet.logging.CustomELFLogger;
import weblogic.servlet.logging.FormatStringBuffer;
import weblogic.servlet.logging.HttpAccountingInfo;
/* This example outputs the User-Agent field into a
custom field called MyCustomField
*/
public class MyCustomField implements CustomELFLogger{
public void logField(HttpAccountingInfo metrics,
FormatStringBuffer buff) {
buff.appendValueOrDash(metrics.getHeader("User-Agent"));
}
}
```

防止通过 POST 方式的“拒绝服务攻击”

“拒绝服务攻击”是通过假冒请求造成服务器过载的一种恶意攻击。这种攻击类型通常采用 HTTP 的 POST 方法发送大量数据。在 WebLogic 服务器中，你可以通过设置三个属性来防止这种攻击。这三个属性可以在管理控制台的 Servers 或 virtural hosts 节点下设置。如果为一个虚拟主机定义了这些属性，那么所设置的值将覆盖 Servers 节点中的设置

PostTimeoutSecs

通过这个属性，你可以限制 WebLogic 服务器接收 HTTP POST 数据段之间的等待时间

MaxPostTimeSecs

限制 WebLogic 服务器用于接收 POST 数据的时间。如果触发了该限制，那么将会引发 PostTimeoutException 异常并将以下信息发送到服务器日志中。

Post time exceeded MaxPostTimeSecs

MaxPostSize

限制每个请求中的 POST 数据量。当请求中的 POST 数据量超过该限制时，将引发 MaxPostSizeExceeded 异常并将以下信息发送到服务器日志中。

POST size exceeded the parameter MaxPostSize

同时向客户端返回 HTTP 413 错误代码

如果客户端使用了监听模式，那么它将得到这个信息。如果没有采用监听模式，则

断开连接。

设置 WebLogic 服务器的 HTTP 隧道

如果你只能采用 HTTP 协议连接 WebLogic 服务器与 Java 客户端，那么可以用 HTTP 隧道功能来模拟有状态的套接字连接。HTTP 隧道通常被用来穿越安全防火墙的 HTTP 端口。HTTP 是一种无状态协议，因此 WebLogic 服务器所提供的隧道功能使得连接看上去象是一个 T3 连接。但比起标准的套接字连接，这会造成性能的下降。

配置 HTTP 隧道连接

当使用 HTTP 协议时，客户端只能发送请求，然后接收服务器的响应。服务器不能主动地与客户端通信，并且 HTTP 协议是无状态的，这就意味这服务器与客户端之间不可能发生连续的双向连接。

WebLogic HTTP 隧道通过 HTTP 协议模拟 T3Connection 协议，从而克服上述限制。有两个属性与隧道连接的性能有关。这两个属性可以通过管理控制台的 Servers 部分的 Configuration 标签页的 Tunning 标签页来设置。大多数情况下，你应该使用这两个属性的缺省值，除非你对在处理连接问题方面非常有经验。服务器通过这些属性来决定一个客户端连接是否有效以及客户端是否是活动的。

Enable Tunneling

启用或禁用 HTTP 隧道。缺省情况下，HTTP 隧道是被禁用的

Tunneling Ping

在建立 HTTP 隧道连接时，客户端会自动地向服务器发一个请求，使服务器可以主动响应客户端。客户端可以在请求中包含指示，无论客户端应用是否需要与服务器进行通信。如果服务器没有在由该属性所设置的时间内响应客户端，那么它就会主动地给客户端一个响应，客户端接收该响应并立即发送另一个请求。

缺省值为 45 秒；有效的范围为 20 至 900 秒

Tunning Timeout

在继上次客户端请求之后，如果在该属性所指定的时间内，客户端还没有发出新的请求，那么服务器就认为客户端已经死了并结束该 HTTP 隧道连接。在服务器响应客户端请求时，它会按该属性所指定的时间间隔来检测继上一次请求后所过去的时间。

缺省值为 40 秒，有效范围为 10 至 900 秒

建立客户端与 WebLogic 服务器之间的连接

如果客户端要与 WebLogic 服务器建立 HTTP 隧道连接，那么只需要 URL 中指定 HTTP 协议。例如：

```
Hashtable env = new Hashtable();
env.put(Context.PROVIDER_URL, "http://wlhost:80");
```

Context ctx = new InitialContext(env);

客户端会在 HTTP 协议后面附加一个特殊标签，这样 WebLogic 服务器就知道这是一个隧道连接而不是普通的 HTTP 请求，这些都是自动的，你的应用不需要做额外的工作。

客户端必须在 URL 中指定端口号，即便是使用 80 端口。尽管 WebLogic 服务器可以在任何端口监听 HTTP 请求，但一般使用 80 端口，因为通常只有该端口才能穿越防火墙。

你可以在管理控制台的 Servers 节点的 Network 标签页配置 WebLogic 服务器的简听端口。

用本地 I/O 提供静态文件服务（只适用于 Windows

当你在 Windows NT/2000 上运行 WebLogic 服务器，可以让 WebLogic 服务器通过本地操作系统调用 TramsmitFile 来代替 Java 方法服务静态页面，例如 HTML 文件，文本文件以及图象文件。在服务较大的静态页面时，使用本地 I/O 会提高服务器的性能。

要使用本地 I/O，那么需要在 Web 应用的分发描述符 web.xml 中增加两个 I/O 参数。参数 weblogic.http.nativeIOEnabled 应该设置为 TRUE 以启用本地 I/O 文件服务。参数 weblogic.http.minimumNativeFileSize 指定要用本地 I/O 进行服务的文件的最小长度。如果一个文件的长度大于该参数所设置的值，那么就会用本地 I/O 来服务该文件。如果没有指定该参数的值，那么默认为 400 个字节。

一般而言，对于较大的文件，使用本地 I/O 会提高服务的效率。但随着运行服务器的计算机的负载的增大，这种效益会消失。因此需要通过实验来寻求 weblogic.http.minimumNativeFilesize 参数的最佳值。

下面的例子演示了如何在 web.xml 中添加本地 I/O 的相关参数的配置。这些设置必须放在<distributable>元素与<servlet>元素之间。

```
<context-param>

<param-name>weblogic.http.nativeIOEnabled</param-name>

<param-value>TRUE</param-value>

</context-param>

<context-param>

<param-name>weblogic.http.minimumNativeFileSize</param-name>

<param-value>500</param-value>

</context-param>
```

有关编写分发描述符的详细内容，请参见 “Writing Web Application Deployment Descriptors”的内容，位于：

http://e-docs.bea.com/wls/docs60/programming/webappdeployment.html.

第九章 代理对另一个 HTTP 服务器的请求

本章讨论如何将 HTTP 请求代理到其它 HTTP 服务器。其中涵盖以下主题：

- 概述
- 建立一个到辅助 HTTP 服务器的代理
- ProxyServlet 的分发描述符范例

概述

如果把 WebLogic 服务器作为主 Web 服务器，通过配置，可以让它把请求委托给另一个 HTTP 服务器处理，例如 Netscape Enterprise 服务器,Apache 或者是 Microsoft Internet Information 服务器。任何被代理的请求都会重定向到一个指定的URL。被代理的Web服务器可以位于其它机器上。我们一般基于请求的 URL 来决定把请求委托给哪个服务器处理。

HttpProxyServlet 将 HTTP 请求重定向到代理 URL 上，然后通过 WebLogic 服务器将响应返回给浏览器。要使用代理功能，必须在 Web 应用中配置该功能，并把它分发到一个重定向请求的 WebLogic 服务器中。

设置从服务器的代理

要设置从服务器的代理：

1．在 Web 应用分发描述符（见“使用 ProxyServet 的 web.xml 示例”）注册代理 servlet Web 应用必须是响应请求的服务器的缺省应用。代理 servlet 的类名为 weblogic.t3.srvr.HttpProxyServlet。详细信息可以参见“组装和配置 Web 应用”的内容，位于：http://e-docs.bea.com/wls/docs60/adminguide/config_web_app.html

2 定义 ProxyServlet 的初始化参数。<param-name>为 redirectURL，<param-value>为从服务器的 URL

3 在 Web 应用分发描述符 web.xml 中使用<servlet-mapping>元素把 ProxyServlet 映射到<url-pattern>特别地，为想代理的这些文件扩展名设置映射，例如：*.jsp 或*.html

如果<url-pattern>设置为“/”，那么任何不能被 WebLogic 服务器所解析的请求都会委托给远程服务器处理。如果你还希望代理特定扩展名的文件，那么必须设定对扩展名的映射，如：*.jsp， .html 以及*.htm

ProxyServlet 的分发描述符示例

以下是使用 Proxy servlet 的 Web 应用分发描述符示例。

列表 9-1 使用 ProxyServlet 的 web.xml 示例

```
<!-- DOCTYPE web-app PUBLIC "-//Sun Microsystems, Inc.
//DTD Web Application 1.2//EN"
"file:///weblogic/dev/myserver/servlet2.2/WEB-INF/web-jar.dtd"
```

```
-->
<web-app>

<servlet>
<servlet-name>ProxyServlet</servlet-name>
<servlet-class>weblogic.t3.srvr.ProxyServlet</servlet-class>
<init-param>
<param-name>redirectURL</param-name>
<param-value>
tehama1:7736:7737|tehama2:7736:7737|tehama:7736:7737
</param-value>
</init-param>
</servlet>

<servlet-mapping>
<servlet-name>ProxyServlet</servlet-name>
<url-pattern>/</url-pattern>
</servlet-mapping>

<servlet-mapping>
<servlet-name>ProxyServlet</servlet-name>
<url-pattern>*.jsp</url-pattern>
</servlet-mapping>

<servlet-mapping>
<servlet-name>ProxyServlet</servlet-name>
<url-pattern>*.htm</url-pattern>
</servlet-mapping>

<servlet-mapping>
<servlet-name>ProxyServlet</servlet-name>
<url-pattern>*.html</url-pattern>
</servlet-mapping>

</web-app
```

第十章 代理对 WebLogic 集群的请求

本章将介绍如何代理对 WebLogic 集群的请求，主要有以下内容：

概述

设置 HttpClusterServlet

HttpClusterServlet 分发描述符示例

概述

WebLogic 服务器软件中自带的 HttpClusterServlet 通过一个 WebLogic 服务器代理对 WebLogic 集群中的其他服务器成员的 HTTP 请求， 同时 HttpClusterServlet 还为代理的 HTTP 请求提供负载平衡与容错处理。有关 servlets 与 WEbLogic 集群的详细内容，可以参见 “理解 HTTP 会话状态复制”的部分，位于：

http://e-docs.bea.com/wls/docs60/cluster/servlet.html

设置 HttpClusterServlet

以下是使用 HttpClusterServlet 所需要的配置：

1．在 WebLogic 服务器的管理控制台配置一个代理 HTTP 请求的 WebLogic 服务器实例，该服务器实例负责把请求重定向到 WebLogic 服务器。关于管理控制台的信息，参见 http://e-docs.bea.com/wls/docs61/adminguide/index.html

a. 新建一个应用

b. 在域中新建一个服务器，或使用缺省的服务器

c. 把新创建的 Web 应用设置为新建服务器的缺省 Web 应用。

2．在步骤 1 所创建的 Web 应用的分发描述符中注册 HttpClusterServlet。（见本章的“HttpServlet 的分发描述符示例”）。该 Web 应用必须是响应请求的服务器的缺省 Web 应用。相关信息参见第 8 章的“指定一个缺省 Web 应用”。

HttpClusterServlet 的类名是 weblogic.servlet.internal.HttpClusterServlet HttpClusterServlet 的分发描述符示例见下文

3. 定义 HttpClusterServlet 的初始化参数。初始化参数用 web.xml 的<init-param>元素定义。defaultServers 参数是必须定义的，其它参数视需要而定。有关 HttpClusterServlet 的参数可以参见本章表 10-1 “HttpClusterServlet Parameters”中的内容。

4．把代理 servlet 映射到一个<url-pattern>。特别地，配置映射需要代理的文件扩展名，例如“*.jsp”或*.html

如果<url-pattern>设置为“/”，那么任何不能被 WebLogic 服务器所解析的请求都将交给远程服务器处理。如果你还希望代理特定扩展名的文件，如：*.jsp， .html 以及*.htm，那么应该配置对扩展名进行映射。

设置 URL 模式的另一方式是：首先映射一个 url 模式，例如/foo，然后把 pathTrim 参数设置为 foo，该设置的作用是把 foo 从被代理的 URL 中删除。

表 10-1 HttpClusterServlet 的参数

<param-name>	<param-value>	缺省值
defaultServlets	（必须设置该参数）一组代理请求的服务器，采用以下形式：hostname1:HTTP port1:HTTPS port1 | hostname2:HTTP port2:HTTPS port2 如果 secureProxy 参数设置为 ON（见下面的 secureProxy 参数），那么 HTTPS 端口将在运行 HttpClusterServlet 的 WebLogic 服务器与集群的成员服务器之间使用 SSL 协议。即使把 secureProxy 参数设为 OFF，你也必须定义 HTTPS 端口	None
secureProxy	ON/OFF。如果设置为 ON，那么 HTTPClusterServlet 与 WebLogic 服务器集群成员之间的连接将使用 SSL 协议	OFF
DebugConfigInfo	ON/OFF。如果设置为 on，那么在请求中加上一个请求参数?_WebLogicBridgeConfig 就可以查询 HttoClusterServlet 的调试信息。因为安全性的原因，建议在生产环境中把该参数设置为 OFF	OFF
connectionTimeout	套接字等待读入数据的时间，单位为微秒。超时会引发 java.io.InterruptedIOException 异常	0（即没有限制）
numOfRetries	HttpClusterServlet 重试一个失败连接的次数	5
PathTrim	从原始 URL 的开头部分裁减的字符串	None
TrimExt	从原始 URL 的最后裁减掉的文件扩展名	None
pathPrepend	在 PathTrim 被裁减后，在请求被代理到一个 WebLogic 服务器集群成员之前，附加在原始 URL 之前的字符串。	None

HttpClusterServlet 的分发描述符示例

以下是使用 HttpClusterServlet 的 Web 应用分发描述符示例：

列表 10-1 使用 HttpClusterServlet 的 web.xml 示例

```
<!-- DOCTYPE web-app PUBLIC "-//Sun Microsystems, Inc.
//DTD Web Application 1.2//EN"
"file:///weblogic/dev/myserver/servlet2.2/WEB-INF/web-jar.dtd"
-->
<web-app>

<servlet>
<servlet-name>HttpClusterServlet</servlet-name>
<servlet-class>
weblogic.servlet.internal.HttpClusterServlet
</servlet-class>

<init-param>
<param-name>defaultServers</param-name>
```

```
<param-value>
myserver1:7736:7737|myserver2:7736:7737|myserver:7736:7737
</param-value>
</init-param>

<init-param>
<param-name>DebugConfigInfo</param-name>
<param-value>ON</param-value>
</init-param>

</servlet>

<servlet-mapping>
<servlet-name>HttpClusterServlet</servlet-name>
<url-pattern>/</url-pattern>
</servlet-mapping>

<servlet-mapping>
<servlet-name>HttpClusterServlet</servlet-name>
<url-pattern>*.jsp</url-pattern>
</servlet-mapping>

<servlet-mapping>
<servlet-name>HttpClusterServlet</servlet-name>
<url-pattern>*.htm</url-pattern>
</servlet-mapping>

<servlet-mapping>
<servlet-name>HttpClusterServlet</servlet-name>
<url-pattern>*.html</url-pattern>
</servlet-mapping>

</web-app>
```

第十一章 安装和配置 Apache HTTP 服务器插件

本章将介绍如何配置 Apache-WebLogic Server 插件。主要有以下内容：

概述

平台支持

安装 Apache HTTP 服务器插件

配置 Apache HTTP 服务器插件

Apache-WebLogic Server 插件的配置参数

在 Apache 插件中使用 SSL

与 SSL-Apache 配置有关的问题

httpd.conf 文件模板示例

连接错误和集群容错

概述

通过 Apache HTTP 服务器插件允许 Apache 服务器代理对 WebbLogic 服务器的访问请求。通过插件可以允许 WebLogic 服务器处理访问 WebLogic 服务器动态功能得到请求，从而增强了原有 Apache 服务器的功能。

插件用于这样一种环境，其中 Apache 服务器提供静态页面的服务，而把文档树中的另一部分内容（由 HTTPServlets 和 JSP 产生的动态页面）转交给工作在另一个进程也可能是在另一个主机中的 Weblogic 服务器。对于终端用户即浏览器而言，转交给 webLogic 服务器的 HTTP 请求是依然是来自相同的源

HTTP-隧道技术部件可以通过插件来运行，使得非流览器用户可以访问 WebLogic 服务器的服务。

Apache HTTP 服务器插件在 Aparhe HTTP 服务器中作为一个 Apache 模块运行。Apache 服务器在启动时会装载 Apache 模块，然后把 HTTP 请求交给被装载的 Apache 模块来处理。Apache 模块类似于 HTTP servlets，只是 Apache 模块是采用与平台相关的本地代码编写。

Apache 1.3.x 中的 Keep-Alive 连接

Apache HTTP 服务器插件对每一个请求都建立一个套接字，并在读完请求后关闭。由于 Apache HTTP 服务器是多进程的，它不支持连接缓冲池和 WebLogic 服务器与 Apache HTTP 服务器之间的 Keep-Alive 连接。

Apache 2.x 中的 Keep-Alive 连接

通过 Apache HTTP 服务器插件到 WebLogic 服务器之间的可重用的连接缓冲池，提高了 Apache HTTP 服务器到 WebLogic 服务器的访问效率。插件在其与 WebLogic 服务器之间实现了 HTTP1.1 Keep-alive 连接，它对来自相同客户端的一系列请求重用了缓冲池的同一连接。如果一个连接处于非活动状态时间超过 30 秒（或用户定义的时间），连接会被关闭并释放回缓冲池中。可以根据需要禁止该设置，有关信息参见附件中的“Keep-Alive Enabled”

代理请求

插件根据你指定的设置来代理对 WebLogic 服务器的请求，你可以根据请求的 URL（或其中一部分）来代理请求，这种方式称为依据路径代理。你也可以通过被请求的文件的 MIME 类型来代理请求。你还可以将两种方式混合使用，如果一个请求满足两种请求，请求会首先通过路径方式代理，你也可以对这些请求设定附加参数，它定义了插件的一些附加功能，有关信息参见“配置 Apache HTTP 服务器插件”

平台支持

Linux，Solaris 与 HPUX11 等平台都支持 Apache-WebLogic Server 插件。关于 Apache 版本支持的详细信息，参见 BEA WebLogic 服务器平台支持，位于：

http://e-docs.bea.com/wls/platforms/index.html#apach

安装 Apache HTTP 服务器插件

在 Apache HTTP 服务器中，插件是以 Apache 模块方式安装。模块可以以动态共享对象（DSO 或静态链接模块方式安装 Apache 模块。（只适用于 1.3.x 版本的 Apache）。本节将分别对每种安装模式介绍安装步骤。

以动态共享对象方式安装

按照动态共享对象方式安装 Apache HTTP 服务器插件，步骤如下：

1. 确认平台上的共享对象文件

   Apache HTTP 服务器插件以共享对象文件（.so）形式在 Solaris、Linux 和 HPUX 平台上分发。每一共享对象文件，依据平台不同，是否客户端和 Apache 之间是否使用 SSL，SSL 加密程度不同（通常是 128 位，128 位版本只适用于 128 位版本的 WebLogic 的服务器），而分发的版本不同。这些共享文件存放于 WebLogic 安装的下列目录中。

   Solaris

   lib/solaris

   Linux

   lib/linux

   HPUX11

   lib/hpux11

   Windows (Apache 2.0 only)

   bin/apache20

   从下表中选择相应的共享文件

Apache 版本	常规加密程度	128 位加密
Standard Apache Version 1.x	mod_wl.so	mod_wl128.so

Apache w/ SSL/EAPI Version 1.x (Stronghold, modssl etc.)	mod_wl_ssl.so	mod_wl128_ssl.so
Apache + Raven Version 1.x 必须的 因为Raven版本使用frontpage 补丁程序，使得其无法兼容 标准的共享对象	mod_wl_ssl_raven.so	mod_wl128_ssl_raven.so
Standard Apache Version 2.x	mod_wl_20.so	mod_wl28_20.so

2. 允许共享对象

Apache HTTP 服务器插件将以 Apache 动态共享对象的方式（DSO）安装，Apache 所中支持的 DSO 是基于一个 mod_so.c 文件，并且该文件需要在 mod_wl.so 加载前处于可用状态。如果你是使用脚本安装的 Apache，那么 mod_so.c 已经可用了。若要确认 mod_so.c 文件是否可用，执行下面命令：

*APACHE_HOM*E/bin/httpd -l

（其中 *APACHE_HOM*E 是 Apache HTTP 服务器安装的路径。）

这个命令列出了所有允许的模块，若 mod_so.c 没有列在其中，利用源代码重新编译 Apache HTTP 服务器，并确认配置了下列选项：

```
...
--enable-module=so
--enable-rule=SHARED_CORE

...
```

3. 在安装 Apache HTTP 服务器插件同时，安装一个名为 apxs 的支持程序（Apache eXtension），它在 Apache 代码树外部编译基于 DSO 的模块，并在 httpd.conf 文件内加入如下的行：

AddModule mod_so.c

在 WebLogic 服务器安装中，通过命令行界面流览到包含你所在平台所支持的共享对象的目录，通过该命令激活 weblogic_module（注：你必须安装 perl 以运行 perl 脚本）：

perl *APACHE_HOM*E/bin/apxs –i –a –n weblogic mod_wl.so

这个命令将mod_wl.so文件复制到*APACHE_HOME*/libexec目录下。它在httpd.conf文件中为weblogic_module添加了两行指令。确认下列行添加到了*APACHE_HOM*E/conf/httpd.conf 文件中：

LoadModule weblogic_module

AddModule mod_weblogic.c

注：若你使用了 Apache 2.0 beta 产品，该过程可能会不同。详细信息，请参见 Apache HTTP 服务器 2.0 的文档，位于：http://httpd.apache.org/docs-2.0/.

4. 用以下命令确认 *APACHE_HOM*E/conf/httpd.conf 的语法

   APACHE_HOME/bin/apachect1 configtest

   这个命令的输出表示了 httpd.conf 文件中的任何错误。

5. 在 httpd.conf 文件中配置和添加参数，参见“配置 Apache HTTP 服务器插件”，httpd.conf 文件允许你定制插件的行为。

6. 启动 WebLogic 服务器。

7. 启动（或若修改了配置需要重新启动）Apache HTTP 服务器。

8. 打开流览器测试 Apache 插件，发送 URL +/weblogic/ 请求到 Apache HTTP 服务器，它会调用 WebLogic 缺省的 HTML 欢迎页面，或者缺省的 servlet（在 WebLogic 缺省 Web 应用中定义的。）

以静态链接模块方式安装

按照静态链接模块方式安装 Apache HTTP 服务器插件，步骤如下：

1. 确认你所在平台的链接库文件位置。

   考虑到平台的不同，SSL 加密程度（通常是 128 位，128 位版本只适用于 128 位版本的 WebLogic 的服务器），每个库文件按照不同版本分发。库文件位于 WebLogic 服务器软件安装的下列目录下：

   Solaris

   lib/solaris

   Linux

   lib/linux

   HPUX11

   lib/hpux11

   选择下表中相应的共享对象

Apache 版本	常规加密程度	128 位加密
Standard Apache Version 1.3.x	libweblogic.a	libweblogic128.a

2. 用下列命令对 Apache 分发的包进行解压：

   tar -xvf apache_1.3.x.tar

3. 在解压的包中转换到/src/modules 目录

4. 创建一个名为 weblogic 的目录

5. 从 WebLogic 服务器上的 lib 目录中将 Makefile.libdir, Makefile.tmpl 文件复制到 /src/modules/weblogic 目录
6. 从包含连接库文件的同一目录下，将文件 libweblogic.a 文件（如果使用 128 位安全机制，则用 libweblogic128.a）复制到/src/modules/weblogic 目录
7. 如果使用常规加密，从 Apache1.3 根目录下执行下列命令：

   configure --activate-module=src/modules/weblogic/libweblogic.a
8. 执行 make 命令
9. 执行 make install 命令
10. 按照“按动态共享对象方式安装”步骤 4 继续

配置 Apache HTTP 服务器插件

安装完库后，需要修改 httpd.conf 文件来配置 Apache 插件。修改 httpd.conf 文件是为了通知 Apache web 服务器应该以 Apache 模块的形式来装载插件的本地库，并描述模块应如何处理请求。

编辑 httpd.conf 文件

通过编辑 httpd.conf 文件来配置 Apache HTTP 服务器插件：

1. 打开 httpd.conf 文件，该文件位于 *APACHE_HOME*/conf/httpd.conf （其中 *APACHE_HOME* 是 Apache 安装的根路径）。
2. 当使用 apxs 时，确认下列两行加入到 httpd.conf 文件中：

   LoadModule weblogic_module libexec/mod_wl.so

   AddModule mod_weblogic.c
3. 加入定义下列之一的 IfModule 模块

   对非集群的 WebLogic 服务器：

   需要参数 WebLogicHost 和 WebLogicPort

   对集群的 WebLogic 服务器：

   需要参数 WebLogicCluster

   例如：

   ```
   <IfModule mod_weblogic.c>
         WebLogicHost myweblogic.server.com
         WebLogicPort 7001
    </IfModule>
   ```
4. 若通过 MIME 类型代理请求，还应在 IfModule 模块中添加 MatchExpression 行（你也可以依据路径代理，依据路径代理优先于基于 MIME 代理请求，若只依赖于路径代理请求，跳过步骤 5

   例如，先烈 IfModule 块用于非集群的 WebLogic，并指明所有 MIME 类型为.jsp 的文件被代理：

```
<IfModule mod_weblogic.c>
    WebLogicHost myweblogic.server.com
    WebLogicPort 7001
    MatchExpression *.jsp
</IfModule>
```

你也可以使用多个 MatchExpression 块，例如：

```
<IfModule mod_weblogic.c>
    WebLogicHost myweblogic.server.com
    WebLogicPort 7001
    MatchExpression *.jsp
    MatchExpression *.xyz
</IfModule>
```

如果你使用基于 MIME 类型代理一个 WebLogic 集群的请求，使用 WebLogicCluster 参数替代参数 WebLogicHost 和 WebLogicPort。例如：

```
<IfModule mod_weblogic.c>
    WebLogicCluster w1s1.com:7001,w1s2.com:7001,w1s3.com:7001
    MatchExpression *.jsp
    MatchExpression *.xyz
</IfModule>
```

5. 若依据路径方式代理请求，使用 Location 块和 SetHandler 描述。SetHandler 指明 Apache HTTP 服务器插件模块的句柄。例如下例中的 Location 块描述 URL 中包含/weblogic 请求将被代理：

```
<Location /weblogic>
    SetHandler weblogic-handler
</Location>
```

6. 定义 Apache HTTP 服务器插件使用的其他参数。

   Apache HTTP 服务器插件可以识别列在“Web Server 插件的通用参数”一节中所列的参数。若想修改 Apache HTTP 服务器插件的行为，需要定义下面其中一种参数：

   - 对于基于路径方式代理，定义在 Location 块中，
   - 对于基于 MIME 类型方式的代理，定义在 IfModule 块中。

编辑 httpd.conf 文件的注意事项

- 作为“编辑 httpd.conf 文件”步骤的另一种选择，你可以在 weblogic.conf 文件中定义参数。在 httpd.conf 文件中的 IfModule 块中包含 weblogic.conf 文件。这样做，可以使文件配置模块化。例如：

  ```
  <IfModule mod_weblogic.c>
  ```

```
    # Config file for WebLogic Server that defines the parameters
    Include conf/weblogic.conf
</IfModule>
```

注：当插件和 WebLogic 之间采用了 SSL，不能采用这种方式。

- 每个参数应以一个新行录入。不需要在参数和值之间添加“=”。例如：

```
PARAM_1 value1
PARAM_2 value2
PARAM_3 value3
```

- 若请求同时满足 MatchExpression 中定义的 MIME 类型，以及 Location 块中定义的路径，Location 块定义的行为优先执行。
- 如果你定义了 CookieName 参数，你还必须在 IfModule 块中定义。

在 Apache 插件中使用 SSL

使用安全套接层（SSL）协议可以保护 WebLogic 服务器代理插件以及 Apache 服务器之间的连接，保护在 WebLogic 服务器代理插件与 Apache 服务器之间所传输的数据的机密性与完整性。此外，使用 SSL 协议还可以让 WebLogic 服务器代理插件向 Apache 服务器进行自我身份验证，从而保证信息被传递到一个可信任原点。

Apache HTTP 服务器插件不会依赖用流览器发出的 HTTP 请求所用的传输协议 http 或 https 来确定代理插件与 Apache 插件之间的连接是否使用了 SSL 协议保护。

在 Apache HTTP 服务器插件和 WebLogic 间配置 SSL

要在 Apache HTTP 服务器插件和 WebLogic 间使用 SSL

1. 配置 WebLogic 服务器的 SSL，详细信息参见“配置 SSL 协议”。
2. 配置 WebLogic 服务器中 SSL 的侦听端口，详细信息参见“配置侦听端口”。
3. 在 httpd.conf 文件中 WebLogicPort 设为步骤 2 中所定的端口
4. 在 httpd.conf 文件中的 SecureProxy 设为 ON
5. 在 httpd.conf 文件中定义关于 SSL 连接的参数，详细信息参见“Web Server 插件的参数”。

与 SSL-Apache 配置有关的问题

在 Apache 插件中使用 SSL 时，应该注意以下两个问题：

<Location>标签中必须使用 PathTrim

以下的配置是**不正确**的：

```
<Location /weblogic>
SetHandler weblogic-handler
</Location>
```

<IfModule mod_weblogic.c>

WebLogicHost localhost

WebLogicPort 7001

PathTrim /weblogic

</IfModule>

以下才是**正确**的配置

<Location /weblogic>

SetHandler weblogic-handler

PathTrim /weblogic

</Location>

Include 指令对 Apache SSL 不起作用。必须在 httpd.conf 文件中直接配置，当使用 SSL 时，**不能**使用下面的配置：

<IfModule mod_weblogic.c>

MatchExpression *.jsp

Include weblogic.conf

</IfModule>

连接错误和集群容错

当 Apache HTTP 服务器插件试图连接 WebLogic 服务器时，插件会使用几个配置参数，以决定连接 WebLogic 服务器主机的等待时间，在连接建立后，插件等待响应的时间长度等。如果插件不能连接或者没有收到响应，插件会试图连接并发送请求到集群中其他的 WebLogic 服务器。

若连接失败或集群中任何一台 WebLogic 服务器都无响应，则会发出出错信息。

图 11-1 连接容错演示了插件如何实现容错的。

连接失败

如果主机对于连接请求没有响应，则表明问题原因可能是主机、网络故障，或者服务器出错。

如果 WebLogic 服务器没有响应，可能表示 WebLogic 服务器没有运行或已经挂起，数据库出现问题或者应用出错。

单机，非 WebLogic 集群的容错

如果有且只有一台 WebLogic 服务器运行，除非插件只采用 WebLogicHost 定义参数试图连接，否则采用这里所描述的逻辑。若连接失败，会返回 HTTP503 出错信息。插件会继续试图连接 WebLogic 服务器，直至超过 ConnectTimeoutSecs 所定义的时间。

动态服务器列表

如果在 WebLogicCluster 参数中指定了 WebLogic 服务器列表，插件会使用该列表作为成员间负载均衡的起点。在第一次请求路由到一个服务器后，一个包含集群中的更新的服务器动态列表会返回。更新的列表会添加新的服务器，并删除不再属于该集群或不再响应请求的 WebLogic 服务器。当集群中有变化时，该动态列表会通过 HTTP 响应自动更新。

容错、Cookie 和 HTTP 会话

当一个请求包含存储于 cookie 中的会话信息，它可能以 POST 方式或 URL Encoding 方式传递，那么 sessionID 会包含指向会话建立的服务器（主服务器）的引用，以及保存原始会话复制信息的服务器（辅助服务器）。包含 cookie 信息的请求首先会连接主服务器，如果连接失败，那么请求会路由到辅助服务器。如果主服务器和辅助服务器均出错，会话信息会丢失，插件回建立一个到动态服务器列表中的另一个服务器的新连接。详细信息参见“图 11-1 连接容错”。

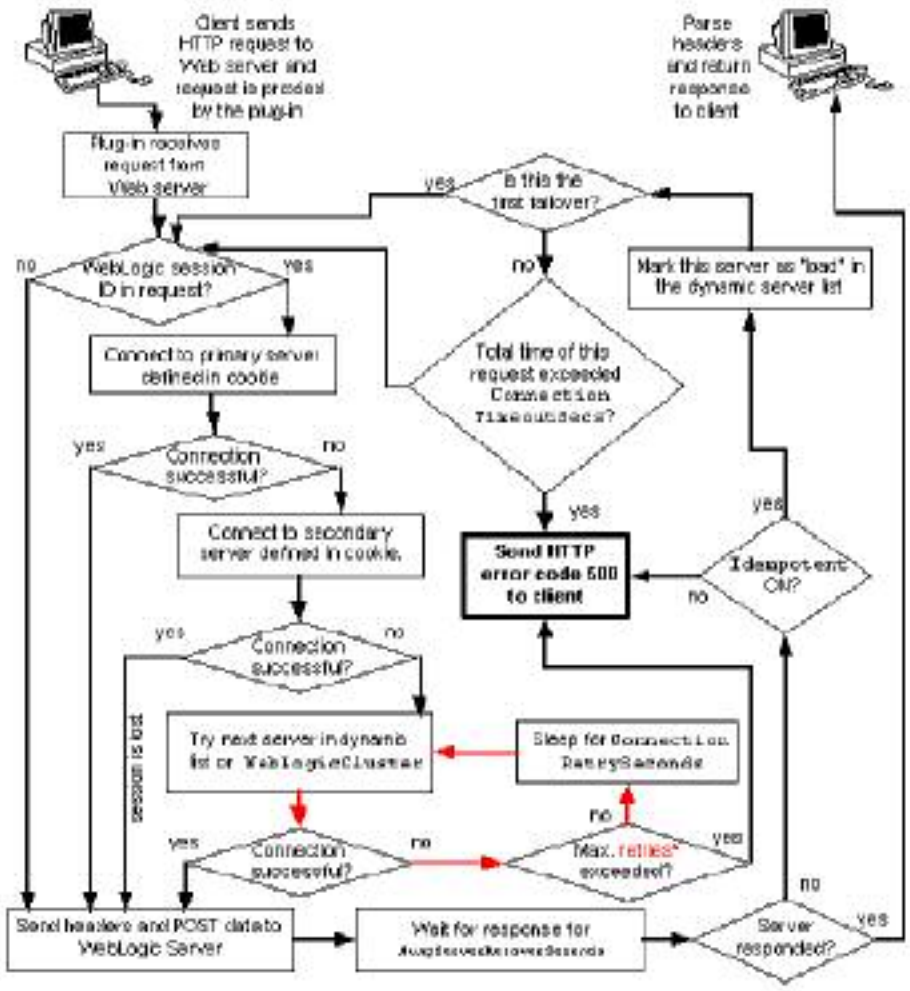


图 11-1 连接容错

* 红色循环中，允许最大重试次数 = ConnectTimeoutSecs÷ConnectRetrySecs

Httpd.conf 文件示例

下面是一个 httpd.conf 文件的例子。你可以把它作为模板，然后在此基础上对该文件进行修改，以与你的环境与服务器相适应。以 # 开始的行是注释行。请注意 Apache 不区分大小写，apxs 工具会自动添加 LoadModule 与 AddModule 行。

```
##################################################
APACHE-HOME/conf/httpd.conf file
##################################################
LoadModule weblogic_module libexec/mod_wl.so
<Location /weblogic>
SetHandler weblogic-handler
PathTrim /weblogic
ErrorPage http://myerrorpage1.mydomain.com
</Location>

<Location /servletimages>
SetHandler weblogic-handler
PathTrim /something
ErrorPage http://myerrorpage1.mydomain.com
</Location>

<IfModule mod_weblogic.c>
MatchExpression *.jsp
WebLogicCluster w1s1.com:7001,w1s2.com:7001,w1s3.com:7001
ErrorPage http://myerrorpage.mydomain.com
</IfModule>

# The following line is not always added:

AddModule mod_weblogic.c
```

配置文件示例

当然，也可以不在 httpd.conf 文件中的 Location 块中定义 Weblogic 参数，而是在 weblogic.conf 文件中定义参数，该文件由 httpd.conf 文件的 IfModule 装载。你可以将以下的示例作为模板，然后针对你所使用的环境与服务器做适当的修改。以 # 开始的行是注释。

使用 WebLogic 集群的范例

```
# These parameters are common for all URLs which are
# directed to the current module. If you want to override
# these parameters for each URL, you can set them again in
# the <Location> or <Files> blocks. (Except WebLogicHost,
# WebLogicPort, WebLogicCluster, and CookieName.)

<IfModule mod_weblogic.c>
WebLogicCluster w1s1.com:7001,w1s2.com:7001,w1s3.com:7001
ErrorPage http://myerrorpage.mydomain.com
MatchExpression *.jsp
</IfModule>
#####################################################
```

使用多 WebLogic 集群的范例

```
# These parameters are common for all URLs which are
# directed to the current module. If you want to override
# these parameters for each URL, you can set them again in
# the <Location> or <Files> blocks (Except WebLogicHost,
# WebLogicPort, WebLogicCluster, and CookieName.)
<IfModule mod_weblogic.c>
    MatchExpression *.jsp WebLogicHost=myHost|WebLogicPort=7001|Debug=ON
    MatchExpression *.html WebLogicCluster=myHost1:7282,myHost2:7283|ErrorPage=
    http://www.xyz.com/error.html
    </IfModule>
```

非 WebLogic 集群的范例

```
# These parameters are common for all URLs which are
# directed to the current module. If you want to override
# these parameters for each URL, you can set them again in
```

```
# the <Location> or <Files> blocks (Except WebLogicHost,
# WebLogicPort, WebLogicCluster, and CookieName.)

<IfModule mod_weblogic.c>
WebLogicHost myweblogic.server.com
WebLogicPort 7001
MatchExpression *.jsp
</IfModule>
```

配置基于 IP 的虚拟主机的例子

```
NameVirtualHost 172.17.8.1
<VirtualHost goldengate.domain1.com>
WebLogicCluster tehama1:4736,tehama2:4736,tehama:4736
PathTrim /x1
ConnectTimeoutSecs 30
</VirtualHost>
<VirtualHost goldengate.domain2.com>
WeblogicCluster green1:4736,green2:4736,green3:4736
PathTrim /y1
ConnectTimeoutSecs 20
</VirtualHost>
```

基于命名的虚拟主机的单 IP 配置范例

```
<VirtualHost 162.99.55.208>
ServerName myserver.mydomain.com
<Location / >
SetHandler weblogic-handler
WebLogicCluster 162.99.55.71:7001,162.99.55.72:7001
```

Idempotent ON

Debug ON

DebugConfigInfo ON

</Location>

</VirtualHost>

<VirtualHost 162.99.55.208>

ServerName myserver.mydomain.com

<Location / >

SetHandler weblogic-handler

WebLogicHost russell

WebLogicPort 7001

Debug ON

DebugConfigInfo ON

</Location>

</VirtualHost>

第十二章 安装和配置 Microsoft-IIS 插件（ISAPI

本章将介绍如何在 Microsoft Internet Information Server 中安装与配置 Microsoft-IIS 插件。

主要介绍以下内容：

Microsoft-IIS 插件概述

Microsoft-IIS 插件安装

iisproxy.ini 文件示例

Microsoft-IIS 插件中使用 SSL 协议

将 servlets 从 IIS 代理到 WebLogic 服务器

安装测试

连接出错和集群容错

Microsoft-IIS 插件概述

通过 Microsoft-IIS 插件允许 Microsoft-IIS 服务器代理对 WebbLogic 服务器的访问请求。通过插件可以允许 WebLogic 服务器处理访问 WebLogic 服务器动态功能得到请求，从而增强了原有 Microsoft-IIS 服务器的功能。

插件用于这样一种环境，其中 Microsoft-IIS 服务器提供静态页面的服务，动态页面的内容（由 HTTPServlets 和 JSP 产生的）转交给工作在另一个进程也可能是在另一个主机中的 Weblogic 服务器。对于终端用户即浏览器而言，转交给 webLogic 服务器的 HTTP 请求是依然是来自相同的源。HTTP-隧道技术部件可以通过插件来运行，使得非流览器用户可以访问 WebLogic 服务器的服务。

连接缓冲池以及 Keep-Alive

通过 IIS 插件到 WebLogic 服务器之间的可重用的连接缓冲池，提高了 Microsoft IIS 服务器到 WebLogic 服务器的访问效率。插件在其与 WebLogic 服务器之间实现了 HTTP1.1 Keep-alive 连接，它对来自相同客户端的一系列请求重用了缓冲池的同一连接。如果一个连接处于非活动状态时间超过 30 秒（或用户定义的时间），连接会被关闭并释放回缓冲池中。可以根据需要禁止该设置，有关信息参见附件中的“Keep-Alive Enabled”

代理请求

插件根据你指定的设置来代理对 WebLogic 服务器的请求，你可以根据请求的 URL（或其中一部分）来代理请求，这种方式称为依据路径代理。你也可以通过被请求的文件的 MIME 类型来代理请求。你还可以将两种方式混合使用，如果一个请求满足两种请求，请求会首先通过路径方式代理，你也可以对这些请求设定附加参数，它定义了插件的一些附加功能，有关信息参见“安装 Microsoft IIS 插件”。

平台支持

关于操作系统以及 IIS 版本与 IIS 插件的兼容性的最新信息，请参阅位于 http://e-docs.bea.com/wls/platforms/index.html#iis 的平台支持，

Microsoft-IIS 插件安装

要安装 Microsoft-IIS 插件：

1. 将 iisproxy.dll 文件从 WebLogic 安装目录下的/lib 目录中复制到一个可以被 IIS 所使用的目录中，而且该目录中应该包含 iisporix.ini 文件

2. 选择 Microsoft IIS 启动菜单，启动 IIS 服务管理器

3. 在服务管理器的左窗格中，选择你的 web 站点（缺省为“缺省 Web 站点”）。

4. 选工具栏中的“Play”（编者注:中文版应该为"启动项目”）箭头来启动站点。

5 在左窗格中，用鼠标右键点选你所选择的 web 站点，打开这个站点的属性设置。

6. 在 Properties （属性）窗格中，选择 Home Directory （编者注:中文版为"主目录"）标签，在 Applications Setting(编者注:中文版为"应用程序设置")部分点 Configuration(编者注:中文版为"配置")按钮。

7. 配置代理文件类型的扩展名

a. 在 App Mapping（应用程序映射）标签页中，点 Add 按钮选择要转交给 WebLogic 服务器处理的文件类型。

b. 在对话框中，找到“iisproxy.dll”文件

c. 设置代理 WebLogic 服务器处理的文件类型的扩展名

d. 不选“Script engine”（编者注:中文版为“脚本引擎”）复选框。

e. 不选“Check that file exists”（编者注:中文版为“检查文件是否存在”）复选框。

f. 设定 “Method exclusions”以符合安装的安全惯例。

g. 完成上述步骤后，点 OK 按扭保存所做的设置，然后对每一种要交给 WebLogic 处理的文件类型都重复上述步骤。

h. 当完成文件类型的配置，点击 OK 按钮关闭属性窗口。

注：在 URL 中，服务器后添加的信息会直接传到 WebLogic，例如你可以通过 IIS 请求一个文件，其 URL 为：

http://myiis.com/jspfiles/myfile.jsp

它将请求代理到 WebLogic 服务器上，其 URL 为：

http://mywebLogic:7001/jspfiles/myfile.jsp

8. 创建 iisproxy.ini 文件

iisproxy.ini 文件包含一对对的“名称 = 值”的设置，它定义了插件的配置参数。参数信息参见“Web Server 插件通用参数”。

注：参数的修改后，直到重起 IIS 后才生效。

BEA 建议将 iisproxy.ini 文件和 iisproxy.dll 存放于同一个目录下，你也可以存放于其他位置，若存放于其他位置，注意 WebLogic 会按照一定的搜索顺序寻找 iisproxy.ini 文件：

a. iisproxy.dll 所在的目录

b. 被 Windows 注册表所引用的 WebLogic 服务器最新版本所在的 home 目录。如果 WebLogic 服务器在这个目录下没有找到 iisproxy.ini 文件，那么它会在 Windows 注册表中所引用老 WebLogic 服务器版本所在的安装目录下查找 iisproxy.ini 文件。

c. c:\weblogic，若该目录存在。

9. 定义 IIS 代理请求的 WebLogic 服务器的主机名和端口号，基于你的设置，有两种方式配置主机名和端口号。

如果代理对一台 WebLogic 服务器的请求，在 iisproxy.ini 文件中定义 WebLogicHost 和 WebLogicPort 参数，例如：

WebLogicHost=localhost

WebLogicPort=7001

如果代理对一 WebLogic 服务器集群的请求，在 iisproxy.ini 文件中定义 WebLogicCluster 参数，例如：

WebLogicCluster=myweblogic.com:7001,yourweblogic.com:7001

其中 myweblogic.com 和 yourweblogic.com 是集群中运行的两个实例。

10. 配置基于路径的代理请求。除了基于文件类型代理请求，你还可以配置 Microsoft-IIS 插件基于路径代理请求。你需要在 iisproxy.ini 文件中指明一些附加参数。基于路径代理请求优先于基于 MIME 类型代理请求。

要配置基于路径的代理：

a. 把 iisforward.dll 放在 iisproxy.dll 所在的目录中，并且在 IIS 中加入 iisfoward.dll 作为过滤器服务 WebSite *Propertie*s → ISAPI Filters → Add the iisforward.dll

b. 把 .wlforward 注册为由 iisproxy.dll 处理的一种特殊文件类型。

c. 在 iisproxy.ini 文件中添加 wlForwardPath 属性。例如：WlForwardPath=/weblogic

d. 如果需要，在 iisproxy.ini 文件加入 PathTrim 参数。

例如，使用

WlForwardPath=/weblogic

PathTrim=/weblogic

它对要转发到 WebLogic 服务器的请求的 URL 进行裁减，因此

/weblogic/session 转变为 /session

e. 如果你不希望请求中包含附加的路径信息（换句话说，就是请求中只包含主机名），设置 DefaultFileName 参数，其值为响应代理请求的 Web 应用的欢迎页面 该参数的值会附加在 URL 后。

f. 如果想对应用进行调试，在 iisproxy.ini 文件中将 Debug 参数设置为 ON，那么会生成一个用于调试的“c:\tmp\iisfoward.log”文件，它记录的插件的动作。

11. 在 iisproxy.ini 文件中设定其他附加参数，详细参数列表请见“Web Server 插件通用参数”

12. 如果不按照基于路径代理的方式，利用 IIS 代理对 WebLogic Servlets 的请求时，参见“将 servlet

从 IIS 代理到 WebLogic”。

iisproxy.ini 文件示例

这是一个非 WebLogic 集群情况下的 iisproxy.ini 文件示例。注释行以‘#’号开始。

```
# This file contains initialization name/value pairs
# for the IIS/WebLogic plug-in.
WebLogicHost=localhost
WebLogicPort=7001
ConnectTimeoutSecs=20
ConnectRetrySecs=2
```

下面是使用了 WebLogic 服务器集群情况下的 iisproxy.ini 文件示例。注释行以‘#’号开始。

```
# This file contains initialization name/value pairs
# for the IIS/WebLogic plug-in.
WebLogicCluster=myweblogic.com:7001,yourweblogic.com:7001
ConnectTimeoutSecs=20
ConnectRetrySecs=2
```

注：若在插件和 WebLogic 服务器之间使用了 SSL，端口号需要设定为 SSL 侦听的端口号。

Microsoft-IIS 插件中使用 SSL 协议

使用安全套接层（SSL）协议可以保护 WebLogic 服务器代理插件以及 IIS 服务器之间的连接，保护在 WebLogic 服务器代理插件与 IIS 服务器之间所传输的数据的机密性与完整性。此外，使用 SSL 协议还可以让 WebLogic 服务器代理插件向 IIS 服务器进行自我身份验证，从而保证信息被传递给可信任原点。

WebLogic 服务器代理插件不会基于传输协议（http 或 https）来确定 IIS 插件与 WebLogic 服务器之间的连接是否使用了 SSL 协议保护。要使用 SSL 协议，必须把 WebLogic 服务器配置为使用 SSL 协议，用以接收代理插件的发送来的请求。WebLogic 服务器插件代理使用 WebLogic 服务器监听 SSL 通信的端口与 IIS 服务器通信。

若在 IIS 和 WebLogic 服务器之间使用 SSL 协议：

1. 为 WebLogic 服务器配置 SSL，详细信息参见“配置 SSL 协议”。
2. 配置 SSL 侦听端口，详细信息参见“配置侦听端口”。
3. 在 iisproxy.ini 文件中，将 WebLogicPort 参数设为步骤 2 中设定的侦听端口。
4. 在 iisproxy.ini 文件中，将 SecureProxy 设为 ON
5. 在 iisproxy.ini 文件中，定义其他 SSL 连接所用参数，有关参数的详细信息，将附件中的“Web Server 插件中的 SSL 参数”。

例如：

```
WebLogicHost=myweblogic.com
```

WebLogicPort=7002

SecureProxy=ON

将 servlets 从 IIS 代理到 WebLogic 服务器

如果把 iisforward.dll 注册成一个过滤器，就可以使用路径方式代理 Servlets 请求。你可以用类似以下格式的 URL 调用 servlet

http://IISserver/weblogic/myServlet

如果没有注册 iisforward.dll 为过滤器，那么必须以扩展名方式代理 servlets 请求。以下是使用扩展方式将 servlet 请求转交给 WebLogic 服务器处理的三个步骤：

1. 在 IIS 注册一种文件类型（扩展名）。IIS 将该类型的文件转发给 WebLogic，详细信息参见“安装 Microsoft-IIS 插件”。

2. 在Web应用中注册相应的servlet　有关注册servlets的详细内容，可以参见“配置Servlets”中的内容。位于：http://e-docs.bea.com/wls/docs61/webapp/components.html#configuring-servlets.

3. 用以下模式的 URL 访问 servlet

http://www.myserver.com/**virtualName**/anyfile.**ext**

virturalName 是在 Web 应用分发描述符（web.xml）中的<servlet-mapping>元素中为 servlet 定义的 URL 模式。 ext 是在 IIS 中注册的文件类型（文件扩展名），这种文件类型由 WebLogic 服务器处理。URL 的 anyfile 部分被忽略。

注：

servlet 调用的所有图形链接作为 Web 应用的一部分，那么必须在 IIS 中为图形文件注册相应类型（例如：.gif 和.jpg）以能够代理。但是，你也可以直接通过 IIS 来提供这些文件请求。

如果被调用的 servlet 调用了其它 servlet，那么这些链接也必须使用上面所提到的 URL 模式。

安装测试

在安装并配置了 Microsoft-IIS 插件后，按以下步骤部署并测试 ISAPI 插件：

1. 确认运行了 WebLogic 与 IIS

2. 在缺省的 Web 应用的文档根目录下保存一个 JSP 文件。

3. 打开浏览器，URL 设置为 IIS+filename.jsp。例如：

http://myii.server.com/*filename*.jsp

如果 filename.jsp 在流览器中显示，证明插件工作正常。

连接错误和集群容错

当 Microsoft IIS 插件试图连接 WebLogic 服务器时，插件会使用几个配置参数，以决定连接 WebLogic 服务器主机的等待时间，在连接建立后，插件等待响应的时间长度等。如果插件不能连接或者没有收到响应，插件会试图连接并发送请求到集群中其他的 WebLogic 服务器。

若连接失败或集群中任何一台 WebLogic 服务器都无响应，则会发出出错信息。

图 12-1 连接容错演示了插件如何实现容错的。

连接失败

如果主机对于连接请求没有响应，则表明问题原因可能是主机、网络故障，或者服务器出错。

如果 WebLogic 服务器没有响应，可能表示 WebLogic 服务器没有运行或已经挂起，数据库出现问题或者应用出错。

单机，非 WebLogic 集群的容错

如果有且只有一台 WebLogic 服务器运行，除非插件只采用 WebLogicHost 定义参数试图连接，否则采用这里所描述的逻辑。若连接失败，会返回 HTTP503 出错信息。插件会继续试图连接 WebLogic 服务器，直至超过 ConnectTimeoutSecs 所定义的时间。

动态服务器列表

如果在 WebLogicCluster 参数中指定了 WebLogic 服务器列表，插件会使用该列表作为成员间负载均衡的起点。在第一次请求路由到一个服务器后，一个包含集群中的更新的服务器动态列表会返回。更新的列表会添加新的服务器，并删除不再属于该集群或不再响应请求的 WebLogic 服务器。当集群中有变化时，该动态列表会通过 HTTP 响应自动更新。

容错、Cookie 和 HTTP 会话

当一个请求包含存储于 cookie 中的会话信息，它可能以 POST 方式或 URL Encoding 方式传递，那么 sessionID 会包含指向会话建立的服务器（主服务器）的引用，以及保存原始会话复制信息的服务器（辅助服务器）。包含 cookie 信息的请求首先会连接主服务器，如果连接失败，那么请求会路由到辅助服务器。如果主服务器和辅助服务器均出错，会话信息会丢失，插件回建立一个到动态服务器列表中的另一个服务器的新连接。详细信息参见“图 12-1 连接容错”。

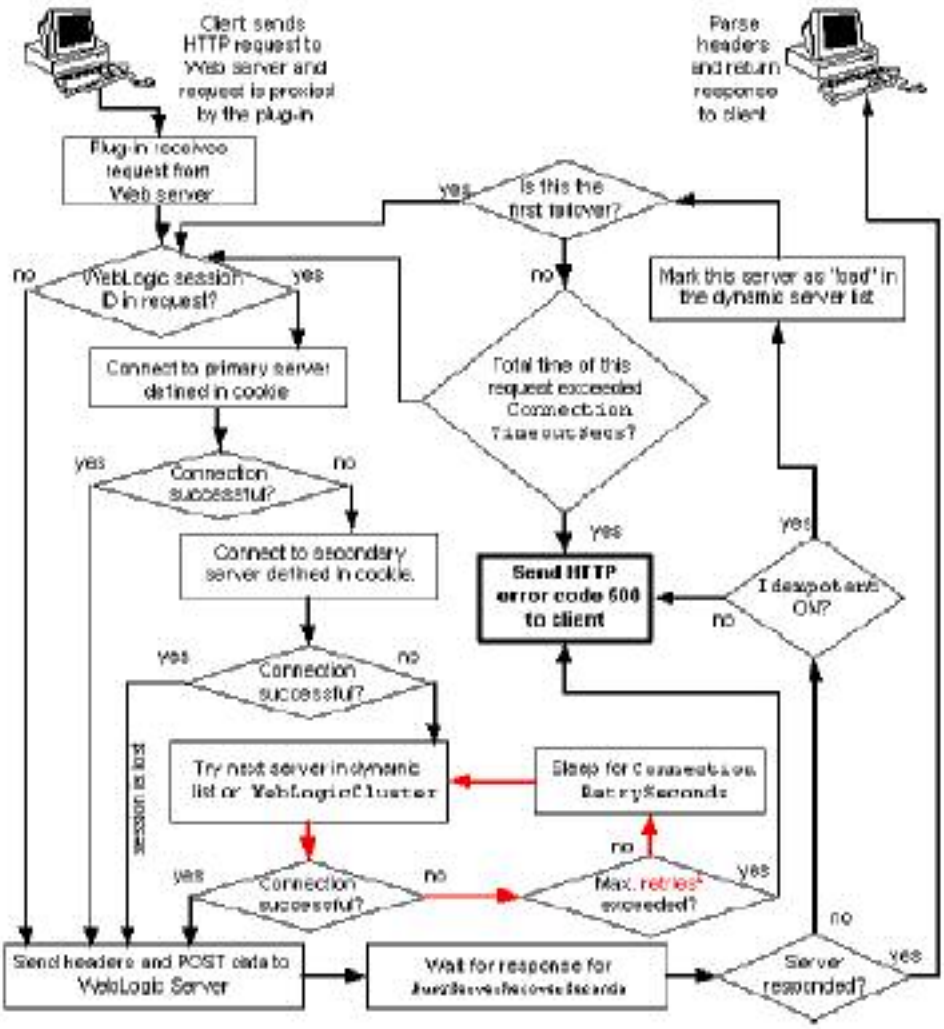


图 12-1 连接容错

* 红色循环中，允许最大重试次数 = ConnectTimeoutSecs÷ConnectRetrySecs

第十三章 安装和配置 Netscape 企业服务器插件（NSAPI

本章将介绍如何配置 Netscape 企业服务器 NES）代理插件。

主要介绍以下内容：

Netscape 企业服务器（NES）插件概述

安装和配置 Netscape 企业服务器（NES）插件

在 NSAPI 插件中使用 SSL

连接出错和集群容错

在使用放火墙和负载指示器时的容错处理

obj.con 文件示例（非 WebLogic 集群）

obj.con 文件示例（使用 WebLogic 集群）

Netscape 企业服务器（NES）插件概述

NES 插件允许 Netscape 企业服务器代理对 WebLogic 服务器的请求，插件通过 WebLogic 处理访问 WebLogic 服务器的动态资源，而提高了 Netscape 企业服务器的性能。

Netscape 企业服务器插件用于这样的环境之中，其中 Netscape 企业服务器提供静态网页服务，而 WebLogic 服务器（运行在另一个进程之中，也可能是另一主机之中）则负责提供对动态页面的服务（例如 JSP 以及 HTTP Servlet 生成的页面）。WebLogic 服务器与 NSAPI 插件之间的通信可以是明文的也可以使用 SSL。对于终端用户即浏览器而言，提交给 WebLogic 服务器的 HTTP 请求与静态网页来自相同的源。而且 WebLogic 服务器 HTTP 隧道功能也可以通过插件运行，因此，通过它可以访问所有 WebLogic 服务。

Netscape 企业服务器插件在 Netscape 企业服务器中被当作一个 NSAPI 模块运行（见：http://home.netscape.com/servers/index.html NES 在启动时会装载 NSAPI 模块，然后某些类型的 HTTP 请求就交由该模块处理。NSAPI 类似于 HTTP(Java) servlet，只是 NSAPI 模块是采用与平台有关的代码编写。

有关 Netscape 企业服务器和 iPlanet 服务器版本支持信息，参见 BEA WebLogic 服务器平台支持，位于：http://e-docs.bea.com/wls/platforms/index.html#plugin

连接缓冲池和 Keep-Alive

通过 NES 插件到 WebLogic 服务器间的可重用的连接缓冲池，提高了 NES 插件到 WebLogic 服务器的访问效率。NES 插件自动实现了在其与 WebLogic 服务器之间 Keep-alive 连接。如果一个连接处于非活动状态时间超过 30 秒（或用户定义的时间），连接会被关闭并释放回缓冲池中。可以根据需要禁止该设置，有关信息参见附件中的“Keep-Alive Enabled”

代理请求

插件根据你指定的设置来代理对 WebLogic 服务器的请求，你可以根据请求的 URL（或其中一部分）来代理请求，这种方式称为依据路径代理。你也可以通过被请求的文件的 MIME 类型

来代理请求。你还可以将两种方式混合使用，如果一个请求满足两种请求，请求会首先通过路径方式代理，你也可以对这些请求设定附加参数，它定义了插件的一些附加功能，有关信息参见下节。

安装和配置 Netscape 企业服务器（NES）插件

安装和配置 NES 插件需要：

1. 复制库

   在 UNIX 平台上 WebLogic NSAPI 插件模块作为共享对象（.so 文件）分发，在 Windows 平台上作为动态连接库分发（.dll 文件）。这些文件分别位于 WebLogic 软件的/lib 或/bin 目录中。

   从平台支持表（http://e-docs.bea.com/wls/platforms/index.html#plugin 中选择与你环境相应的库文件。然后将所选择的库文件复制到 NES 所在的文件系统中。

2. 修改 obj.conf 文件，obj.conf 文件定义了代理哪些对 WebLogic 的访问请求以及其他的一些配置信息。详细信息参见“修改 obj.conf 文件”。

3. 若基于 MIME 类型代理请求

   a. 在 obj.conf 文件添加相应的行，详细信息参见“修改 obj.conf 文件”。

   b. 在 obj.conf 文件中添加指向 MIME.types 文件的 MIME 类型引用。可以 NES 的管理控制台或者直接编辑 MIME.types 文件。

      要直接编辑 MIME.types 文件，打开文件，并输入如下的行：

      type=text/jsp exts=jsp

      注意：对于 NES4.0 iPlanet），除了添加对 JSP 访问的 MIME 类型，还需要修改已存在的 MIME 类型：

      将 magnus-internal/jsp

      改为 text/jsp

      如果使用 NES 管理控制台，则选择 Manage Preferences → MIME Types，然后进行修改或添加操作。

4. 部署和测试 NES 插件

   a. 启动 WebLogic 服务器

   b. 启动 Netsacpe 企业服务器。如果 NES 已经启动，你或者必须重新启动或者起用通过控制台所进行的修改，以便让新的设置生效。

   c. 要测试 NES 插件，打开流览器，将 URL 设为 NES + /weblogic/，这将会调用 WebLogic 服务器缺省 Web 应用中的缺省的 WebLogic 服务器 HTML 页面，（欢迎页面或缺省的 servlet）。例如：

      http://myenterprise.server.com/weblogic/

修改 obj.conf 文件

为了使用 NES 插件，你需要对 NES 的 obj.conf 文件做一些修改。这些修改指明了请求是如何被代理到 WebLogic 服务器上。你可以基于 URL 或者 MIME 类型来代理请求。关于每种方式的

设置过程将在下面章节讲述。

Netscape 的 obj.conf 文件对文本中的空格要求很严格。为了避免出现问题，请注意下列关于 obj.conf 文件的要求：

去除无关的前导与尾部的空格，多余的空格会让 NES 出错。

如果所输入的字符必须跨越多行，那么在行的末尾加上反斜杠（\），然后继续在下一行输入。反斜杠会直接把前后两行连接在一起。如果要在第一行的结尾与第二行的开始之间必须要有一个空格的话，务必只使用一个空格，或者是在第一行的末尾（在反斜杠之前）或者是在第二行的开始之前。

属性不能跨越多行。（例如，你必须在 WebLogicCluster 参数后面列出集群中的所有的服务器）

若配置中丢失了所需要的参数，当对象调用时，它会产生一个 HTML 错误，并注明配置中所丢失的参数名称。

要配置 obg.conf 文件：

1. 找到并打开 obj.conf 文件

用于 NES 实例的 obj.conf 文件位于以下位置：

NETSCAPE_HOME/https-INSTANCE_NAME/config/obj.conf

其中，*NETSCAPE_HOME* 是安装 NES 的根目录，*INSTANCE_HOME* 是正在使用的“特定实例”或服务器配置。例如，在一台叫作 myunixmachine 的 UNIX 机器中，你可以在以下位置找到 obj.conf 文件：

/usr/local/netscape/enterprise-351/https-myunixmachine/config/obj.conf

2. 指示 NES 将本地库装载为一个 NSAPI 模块

在 obj.conf 文件的开头加入以下代码行。这些行指示 NES 应该把本地库（共享对象或 dll）装载为一个 NSAPI 模块。

```
Init fn="load-modules" funcs="wl-proxy,wl-init"\
shlib=/usr/local/netscape/plugins/SHARED_LIBRARY
Init fn="wl-init"
```

其中 *SHARED_LIBRARY* 是你在步骤 1 中所安装的共享对象或 dll（例如 libproxy.so）。功能“load-modules”表明在 NES 启动时要装载的共享库。值“wl-proxy”与“wl-init”标识了 NSAPI 插件将执行的功能

3. 若通过 URL 代理请求（基于路径方式代理），那么对每一个要代理的 URL 分别创建一个<Object>标记。它定义了 PathTrim 参数（基于 MIME 类型的代理参见步骤 4）。基于路径的代理要优先于基于 MIME 类型的代理。下面是一个<Object>标记，它定义了代理包含字符串*/weblogic/*的请求。

要建立一个<Object>标记来通过 URL 代理请求：

a. 在<Object>标记开头处，利用 name 属性指明该对象（可选）。name 属性只是表示信息，它并不被 NES 插件使用。例如：

< Object name=myObject … >

b. 在<Object>标记中，通过 ppath 属性指明所代理的 URL。例如：

<Object name=myObject ppath="*/weblogic/*">

ppath 属性值可以是表示访问 WebLogic 服务器请求的任何字符串。当使用 ppath，每个请求（包含该路径）都进行转发。如：一个为 */weblogic/* 的 ppath，对于http://enterprise.com/weblogic的每个请求都会转发给 NES 插件，并由它发送请求到指定的 WebLogic 主机或集群。

c. 在<Object>和</Object>标记间添加 Service 指令。在 Service 指令中，你可以通过“名称=值”的方式指明有效的参数。在多个“名称=值”之间，以一个且最多最多一个空格分隔。例如：

Service fn=wl_proxy WebLogicHost=myserver.com\
WebLogicPort=7001 PathTrim="/weblogic"

有关参数的详细列表，参见附件中的“Web Server插件通用参数”。你必须指定下列参数：

对非集群的WebLogic服务器：

WebLogicHost 和 WebLogicPort 参数

对集群的WebLogic服务器：

WebLogicCluster参数

Service指令永远以fn=wl_proxy开始，其后添加有效的“名称=值”表示的参数。

下面的这个例子在 obj.conf 文件中添加了两个对象定义，它们分别定义了两个 ppath。这两个 ppath 标识了交由不同 WebLogic 服务器实例处理的请求。

<Object name="weblogic" ppath="*/weblogic/*">
Service fn=wl-proxy WebLogicHost=myserver.com\
WebLogicPort=7001 PathTrim="/weblogic"
</Object>
<Object name="si" ppath="*/servletimages/*">
Service fn=wl-proxy WebLogicHost=otherserver.com\
WebLogicPort=7008
</Object>

注：不是必须配置的参数，如PathTrim，可以用来更进一步的配置ppath的传送给NES插件的方式。有关参数详细信息，参见“Web Server插件通用参数列表”。

4. 如果使用基于MIME类型来代理请求，MIME类型必须列在MIME.types文件中。有关修改该文件的指令，参见“安装和配置Netscape 企业服务器（NES）插件”.

所有指定MIME类型的请求（如：.jsp），将会转发给WebLogic服务器，而不管其URL为何种形式。

要代理WebLogic服务器对某一类型文件的所有请求：

a. 在现存的defaultObject定义（<Object name=default ... >）中，添加Service指令。

例如：要将所有jsp文件转发给WebLogic，下面的Service指令应该添加在以开始NameTrans fn= .... 的行结尾。同时在以开始 PathCheck 的行之前。

Service method="(GET|HEAD|POST|PUT)" type=text/jsp fn=wl-proxy\

WebLogicHost=192.1.1.4 WebLogicPort=7001 PathPrepend=/jspfiles

该 Service 指令语句将把所有以 .jsp 为扩展名的文件代理到所指定的 WebLogic 服务器，其请求的 URL 类似于：

http://WebLogic:7001/jspfiles/myfile.jsp

pathPrepend 的值应该对应于 Web 应用上下文的文档根目录，该应用部署于被代理的一个 WebLogic 服务器或一个 WebLogic 集群上。

在增加了对NES插件所有条目的语句后，缺省的Object 定义类似于下面，所添加的部分用**黑体**显示。

<Object name=default>

NameTrans fn=pfx2dir from=/ns-icons\

dir="c:/Netscape/SuiteSpot/ns-icons"

NameTrans fn=pfx2dir from=/mc-icons\

dir="c:/Netscape/SuiteSpot/ns-icons"

NameTrans fn="pfx2dir" from="/help" dir=\

"c:/Netscape/SuiteSpot/manual/https/ug"

NameTrans fn=document-root root="c:/Netscape/SuiteSpot/docs"

Service method="(GET|HEAD|POST|PUT)" type=text/jsp fn=wl-proxy\

WebLogicHost=localhost WebLogicPort=7001 PathPrepend=/jspfiles

PathCheck fn=nt-uri-clean

PathCheck fn="check-acl" acl="default"

PathCheck fn=find-pathinfo

PathCheck fn=find-index index-names="index.html,home.html"

ObjectType fn=type-by-extension

ObjectType fn=force-type type=text/plain

Service method=(GET|HEAD) type=magnus-internal/imagemap fn=imagemap

Service method=(GET|HEAD) \

type=magnus-internal/directory fn=index-common

Service method=(GET|HEAD) type=*~magnus-internal/* fn=send-file

AddLog fn=flex-log name="access"

</Object>

b. 所有要转交给WebLogic服务器处理的MIME类型都应该以上面这种方式，在缺省对象中添加相似的Service语句。

5. 若需要使用 HTTP 隧道功能

你应该在 obj.conf 文件中添加以下对象定义，表示希望使用 WebLogic 主机和端口,或者 WebLogic 集群名来处理关于 HTTP 隧道的请求。

<Object name="tunnel" ppath="*/HTTPClnt*">

Service fn=wl-proxy WebLogicHost=192.192.1.4 WebLogicPort=7001

</Object>

在 NSAPI 插件中使用 SSL

使用安全套接层（SSL）协议可以保护 WebLogic 服务器代理插件以及 NSAPI 服务器之间的连接，保护在 WebLogic 服务器代理插件与 NSAPI 服务器之间所传输的数据的机密性与完整性。此外，使用 SSL 协议还可以让 WebLogic 服务器代理插件向 NSAPI 服务器进行自我身份验证，从而保证信息被传递回可信任原点。

WebLogic 服务器代理插件不会基于传输协议（http 或 https）来确定 NES 插件与 WebLogic 服务器之间的连接是否使用了 SSL 协议保护。

若在 NES 插件和 WebLogic 服务器之间使用 SSL 协议：

1. 为 WebLogic 服务器配置 SSL，详细信息参见“配置 SSL 协议”。
2. 配置 SSL 侦听端口，详细信息参见“配置侦听端口”。
3. 在 obj.conf 文件中，将 WebLogicPort 参数设为步骤 2 中设定的侦听端口。
4. 在 obj.conf 文件的 Service 指令中，将 SecureProxy 设为 ON
5. 在 obj.conf 文件中，定义其他 SSL 连接所用参数，有关参数的详细信息，将附件中的“Web Server 插件中的 SSL 参数”。

连接错误和集群容错

当 Netscape 企业服务器（NES）插件试图连接 WebLogic 服务器时，插件会使用几个配置参数，以决定连接 WebLogic 服务器主机的等待时间，在连接建立后，插件等待响应的时间长度等。如果插件不能连接或者没有收到响应，插件会试图连接并发送请求到集群中其他的 WebLogic 服务器。

若连接失败或集群中任何一台 WebLogic 服务器都无响应，则会发出出错信息。

图 13-1 连接容错演示了插件如何实现容错的。

连接失败

如果主机对于连接请求没有响应，则表明问题原因可能是主机、网络故障，或者服务器出错。

如果 WebLogic 服务器没有响应，可能表示 WebLogic 服务器没有运行或已经挂起，数据库出现问题或者应用出错。

单机，非 WebLogic 集群的容错

如果有且只有一台 WebLogic 服务器运行，除非插件只采用 WebLogicHost 定义参数试图连接，否则采用这里所描述的逻辑。若连接失败，会返回 HTTP503 出错信息。插件会继续试图连接 WebLogic 服务器，直至超过 ConnectTimeoutSecs 所定义的时间。

动态服务器列表

如果在 WebLogicCluster 参数中指定了 WebLogic 服务器列表，插件会使用该列表作为成员间负载均衡的起点。在第一次请求路由到一个服务器后，一个包含集群中的更新的服务器动态列表会返回。更新的列表会添加新的服务器，并删除不再属于该集群或不再响应请求的 WebLogic 服务器。当集群中有变化时，该动态列表会通过 HTTP 响应自动更新。

容错、Cookie 和 HTTP 会话

当一个请求包含存储于 cookie 中的会话信息，它可能以 POST 方式或 URL Encoding 方式传递，那么 sessionID 会包含指向会话建立的服务器（主服务器）的引用，以及保存原始会话复制信息的服务器（辅助服务器）。包含 cookie 信息的请求首先会连接主服务器，如果连接失败，那么请求会路由到辅助服务器。如果主服务器和辅助服务器均出错，会话信息会丢失，插件回建立一个到动态服务器列表中的另一个服务器的新连接。详细信息参见“图 13-1 连接容错”。

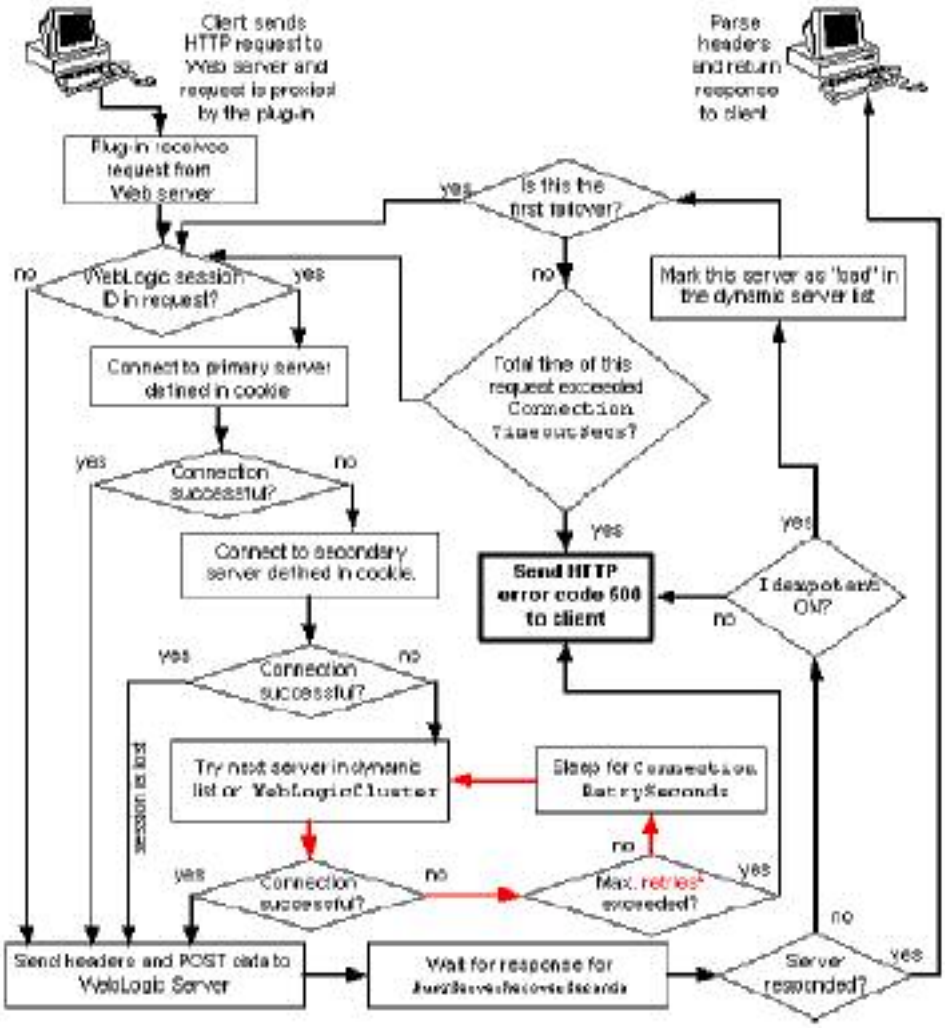


图 13-1 连接容错

* 红色循环中，允许最大重试次数 = ConnectTimeoutSecs÷ConnectRetrySecs

在使用放火墙和负载指示器时的容错处理

在大多数配置中，如果 NSAPI 插件把请求发送到集群的主实例，而该实例不可用时，那么请求会转移到辅助实例。然而，在一些综合使用了防火墙与负载指示器的系统中，那么当 WebLogic 服务器的主实例不可用时，其中的任何一个服务器（防火墙或负载指示器）都可能会接受这个请求（并返回一个成功的连接）。在尝试着把请求转交到 WebLogic 服务器的主实例之前（该实例不可用），请求会作为“连接重设”返回到 NSAPI 插件中。

穿越防火墙（包括或不包括负载指示器）的 NSAPI 请求将由 WebLogic 服务器来处理。换句话说，“连接重设”的响应将转移到 WebLogic 服务器的辅助实例。因为在这些配置中“连接重设”响应可以进行容错处理，因此 servlets 必须是幂等的否则将导致事务的重复处理。

obj.conf 文件示例（非 WebLogic 集群）

下面这个例子演示了在非集群的情况下需要在 obj.conf 文件加入的行。你可以把这个例子作为一个模板，然后根据所使用的环境与服务器对这个文件作适当的修改。以‘#’开头的行都是注释行。

注：确认你没有在 obj.conf 文件中添加了多余的空格，从例子中复制和粘帖，有时会增加出多余的空格，这会在读取文件时产生错误。

你可以从 NetSacpe 企业服务器插件文档中阅读有关企业服务器配置的内容。

```
## ---------- BEGIN SAMPLE OBJ.CONF CONFIGURATION ---------
# (no cluster)
# The following line locates the NSAPI library for loading at
# startup, and identifies which functions within the library are
# NSAPI functions. Verify the path to the library (the value
# of the shlib=<...> parameter) and that the file is
# readable, or the server fails to start.
Init fn="load-modules" funcs="wl-proxy,wl-init"\
shlib=/usr/local/netscape/plugins/libproxy.so
Init fn="wl-init"
# Configure which types of HTTP requests should be handled by the
# NSAPI module (and, in turn, by WebLogic). This is done
# with one or more "<Object>" tags as shown below.
# Here we configure the NSAPI module to pass requests for
# "/weblogic" to a WebLogic Server listening at port 7001 on
# the host myweblogic.server.com.
<Object name="weblogic" ppath="*/weblogic/*">
Service fn=wl-proxy WebLogicHost=myweblogic.server.com\
WebLogicPort=7001 PathTrim="/weblogic"
</Object>
# Here we configure the plug-in so that requests that
# match "/servletimages/" is handled by the
# plug-in/WebLogic.
<Object name="si" ppath="*/servletimages/*">
Service fn=wl-proxy WebLogicHost=192.192.1.4 WebLogicPort=7001
</Object>
# This Object directive works by file extension rather than
```

```
# request path. To use this configuration, you must also add
# a line to the mime.types file:
#
# type=text/jsp exts=jsp
#
# This configuration means that any file with the extension
# ".jsp" are proxied to WebLogic. Then you must add the
# Service line for this extension to the Object "default",
# which should already exist in your obj.conf file:
<Object name=default>
NameTrans fn=pfx2dir from=/ns-icons\
dir="c:/Netscape/SuiteSpot/ns-icons"
NameTrans fn=pfx2dir from=/mc-icons\
dir="c:/Netscape/SuiteSpot/ns-icons"
NameTrans fn="pfx2dir" from="/help" dir=\
"c:/Netscape/SuiteSpot/manual/https/ug"
NameTrans fn=document-root root="c:/Netscape/SuiteSpot/docs"
Service method="(GET|HEAD|POST|PUT)" type=text/jsp fn=wl-proxy\
WebLogicHost=localhost WebLogicPort=7001 PathPrepend=/jspfiles
PathCheck fn=nt-uri-clean
PathCheck fn="check-acl" acl="default"
PathCheck fn=find-pathinfo
PathCheck fn=find-index index-names="index.html,home.html"
ObjectType fn=type-by-extension
ObjectType fn=force-type type=text/plain
Service method=(GET|HEAD) type=magnus-internal/imagemap\
fn=imagemap
Service method=(GET|HEAD) \
type=magnus-internal/directory fn=index-common
Service method=(GET|HEAD) type=*~magnus-internal/* fn=send-file
AddLog fn=flex-log name="access"
</Object>
# The following directive enables HTTP-tunneling of the
# WebLogic protocol through the NSAPI plug-in.
<Object name="tunnel" ppath="*/HTTPClnt*">
```

```
Service fn=wl-proxy WebLogicHost=192.192.1.4 WebLogicPort=7001
</Object>
#
## ------------ END SAMPLE OBJ.CONF CONFIGURATION ---------
```

obj.conf 文件（使用 WebLogic 集群）

下面这个例子演示了在使用 WebLogic 集群时需要在 obj.conf 文件中所要添加的行。你可以把这个例子作为一个模板，然后对它作适当的修改以适合你所使用的环境与服务器。以‘#’号开始的行都是注释行。

注：确认你没有在 obj.conf 文件中添加了多余的空格，从例子中复制和粘帖，有时会增加出多余的空格，这会在读取文件时产生错误。

```
## ------------ 开始示例 OBJ.CONF 配置 ---------
# (使用 WebLogic 集群)
#
# 下面的行定位 NSAPI 库用于
# 启动时载入, and identifies which functions within the library are
# NSAPI functions. Verify the path to the library (the value
# of the shlib=<...> parameter) and that the file is
# readable, or the server fails to start.
Init fn="load-modules" funcs="wl-proxy,wl-init"\
shlib=/usr/local/netscape/plugins/libproxy.so
Init fn="wl-init"
# Configure which types of HTTP requests should be handled by the
# NSAPI module (and, in turn, by WebLogic). This is done
# with one or more "<Object>" tags as shown below.
# Here we configure the NSAPI module to pass requests for
# "/weblogic" to a cluster of WebLogic Servers.
<Object name="weblogic" ppath="*/weblogic/*">
Service fn=wl-proxy \
WebLogicCluster="myweblogic.com:7001,yourweblogic.com:7001,\
theirweblogic.com:7001" PathTrim="/weblogic"
</Object>
# Here we configure the plug-in so that requests that
# match "/servletimages/" are handled by the
# plug-in/WebLogic.
```

```
<Object name="si" ppath="*/servletimages/*">
Service fn=wl-proxy \
WebLogicCluster="myweblogic.com:7001,yourweblogic.com:7001,\
theirweblogic.com:7001"
</Object>
# This Object directive works by file extension rather than
# request path. To use this configuration, you must also add
# a line to the mime.types file:
#
# type=text/jsp exts=jsp
#
# This configuration means that any file with the extension
# ".jsp" is proxied to WebLogic. Then you must add the
# Service line for this extension to the Object "default",
# which should already exist in your obj.conf file:
<Object name=default>
NameTrans fn=pfx2dir from=/ns-icons\
dir="c:/Netscape/SuiteSpot/ns-icons"
NameTrans fn=pfx2dir from=/mc-icons\
dir="c:/Netscape/SuiteSpot/ns-icons"
NameTrans fn="pfx2dir" from="/help" dir=\
"c:/Netscape/SuiteSpot/manual/https/ug"
NameTrans fn=document-root root="c:/Netscape/SuiteSpot/docs"
Service method="(GET|HEAD|POST|PUT)" type=text/jsp fn=wl-proxy\
WebLogicCluster="myweblogic.com:7001,yourweblogic.com:7001,\
theirweblogic.com:7001",PathPrepend=/jspfiles
PathCheck fn=nt-uri-clean
PathCheck fn="check-acl" acl="default"
PathCheck fn=find-pathinfo
PathCheck fn=find-index index-names="index.html,home.html"
ObjectType fn=type-by-extension
ObjectType fn=force-type type=text/plain
Service method=(GET|HEAD) type=magnus-internal/imagemap\
fn=imagemap
Service method=(GET|HEAD) \
```

```
type=magnus-internal/directory fn=index-common
Service method=(GET|HEAD) type=*~magnus-internal/* fn=send-file
AddLog fn=flex-log name="access"
</Object>
# The following directive enables HTTP-tunneling of the
# WebLogic protocol through the NSAPI plug-in.
<Object name="tunnel" ppath="*/HTTPClnt*">
Service fn=wl-proxy WebLogicCluster="myweblogic.com:7001,\
yourweblogic.com:7001,theirweblogic.com:7001"
</Object>
#
## ------------- END SAMPLE OBJ.CONF CONFIGURATION ---------
```

第十四章 安全管理

本章将介绍以下内容：

安全配置步骤

改变系统口令

指定一个安全域

定义用户

定义组

定义 ACL

配置 SSL 协议

配置双向验证

配置基于 IIOP 上的 RMI 的 SSL

口令的保护

安装审计提供者

安装连接过滤器

设置 Java 安全管理器

配置安全上下文传播

安全配置步骤

在 WebLogic 服务器中，安全管理主要通过配置定义安全策略的属性来实现。可以用管理控制台定义安全策略。在管理控制台中，你应该为所分发的应用指定设置与安全相关的属性：

域

用户与组

访问控制列表（ACLs）以及对 WebLogic 服务器资源的访问权限

SSL 协议

双向认证

主机名验证器

审计提供者

定制过滤器

安全上下文传播

因为各安全部件之间是紧密关联的，因此，在进行安全配置时，很难确定从哪开始。事实上，安全配置是一个重复的过程。尽管在进行安全配置时，可能会有很多种流程，但我们还是建议你遵照以下步骤进行：

1．改变 system 用户的口令以保护 WebLogic 服务器，见“修改系统口令”。

2．指定一个安全域。WebLogic 服务器有一个缺省的 File 安全域。但你可能更愿意用别的安全域

或者是一个定制的安全域，见“指定安全域”。

3．定义安全域的用户。你可以在安全域中用组来组织用户，见“定义用户”。

4．定义 ACL 以及对资源的访问权限，见“定义 ACLs”.

5．客户端与 WebLogic 服务器之间的通信采用 SSL 协议，这样可以保护网络连接。当使用 SSL 协议，WebLogic 服务器会使用由可靠的证书认证机构所发放的数字证书对客户进行验证。这一步是可选的，但我们建议你实施这一步，见“配置 SSL 协议”。

6．通过实施双向认证进一步保护你的 WebLogic 服务器。当使用了双向验证， WebLogic 服务器在对客户端进行验证，然后客户端会对 WebLogic 服务器进行验证，这个步骤也是可选的，但 BEA 建议你采用，见“配置双向验”。

本章将介绍上述安全配置步骤，以及如何在管理控制台中设置那些与安全相关的属性。有关 WebLogic 服务器安全特征的详细描述，请参见“WebLogic 安全介绍”与“安全基础”。

注意：本章所描述的所有配置步骤都是在管理控制台中进行的。

有关如何为 WebLogic EJB 分配安全角色的内容，可以参见“WebLogic Server 6.0 的分发属性”中的内容。

有关 web 应用安全的内容，可以参见“分发与配置 Web 应用”中的内容

改变系统口令

在安装 WebLogic 服务器时，安装程序会要求你指定系统用户的口令。该口令是 WebLogic 服务器中 system 用户的口令，被存储在\wlserver6.1\config\mydomain 目录中的 fileRealm.properties 文件中。这个口令可以用于这个域的管理服务器以及所有与这个管理服务器关联的受管服务器。

注意：WebLogic 服务器只能用 system 用户来启动。

保存在 fileReamln.properties 文件中的口令是经过加密的。WebLogic 服务器对被加密的口令进行散列化处理，从而进一步保护了口令。为了提高安全性，我们建议你经常更新系统口令。每个部署的 WebLogic 服务器必需有唯一的口令。

以下是更改系统口令的步骤：

1．在管理控制台中打开 Users 窗口

2．在 User 属性中输入 system

3．在 Password 属性中输入一个新的口令

4．确认你输入的口令

在一个域中，所有受管服务器的口令与管理服务器的口令相同。你应该用管理控制台经常地改变管理服务器的口令，新口令会传播到同一域中的所有受管服务器上。记住，一个域中的所有服务器成员的系统口令必须相同。

注意：Petstore 以及 ExampleServer 域仍然把系统口令保存在 password.ini 文件中。当使用这些域时，你可以通过修改 password.ini 文件中的口令信息来修改 examples 服务器的系统口令。Password.ini 文件中的口令是以明文形式保存的。

保持 WebLogic 服务器的口令不公开，是你部署 WebLogic 服务器和数据安全的关键。为了保

护你应用的安全，BEA 建议你不要公开 WebLogic 服务器的口令。

配置一个安全域

本节描述配置 WebLogic 应用部署的安全域，关于安全域的介绍以及如何使用，参见“WebLogic 安全编程指南”中的“安全域”。下面介绍如何指定安全域：

- 配置文件域
- 配置缓存域
- 配置 LDAP 安全域
- 配置 Windows NT 安全域
- 配置 UNIX 安全域
- 配置 RDBMS 安全域
- 安装自定义的安全域
- 安全域迁移

配置文件域

缺省情况下，安装完 WebLogic 服务器后，WebLogic 服务器就有一个 File 域（File realm）。在使用这个域之前，需要先设置几个属性来管理 File 域的使用。可以在管理控制台的 Security 窗口的 filerealm 标签中设置这些属性。下表是这些属性的描述。

表 14-1 File 域的属性

属性	描述
Cache Realm	所使用的缓存域（Caching realm）的名字。 ● 当使用 File 域时，该属性必须设置为 None ● 若使用其他或自定义的安全域，将该属性设置为缓存域所用的名字，然后一组可用的安全域会显示在下拉菜单中。
Max Users	该属性设置了 File 域的最大用户数。File 域的最大用户数为 10,000，最小为 1，缺省为 1,000
Max Groups	该属性设置了 File 域的最大组数，最大值为 10,000，最小为 1，缺省为 1,000
Max ACLs	指定 File 域中最多可以有多少 ACL，最小设置为 1，最大为 10,000，缺省为 1,000

注意： 如果 fileRealm.properties 文件被破坏，那么你需要重新配置 WebLogic 服务器的所有安全信息。因此，我们建议你完成以下任务：

对 fileRealm.properties 文件进行备份，然后把备份放在一个安全的地方。

设置 fileRealm.properties 文件的访问权限，只有 WebLogic 服务器管理员才有该文件的读写该权限，其它用户不能读写这个文件。

注：你还应该备份 File 域的 SerializedSystemIni.dat 文件。有关 SerializedSystemIni.dat 文件的更多内容，你可以参见“口令的保护”中的内容。

如果你不想使用 File 域，而是想使用 WebLogic 服务器提供的其它安全域或者是定制安全域，那么在对所用的安全域设置了上述属性后，重启服务器使你的设置生效。若使用其他的安全

域，则必需启用缓存域。

有关 WebLogic 服务器安全域的更多内容，你可以参见“安全域”中的内容。

配置缓存域

缓存域（Caching realm）与 File 域、WebLogic 服务器的其它安全域以及定制安全域一起实现对客户端请求的验证与授权。无论是成功的还是失败的域搜索，缓存域都将保存搜索结果。缓存域要管理以下内容的缓存：用户，组，权限，ACL 以及验证请求。缓存域通过缓存搜索结果，减少对其它安全域的调用次数，从而提高了 WebLogic 服务器的性能。有关 WebLogic 服务器安全域的更多内容，你可以参加“安全域”中的内容。

安装 WebLogic 服务器时会自动安装缓存域：缓存域被设置为其他安全域的代表，但不启用缓存。 你可以用管理控制台启用缓存。如果使用其他的安全域不是文件域，那么必须启用缓存域。

启用缓存域后，缓存域会把对一个域的搜索结果保存在它的缓冲区中，所保存的时间由存活期（time-to-live，简写为 TTL）属性的值而定或者看缓存区是否已经满了。如果缓存区已经满了，新的搜索结果将替换最老的搜索结果。TTL 属性决定了一个缓存对象的有效时间。你把这些值设得越高，缓存区调用从其他安全域（secondary realm）的次数越少。减少调用的频率会提高性能，代价是对从安全域的改变只有等到缓存对象过期了才能被识别。

注意：当你从安全域获得一个对象时，所获得的对象只是这个对象的一个快照。因此，要更新对象，必须再次调用 get()方法。例如，当你调用 getGroup()方法从安全域获得一个组时，你设置了这个组的成员关系，如果要更新组的成员，你必须再次调用 getGroup()方法。

缺省情况下，缓存域认为从安全域是区分大小写的。例如，在区分大小的安全域中，用户名 bill 与 Bill 代表不同的用户。Windows NT 安全域以及 LDAP 安全域不区分大小。如果使用一个不区分大小写的安全域，那么必须禁用 CacheCaseSensitive 属性。在禁用这个属性后，缓存域把所有用户名都转换为小写字母，这样 WebLogic 服务器在执行区分大小写的比较操作时就能得出正确的结果。在区分大小写的安全域中定义或引用用户或组时，在输入用户名时应该用小写字母。

要配置缓存域：

1. 在管理控制台左侧窗格中选择 Security →Caching Realms 节点。
2. 在管理控制台右侧窗格中，点击 New Caching Realm 链接
3. 在缓存域窗口的 General 标签页中定义属性

下表描述了 General 标签页上的各个属性。

表 14-2 缓存域的配置属性

属性	描述
Name	显示了活动的安全域。不要改变这个属性。
Basic Realm	与缓存域一同工作的其他安全域或定制安全域的类名，可配置的域名以下拉菜单方式显示。
Case Sensitive Cache	所指定的安全域是否区分大小写。缺省情况下，该属性是开启的，即域是区分大小写的。如果要使用一个不区分大小写的域（例如 Windows NT 与 LDAP 安全域），应该禁用这个属性。

4. 点击 Create

5. 要启用或配置 ACL 缓存，需要定义 Caching Realm configuration 窗口的 ACL 标签页上的属性。

下表描述了 ACL 标签页上的各个属性。

表 14-3 ACL 缓存的配置属性

属性	描述
Enable ACL Cache	是否开启 ACL 缓存
ACL Cache Size	指定最多缓存多少 ACL 查找结果。缺省为 211　该属性应设置为一个质数以获得最好的查找性能。
ACL Cache Positive TTL	设置一个成功的查找结果保存在缓存中的时间。缺省为 60 秒。
ACL Cache Negative TTL	设置一个失败的查找结果保存在缓存中的时间。缺省为 10 秒

6. 要保存所作的设置，点 Apply 按钮。

7. 要启用或配置 Authentication 缓存，你需要定义 Caching Realm configuration 窗口的 Authentication 标签页上的那些属性。

下表描述了 Authentication 标签页上的各个属性。

表 14-4 Authentication 缓存的配置属性

属性	描述
Enable Authentication Cache	该选项用于开启 Authentication 缓存
Authentication Cache Size	指定最多能缓存多少 Authentication 请求。缺省为 211。该属性应该设置为一个质数以获得最好的查找性能。
Authentication Cache TTL Positive	设置一个成功的查找结果保存在缓存中的时间。缺省为 60 秒。
Authentication Cache TTL Negative	设置一个失败的查找结果保存在缓存中的时间。缺省为 10 秒

8. 要保存所做的设置，点 Apply 按钮.

9. 要启用或配置 Group 缓存，需要定义 Caching Realm configuration 窗口的 Group 标签页上的那些属性。

下表描述了 Group 标签页上的各个属性。

表 14-5 Group 缓存的配置属性

属性	描述
Group Enable Cache	该选项用于开启 Group 缓存
Group Cache Size	指定最多能缓存多少 Group 查找。缺省为 211。该属性应该设置为一个质数以获得最好的查找性能。
Group Cache Positive TTL	设置一个成功的查找结果保存在缓存中的时间。缺省为 60 秒。
Group Cache Negative TTL	设置一个失败的查找结果保存在缓存中的时间。缺省为 10 秒
Group Membership Cache TTL	隔多长时间个更新缓存组的组成员。缺省为 10 秒

10. 要保存所作的设置，点 Apply 按钮.

11. 要启用或配置 User 缓存，需要定义 Caching Realm configuration 窗口的 User 标签页上的那些属性。

下表描述了 User 标签页上的各个属性。

表 14-6 User 缓存的配置属性

属性	描述
Enable User Cache	该选项用于开启 User 缓存
User Cache Size	指定最多能缓存多少 User 查找。缺省为 211。该属性应该设置为一个质数以获得最好的查找性能。
User Cache TTLPositive	该属性设置一个成功的查找结果保存在缓存中的时间。缺省为 60 秒。
User Cache TTLNegative	该属性设置一个失败的查找结果保存在缓存中的时间。缺省为 10 秒

12. 要保存所作的设置，点 Apply 按钮.
13. 要启用或配置 Permission 缓存，你需要定义 Caching Realm configuration 窗口的 Permission 标签页上的那些属性。

下表描述了 Permission 标签页上的各属性。

表 14-7 Permission 缓存的配置属性

属性	描述
Enable Permission Cache	该选项用于开启 Permission 缓存
Permission Cache Size	指定最多能缓存多少 Permission 查找。缺省为 211。该属性应该设置为一个质数以获得最好的查找性能。
Permission Cache TTLPositive	该属性设置一个成功的查找结果保存在缓存中的时间。缺省为 60 秒。
Permission Cache TTLNegative	该属性设置一个失败的查找结果保存在缓存中的时间。缺省为 10 秒

14. 要保存所作的设置，点 Apply 按钮。
15. 当完成对缓存域中属性的定义时，重启 WebLogic 服务器。

配置 LDAP 安全域

LDAP 安全域通过轻量级目录访问协议（LDAP）服务器对客户端请求进行验证。该服务器把组织中的所有用户都放在 LDAP 目录中以进行集中管理。目前 LDAP 安全域支持 Open LDAP，Netscape iPlanet, Microsoft Site Server 与 Novell NDS

目前，WebLogic 服务器支持以下两个版本的 LDAP 安全域。

LDAP realm V1 — 打包在以前 WebLogic 服务器版本中的 LDAP 安全域。LDAP security realm V1 可以与所有所支持的 LDAP 服务器一同工作，该版本主要是为那些仍然使用老版本 WebLogic 服务器的 BEA 用户提供的。但是这一版本已经不推荐使用 LDAP realm V1，所以 BEA 建议用户升级到 LDAP realm V2

LDAP realm V2 — 最新的 LDAP 安全域在性能与可配置性方面都得到了提高。与 WebLogic Server 6.0 Service Pack 1.0 中提供的 LDAP 安全域是一样的。

注意：在使用 LDAP 安全域 v1 时，可以通过管理控制台查看存储于 LDAP 目录服务器中的用户与用户组，但使用 LDAP 安全域 v2 时，只能通过管理控制台查看存于 LDAP 服务器中的用户组信息。

对用户以及用户组的管理（例如，增加与删除用户或用户组，或者是在组中增加成员），你

需要使用 LDAP 服务器所提供的管理工具。如果你修改了 LDAP 存储中的内容，重置用户和组的信息以便查看改动情况。

LDAP 安全域的配置主要是确定 WebLogic 服务器中的 LDAP 安全域如何与 LDAP 服务器进行通信以及描述用户与用户组如何保存在 LDAP 目录中。如果使用 LDAP realm V2，那么可以使用 WebLogic 服务器所提供的模板来配置 LDAP 安全域。

对使用 LDAP 安全域的限制

在使用 LDAP 安全域时，应该注意以下限制：

在安装 Microsoft Site Server 中的 LDAP 服务器并创建了 LDAP 目录的根时，将缺省地创建若干个组织单元。在 Groups 下面有一个缺省的组织单元（名字为 NTGroups）以及一个名字为 Administrators 的缺省空用户组。缺省情况下，WebLogic 服务器也提供一个名字为 Administrators 的用户组，这个组中包含了一个启动 WebLogic 服务器的成员：System。如果你使用了 Microsoft Site Server 的缺省设置，并在缺省的组织单元中创建自己的用户组，那么 WebLogic 服务器将不能被启动。因此你应该在 LDAP 目录服中创建自己的组织单元并在这个组织单元中创建自己的用户组。

如果 LDAP 目录中存在两个同名的用户组，那么 WebLogic 服务器将不能正确地对用户组中的用户进行验证。LDAP 安全域使用用户组的 DN 来定位 LDAP 目录中的用户组。如果你创建了多个同名的用户组，WebLogic 服务器只对它所找到的第一个组中的用户进行验证。因此在使用 LDAP 安全域时，每个用户组的名字应该是唯一的。

LDAP realm V1:所提供的以下功能在 LDAP realm V2 中将不再提供。

列出所有用户

列出用户组的成员

authProtocol 与 userAuthentication 机制被删除。你应该用 JNDI 绑定机制将安全证书传递到 LDAP 服务器。

配置 LDAP Realm V1

如果要使用 LDAP Security realm V1 而不是文件域：

1. 那么进入管理控制台左窗格中的 Security->Realms 节点。
2. 在管理控制台的右窗格中，点击 Configure a New LDAP Realm V1 链接。

   实施 LDAP 安全域的类名将显示。
3. 点击 Create 按钮。
4. 要开启 LDAP 服务器与 WebLogic 服务器之间的通信，需要定义 LDAP Realm Create 窗口的 LDAP Realm V1 标签页上的属性。

下表描述了 LDAP Realm V1 标签页上的各个属性。

表 14-8 LDAP 标签页上的 LDAP 安全域配置属性

属性	描述
LDAPURL	LDAP 服务器的位置。把 URL 该为运行 LDAP 服务器的机器名与监听端口号。如果 WebLogic 服务器与 LDAP 服务器之间的连接使用 SSL 协议，那么

	在 URL 中使用 LDAP 服务器的 SSL 端口。 如 ldap://ldapserver:385
Principal	WebLogic 服务器连接到 LDAP 服务器所使用的 LDAP 用户的区分名（distinguished name，简称为 DN）。这个用户必须有列出 LDAP 用户与用户组的权限
Credential	用来验证 Principal 属性所定义的 LDAP 用户的口令
Enable SSL	是否启用 SSL 的选项。用 SSL 可以保护 LDAP 服务器与 WebLogic 服务器之间的通信。记住以下规则： 1．如果不使用 SSL 协议，那么就禁用该属性 2 LDAP Users 标签页上的 UserAuthentiation 属性设为 external，那么必须开启该属性。
AuthProtocol	对 LDAP 服务器进行验证的验证类型。该属性应该设置为以下值： ● 如果不需要进行验证，那么就设为 None ● 如果使用口令验证，那么设为 Simple ● 如果采用证书验证，那么就设为 CRAM-MD5 Netscapte iPlanet 支持 CRAM-MD5 验证。Netscapte iPlanet,Microsoft Site Server, Novell NDS 支持 Simple 验证

5. 点 Apply 按钮保存设置。
6. 要设置如何把用户保存在 LDAP 目录中，需要定义 LDAP Realm Create 窗口的 Users 标签页中的属性。

下表描述了 Users 标签页上的各个属性。

表 14-9 Users 标签页上的 LDAP 安全域配置属性

属性	描述
User Authenticatione	设置用户验证的方式，可以设置为以下方式之一： 1.Bind 方式 l LDAP 安全域从 LDAP 目录服务器获得包括口令在内的用户数据，在 WebLogic 服务器中核对口令。（Local 设置适用于 Netscape Directory 与 Microsoft Site Server 2.设置为 External LDAP 安全域把 WebLogic 服务器的客户端所提供的用户名与口令绑定到 LDAP 目录服务器来对用户进行验证。如果选择 External 设置，那么必须使用 SSL 协议。 External 设置适用于 Novell NDS 3.设置为 Local LDAP 安全域查找 LDAP 目录的 UserPassword 属性并将它与 WebLogic 服务器中的口令进行核对。
User Password Attribute	如果 User Authentication 属性设置为 Local，那么该属性被用来查找包含 LDAP 用户口令的 LDAP 属性。
User DN	是一组属性，结合 User Name Attribute 属性可以唯一定义一个 LDAP 用户。
User Name Attribute	LDAP 用户的登录名。该属性可以设置 LDAP 用户的普通名字，通常使用简写字符串作为普通名字。

7. 点 Apply 按钮保存设置。
8. 为了指定如何把用户组保存在 LDAP 目录中，你应该设置 LDAP Realm Create 窗口的 Group 标签页中的属性。

下表描述了 Groups 标签页上的各个属性。

表 14-10 Groups 标签页上的 LDAP 安全域配置属性

属性	描述
Group DN	是一组属性，该属性与 Name Attritute 属性一起唯一地定义了 LDAP 目录中的用户组。
Group Name Attribute	LDAP 目录中的用户组的名字。这一般是一个普通名字
Group Is Context	该复选框指定了组成员如何记录在 LDAP 目录中。 如果每个组包含一个成员，那么选中该复选框。该属性缺省为开启的。 如果一个用户组条目中包含用户组成员的属性，那么不要选该复选框。
Group Username Attribute	组条目中组成员的 LDAP 属性的名字。

9. 点 Apply 按钮保存设置。
10. 当完成属性的定义，重启 WebLogic 服务器。
11. 有关配置缓存域的信息，参见“缓存域配置”。

    当配置缓存域时，在 General 标签页中的 Basic 属性下拉菜单中选取 LDAP Realm V1 Basic 属性定义了缓存域和其他安全域之间的关联（本例中为 LDAP Realm V1

缓存域将把用户与用户组保存在缓冲区中以避免频繁地查找 LDAP 目录。用户缓冲区与组缓冲区中的每个对象都具有 TTL 属性，该属性是你在配置缓存域时指定的。如果你更改了 LDAP 目录，直到缓存对象过期或者缓冲区满了，所作的更改才会反映到 LDAP 安全域中。成功查找的默认 TTL 值为 60 秒，不成功查找的默认 TTL 值为 10 秒。除非你改变了 User 缓冲区与 Group 缓冲区的 TTL 属性，否则对 LDAP 目录只有等到 60 秒后才能反映到 LDAP 安全域中。

某些服务器端程序会执行对 LDAP 安全域的查找操作。例如在 LDAP 安全域中调用 getUser() 方法，那么对象只有等到程序释放了它之后，才能被域返回，否则，即使是你从 LDAP 目录删除了某个被 WebLogic 服务器验证了的用户，这个用户在连接被关闭之前会一直有效。

配置 LDAP Realm V2

WebLogic 服务器为所支持的 LDAP 服务器提供了模板。这些模板定义了代表所支持的 LDAP 服务器中的用户与用户组信息的缺省属性。选择一个对应 LDAP 服务器的模板并填写 LDAP 服务器的主机名与端口号。以及 Group DN、UserDN、Priinciple 和 Credential 属性等，详细信息参见“配置 LDAP Realm V1”。

为了使用 LDAP Security realm V2

1. 进入管理控制台左窗格中的 Security->Realms 节点。
2. 选择要使用的 LDAP 服务器。有以下选项：

defaultLDAPRealmforOpenLDAPDirectoryServices

defaultLDAPRealmforNovellDirectoryServices

defaultLDAPRealmforMicrosoftSiteServer

defaultLDAPRealmforNetscapeDirectoryServer

   选择一个 LDAP 服务器，其配置窗口然后就显示出来。

3. 在 Configuration Data 框中的 server.host 与 server.port 属性中输入 LDAP 服务器的主机名与端口号。

4. 如果需要，在 Configuration Data 框中为 LDAP 目录服务器，更新 Group DN、UserDN、Priinciple 和 Credential 属性的定义。
5. 可选的，为 LDAP 服务器定义口令，Password 属性为 Principle 定义了口令，一旦定义了口令，WebLogic 就对其进行加密。
6. 点 Apply 按钮保存设置。
7. 当设置完所有属性后重启服务器。
8. 配置缓存域，详细信息参见“配置缓存域”。

当配置缓存域时，在 General 标签页中的 Basic 属性下拉菜单中选取 LDAP Realm V2　Basic 属性定义了缓存域和其他安全域之间的关联（本例中为 LDAP Realm V2

配置 Windows NT 安全域

Windows NT 安全域使用 Windows NT 域中的帐号信息对用户与用户组进行验证。你可以通过控制台查看 Windows NT 安全域中的用户与组，但要对用户或组进行管理则只能使用 Windows NT 所提供的工具。

Window NT 安全域提供身份（用户与组）验证，但不进行授权（ACLs）。要更新 filerealm.properties 文件中的 WebLogic 使用的 ACL 信息，在你更改了 ACL 信息后，点击 General 标签页中的 Refresh 按钮。如果在 ACL 中使用了组，你可以就可以减少刷新信息的频率。改变 Windows 组的成员，可以允许你动态的管理单一用户访问 WebLogic 服务器资源的权限。

定义在 WebLogic 服务器中的用户 system 必须同时在 Windows NT 域中声明。在 Windows NT 平台上，必须用 system 帐号运行 WebLogic 服务器，客户端必须提供 system 用户的口令才能通过身份验证。当你在 Windows NT 中定义 system 用户帐号时，要确保该帐号的拥有者具有管理权限并且可以从 Windows NT 域控制器中读取与安全相关的信息。

要使用 Windows NT 安全域，WebLogic 服务器必须作为 NT 的一个服务运行。你不必只在域控制器中运行 WebLogic 服务器。

要使用 Windows NT 安全域：

1. 到管理控制台的 Security->Realm 节点.
2. 在右窗格中，选 Create a New NT Realm 链接。
3. 对 Windows NT 安全域的配置包括定义域的名字以及运行 Window NT 域的机器。域的名字与所在机器通过管理控制台的 NT Realm Create 窗口中的属性来定义的。

   下表描述了 NT Realm Configuration 窗口中的各属性。

表 14-11 Windows NT 安全域的配置属性

属性	描述
名字	Window NT 安全域的名字，例如 AccountingRealm
主域	定义用户与用户组的 Window NT 域所在机器的主机名与端口号。多个主机名与端口号之间用逗号分开。

4. 点 Apply 按钮保存设置。
5. 当完成属性设置后，重启服务器。
6. 配置缓存域，详细信息参见“配置缓存域”。

当配置缓存域时，在 General 标签页中的 Basic 属性下拉菜单中选取 Windows NT 安全域。Basic 属性定义了缓存域和其他安全域之间的关联（本例中为 Windows NT 安全域）

在配置完 Windows NT 安全域后，需要在 Windows NT 中定义 system 用户。

1．使用管理员帐号进入到 WebLogic 服务器所在的 Windows NT 域。

2．进入 Program->Administrative 工具

3．选择 User Manager

4．定义 System 用户

5．选中 Show Advanced User Rights 选项

6．从 Rights 下拉菜单选择 Act as part of the operating system 选项

7．选 Add 按钮

8．确保 Window NT 的 PATH 环境变量中包含\wlserver6.1\bin 目录。（WebLogic 服务器从这个目录装载 Wlntrealm.dll.

配置 UNIX 安全域

Unix 安全域通过执行一个小程序 wlauth 来查找用户与组并基于用户的 UNIX 登录名与口令对用户进行身份验证。在有些平台上，wlauth 可以使用 PAM（插件式验证模块）。通过 PAM，你可以配置操作系统的验证服务而不需要改变使用这个服务的应用。在没有 PAM 的平台上，wlauth 使用标准的登录机制，例如 shadow password

改变了 ACL 后，点击 Security 节点中的 General 标签页中的 Refresh 按钮，从而更新保存在 filerealm.properties 文件中，为 WebLogic 服务器使用的信息。如果在 ACL 中使用组，那么可以减少刷新服务器的频率。通过改变 UNIX 组中的成员，就可以动态地管理单个用户对 WebLogic 资源的访问权限。

Wlauth 程序运行 setuid root。你需要拥有 root 的权限以便具有修改 wlauth 程序的所有权与文件属性。并为 wlauth 程序设定 PAM 配置文件。

以下是为 UNIX 安全域配置 wlauth 程序的步骤：

1．如果 WebLogic 服务器安装在一个网络驱动器上，那么把 wlauth 文件复制到运行 WebLogic 服务器的机器的文件系统上，例如复制到/usr/sbin 目录下。Wlauth 文件在 weblogic/lib/arch 目录中，其中 arch 是你所使用的机器平台的名字。

2．用 root 用户帐号运行以下命令来改变 wlauth 文件的所有人与权限：

```
# chown root wlauth
# chmod +xs wlauth
```

3．在 PAM 平台上（如 Solaris 与 Linux），对 wlauth 进行 PAM 配置

在 Solaris 上，把以下行加到/etc/pam.conf 文件中：

```
# Setup for WebLogic authentication on Solaris machines
#
wlauth auth required /usr/lib/security/pam_unix.so.1
```

wlauth password required /usr/lib/security/pam_unix.so.1

wlauth account required /usr/lib/security/pam_unix.so.1

在 Linux 上，创建一个名为/etc/pam.d/wlauth 的文件，它包含以下内容：

#%PAM-1.0

#

File name:

/etc/pam.d/wlauth

#

If you do not use shadow passwords, delete "shadow".

auth required /lib/security/pam_pwdb.so shadow

account required /lib/security/pam_pwdb.so

注意：如果不使用 shadow password，那么省略上文中的 shadow

要在 WebLogic 服务器中使用 UNIX 安全域：

1. 进入管理控制台左窗格中的 Security->Realms 节点.
2. 在右窗格中，点 Create a New UNIX Realm 链接。
3. UNIX 安全域的配置涉及以下属性：定义域的名字，提供验证服务的应用程序。这些属性在管理控制台的 UNIX Realm Create 窗口中定义。

下表描述了 UNIX Realm Create 窗口上的各属性。

表 14-12 UNIX 安全域的配置属性

属性	描述
Name	UNIX 安全域的名字，例如 AccountingRealm
AuthProgram	用来对 UNIX 安全域中的用户进行验证的程序名。大多数情况下，该程序的名字都是 wlauth.

4. 点 Apply 按钮可以保存设置。
5. 在定义完属性之后，重启 WebLogic 服务器。
6. 配置缓存域，详细信息见“配置缓存域”。

当配置缓存域时，在 General 标签页中的 Basic 属性下拉菜单中选取 UNIX 安全域。Basic 属性定义了缓存域和其他安全域之间的关联（本例中为 UNIX 安全域）

如果 wlauth 不在 WebLogic 服务器的类路径中，或者所给的程序名不是 wlauth，那么你在启动 WebLogic 服务器时，必须加上一个 Java 命令行属性。在启动 WebLogic 服务器的脚本中的 java 命令后加上以下选项：

-Dweblogic.security.unixrealm.authProgram=*wlauth_prog*

用 wlauth 程序的名字代替上面的 wlauth_prog，如果该程序不在搜索路径中，那么 wlauth_prog 应该使用全路径的文件名。 启动 WebLogic 服务器。 如果 wlauth 程序在 WebLogic 服务器的路径中并且名字是 wlauth，那么可以省略上面这个步骤。

配置 RDBMS 安全域

RDBMS 安全域是 BEA 提供的一个定制安全域，它把用户、组以及 ACL 存储在关系数据库中。可以用管理控制台来管理 RDBMS 安全域。

注： RDBMS 例子不能运行在允许自动提交功能的数据库中。你可以使用该范例作为你实施 RDBMS 安全域的起点，在你的代码中利用明确的 commit 语句，同时禁用数据库中的自动提交功能。

要使用 RDBMS 安全域：

1. 进入管理控制台左窗格的 Security->Realms 节点。
2. 在右窗格中，点 Create a New RDBMS Realm 链接。
3. 定义实施 RDBMS 类名信息。

下表描述了 General 标签页上的各个属性。

表 12-13 General 标签页上的 RDBMS 安全域配置属性

属性	描述
Name	RDBMS 安全域的名字，例如 AccountingRealm
Realm Class	实现 RDBMS 安全域的类名。你应该把这个类包含在 WebLogic 服务器的 CLASSPATH 中。

4. 点 Apply 按钮可以保存设置。
5. 定义用于连接数据库的 JDBC 驱动的属性

下表描述了 Database 标签页上的各个属性。

表 14-14 Database 标签页上的 RDBMS 安全域配置属性

属性	描述
Driver	JDBC 驱动程序的完整类名。这个类必须位于 WebLogic 服务器的 CLASSPATH 中
URL	RDBMS 安全域所使用的数据库的 URL，URL 的模式可以参考所使用的 JDBC 驱动程序的文档。
User Name	数据库的缺省用户名
Password	数据库缺省用户的口令

6. 点 Apply 按钮可以保存设置。
7. 在 Schema 属性框中定义用于保存用户、组以及 ACLs 至数据库的模式。

   列表 14-1 包含了随 WebLogic 服务器分发而附带的 RDBMS 代码范例，它被输入到 Schema 属性框中作为数据库描述。

列表 14-1 RDBMS 安全域模式范例：

```
“getGroupNewStatement=true;getUser=SELECT U_NAME, U_PASSWORD FROM
users WHERE U_NAME = ?;
getGroupMembers=SELECT GM_GROUP, GM_MEMBER from groupmembers WHERE
```

GM_GROUP = ?;

getAclEntries=SELECT A_NAME, A_PRINCIPAL, A_PERMISSION FROM

aclentries WHERE A_NAME = ? ORDER BY A_PRINCIPAL;

getUsers=SELECT U_NAME, U_PASSWORD FROM users;

getGroups=SELECT GM_GROUP, GM_MEMBER FROM groupmembers;

getAcls=SELECT A_NAME, A_PRINCIPAL, A_PERMISSION FROM aclentries

ORDER BY A_NAME, A_PRINCIPAL;

getPermissions=SELECT DISTINCT A_PERMISSION FROM aclentries;

getPermission=SELECT DISTINCT A_PERMISSION FROM aclentries WHERE

A_PERMISSION = ?;

newUser=INSERT INTO users VALUES (? , ?);

addGroupMember=INSERT INTO groupmembers VALUES (? , ?);

removeGroupMember=DELETE FROM groupmembers WHERE GM_GROUP = ? AND

GM_MEMBER = ?;

deleteUser1=DELETE FROM users WHERE U_NAME = ?;

deleteUser2=DELETE FROM groupmembers WHERE GM_MEMBER = ?;

deleteUser3=DELETE FROM aclentries WHERE A_PRINCIPAL = ?;

deleteGroup1=DELETE FROM groupmembers WHERE GM_GROUP = ?;

deleteGroup2=DELETE FROM aclentries WHERE A_PRINCIPAL = ?”

---

8. 点 Apply 按钮可以保存设置。
9. 当完成属性定义时，重启 WebLogic 服务器。
10. 配置缓存域，详细信息见“配置缓存域”。

当配置缓存域时，在 General 标签页中的 Basic 属性下拉菜单中选取 RDBMS 安全域。Basic 属性定义了缓存域和其他安全域之间的关联（本例中为 RDBMS 安全域）

安装一个定制安全域

你可以利用现有的用户存储库（例如网上的目录服务器）创建一个定制安全域。要使用自行定制的安全域，必须先创建一个实现 WebLogic.security.acl.AbstractListableRealm 接口或者 weblogic.security.acl.AbstractManageableRealm 接口的类，然后在管理控制台中安装这个类。

要安装一个定制安全域：

1. 进入管理控制台左窗格 Security->Realms 节点。
2. 点击右窗格中的 Create a New Custom Realm 链接。
3. 定义安全域的名字以及实现安全域的类，指定安全域保存用户、组与 ACL 的方式。这些

配置可以通过管理控制台的 Custom Realm Create 窗口上的属性来进行。

下表列出了 Custom Security Realm Create 窗口中必须设置的属性。

表 14-15 定制安全域的配置属性

属性	描述
Name	定制安全域的名字，例如 AccountingRealm
Realm Class Name	实现定制安全域的类名。该 Java 类必须在 WebLogic 服务器的类路径中
Configuration Data	连接安全存储器所需要的信息
Password	连接定制安全域的口令，若提供了口令，WebLogic 对其进行加密。

4. 点击 Apply 按钮保存设置。
5. 当完成属性设置后，重启服务器。
6. 配置缓存域，详细信息见“配置缓存域”。

当配置缓存域时，在 General 标签页中的 Basic 属性下拉菜单中选取定制的安全域。Basic 属性定义了缓存域和其他安全域之间的关联（本例中为定制的安全域）

有关编写定制安全域的信息，可以参见“编写定制安全域”中的内容。

迁移安全域

WebLogic Server 为安全域提供了一个管理体系结构。该管理体系结构通过 Mbeans 实现，因此可以从管理控制台来管理安全域。如果你使用的是老版本 WebLogic 服务器的安全域，你应该按照以下步骤将安全域迁移到新的体系结构。

如果你使用的是 Window NT, UNIX 或者 LDAP 安全域，那么你可以用管理控制台的 Convert weblogic.properties 选项把旧的安全域转换到新的体系结构中。注：在管理控制台，你只能查看 Window NT, UNIX 或者是 LDAP 安全域中的用户、组以及 ACL，要管理用户与组，仍然需用 Window NT、UNIX 或者是 LDAP 环境所提供的工具。

如果使用的是定制安全域，那么你可以按照“安装定制安全域”中的步骤来指定有关用户、组以及 ACL 如何存储在定制安全域中的信息。

WebLogic 服务器不再使用委托安全域（Delegating security realm）。如果你使用的是委托安全域，那么你应该选择另一种安全域来保存用户、组与 ACL

如果你使用的是 RDBMS 安全域，那么你应该用以下方式中的一种来转换你所使用的安全域。

如果你不想改变 RDBMS 安全域的源，那么你应该按照“配置 RDBMS 安全域”中的步骤新建一个 RDBMS 安全域类的实例并定义有关连接数据库的 JDBC 驱动程序的信息以及有关安全域所使用的模式的信息。这种情况下，实际上在 WebLogic Server 6.0 中为 RDBMS 安全域创建了一个 Mbean

如果你对 RDBMS 安全域进行了定制，那么需要使用 Mbeans 来转换源。你可以按照 \samples\example\security\rdbmsrealm 目录中的例子来转换 RDBMS 安全域。把 RDBMS 安全域转换到 Mbeans 后，按照“配置 RDBMS 安全域”中的说明来定义连接数据库的 JDBC 驱动程序信息及安全域所使用的模式信息。

定义用户

注意：本节将说明如何在文件域中添加用户，如果使用其他的安全域，那么需要使用相应安全域所提供的管理工具来定义用户。

用户是可以在 WebLogic 服务器的安全域中进行身份验证的实体。用户可以是人也可以是一个软件实体，例如 Java 客户端。在 WebLogic 服务器的安全域中每个用户都有唯一的标识，因此管理员必须保证安全域中不存在相同的用户。

在安全域中定义用户需要为每个访问 WebLogic 服务器安全域中的资源的用户指定一个唯一的名字与口令，你可以用管理控制台的 Users 窗口来定义用户。

WebLogic 服务器有两个特殊的用户：system 与 guest

system 用户具有管理权限。该用户控制系统级的 WebLogic 服务器操作，如启动与停止服务器，锁定与解锁资源。system 用户及其口令是在安装 WebLogic 服务器时定义。作为一个安全措施，BEA 建议你经常更换 system 用户的，参见“改变系统口令”。

guest 用户是 WebLogic 服务器自动提供的。在不需要身份验证时，WebLogic 将 guest 标识分配给一个客户，该客户就可以访问任何 guest 用户可以使用的资源。如果浏览器或者 java 客户端提示需要输入用户名与口令，那么客户可以作为 guest 用户登录。guest 用户的用户名与口令分别为 guest 与 guest，缺省情况下系统启用 guest 帐户。

为了提供 WebLogic 服务器的安全性，BEA 建议在 WebLogic 服务器运行时，禁用 guest 用户。打开管理控制台的 Security 窗口，复选 General 标签页的 Guest Disabled opetion 选项，就可以禁用 guest 用户。禁用 guest 用户只是禁止以 guest 用户的身份登录到 WebLogic 服务器，它并没有禁止未验证的用户访问 WebLogic 服务器的资源。

system 用户及 guest 用户与 WebLoigc 服务器安全域的用户具有以下相似之处：

如果这两个用户要访问 WebLogic 服务器的资源，那么需要有合适的 ACL

要对 WebLogic 服务器的资源进行操作，那么需要提供用户名与口令（或者是数字签名）

要定义一个用户：

1. 进入管理控制台，在左侧窗格中选取 Security→Users 节点。
2. 在用户配置窗口，在 Name 属性中输入用户的名字。
3. 在 Password 属性中，输入口令。
4. 在 Confirm Password 属性中输入口令，加以确认。
5. 点击 Create

要删除一个用户：

1. 在用户配置窗口中的 Delete Users 框中输入要删除的用户名。
2. 点击 Delete

要改变一个用户的口令：

1. 在用户配置窗口中的 Name 属性输入用户名字。
2. 在 Old Password 属性中输入旧的口令。
3. 在 New Password 属性中输入新的口令。

4. 再次输入新口令以确认。

在 WebLogic 服务器运行时，某用户被锁定。执行以下步骤以解除：

1. 打开管理控制台中的用户窗口。
2. 点击 Unlock Users 链接。
3. 在 Users to Unlock 字段中输入希望解除锁定的用户的名字。
4. 选择在哪一台服务器上解除锁定。
5. 点击 Unlock.

有关 WebLogic 服务器中的用户以及访问控制模型，你可以参见“WebLogic 安全简介”与“安全基础”中的内容。

定义用户组

注意：本节将说明如何在文件域中添加用户组，如果使用其他的安全域，那么需要使用相应安全域所提供的管理工具来定义用户组。

用户组代表了一组具有某些共同点的用户，例如，在公司的同一部门工作。用户组的作用主要是为了对一组用户进行有效的管理。如果在 ACL 中一个组被授予了一个权限，那么组中的所有成员都具有该权限。

缺省情况下，WebLogic 服务器包含以下用户组：

安全域中的所有用户都是 everyone 组的成员

system 用户是 Administrators 组的成员。该用户组给那些负责启动与停止服务器、维护 WebLogic 服务器运行的用户分配相应的权限。对这个成员组的访问应该被限制。

按以下步骤可以在 WebLogic 服务器的安全域中注册一个组：

1．点击管理控制台的 Group 节点

2．点击 Create a New Group 链接

组配置窗口会显示。

3. 在管理控制台的 Group 窗口中设置 Name 属性。BEA 建议组的名字是复数的。如 Administrators 而非 Administrator

4．选择 Users 属性并选择需要加入到用户组的 WebLogic Server Users

5．选择 Groups 属性并选择要加入到用户组的 WebLogic Server Groups

6．点 Apply 按钮创建一个新的用户组。

在组配置窗口中的 Remove These Groups 列表框中输入要删除的组的名字，点 Remove 按钮就可以删除一个组。

有关 WebLogic 服务器中的组以及访问控制模型的更多信息，可以参见“WebLogic 安全简介”以及“安全基础”中的内容。

定义 ACL

用户可以访问在 WebLogic 安全域中的资源。用户是否具有访问的资源的权限取决于该资源的

访问控制列表ACLs。资源的ACL定义了某个用户访问该资源的许可。要定义一个资源的ACLs，首先创建该资源的一个 ACL，然后指定资源的许可，然后将许可分配给指定的用户与组。

每个 WebLogic 资源都具有一到多个许可可以分配，下表总结了 WebLogic 服务器不同资源可以用 ACL 限制的功能项。

表 14-16 WebLogic 服务器资源的 ACL

对 WebLogic 服务器该资源…	ACL…	可赋予权限的功能…
WebLogic 服务器	weblogic.server weblogic.server.*servername*	启动
命令行管理工具	weblogic.admin	终止服务器 对服务器加锁 对服务器解锁
MBeans	weblogic.admin.mbean.*mbeaninstancename*	访问
WebLogic 事件	weblogic.servlet.*topicName*	发送事件 接收事件
WebLogic JDBC 连接池	weblogic.jdbc.connectionPool.*poolname*	保留 管理
WebLogic 的口令	weblogic.passwordpolicy	对用户解锁
WebLogic JMS 目的方	weblogic.jms.topic.*topicName* weblogic.jms.queue.*queueName*	发送，接收
WebLogic JNDI 上下文	weblogic.jndi.*path*	查找 更改 列表

对某一 WebLogic 服务器资源建立 ACL，打开管理控制台执行以下步骤：

进入管理控制台，在左侧窗格中选择 Security→ACLs

在管理控制台右侧窗格中，点击 Create a New ACL 链接，ACL 配置窗口即显示。

在 New ACL Nane 字段中指定需要用 ACL 保护的 WebLogic 服务器资源的名字

例如，为一个名字为 demopool 的连接池创建一个 ACL

点击 Create

点击 Add a New Permission 链接。

指定资源的权限

可以为资源的每个权限单独创建 ACL，也可以用一个 ACL 包含资源的所有权限。例如，你可以为 JDBC 连接池——demopool 创建三个 ACLs，一个用于 reserve 权限，一个用于 reset 权限，一个用于 shrink 权限。或者只创建一个有 reserve,reset 与 shrink 的 ACL

指定哪些用户或用户组具有该资源的哪些权限。

点击 Apply 按钮。

在创建 WebLogic 服务器资源的 ACL 时，应该使用表 14-16 中所列出的语法来指定资源。例如，如果你要指定 JDBC 连接池 demopool 应该是 welogic.jdbc.connectionPool.demopool

如果修改了一个现存的 ACL，点击 Security 节点中 General 标签页上的 Refresh 按钮，将信息更新到 filerealm.properties 文件中。

在启动 WebLogic 服务器之前，你应该将启动服务器的权限分配给一组用户，以防止未授权的用户启动 WebLogic 服务器。

缺省情况下，只有系统用户有改变 MBeans 的权限。BEA 建议你限制访问或改变 MBeans 的用户数。用以下命令访问所有 WebLogic 服务器中的 MBeans

access.weblogic.admin.mbean=*User or Group name*

如果用户对 Mbean 的访问没有成功，那么将返回 weblogic.management.NoAccessRuntimeException 服务器日志中详细记载了是哪一个用户访问 Mbean 的信息。

配置 SSL 协议

下面的章节介绍了如何获取数字证书和 SSL 配置：

获得一个 WebLogic 服务器使用的私钥与数字证书

保存 WebLogic 服务器的私钥与数字证书。

定义可信认证机构

定义 SSL 协议的相关属性

有关 SSL 协议的详细内容，你可以参见“WebLogic 服务器安全简介与安全基础”中的内容。

获得私钥与数字证书

对每一个部署的 WebLogic 服务器都需要一个私钥以及数字证书。要从证书认证机构获得一个数字证书，你应该使用一种称为证书签名请求（Certificate Signature Request，简称 CSR）的特殊格式提交你的请求。WebLogic 服务器有一个证书请求生成器 servlet，你可以用这个 servlet 创建 CSR。证书生成器从根据你所提供的信息生成一个私钥与证书请求文件。然后把这个 CSR 提交给一个证书认证机构，如 VeriSign 或 Entrust.net。在使用证书请求生成器前，必须先安装并运行 WebLogic 服务器。

注：如果你没有通过证书请求生成器 servlet 而是其他途径获得一个私钥文件，确认该文件应属于 PKCS#5/PKCS#8 PEM 的格式。

以下是生成 CSR 的步骤：

启动证书请求生成器 servlet。该 servlet 的.war 文件位于:

\wlserver6.0\config\mydomain\applications 目录中。WebLogic 服务器在启动时自动安装该.war 文件。

2 在 Web 浏览器中，输入证书请求生成器 Servlet 的 URL

https://*hostname*:*port*/certificate/

URL 中的各部分描述如下：

hostname 是运行 WebLogic 服务器的主机的名字

port 是 WebLogic 服务器监听 SSL 连接的端口号。缺省为 7002

例如，运行 WebLogic 服务器的主机的名字是 ogre，7002 是它监听 SSL 通信的端口，那么在浏览器中输入以下 URL 来访问证书请求生成器。

https://ogre:7002/certificate/

3．证书生成器在 web 浏览器中装载进一个表单。你需要在表单中填写下表所描述的信息。

表 14-17 证书生成器表单中的字段

字段	描述
Country code	国家的 ISO 代码，由两个字母组成。美国的 ISO 国家代码是 US
Organizational unit name	你所在部门或所在机构中的组织单位的名称
Organization name	你所在机构的名字。证书认证机构可能会要求你在这个字段中输入在该机构注册了的域的所有主机名
E-mail address	管理员的 e-mail 地址。数字证书将通过邮件发送到这个地址。
Full host name	要安装数字证书的 WebLogic 服务器的完整主机名。这个名字用来对 WebLogic 服务器进行 DNS 查找。例如，node.mydomain.com web 浏览器会对 URL 中的主机名与数字证书中的名字进行比较。如果改变了主机名，需要重新申请一个数字证书。
Locality name (city)	你所在城镇的名字。如果所使用的许可证按城市授权，那么必须输入授权许可证的城市。
State name	如果你所在的机构在美国，那么在这个字段中输入你所在州的洲名。如果是加拿大，输入你所在省的名字。不要用简写
Private Key Password	对私钥进行加密的口令。 如果想使用一个被保护的密钥，那么在这个字段中输入一个口令。当每次使用这个密钥时，你会被要求输入一个口令。如果你指定了一个口令，那么所得到的是一个 PKCS-8 加密私钥。BEA 建议使用口令来保护私钥。 如果不使用被保护的密钥，那么不需要设置该字段。 如果使用被保护的私钥，那么需要在管理控制台的 Server 窗口中，启用 SSL 标签页上的 Key Encrypted 属性。
RandomString	加密算法所使用的字符串。你必须记住该字符串。该字符串被加密算法使用，加大解密的难度。因此，你所输入的这个字段应该不易被猜中。你最好在这个字段中使用大写与小写字符、数字、空格以及标点字符。（该字段是可选的）
Strength	所生成的私钥的长度（以位计），私钥越长，就越难被人解密。 如果你使用 WebLogic 服务器的国内版，你应该选择 512-, 768-,或者是 1024 位的私钥。建议你使用 1024 位私钥。

4．点 Generate Request 按钮。

如果你在表单中填了无效的值或没有填写必须的字段，那么证书生成器 servlet 会给出提示信息，按浏览器的回退按钮更正错误。

如果所有的字段都填写正确，证书生成器 servlet 在 WebLogic 服务器的启动目录下生成以下文件：

www_mydomain_com-key.der — 这是私钥文件。你应该在管理窗口的 SSL 标签页上的 Server Key File Name 字段中输入该文件的名字。

www_*mydomai*n_com-request.dem — 证书请求文件，使用二进制格式。

www_*mydomai*n_com-request.pem — 提交到证书认证机构的 CSR 文件。它包含的数据与.dem 文件相同，但是用的是 ASCII 码，因此你可以把它复制到邮件或者 Web 表单中。

5．选择一家证书认证机构，按以下步骤从证书认证机构的 web 站点中购买数字证书。

VerSign.Inc.为 WebLogic 服务器提供了两个选项：Global Site Services 为美国境内销售的与出

口的 Web 浏览器提供了健壮的 128 位加密选项。Secure Site Services 为在美国境内销售的 Web 浏览器提供 128 位加密选项，为出口的 Web 浏览器提供 40 位加密选项。

Entruest.net 数字证书为国内使用的浏览器版本提供 128 位加密，为国外使用的浏览器版本提供 40 位加密。

6．在选择服务器类型时，请选择 BEA WebLogic Server 以确保所获得的数字证书与 WebLogic 服务器兼容。

7．在收到证书认证机构所提供的数字证书后，把它保存在\wlserver6.1\config\mydomain 目录中。

8．配置 WebLogic 服务器以使用 SSL 协议，在 Server Configuration 窗口的 SSL 标签页中输入以下信息：

在 Server Certificate File Name 字段中输入确定 WebLogic 服务器身份的数字证书所在的全路径名与文件名。

在 Trusted CA File Name 字段中输入证书认证机构发放的数字证书所在的完整路径名。

在 Server Key File Name 字段中，输入 WebLogic 服务器的私钥文件所在的完整路径名。

有关配置 SSL 协议的更多信息，请参见“定义 SSL 协议的配置字段”。

9．如果使用受保护的私钥文件，那么用以下命令行选项启动 WebLogic 服务器。

-Dweblogic.management.pkpassword=password

其中 password 是在私钥的口令

保存私钥与数字签名

获得私钥与数字证书后，把这两个文件放在\wlserver6.1\config\mydomain 目录中。

所生成的私钥文件可以是 PEM 格式的，也可以是 Definite Encoding Rules(DER)格式的。文件扩展名表明了数字证书使用的格式。

PEM(.pem)格式的私钥分别以下面的行开头与结尾：

-----BEGIN ENCRYPTED PRIVATE KEY-----

-----END ENCRYPTED PRIVATE KEY-----

PEM(.pem)格式的数字证书以下面的行开头与结尾：

-----BEGIN CERTIFICATE-----

-----END CERTIFICATE-----

注意：一个文件中可能包含多个数字证书，每个证书以 BEGIN CERTIFICATE 与 END CERTIFICATE 作为界限，你使用其中的一个证书。通常情况下，WebLogic 服务器的数字证书文件的后缀名可以是.pem 或.der，数字证书与证书链分别位于不同的文件。使用两个文件的原因是多个 WebLogic 服务器可能共享同一个证书链。

数字证书文件中的第一个数字证书是 WebLogic 服务器证书链中的第一个数字证书，文件中的第二个证书是数字证书链中的第二个数字证书，以此类推。文件中的最后一个证书是结束证书链的一个自签名数字证书。

Der(.der)格式的文件包含二进制数据。WebLogic 服务器要求文件的扩展名与证书文件的内容相匹配，从而确保从证书认证机构所获得的文件具有正确的文件扩展名。

应该对私钥文件以及数字证书文件进行保护，只有 WebLogic 服务器的系统用户才有这两个文件的读权限，其它用户没有这两个文件的访问权限。如果你将多个证书认证机构发放的数字证书创建成一个文件，或者创建一个包含证书链的文件，那么你必须使用 PEM 格式。WebLogic 服务器提供了 DER 格式与 PEM 格式相互转换的工具。更多信息，可以参见“WebLogic 工具”中的内容。

定义可信的证书认证机构

WebLogic 服务器在建立 SSL 连接时，它会在一组可信的证书认证机构列表中检查证书认证机构的身份，从而保证当前所使用的证书认证机构是可靠的。

把证书认证机构的根证书复制到 WebLogic 服务器的\wlserver6.1\config\mydomain 目录中，然后设置“定义 SSL 协议的配置字段”一节中所描述的属性。

如果使用证书链，那么需要把其它 PEM 编码的数字证书附加到由数字证书认证机构所发放给 WebLogic 服务器的数字证书的后面。文件中的最后一个数字证书应该是一个自签名的数字证书（即 rootCA 证书）。

如果要用双向验证，那么应该在可信 CA 文件（trusted CA file）中包含证书认证机构的根证书。

定义 SSL 协议的配置字段

安全套接层协议（SSL），提供了安全的网络连接，它允许两个通过网络连接的应用可以相互进行身份验证，对需要交换的数据进行加密。SSL 协议提供了服务器验证，可选的客户端验证、加密以及数据完整性测试。

按以下步骤定义 SSL 协议的配置属性：

1．打开管理控制台

2．打开 Server Configuration 窗口

3．选择 SSL 标签页，定义该页中的属性字段。（请参见下面的表格）

4．点 Apply 按钮保存设置。

5．重启 WebLogic 服务器

注：如果使用 PKCS-8 来保护私钥，那么在启动 WebLogic 服务器的命令行中需要该指定私钥的口令。

下表描述了 Server Configuration 窗口 SSL 标签页上的属性字段。

表 14-18 SSL 协议的配置属性

属性	描述
Enabled	允许使用 SSL 协议。该字段缺省为启用的。
Listen Port	WebLogic 服务器监听 SSL 协议的端口号，缺省为 7002
Server Key File Name	WebLogic 服务器私钥文件所在的全路径和文件名。 路径应该以 WebLogic 服务器安装的根路径为起始。 例如：\wlserver6.1\config\myapp\privatekey.pem 文件扩展名（.DER 或.PEM）表明了 WebLogic 服务器用哪种方式读取

	文件的内容。
Server Certificate File Name	表明 WebLogic 服务器身份数字证书所在的全路径。文件扩展名（.DER 或.PEM）表明了 WebLogic 服务器用哪种方式读取文件的内容。
Server Certificate Chain File Name	对 WebLogic 服务器数字证书签名的数字证书所在的目录。 路径应该以 WebLogic 服务器安装的根路径为起始。 文件扩展名（.DER 或.PEM）表明了 WebLogic 服务器用哪种方式读取文件的内容。 如果使用数字证书链，文件中的第一个成员应该是 WebLogic 服务器数字证书签名的数字证书，第二个成员应该包含是对第一个数字证书签名的数字证书，以此类推。文件中的最后一个数字证书是一个自签名的数字证书。 Server Certificate Chain File Name 属性要求至少一个数字证书。如果文件只有一个数字证书，那么数字证书必须是自签名的（即，必须是一个根 CA 数字证书）
Client Certificate Enforced	定义了客户端是否需要向 WebLogic 服务器出示由可靠的证书认证机构发放的数字证书。
Trusted CA File Name	是一个文件，包含 WebLogic 服务器所信任的证书认证机构发放的数字证书。该文件包含证书认证机构发放的一或多个的数字证书。WebLogic 服务器根据文件扩展名（.DER 或.PEM）决定用那种方式读文件的内容。
CertAuthenticator	实现 CertAuthenticator 接口的 Java 类名，关于 weblogicl.security.acl.CertAutherticator 接口的使用方法，可以参见“把数字证书映射到 WebLogic 用户”中内容。
Key Encrypted	该属性指定是否需要用口令加密 WebLogic 服务器的私钥文件。缺省为 false 如果设置了该属性，那么必须使用受保护的 WebLogic 服务器私钥文件。在重启 WebLogic 服务器时，使用以下命令行选项启动 WebLogic 服务器： -Dweblogic.management.pkpassword=*password* 其中 *password* 是私钥的口令
Use Java	启用本地 Java 库的复选框。WebLogic 服务器允许使用本地 java 库。 WebLogic 服务器提供了纯 Java 实现的 SSL 协议；但在 Solaris, Windows NT, IBM AIX 平台上，使用本地 Java 库可以提高 SSL 协议的性能。该字段缺省为不启用本地 Java 库。
Handler Enabled	该字段决定 WebLogic 服务器是否拒绝因为以下原因而引起客户验证失败的 SSL 连接： 1．没有提供所要求的客户数字证书 2．客户端没有提交数字证书 3．客户端所提交的数字证书不是由 Trusted CA Filename 字段所指定的证书认证机构发放的。 缺省情况下，SSL Handler 允许 WebLogic 服务器把 SSL 连接从该实例引向另一个 WebLogic 服务器。例如，WebLogic 服务器中的一个 EJB 可以打开另一个 Web 服务器的 HTTPS 流。当 HandlerEnabled 字段启用时，WebLogic 服务器的角色是 SSL 连接的客户端。该字段缺省为启用的。 只有在你要提供 SSL 连接引出（SSL connection outgoing 的实现时，才可以禁用该字段。 **注意：** WebLogic 服务器管理接入 SSL 连接不受 SSL Hander 的影响。
Export Key Lifespan	一个可出口的私钥被 WebLogic 服务器在美国境内服务器与出口的客

	户端之间使用的次数。当超过这个次数时，WebLogic 服务器将产生一个新的私钥。如果想使 WebLogic 服务器更安全，那么应该减少私钥被使用的次数。缺省值为 500 次。
Login Timeout Millis	SSL 连接的超时时限，单位为微秒。缺省为 25,000 微秒。建立 SSL 连接要比建立常规连接花更多的时间。如果客户端通过 Internet 连接，那么考虑到附加的网络滞后，应该调高该字段的值。
Certificate Cache Size	WebLogic 服务器存储并标记的数字证书的个数，缺省为 3 个。
Ignore HostName Verification	当 WebLogic 服务器作为另一个 WebLogic 服务器的客户端时，禁用所安装的 HostName Verifier
HostName Verifier	实现 HostName Verifier 接口的 Java 类的名字。有关如何使用 weblogic.security.SSL.HostNameVerifier 接口的信息，可以参见“使用定制的 HostName Verifier”中的内容。

注：在以前的 WebLogic 服务器版本中，Server Certificate File Name 属性（或者在 weblogic.security.certificate.server 属性中）所定义的数字证书可能是一个自签名，但无效的数字证书。这不是一个很好的安全策略。目前的 WebLogic 服务器版本要求同时定义 Server Certificate File Name 以及 Server Certificate Chain File Name 属性。

配置双向验证

如果 WebLogic 服务器配置为使用双向认证，那么客户端需要向 WebLogic 服务器出示数字证书，WebLogic 服务器根据一个可信的证书认证机构列表来验证数字证书。

关于如何配置 WebLogic 服务器的 SSL 协议以及证书验证的内容，可以参见“配置 SSL 协议”部分。

把 WebLogic 服务器使用的数字认证机构根证书复制到\wlserver6.1\config\mydomain 目录中。在双向认证中，客户端需要出示由这些可信的证书认证机构之一所发放的数字证书。

要启用双向认证，在管理控制台的 Server Configuration 窗口中，选 SSL 标签页中的 Client Certificate Enforced 选项。该选项缺省为禁用的。

配置基于 IIOP 之上的 RMI 所用的 SSL

可以用 SSL 协议保护与 RMI 远程对象的 IIOP 连接。SSL 协议通过身份验证以及加密对象之间所交换的数据来保护连接。要用 SSL 协议保护基于 IIOP 上的 RMI 连接，你需要：

1．配置 WebLogic 服务器使用 SSL 协议。详细信息，可以参见“配置 SSL 协议”中的内容。

2．把客户端 ORB Object Request Broker，对象请求代理）设置为使用 SSL 协议。有关如何配置客户端 ORB 的 SSL 协议，可以参考所使用的 ORB 产品的文档中相关章节。

3．用 host2ior 工具把 WebLogic 服务器的 IOR 打印到控制台中。host2ior 工具打印两个版本的可互操作对象引用（interoperable object reference IOR：一个是基于 SSL 连接的，一个是基于非 SSL 连接的。IOR 的头指定了该 IOR 是否可以用于 SSL 连接。

4．在获得 CosNaming 服务的初始引用时，使用 SSL IOR CosNaming 服务被用来访问 WebLogic 服务器的 JNDI 树。

有关如何在 IIOP 上使用 RMI 的更多内容，可以参考 Programming WebLogic IIOP over RMI 中的内容。

口令的保护

保护用来访问 WebLogic 服务器资源的口令是很重要的。在过去，用户名与口令以明文的形式存储在 WebLogic 服务器的安全域中。现在 WebLogic 服务器对所有口令进行散列化。当 WebLogic 服务器获得一个客户端请求时，客户端所输入的口令也被散列化，然后把散列化结果与所保存的散列化口令进行比较，看它们是否相互匹配。

每个 filerealm.properties 文件都有一个与它关联的 SerializedSystemIni.dat 文件，这个文件被用来散列化口令。在安装时，SerializedSystemIni.dat 文件保存在\wlserver6.1\config\mydomain 目录下，如果该文件被破坏，那么就需要重新配置 WebLogic 服务器。

我们建议你采用以下预防措施：

备份 SerializedSystemIni.dat 文件，并将其与 filerealm.properties 文件的备份存放于同一目录下。

设置 SerializedSystemIni.dat 文件的访问权限，例如只允许 WebLogic 服务器的管理员有读写这个文件的权限，其它用户没有这个文件的任何权限。

如果你使用的是 weblogic.properties 文件，并且想散列化这个文件中的口令，你可以用管理控制台主窗口的 Convert weblogic.properties 选项将 weblogic.properties 文件转换为 config.xml 文件。一旦文件被转换，则所有现存的口令就已经被保护了。

Config.xml 文件中的不再含有明文的口令，若在其中保存明文的口令，config.xml 会对其进行加密并被散列化该口令。被加密的口令不能从一个域复制到另一个域，而是应该使用明文口令替换 config.xml 中被加密的散列化口令，然后再将文件复制到另外一个域中。管理控制台在下一次写文件时会加密并散列化口令。

要保护 WebLogic 服务器的口令，执行以下操作：

1. 打开管理控制台
2. 点击 Security 节点
3. 在管理控制台右侧窗格中选择 Passwords 标签页。
4. 定义该标签页中需要配置的属性，按照相应的提示输入值，并选中必须选择的复选框（详细信息，参见下表）。
5. 点 Apply 按钮保存所做的设置
6. 重启 WebLogic 服务器。

下表详细描述了 Security Configuration 窗口的 Password 标签页上的各个属性。

Table 14-19 口令保护属性

属性名	描述
Minimum Password Length	口令所需的长度，至少为 8 个字符。缺省为 8 个字符
Lockout Enabled	在超过 Lockout Threshold 次尝试后，是否需要锁住某个登录无效的帐号。缺省情况下，该属性被启用。
Lockout Threshold	当一个用户试图登录到一个用户帐户，因口令不对而失败，那么多少次这样的失败登录后将锁住这个帐号。以后对该帐号的访问 （即使是 username/password 是正确）也将引发 Security 异常；在管理员对该帐号进行解锁前，或在锁住期限内，该帐号一直处于锁住的状态。注意非法登录必须在 Lockout Reset Duration 属性所定义范围内。默认为 5

Lockout Duration	该属性定义了当某一帐户因为在 Lockout Reset Duration 期限内发生非法登录而被锁住后，多长时间范围内该用户帐号不能被使用。 要解开一个被锁住的用户帐号，你必须拥有 weblogic.passwordpolicy 中的 unlockuser 权限。 默认为 30 分钟。
Lockout Reset Duration	该属性定义了在多长时间里，非法登录某一帐号将导致该帐号被锁住。 在本属性定义的时间范围内，当非法登录的次数超过了 Lockout Threshold 属性所定义的值，那么该帐号将被锁住，例如，该属性被设置为 5 分钟，当帐号在 6 分钟内被非法登录了 3 次，那么该帐号不会被锁住，但是，如果在 5 分钟内，发生了 5 次非法登录，那么该帐号将被锁住。 缺省为 5 分钟。
Lockout Cache Size	指定无效的或非法的登录意图的缓存大小。缺省为 5

安装审计提供者

在 WebLogic 服务器中，你可以创建一个审计提供者来接收与处理安全事件布告，例如身份验证请求、失败的或成功的授权以及无效数字证书等。

要使用审计提供者，首先要实现 weblogic.security.audit.AuditProvider 接口。然后用管理控制台安装并激活该接口的实现类。

要安装审计提供者，需要在管理控制台中的 Security 节点上的 General 标签页中的 Audit Provider Class 字段中输入实现 AuditProvider 接口的类名。然后重启 WebLogic 服务器。

有关编写审计提供者的更多信息，可以参考“审计安全事件”中的内容。有关创建连接过滤器的例子，可以参考 WebLogic 服务器的\samples\examples\security 目录中的 LogAuditProvider 例子。

安装连接过滤器

连接过滤器根据客户端的源以及协议来决定是否接受客户端的连接请求。当客户端连接 WebLogic 服务器后，在实施任何动作之前，WebLogic 服务器把客户端的 IP 地址、端口号、协议（HTTP, HTTPS, T3, T3s 或 IIP）以及 WebLogic 服务器的端口号传递给连接过滤器。检验了这些信息后，你可以决定是允许连接，还是引发 FilterException 来结束连接。

要使用连接过滤器，必须先实现 weblogic.security.net.ConnectionFilter 接口。然后使用管理控制台安装实现该接口的 Java 类。

具体的步骤为：打开管理控制台的 Security Configuration 窗口，设置 Advanced 标签页中的 Connect Filter 字段，填入你实现 weblogic.security.net.ConnectionFilter 接口的类名，然后重启 WebLogic 服务器。

有关编写连接过滤器的详细信息，可以参考“过滤网络连接”中的内容。有关创建连接过滤器的例子，可以参考 WebLogic 服务器的\samples\examples\security 目录中的 SimpleConnectionFilter 例子。

设置 Java 安全管理器

当你在 Java 2 (JDK 1.2 或 JDK 1.3) 上运行 WebLogic 服务器时，WebLogic 服务器使用 Java 安全管理器进一步控制对 WebLogic 服务器资源的访问。JVM 内建的安全机制需要一个安全策略文件。Java 安全管理器使用一组授予给 CodeSource 或 SignedBy 类的权限。这些权限确定了运行在一个 JVM 实例中的类是否可以执行某项运行时操作。多数情况下，威胁安全的不大可能是运行在 JVM 中的恶意代码，因此 Java 安全管理器并不是必要的。当一个应用服务提供者使用了 WebLogic 服务器而且运行了未知的类，这种情况下 Java 安全管理器是必要的。

注：在以前的版本中，启用 Java 安全管理器通过在启动 WebLogic 服务器时设置 -DwebLogic.security.manager 属性来实现。注意 WebLogic Server 6.0 及以后的版本在一点的变更。

如果要在 WebLogic 服务器中使用 Java 安全管理器，那么启动 WebLogic 服务器时应该使用 -Djava.security.manager 属性。

Java 安全管理器需要一个安全策略文件来定义权限。在启动 WebLogic 服务器时用 -Djava.security.policy 属性指定安全策略的全路径名。如果启用了 Java 安全管理器但没有指定策略文件，java 安全管理将使用$Java_HOME/lib/security 目录下的 java.security 与 java.policy 文件中所定义的安全策略。

WebLogic 服务器包含了一个名字为 weblogic.policy 的安全策略文件示例，该文件提供了一组缺省的权限。

要在 WebLogic 部署中使用 Java 安全管理器安全策略文件：

1. 编辑 webLogic.policy 文件中的以下行，用你的 WebLogic 服务器所在的目录替换相应的部分

```
grant codebase “file:.//BEA/-”{
permission java.io.FilePermission “D:${/}BEA${/}=”,...
```

注：该变更假定 WebLogic 服务器的安装目录结构与 BEA WebLogic 服务器安装指南中对 BEA 主目录中所描述的一样

2. 如果要运行管理控制台，在 weblogic.policy 文件中加入以下内容：

```
grant {
permission java.io.FilePermission
"D:{/}BEA${/}wlserver6.1${/}weblogic${/}management${/}console${
/}-", "read";
permission java.io.FilePermission
"D:{/}BEA${/}wlserver6.1${/}config${/}mydomain${/}applications$
{/}.wl_temp_do_not_delete${/}weblogic${/}management${/}console$
{/}-", "read";
permission java.util.PropertyPermission "user.*", "read";
};
```

3. 如果 CLASSPATH 中包含其它目录或者在别的目录中部署应用，那么应该将这些目录的权限控制加到 weblogic.policy 文件。

4. BEA 建议你采取以下防范措施：

对 weblogic.policy 文件进行备份，并将备份放在一个安全的地方。

设置 weblogic.policy 文件的访问权限，只有 WebLogic 服务器管理员才有此文件的读写权限，其它用户不具有 weblogic.policy 文件的读写权限。

5. 在启动 WebLogic 服务器的 Java 命令行上设置 java.security.manager 与 java.security.policy 属性。

```
$ java ... -Djava.security.manager\
        -Djava.security.policy==D:/BEA/wlserver600/lib/weblogic.policy
```

有关 Java 安全管理器的详细信息，请参见 Java2 的 Javadoc 文档。

使用 Recording Security Manager 工具

可以用 Recording Security Manager 工具发现 WebLogic 服务器在启动与运行过程中所出现的权限分配问题。该工具输出那些能够加到安全策略文件中的权限以解决该工具所发现的权限问题。你可以从 BEA Developer's Center 获得 Recording Security Manager 工具。

配置安全上下文传播

安全上下文传播使得运行在 WebLogic 服务器中的 Java 应用可以访问 BEA Tuxedo 域中的对象和操作。WebLogic 服务器的 WebLogic Enterprise Connectivity 组件提供了安全上下文传播功能。

如果使用安全上下文传播，一个 WebLogic 服务器安全域中用户定义的安全标识将作为 Internet Inter-ORB Protocol (IIOP) 请求的服务环境上下文中的一部分被传送到 BEA TUXEDO 域中，该请求通过作为 WLEC 连接组成部分的网络连接发送给 WLE 域的。WLEC 连接池中的每个网络连接都通过一个已定义的用户标识认证过。

为了使用安全上下文传播，需要为每个从 WebLogic 服务器访问的 BEA TUXEDO 域创建 WLEC 连接池。WebLogic 利用 IIOP 连接来传播 WLEC 连接池。WebLogic 服务器环境中的 Java 应用从 WLEC 连接池取得 IIOP 连接并用这些连接调用或激活 BEA TUXEDO 域中的对象。

在使用安全上下文传播之前，需要在 startAdminWebLogic.sh 或者 startAdminWebLogic.cmd 文件的 CLASSPATH 环境变量中加上 *TUXDIR*/lib/wleorb.jar 和 *TUXDIR*/lib/wlepool.jar

更多资料，请参照“使用 WebLogic Enterprise Connectity”。

要实现安全上下文传播的步骤如下：

选择管理控制台左窗格的 Service->WLEC 节点。

在右窗格单击“Create a new WLEC Connection Pool”链接。

定义下表中的属性：

表 14-20 General 标签页中的 WLEC 连接池属性

属性	说明
Name	WLEC 连接池的名字。每个 WLEC 连接池的名字必须唯一。
Primary Addresses	用来在 WLEC 连接池和 BEA TUXEDO 域之间建立连接的 IIOP 监听器/处理器的地址列表。每个地址的格式是//hostname:port 地址必须要和 UBBCONFIG 文件中定义的 ISL 地址匹配。多个地址使用分号分开，例如：//main1.com:1024;//main2.com:1044

	如果 WLEC 连接池要使用 SSL 协议，那么在 IIOP 的监听器与处理器前加上 corbalocs 前缀。例如： corbalocs://*hostname*:*port*.
Failover Addresses	不能与 Primary Addresse 字段定义的地址建立连接时所使用的 IIOP 监听器/处理器的地址列表。多个地址使用分号分开。这个字段时可选的。
Domain	WLEC 连接池所连接的 BEA TUXEDO 域的名字。一个 BEA TUXEDO 域只能有一个 WLEC 连接池。域的名字必须与 BEA TUXEDO 域的 UBBCONFIG 文件中的 RESOURCES 节中的 domainid 参数匹配。
Minimum Pool Size	在 WebLogic 服务器启动时，WLEC 连接池的初始 IIOP 连接数，缺省为 1
Maximum Pool Size	WLEC 连接池的最大 IIOP 连接数，缺省为 1

4. 点 Create 按钮

5. 在管理控制台中的 WLEC Connection Pool Configuration 窗口的 Security 标签页上，为 WebLogic 服务器安全域中的用户到 BEA TUXEDO 域而定义传播安全上下文的属性。

表 14-21 Security 标签页上的 WLEC 连接池属性

属性	描述
User Name	一个 BEA TUXEDO 用户的用户名。只有当 BEA TUXEDO 域的安全级别为 USER_AUTH, ACL 或者是 MANDATORY_ACL 时，才需要定义这个字段
User Password	用户的口令。只有当你定义了 User Name 字段后，才需要定义这个字段
User Role	BEA TUXEDO 用户的角色。只有当 BEA TUXEDO 域的安全级别为 APP_PW, USER_AUTH, ACL, 或者是 MANDATORY_ACL 时，才需要定义这个字段。
Application Password	BEA TUXEDO 应用的口令。只有当 BEA TUXEDO 域的安全级别为 APP_AUTH, ACL 或者是 MANDATORY_ACL 时，才需要定义这个字段。
Minimum Encryption Level	BEA TUXEDO 域与 WebLogic 服务器之间的通信所使用的最低 SSL 加密级别。可以设置为以下值：0，40，56 以及 128。缺省为 40　0 表示数据被签名但没有加密。40, 56,128 是指密钥的长度（以位为单位）。如果最低的级别都不满足，那么 WebLogic 服务器与 BEA TUXEDO 之间的 SSL 连接将失败。
Maximum Encryption Level	BEA TUXEDO 域与 WebLogic 服务器之间的通信所使用的最高 SSL 加密级别。可以设置以下值：0，40，56 以及 128。缺省为 40　0 表示数据被签名但没有加密。40, 56,128 是指密钥的长度（以位为单位）。如果最低的级别都不满足，那么 WebLogic 服务器与 BEA TUXEDO 之间的 SSL 连接将失败。
Enable Certificate Authentication	这是一个复选框，该字段决定是否启用证书验证。 缺省情况下，不使用证书验证。
Enable Security Context	这是一个复选框，如果要把 WebLogic 服务器的安全上下文传递到 BEA TUXEDO 域，那么应该启用该选项。 该属性缺省为禁用的。

6. 点 Apply 按钮保存所做的设置，然后重启 WebLogic 服务器。

7. 运行 tpusradd 命令。该命令的作用是把 WebLogic 服务器中的用户定义为在 WebLogic Enterprise 域中的授权用户

8. 为 ISL 命令设置 –E 选项。该命令的作用是使 IIOP 监听器/处理器可以察觉并使用来自 WebLogic 服务器的安全上下文。在-E 选项中，需要指定一个 principal 名。principal 名定义了 WLEC 连接池使用什么样的 principal 来登录到 WebLogic Enterpise 域。principal 名应该与建立 WLEC 连接池时的 User Name 属性匹配。

当建立 WebLogic 服务器与 WebLogic Enterprise 环境的 CORBA 对象之间连接时，使用证书验证意味着 WebLogic 服务器环境与 BEA TUXEDO 环境之间的通信执行一次新的 SSL 握手操作。如果要使一个 SSL 网络连接同时支持多个客户端请求，那么必须使用以下步骤来建立证书验证：

1．为 WebLogic 服务器的用户获取数字证书。把私钥文件保存在 BEA TUXEDO 中的 TUXDIR/udataobj/security/keys 目录中。

2　在用于 BEA TUDEDO CORBA 应用的 UBBCONFIG 文件中，使用 tpusradd 命令将 WebLogic 服务器的用户定义为 BEA TUDEXO 的用户。

3．定义 UBBCONFIG 文件的 IIOP 监听器/处理器，使用-E 选项表明用于验证的 WebLogic 服务器用户。

4．在 WebLogic 服务器的管理控制台创建一个 WLEC 连接池时，在 User Name 属性中定义了 WebLogic 服务器用户名。

5．获得 IIOP 监听器/处理器的数字证书。

6．在 ISL 命令中利用 SEC_PRINCIPAL_NAME 选项指定数字证书。使用-s 选项表明 BEA TUXEDO 域与 WebLogic 服务器安全域之间的通信使用安全端口。

有关 UBBCONFIG 文件的更多信息，参见 BEA TUXEDO 文档中的“创建配置文件”部分。

有关 corbalocs 前缀的更多信息，可以参见 BEA TUXEDO 文档中的“理解 Bootstrap 对象的地址格式”部分。

有关 BEA TUXEDO 安全级别的更多内容，可以参见 BEA TUXEDO 文档中的“定义安全级别”部分。

＼测试代用安全域与定制安全域

如果 WebLogic 服务器要使用代用安全域或定制安全域，你应该按以下步骤测试域是否可以正常工作：

1．启动管理控制台。管理控制台显示安全域中的所有用户、组与 ACL

2．在管理控制台中，为例子 HelloWorld 加上一个 ACL， 使安全域中的一个用户与一个组可以有访问 Helloworld 的权限。所选的组不要包含所选的那个用户

3．重启 WebLogic 服务器，然后用以下 URL 访问 HelloWorld

http:///*localhost*:*portnumber*/HelloWorld

输入一个没有包含在 HelloWorld 的 ACL 中的用户名与口令，这时会出现一个消息告诉你没有权限访问 HelloWorld

然后再输入一个包含在 HelloWorld 的 ACL 中的用户名与口令（可以是单个用户也可以是组中的成员），那么 HelloWorld 会被装载并显示 Hello World 的信息。

第十五章 管理事务

本章将介绍以下内容

事务管理概述

事务的配置

对事务的监控与日志记录

将服务器迁移到另一机器

本章将介绍如何用管理控制台来配置及管理事务。有关如何通过 JDBC 连接池使 JDBC 驱动器参与到分布式事务的内容，请参见 “管理 JDBC 连接”章节中的内容。

事务管理概述

Weblogic 服务器的许多功能部件，如 Java 事务编程接口(JTA)可以通过一些工具来配置，管理控制台为这些工具提供了接口。有关启动管理控制台的步骤，可以参见 “配置 Weblogic 服务器与集群”中的内容。通过设置事务属性，你可以配置事务环境的以下方面：

事务超时与期限

事务管理器的行为

事务日志文件的前缀

在配置事务环境之前，你应该熟悉事务所涉及到的 J2EE 组件，如 EJB、 JDBC 与 JMS

EJB Enterprise JavaBeans）通过 JTA 实现对事务的支持。分发描述符文件的部分内容就与事务处理相关的。有关如何用 EJB 与 JTA 进行编程的信息，可以参见 “Weblogic Enterprise JavaBeans 编程”中的内容。

JDBC 是 Java 访问关系型数据库的标准接口。JTA 通过 JDBC 驱动程序与事务数据源提供了对连接的事务支持。有关如何使用 JDBC 与 JTA 编程的信息，可以参见“Weblogic JDBC 编程”中的内容。

JMS Java 消息服务）通过 JTA 来支持涉及多个数据源的事务。Weblogic JMS 是与 XA 兼容的资源管理器。有关任何使用 JMS 与 JTA 编程的信息，请参见“WeblLogic JMS 编程”中的内容。

有关如何配置 J2EE 组件的详细信息，请参见本文档的应用部分以及管理控制台的在线帮助。

配置事务

管理控制台提供了所有 JTA 配置属性的缺省值，如果没有正确地设置事务属性，WebLogic 服务器不能正确启动。

JTA 的配置作用于整个域，即配置属性将影响该域中的所有服务器，但 JTA 的监控以及日志记录在服务器级执行。

在配置了 WebLogic JTA 以及事务参与者后，系统就可以用 JTA API 以及 WebLobi JTA 扩展来执行事务。

事务属性的配置可以是静态的也可以是动态的。静态配置就是指在运行应用运行之前配置事

务属性；动态配置是在应用运行的时候配置事务的属性。在运行应用之前，必须先设置 TransactionFilePrefix 属性。

以下是配置事务属性的步骤：

1　启动管理控制台

2　在左边的窗格中选择一个域节点，将显示缺省的 Configuration 标签页。

3　点击 JTA 标签页

4　指定每一个属性的值，或者使用缺省值

5　点 Apply 按钮保存新的属性值。

6　在配置服务器时必须设置 Transaction Log File Prefix 属性。有关设置日志属性以及有效值和缺省值的更多内容，请参见“事务的监控以及日志记录”中的内容。

表 15-1 描述了 WebLobi 服务器的事务属性，以及它们的有效值与缺省值。有关这些属性的详细信息，请在管理控制台的在线帮助中查看“域”主题下的内容。

表 15-1　事务属性

属性	描述
Timeout Seconds	在系统进行强制回滚之前，所允许的事务活动时间，以秒为单位
Abandon Timeout Seconds	事务协调器完成某一事务的最长时间，以秒为单位
Before Completion Iteration Limit	在完成 beforeCompleteion 次回调后，系统将进行强制性回滚。
Max Transactions	每个服务器上最多能有多少个并发活动事务
Max Unique Name Statices	每个服务器最多能同时跟踪多少个唯一事务名
Forget Heuristics	是一个布尔类型的值。事务管理器根据这个值决定是否让一个资源忘记任何启发式结果的事务。

事务的监控与日志记录

通过管理控制台，你可以对事务进行监控并指定事务日志文件的前缀。事务的监控及日志记录在服务器级进行，每个服务器都有自己的事务统计信息及一个事务日志文件。

显示事务统计信息及设置事务日志文件的前缀的步骤如下：

1．启动管理控制台

2．点击左窗格的 Server 节点

3．选择一个服务器

4．选 Monitoring 标签页

5.选 JTA 页面。事务的统计信息显示在 JTA 对话框中。（你也可以点击 monitoring text 链接，按照资源或名称来监视事务，或者监视所有活动事务）

6．选 Logging 标签页

7．选 JTA 标签页

8．输入事务日志文件的前缀，然后点击 Apply 按钮保存设置。

有关事务监控与日志记录的详细信息，请在管理控制台的在线帮助中查看“服务器”主题下的内容。

将服务器迁移到另一台机器中

当应用服务器迁移到另一台机器时，它必须能在新硬盘中找到事务日志文件。因此，在把服务器迁移到另一台机器后，建议你在重新启动服务器之前先把所有的日志文件迁移到新机器中。这样可以保证恢复的正常进行。在迁移后，如果事务日志文件的路径名不同与先前的路径名，那么在启动服务器前应该把 TransactionLogfilePrefix 属性值设置为日志文件的新路径。

在服务器失败后迁移事务日志时，那么启动新机器上的服务器之前，所有的事务日志文件必须都是可用的。例如，你可以把事务日志保存在一个双口硬盘中，失败前的机器与新机器都能使用它。与有计划的迁移一样，如果新老机器的事务日志文件路径不一样，那么在重启服务器前，首先要把 TransactionLogFilePrefix 属性值设为日志文件新路径。务必要记住，重启服务器之前，所有的事务日志文件都是可用的，否则，系统崩溃时所提交的事务将不能被正确解析，从而导致事务的不一致性。

第十六章 管理 JDBC 连接

本章给出通过 JDBC 组件（数据源，连接池和多池）为本地和分布式事务配置和管理数据库连接：

JDBC 管理概述

JDBC 组件 — 连接池，数据源，和多池

JDBC 配置指南

建立和管理 JDBC 连接

JDBC 管理概述

管理控制台为你配置和管理 WebLogic 服务器功能（包括 JD BC,Java 的数据库连接）用到的工具提供了界面。对于大多数 JDBC 管理功能，包括创建，管理和监视连接，系统管理员都可以使用管理控制台或者命令行。应用开发者可能希望使用 JDBC API

经常执行的设置和管理连接的任务包括：

定义在数据库和 WebLogic 服务器之间 JDBC 连接的属性

管理已建立的连接

监视已经建立的连接

关于管理控制台

首选的建立和管理 JDBC 连接的途径是通过管理控制台。使用管理控制台，你要在服务器启动之前静态的建立连接。更多的信息，请阅读“启动管理控制台”。

除了建立连接，管理控制台允许你管理和监视已建立已连接。

关于命令行界面

命令行界面提供了动态的创建和管理连接池的方法。关于如何使用命令行界面的内容，请阅读附录 B,“WebLogic 服务器命令行接口参考”。

关于 JDBC API

关于方式编程设置和管理连接的信息，请阅读http://e-docs.bea.com/wls/docs61/jdbc/index.html中的 WebLogic JDBC 编程。

附加信息

JDBC驱动程序, 用于本地和在分布式事务中的,以及与其他WebLogic Server 组件的接口和文档在多处出现。例如, 关于JDBC 驱动的信息包含在JDBC, JTA 和WebLogic jDrivers.的文档集中。

下面是 JDBC, JTA 和 Administration的附加资源:

系统管理和管理

- 关于打开管理控制台的指令，请参考 “配置WebLogic服务器和集群”
- 关于完整的JDBC 属性,参照WebLogic Administration Console Online Help ，位于 http://e-docs.bea.com/wls/docs61/ConsoleHelp/index.html.
- 关于使用命令行界面的的相关信息, 阅读附录B,“WebLogic Server 命令行接口参考”

JDBC 和 WebLogic jDrivers

下列文档主要服务于应用开发人员，系统管理员可以阅读入门资料以作为本文档的补充。

- 关于JDBC API的信息, 阅读*WebLogic JDBC 编程* “WebLogic JDBC 简介”一节对JDBC和JDBC驱动进行简要介绍。
- 关于使用WebLogic jDrivers的信息，可参阅 *Installing and Using WebLogic jDriver for Oracle* 在 http://e-docs.bea.com/wls/docs61/oracle/index.html, *Installing and Using WebLogic jDriver for Microsoft SQL Server*在 http://e-docs.bea.com/wls/docs61/mssqlserver4/index.html 或者 *Using WebLogic jDriver for Informix* 在 http://e-docs.bea.com/wls/docs61/informix4/index.html.

Transact i ons (JTA)

- 关于管理JTA的信息, 阅读15-1 页的“管理事务”
- 关于使用第三方的驱动程序，参阅在 WebLogic JTA 编程中的“WebLogic 服务器中使用第三方的 JDBC XA 驱动程序”在http://e-docs.bea.com/wls/docs61/jta/thirdpartytx.html.

下列文档主要服务于应用开发人员，系统管理员可以阅读入门资料以作为本文档的补充。

- 关于分布式事务的信息, 阅读 Programming WebLogic JTA 位于:

  http://e-docs.bea.com/wls/docs61/jta/index.html
- 关于使用WebLogic jDriver for Oracle/XA的信息, 阅读"Using WebLogic jDriver for Oracle/XA in Distributed Transactions"在 *Installing and Using WebLogic jDriver for Oracle* 在 http://e-docs.bea.com/wls/docs61/oracle/trxjdbcx.html

JDBC 组件—连接池，数据源，和多池

下面是针对 JDBC 连接组件的简单介绍，包括连接池、多池和数据源。

连接池

连接池包含一组已命名的 JDBC 连接，这些连接在连接池注册的时候创建，通常是在 WebLogic 服务器启动的时候进行。你的应用可以利用连接池中的连接，使用它，在关闭时将其返回给连接池。关于连接池的更多内容，请阅读 WebLogic JDBC 编程位于

http://e-docs.bea.com/wls/docs61/jdbc/programming.html

所有的你需要在管理控制台进行的设置都是静态的；也就是说，所有的设置都在 WLS 启动之前进行。你可以使用命令行（阅读附录“连接池管理和配置命令”），或者使用编程接口编程（阅读 Programming WebLogic JDBC 的创建动态连接池）动态（服务器启动后）创建连接池。

多池（MultiPools

用于在单一一个 WebLogic 服务器的本地（非分布式）的交易中。多池有助于：

- 负载均衡 — 连接池的增加不依赖于某一个附加的排序方式，同时可以利用轮循方式访问。当在连接间切换时，紧跟在上次所访问的连接池后的一个连接池被选中。
- 高可用性 — 建立一个有序的连接池列表以决定连接池间的切换顺序，例如，列表中的第一个连接池首先被选中，然后是第二个。

连接池中所有特定的连接都是相象的；也就是说，他们都连接到一个数据库。多池中的连接池可以和不同的 RDBMS 连接。关于多池的更多资料，请阅读 WebLogic JDBC 编程位于 http://e-docs.bea.com/wls/docs61/jdbc/programming.html

数据源

数据源对象允许 JDBC 客户端获得一个 DBMS 连接。每个数据源对象指向一个连接池或多池。数据源可以有或没有 JTA 定义，JTA 提供了对分布式事务的支持。阅读 WebLogic JDBC 编程中有关数据源的信息，位于http://e-docs.bea.com/wls/docs61/jdbc/programming.html

注： Tx 数据源不能指向多池，只能是连接池，因为多池不支持分布式交易。

JDBC 配置指南

本节将介绍如何配置本地事务与分布式事务的 JDBC 配置。

JDBC 配置概述

本节介绍了对本地和分布式交易的 JDBC 配置。在配置 JDBC 连接前，，你需要配置连接池、数据源（建议采用，但某些情况下可选），以及多池（可选）。有三种交易方式：

- 本地 —— 非分布式交易
- 带 XA 驱动的分布式交易 — 两阶段提交
- 不带 XA 驱动的分布式交易 — 单资源管理器和但数据库实例。

下表列出了如何在本地以及分布式事务中使用这些对象：

表 16-1 JDBC 配置指南

描述/对象	本地事务	分布式事务 XA 驱动程序	分布式事务 非 XA 驱动程序
JDBC 驱动程序	● Oracle, Microsoft SQL Server, 和 Informix 的 WebLogic jDriver ● 第三方的兼容驱动程序	● WebLogic jDriver for Oracle/XA ● 第三方的兼容驱动程序	● WebLogic jDriver for Oracle/XA ● 第三方的兼容驱动程序
数据源	建议使用 DataSource 对象（若没有，则使用 JDBC API	需要 TxDataSource	建议使用 TxDataSource 注意：如果超过一个资源，那么将 enable two-phase commit 属性设置为 true，参见“对 Non-XA JBDC 驱动的分布式交易”

连接池	在管理控制台配置时，需要数据源对象。	需要 TxDataSource	需要 TxDataSource
多池	在单 WebLogic 服务器配置时，需要连接池和数据源对象。	不支持分布式交易。	不支持分布式交易。

注意：使用 WebLogic jDriver for Oracle/XA 分布式交易，使用 WebLogic jDriver for Oracle 的事务模式

支持本地事务的驱动程序

支持 JDBC Core 2.0 API java.sql）的 JDBC 2.0 驱动程序有以下几种：WebLogic jDrivers for Oracle, Microsoft SQL Server 与 Informix。通过这些 API ，你可以创建类对象，以便通过数据源创建连接、进行查询、修改的信息给数据源以及处理结果集。

支持分布式事务的驱动程序

任何支持 JDBC 2.0 分布式事务标准扩展接口 javax.sql,XADataSource, javax.sql.SAConnection, javax.transaction.xa.SAResource ）的 JDBC 2.0 驱动程序，包括 WebLogic jDriver for Oracle/XA

只支持 JDBC 2.0 核心 API 但不支持 JDBC 2.0 分布式事务标准扩展接口的 JDBC 驱动程序。但一次只能有一个非 XA 的 JDBC 驱动可以参与到分布式交易中。参见“为分布式交易配置非 XA 的 JDBC 驱动程序”。

配置 JDBC 驱动程序

本节将介绍如何配置本地事务驱动程序以及分布式事务驱动程序。

如何配置支持本地事务的驱动程序

要配置支持本地事务 JDBC 驱动程序，应该对 JDBC 连接池作以下设置：

将 Driver Classname 属性设置为支持 java.sql.driver 接口的类的名字。

指定数据属性。这些属性将作为驱动的属性传递给特定的驱动程序。

有关 WebLogic 两层 JDBC 驱动程序的更多信息，可以参见 BEA 文档的以下内容：“安装及使用 WebLogic jDriver for Oracle”位于http://e-docs.bea.com/wls/docs61/oracle/index.html ，“ 安装及使用 WebLogic jDriver for Informix” 位于http://e-docs.bea.com/wls/docs61/informix4/index.html 与 “安装并使用 WebLogic jDriver for Microsoft SQL Server”位于

http://e-docs.bea.com/wls/docs61/mssqlserver4/index.html 。如果使用第三方驱动程序，请参见“WebLogic JTA 编程”中的“在 WebLogic 服务器中使用第三方的 JDBC XA 驱动”位于 http://e-docs.bea.com/wls/docs61/jta/thirdpartytx.html 以及厂商所提供的文档。下列列表演示了使用 WebLogic jDrivers 进行 JDBC 连接池和数据源配置的范例。

下表是使用 WebLogic jDriver for Oracle 的连接池配置示例。

注：增加了新的”Password”属性。该属性将覆盖在 Properties 属性中定义的口令（以名字/值对表示）。在建立物理的数据库连接时，该属性传递给两层 JDBC 驱动。该属性以加密的形式存储于 config.xml 文件中，并避免以明文形式保存于给文件中。

表 16-2 WebLogic jDriver for Oracle　JDBC 连接池配置

属性名	属性值
名字	myConnectionPool
目标	myserver
驱动程序的类名	weblogic.jdbc.oci.Driver
初始容量	0
最大容量	5
容量增长的步长	1
属性	user=scott;server=localdb
口令	tiger（该属性将覆盖在 Properties 属性中以名字/值对表示定义的口令）

下表是一个使用 WebLogic jDriver for Oracle 的数据源配置示例。

表 16-3 WebLogic jDriver for Oracle: 数据源配置

属性名	属性名字
名字	myDatasource
目标	myserver
JNDI 名字	myConnection
连接池的名字	myConnectionPool

下表是使用 WebLogic jDriver for Microsoft SQL Server 的连接池配置示例。

表 16-4 WebLogic jDriver for Microsoft SQL Server　JDBC 连接池配置

属性名	属性值
名字	myConnectionPool
目标	myserver
驱动程序的类名	weblogic.jdbc.mssqlsvr4.Driver
初始容量	0
最大容量	5
容量增长的步长	1
属性	user=sa;password=secret;db=pubs;server=myHost:1433;appname=MyApplication;hostname=myhostName

下表是一个使用 WebLogic jDriver for Microsoft SQL Server 的数据源配置示例。

表 16-5 WebLogic jDriver for Microsoft SQL Server: 数据源配置

属性名	属性名字
名字	myDatasource
目标	myserver
JNDI 名字	myConnection
连接池的名字	myConnectionPool

下表是使用 WebLogic jDriver for Informix 的连接池配置示例。

表 16-6 WebLogic jDriver for Informix　JDBC 连接池配置

属性名	属性值
名字	myConnectionPool
目标	myserver
驱动程序的类名	weblogic.jdbc.informix4.Driver
初始容量	0
最大容量	5
容量增长的步长	1
属性	user=informix;password=secret;server=my DBHost;port=1493;db=myDB

下表是一个使用 WebLogic jDriver for Informix 的数据源配置示例。

表 16-7 WebLogic jDriver for Informix: 数据源配置

属性名	属性名字
名字	myDatasource
目标	myserver
JNDI 名字	myConnection
连接池的名字	myConnectionPool

配置分布式事务的 XA JDBC 驱动程序

要使用 XA 的 JDBC 驱动程序参与到分布式事务中，需要对 JDBC 连接池做以下设置：

将 Driver Classname 属性设置为支持 javax.sql.XADataSource 接口的类的名字。

必须设置数据库属性，这些属性将作为数据源属性传递给所指定的 XADataSource。有关 WebLogic jDrive for Oracle 的数据源属性可以参见“WebLogic jDriver for Oracle/XA 数据源属性”中的内容。有关第三方驱动程序的数据源属性，可以参见供应商所提供的文档。

下表是一个使用 WebLogic jDriver for Oracle/XA 的 JDBC 连接池配置。

表 16-8 WebLogic jDriver for Oracle/XA：连接池配置

属性类型	属性值
名字	fundsXferAppPool
目标	myserver
驱动程序的类名	weblogic.jdbc.oci.xa.XADataSource
初始容量	0
最大容量	5
容量增长的步长	1
属性	user=scott;password=tiger;server=localdb

下表是一个使用 WebLogic jDriver for Oracle/XA 的 TxDataSource 的配置。

表 16-9 WebLogic jDriver for Oracle/XA　TxDataSource 配置

属性类型	属性值
名字	fundsXferData Source
目标	myserver

JNDI 名	myapp.fundsXfer
连接池的名字	fundsXferAppPool

通过配置，JDBC 连接池可以使用第三方 XA 模式的驱动程序。在这种情况下，数据源的属性是通过对 XADataSource 实例的反射（reflection）来配置的。也就是说，对于一个 abc 属性，XADataSource 实例必须支持对该名字的 set 与 get 方法，如 getAbc 与 segAbc

下表是一个使用 Oracle Thin 驱动程序的 JDBC 连接池配置。

表 16-10 Oracle Thin Driver：连接池配置

属性类型	属性值
名字	jtaXAPool
目标	myserver,server1
驱动程序的类名	oracle.jdbc.xa.client.OracleXADataSource
初始容量	1
最大容量	20
容量增长的步长	2
属性	user=scott;password=tiger; url=jdbc:oracle:thin:@baybridge:1521:bay817

下表是一个使用 Oracle Thin driver 的 TxDataSource 配置示例。

表 16-11 Oracle Thin Driver　TxDataSource 配置

属性类型	属性值
名字	jtaXADS
目标	myserver, server1
JNDI 名字	jtaXADS
连接池名字	JtaXAPool

下表是一个使用 Cloudscape 驱动程序的 JDBC 连接池配置的例子：

表 16-12 Cloudscape:连接池配置

属性类型	属性值
名字	jtaXAPool
目标	Myserver, server1
驱动程序的类名	COM.cloudscape.core.XADataSource
初始容量	1
最大容量	10
容量增长的步长	2
属性	databaseName=CloudscapeDB
是否支持本地事务	true

下表是一个使用 Cloudscape 驱动程序的 TxDataSource 配置：

表 16-13 Cloudscape　TxDataSource 配置

属性类型	属性值
名字	jtaZADS
目标	myserver, myserver1

JNDI 名字	JTAXADS
连接池名字	JtaXAPool

WebLogic jDriver for Oracle/XA 的数据源属性

表 16-14 列出了 WebLogic jDriver for Oracle 驱动程序支持的数据源属性。JDBC 2.0 列指的是某一数据源属性是否是 JDBC 2.0 的标准数据源属性（Y）或者是 WebLogic 服务器对 JDBC 的扩展 N

Option 列指某一数据源属性是否是可选的。有 Y*标记的属性对应于表 14-11 所列出的 xa_open 字符串（openString 属性的值）。如果没有指定这些属性的值，它们的缺省值将从 openString 属性获得。如果指定了这些属性的值，那么它们的值应与 openString 属性的值匹配。如果属性不匹配，在进行 XA 连接时，将引发 SQLException 异常。

那些有 N*标记的强制性属性必须与 Oracle 中的 xa_open 字符串的对应字段匹配。你在设置 Oracle xa_open 字符串时指定这些属性。如果没有指定属性或者所指定的属性与 Oracle xa_open 字符串的对应字段不匹配，那么在进行 XA 连接时将引发 SQLException 异常

表 16-14 WebLogic jDriver for Oracle/XA 的数据源属性

属性名	类型	描述	JDBC2.0	是否是可选的	缺省值
databaseName**	String	数据库的名字	Y	Y	None
dataSourceName	String	数据源的名字，用该字段命名 XADataSource	Y	Y	连接池的名字
description	String	对数据源的描述	Y	Y	None
networkProtocol**	String	与服务器通信的网络协议	Y	Y	None
password	String	数据库口令	Y	N*	None
portNumber**	int	服务器监听请求的端口号	Y	Y	None
roleName**	String	初始的 SQL 角色名	Y	Y	None
serverName	String	数据库服务器的名字	Y	Y*	None
user	String	用户的帐号	Y	N*	None
openString	String	Oracle XA 的 open string	N	Y	None
oracleXATrace	String	指明是否需要启用 XA 跟踪输出，如果启用（设置为 true），那么在启动服务器的目录下将生成一个 xa_poolnamedate.trc 形式的文件	N	Y	true

表 16-15 列出了与 Oracle 的 xa_open 字符串中的字段相匹配的数据源属性

表 16-15 与与 Oracle 的 xa_open 字符串中的字段相匹配的 JDBC 数据源属性

Oracle xa_open 字符串中的字段	JDBC 2.0 数据源属性	是否是可选的
acc	user, password	N

sqlnet	ServerName	

注意必须在 Oracle 的 xa_open 字符串中指定 Threads=true。有关 Oracle 的 xa_open 字符串中各字段的描述，可以参见 Oracle 文档

配置分布式事务的非 XA JDBC 驱动程序

在配置连接池时，如果允许非 XA JDBC 驱动程序和其他资源一起参与到分布式事务中，那么必须指定 JDBC Tx 数据源中的允许两阶段提交（Enable Two-Phase Commit）属性。（支持 XAResource 接口的资源会忽略这个参数。）注意：在分布式交易中一次只能有一个非 XA JDBC 驱动。

非 XA 驱动/单资源

若只使用一个非 XA 的驱动，它在交易中是唯一的资源，那么在控制台中不要选中 Enable Two-Phase Commit 属性（缺省为 Enable Two-Phase Commit = false ），在这种情况下交易管理器执行单阶段优化。

非 XA 驱动/对资源

如果使用一个非 XA JDBC 同时还使用其他 XA 资源，在控制台中选中 Enable Two-Phase Commit 属性（Enable Two-Phase Commit = true

如果 enableTwoPhaseCommit 属性设置为 true，那么在 XAResouce.prepare()方法调用时，非 XA JDBC 资源总是返回 XA-OK 在随后的 XAResource.commit()或 XAResource.roolback()调用中，资源会进行提交或回滚。如果资源提交或回滚失败，那么就会得到一个追溯性的错误结果，从而应用数据可能会因为追溯性的失败而出现不一致。

如果 enableTwoPhaseCommit 设置为 false,那么非 XA JDBC 资源会造成 XAResource.prepare()调用的失败。因为 commit() 在这种情况下会引发 SystemException 异常，因此这种机制确保了事务只有一个资源参与。如果只有一个资源参与到事务中，那么单阶段优化会绕过 XAResource.prepare()方法调用，大多数情况下事务都会成功。

下表是一个使用 non-XA JDBC 驱动程序的 JDBC 连接池配置示例。

表 16-16 WebLogic jDriver for Oracle/XA: 连接池配置

属性类型	属性值
名字	fundsXferAppPool
目标	myserver
URL	jdbc:weblogic.:oracle
驱动程序的类名	weblogic.jdbc.oci.Driver
初始容量	0
最大容量	5
容量的增长步长	1
属性	user=scott;password=tiger;server=localdb

下表是一个使用 non-XA JDBC 驱动程序的 TxDataSource 配置示例。

表 16-17 WebLogic jDriver for Oracle/XA: TxDataSource 配置

属性类型	属性值
名字	fundsXferDataSource

目标	myserver, server1
JNDI 名字	myapp.fundsXfer
连接池的名字	fundsXferAppPool
EnableTwoPhaseCommit	true

建立和管理 JDBC 连接

本节讨论了如何建立数据库连接

- 静态的，在服务器启动之前，在管理控制台中设定
- 动态的，在服务器启动之后，通过命令行接口。

一旦连接建立起来，你既可以用管理控制台或者命令行接口来管理和监视连接。详见表16-19所描述的任务配置和管理控制台的在线帮助链接

下表给出过程，并提供了在线帮助的链接。

JDBC 配置

通过管理控制台，你可以静态的设定连接，指定JDBC组件－连接池、数据源、Tx数据源的属性和数据库属性。见通过“管理控制台设置JDBC配置”。

数据源通常与连接池和多池相关，每个数据源应与一个特定的数据源相关联。相关亮的数据源和连接池应分配给同一个目标－或是同一个服务器，或相关的服务器/集群。不能将数据源分配给一个服务器，而连接池分配给另外一个。

更多信息请参考下表

表 16-18 配置和分配的场景

场景	相关...	分配...	目标描述
1	数据源A和连接池A	1. 数据源A分配给受管服务器1， 2. 连接池A分配给受管服务器1	数据源和连接池分配给同一目标。
2	数据源B和连接池B	1. 数据源B分配给集群X，然后 2. 连接池B分配给集群中的受管服务器2	数据源和连接池分配给相关的服务器/集群。
3	数据源C和连接池C	1. 数据源A和连接池A分配给受管服务器1 2. 数据源A分配给集群X，然后连接池A分配给集群中的受管服务器2	数据源和连接池作为一个单元分配给两个不同的目标。

（你可以为一个连接池分配多个数据源，但这没有实际意义。）你可以将数据源/连接池分配给多个服务器，但他们必需是组合分配。例如：你不可以把数据源分配个受管服务器A，而与之相关联的连接池分配给服务器B

在服务器启动后，你可以通过命令行接口动态的配置连接池。参见“用命令行接口配置JDBC”，

也可以以编程的方式动态配置连接池。参见“WebLogic JDBC编程”中的“创建动态连接池”。

服务器或集群的JDBC配置

一旦建立好关联的数据源和连接池（多池），你就可以将每个对象分配给相同的服务器或服务器/集群。通常情景如下：

- 在一个集群环境中，分配数据源给集群，分配与之相关的连接池给受管服务器。
- 在单一服务器配置中，将数据源和与之相关的连接池分配给服务器。
- 若使用多池，将连接池分配给多池，然后将数据源、所有连接池和多池分配给服务器。

见“使用管理控制台的JDBC配置任务”中，关于你所执行任务的描述。

使用管理控制台的JDBC配置任务

通过管理控制台你可以配置、管理和监视JDBC连接。完成下列步骤即可以显示你所要执行任务的标签：

1. 启动管理控制台。
2. 在左侧窗格中找到Services 节点，展开JDBC节点。
3. 选择要配置或管理的足见，如连接池、多池、数据源和Tx 数据源。
4. 按照在线帮助中的指示执行步骤，关于在线帮助的链接，见表16-19.

下表按照典型的顺序，显示建立连接所需的任务，你可以改变顺序，但要记住一定在关联或分配对象前，把该对象建立好。

表 16-19 JDBC 配置任务列表

任务#	JDBC 组件/任务	描述
1	配置连接池	在Configuration标签页，设置连接池的属性，如：名字、URL、数据库属性
2	克隆一个连接池（可选）	该任务拷贝一个连接池。在Configuration标签页中，你修改连接池的名字使之唯一，可以接受或修改其他的属性。该功能用于当你需要配置相似的连接池，其中只是名称不同时情况。例如你需要每个数据库管理员使用不同的连接池来跟踪每一个数据库的变化。
3	配置多池（可选）	在MultiPools标签页，设定属性，包括名字、算法类型（或者是高可用性或是负载均衡）。在Pool标签页上，将连接池分配给该多池。
4	配置数据源（并同连接池关联）	使用Data Source 标签页，设定数据源的属性，包括名字，JNDI名字，连接池的名字（通过它来关联或分配，数据源和某一连接池或多池）。
5	配置Tx 数据源（并同连接池关联）	使用 Tx Data Source 标签页，设定数据源的属性，包括名字，JNDI名字，连接池的名字（通过它来关联或分配，数据源和某一连接池）
6	为服务器/集群指定连接池	在Target标签页中，将连接池分配给服务器或集群。
7	为服务器指定多池	在Target标签页中，将配置的多池分配给服务器。

使用命令行接口的JDBC配置任务

下表显示动态建立连接池的方法。

表16-20　动态建立连接

如果你试图...	那么使用...
创建动态连接池	■ 命令行 — 见B-20“CREATE_POOL” ■ API — 请看“WebLogic JDBC编程”的“配置WebLogic JDBC特性”

详细信息参见B-1中的“WebLogic服务器命令行接口参考”和“WebLogic JDBC编程”中的“动态创建连接池”。

管理和监视连接

对于连接的管理，包括启用，禁止和删除已建立的JDBC组件

使用管理控制台管理JDBC

管理和监视JDBC连接，见下表：

表16-21 JDBC 管理任务

如果你试图...	在管理控制台中做...
删除连接池	见在线帮助中的删除一个连接池。
删除多池	1. 在左侧窗格中选择 MultiPool 节点，域中所有的多池会列在右侧的窗格中。 2. 点击要删除的多池的“Delete”图标，右侧窗格会显示一个删除请求的确认对话框。 3. 点击 yes 删除多池，该多池会在 MultiPool 节点下被删除。
监视连接池	1. 在左侧窗格中选择连接池 2. 在右侧选择 Monitoring 标签页，然后选择 Monitor All Active Pool 链接。
对连接池、多池和数据源修改属性	1. 左侧窗格中选择 JDBC 对象（连接池、多池或数据源） 2. 右侧窗格中选择 Target 标签页，取消对象的分配（将对象从 Chosen 列移动到 Available 列中），然后点击 Apply。该步骤使服务器不再使用 JDBC 对象（连接池、多池或数据源）。 3. 选择要修改或改变的属性标签页。 4. 选择 Target 标签页，然后重新将对象分配给服务器。这将使相应的服务器可以使用

JDBC 对象（连接池、多池或数据源）。

使用命令行接口管理JDBC

下表描述了使用命令行接口对连接池进行管理。详细信息见相应命令。

关于使用连接池命令的更多信息，参见B-1中的“WebLogic 服务器命令行接口参考”。

表16－22 使用命令行接口管理JDBC

如果你试图...	那么会用到命令...
禁止连接池	**DISABLE_POOL**
使能已禁止连接池	**ENABLE_POOL**
删除连接池	**DESTROY_POOL**
确认连接池是否创建	**EXISTS_POOL**
复位连接池	**RESET_POOL**

第十七章 管理 JMS

本章将介绍如何管理 WebLogic JMS，包括以下内容：

配置 JMS

监控 JMS

恢复失败的 WebLogic 服务器

配置 JMS

在管理控制台中可以配置 JMS 的以下属性：

启用 JMS

创建 JMS 服务器

创建或定制 JMS 服务器、连接工厂、目的方(队列与主题)、目的方模板、 目的方主键、存储库、会话池、连接使用者。

建立客户 JMS 应用。

定义阀值和配额

启用需要的 JMS 特性，如服务器集群（见下节），并发消息处理，目的方的排序，以及消息的持久化。

有一些 WebLogic JMS 属性有缺省值，而其它属性必需要配置。如果没有正确配置这些属性或没有提供缺省值时，那么重启时，WebLogic 服务器不会启动 JMS。在 WebLogic 产品中有一个 JMS 配置示例。

当你从老版本迁移到 6.1 版本时，WebLogic 服务器会自动转换配置信息。你可以参见“WebLogic JMS 编程指南”的“应用移植”部分的说明。

以下是配置 WebLogic JMS 属性的步骤：

1．启动管理控制台。

2．在左窗格中选择“Services”下面的“JMS”按钮把列表展开。

3．按照以下内容或管理控制台在线帮助中所描述的步骤来创建及配置 JMS 对象。

配置完 WebLogic JMS 后，应用程序可以开始用 JMS API 发送或接收消息。关如何开发 WebLogic JMS 应用的更多信息，请见“WebLogic JMS 编程指南”。

注意：“WebLogic JMS 编程指南”提供了一个 JMS 属性配置列表，你可以参照这个表中的必选项和支持不同其他 JMS 功能的可选项信息，以便于规划自身应用的 WebLogic JMS 的配置。

配置连接工厂

JMS 客户端可以通过连接工厂（Connection factory）创建 JMS 连接。连接工厂根据预先定义的属性来创建连接。

要配置连接工厂，需要使用管理控制台中的 Connection Factories 节点进行如下步骤：

需要配置的属性包括：

1．连接工厂的名字

2　JNDI 命名空间中的连接工厂的名字。

3．长期订阅方可以使用的客户端标识 (Client ID) (关于长期订阅方的更多内容，请参考“WebLogic JMS 编程”)

4．消息传递的缺省属性（包括：优先级、存活期、传输时间及模式）

5. 异步会话的最大重要消息数与过载策略(即消息数达到该最大值时，对于广播会话所采取的行动)

6. 是否允许在 onMessage()方法中调用 close()方法

7. 事务属性（事务超时以及是否可以使用 JTA 用户事务，连接工厂是否可以返回一个 XA 或非 XA 主题）

连接工厂关联的目标服务器（WebLogic Servers），用以支持集群。可以用目标服务器限制需要部署连接工厂的服务器、组，或者集群。

WebLogic JMS 定义了一个缺省的连接工厂：weblogic.jms.ConnectionFactory。该连接工厂的所有配置属性都使用缺省值。管理控制台在线帮助的“JMS 连接工厂”描述了配置属性的缺省值。

如果缺省连接工厂能满足应用的需求，就不需要为应用配置其它连接工厂。

注意：如果用用缺省的连接工厂，那么你不能控制连接工厂将部署到哪个 JMS 服务器上。如果希望把连接工厂部署到一特定的 JMS 服务器上，那么需要新建一个连接工厂，然后把它部署到这个 JMS 服务器上。

有关创建与配置连接工厂的说明，可以参考管理控制台在线帮助中的“JMS 连接工厂”。

某些连接工厂属性可以动态配置。如果在运行时改变了动态属性的值，那么新配置只对新连接有效而不影响已经建立的连接。

配置模板

可以用模板定义多个具有相似属性设置的目的方（destination）。使用模板有以下好处：

在每次定义新的目的方时，不需要设置所有的属性。你可以先使用模板，然后覆盖那些需要重新设置的属性。

如果要动态改变共享的属性设置，仅需修改模板。

目的方模板的属性在管理控制台的“Templates”节点设置。目的方模板的属性与目的方配置中的相同。通常目的方会继承所使用的目的方模板的属性配置，但不包括以下情况：

如果使用了模板的目的方重新设置了某个属性，那么新值有效。

如果使用了模板的目的方将消息 redelivery 的值设为一个属性，那么就会使用 redelivery 的值。

目的方不会继承目的方模板的 Name 属性，因为 Name 属性只对目的方模板有效。每个目的方必须明确定义一个唯一的名字。

目的方模板没有定义 JNDI Name, Enable Store 以及 Template 属性。

目的方模板没有定义 Multicast 属性，因为该属性只对主题（topics）有效。

任何没有被明确设置的目的方属性使用缺省值。如果一个属性没有缺省值，那么必须为目的

方模板或目的方设置这个属性的值。如果没有设置，那么不完整的配置信息将导致 WebLogic JMS 配置失败，从而使 WebLogic JMS 不能启动。

有关如何创建与配置模板的详细信息，可以参考管理控制台在线帮助的“JMS 模板”。

配置目的方主键

目的方的的排序方式用目的方主键（Destination Key）来定义。

利用管理控制台的 Destination Keys 节点创建一个新的目的方主键，并配置目的方主键的以下属性：

目的方主键的名字

用来排序的属性名

主键的类型

排序的方向（升序还是降序）

有关如何创建与配置目的方主键的详细信息，可以参考管理控制台在线帮助中的“JMS 目的方主键”。

配置存储库

存储库（Backing Store）可以由文件或数据库构成，用于消息的持久保存。

JMS 通过 JDBC 把消息保存在数据库中或从通过 JDBC 连接池来访问持久化的消息。JMS 数据库可以是任何可以用 JDBC 驱动访问的数据库。WebLogic 支持并提供以下数据库的 JDBC 驱动程序。

Cloudscape

Informix

Microsoft SQL(MSSQL) Server(6.5 或 7 版)

Oracle(8.1.6 版)

Sybase(12 版)

你可以用 ACL 限制 JDBC 连接池的使用。若限制了该 ACL，你需要在 ACL 中加入 WebLogic 系统用户以及任何需要发送 JMS 消息的用户。详细内容，参见《**WebLogic 管理指南**》的“安全管理”部分。

注意：WebLogic 的 JMS 例子被设置为使用 Cloudscape Java 数据库。WebLogic 服务器提供了该数据库的评估版本以及一个 demoPool 数据库。

要创建一个文件或数据库存储库，需要在管理控制台的 Store 节点下设置以下属性：

存储库的名字

文件存储库所在的目录（对于文件存储库而言）。

JDBC 连接池以及用于多实例的数据库表名前缀（相对于 JDBC 数据库存储库而言）。

警告：不能将一个 XA 的连接池配置给 JDBC 数据源

注：随着存储的消息增多，使用 JMS 存储库会增加 WebLogic 服务器对内存的需求。在重新

启动 WebLogic 服务器时，可能会因为内存不足而造成 JMS 初试化失败，这时就需要根据 JMS 存储库中有效的消息数量来增加虚拟机中堆内存的大小，然后重新启动 WebLogic 服务器。

关于 JMS 存储库

JMS 数据库中包含了两张自动创建的系统表，它们供 JMS 内部使用。

- <prefix>JMSStore
- <prefix>JMSState

前缀名用于在存储库中唯一确定 JMS 表。指定唯一前缀可以允许在同一个数据库中存在多个存储库。前缀的配置是在管理控制台中配置 JDBC 仓库时设置的。当 DBMS 需要表的全名时或者必需区分两个 WebLogic 服务器使用的 JMS 表时，前缀附加在 JMS 表名前，从而允许在一个 DBMS 中存储多个 JMS 表。

前缀使用下列指定的格式，当它加于 JMS 表名前，会生成一个合法的表名：

[[catalog.]schema.]prfix

警告：不允许两个 JDBC 存储库使用相同的数据表，否则会造成数据损坏。

关于建立和配置存储库，参见管理控制台中的在线帮助中关于“JMS 文件仓库”和“JMS JDBC 备份仓库”部分以获得关于文件和 JDBC 存储库的信息。

关于 JMS 存储库中 JDBC 连接池的建议

若使用 JDBC 存储库，而 DBMS 关闭之后重新启动，直至 WebLogic 关闭并重新启动之前，那么 JMS 则不能再访问存储库。要修正此问题，在配置用于 JMS 存储库的 JDBC 连接池时做如下设定：

TestConnectionsOnReserve = “true”

TestTableName = “ [[[catalog.]schema.]prefix.]JMSState”

否则，如果 JDBC 资源关闭再重起，直至 WebLogic 关闭并重新启动之前，JMS 不能重用它。

配置 JMS 服务器

JMS 服务器管理连接以及用户方的消息请求。

创建 JMS 服务器前，需要在管理控制台的“Servers”节点中定义以下内容：

配置属性：

1 JMS 服务器的名字

2．保存消息的备份库(文件类型的或 JDBC 数据库型的)。如果没有设置备份库，那么服务器不支持消息的持久化。

3．用来创建临时目的方的模板，临时目的方可以是临时队列也可以是临时主题。

4．消息数量和尺寸的阀值和配额 (最大数量，上下限)

指定使用 JMS 服务器的 WebLogic 服务器，当目标 WebLogic 服务器启动，JMD 服务器也随

之启动，若未指定目标的 WebLogic 服务器，则 JMS 服务器不会启动。

注意：部署 JMS 服务器与部署连接工厂或模板不一样。一个 JMS 服务器只能部署在一个服务器上，而连接工厂与模板可以同时在多个服务器上实例化。

有关如何创建以及配置 JMS 服务器的内容，请参见管理控制台在线帮助的“JMS 服务器”部分。

配置目的方

一个目的方（Destination）就是一个队列（Queue）或一个主题（Topic），定义完 JMS 服务器后，可以定义该服务器的目的方。一个 JMS 服务器可以配置多个目的方。

可以直接定义目的方，也可以通过模板生成目的方。前面讲过，目的方模板用来定义具有相似配置的目的方。

你可以在管理控制台的 Destiation 节点直接配置目的方。以下是需要配置的属性：

目的方的名字与类型（队列或主题）

JNDI 命名空间中目的方的名字

是否用存储库保存持久化消息

用来创建目的方的模板

定义目的方排序的主键

消息数量和尺寸的阀值和配额 （最大数量，上下限阀值）

可以被重载的属性（如优先级，存活期（time-to-live），传输时间以及传递模式）

消息重传的属性，包括用于覆盖重发的延时，重发限制，出错的目的方等

广播属性，包括广播地址，端口与存活期(指对主题有意义)

有关如何创建与配置目的方的内容，可以参见管理控制台在线帮助的“JMS 目的方”。

某些目的方属性可以动态配置。在运行时所改变的属性配置，只影响以后的消息会受影响，原来的消息不受影响。

配置会话池

通过会话池（Session Pool），应用可以并发地处理消息。定义了 JMS 服务器后，就可以为其设置会话池。可以为每个 JMS 服务器配置一或多个会话池。

你可以用管理终端的 Session Pools 节点来定义以下属性：

会话池的名字。

会话池所关联的连接工厂。连接工厂用于创建会话。

用于并发接收及处理消息的消息监听类。

事务属性（确认模式以及是否允许会话池创建事务性会话））

最大并发事务数

有关创建及配置会话池的内容，请参见管理控制台在线帮助的“JMS 会话池”部分。

某些会话池属性可以动态设置，但只有重启会话池后新设置才会生效。

配置连接使用者

连接使用者（Connection Consumer）获得服务器会话并对消息进行处理。在定义了会话池后，你可以配置该会话池的连接使用者。可以为每个会话池定义一或多个连接使用者。你可以在管理管理控制台的“Session Pools”节点来配置连接使用者的以下属性：

连接使用者的名字

连接使用者可以收集的最大消息数

过滤消息的 JMS 选择器表达式（selector expression），有关选择器表达式的信息，请参见“WebLogic JMS 编程指南”中的相关内容。

连接使用者所监听的目的方

有关创建及配置连接使用者的详细内容，请参见管理控制台在线帮助的“JMS 连接使用者”部分。

监控 JMS

你可以通过管理控制台了解以下 JMS 对象的统计信息：JMS 服务器，连接，会话，目的方，消息生产者，消息使用者以及服务器会话池，长久订阅者。

你可以通过管理控制台来监视 JMS 的统计信息。

注： 关于监视 WebLogic 服务器的 JMS 连接的指示，请参阅管理控制台的在线帮助中关于服务器的部分。

监视 JMS 对象

按照以下步骤来查看 JMS 的监控信息：

1. 启动管理控制台
2. 在左边窗格中选择 Services 节点下的 JMS 节点以打开 JMS 服务列表。
3. 在左边窗格中选择 JMS 下的 JMS Server 节点。有关 JMS 服务器的信息显示在右边的窗格中。
4. 从 JMS 服务器列表或者右边窗格中的服务器中选择一个要监控的 JMS 服务器。

   右边窗格将显示该 JMS 服务器的信息。
5. 选择 Monitoring 标签页，该页面会显示所选 JMS 服务器的监控数据。

有关 JMS 服务器监控的详细信息，请参见管理控制台的在线帮助。

监视长期订阅者

若监视运行于某一主题的长期订阅者的信息：

1. 执行上一节所讲述的步骤 1-3
2. 早左侧窗格中选 Server 节点，展开列出所有的 JMS 主题和队列的目的方

   JMS 目的方的信息以表格的形式显示在右侧窗格中，在该表中也列出对某一目标主

题的长期订阅者数目（如果有，且运行的话）的信息。

3. 要查看某一特定的主题的长期订阅者信息，点击该主题中 Durable Subscriber Runtimes 列中的图标（或实际数字）

有关 JMS 服务器监控的详细信息，请参见管理控制台的在线帮助。

恢复失败的 WebLogic 服务器

以下部分的内容将介绍如何在系统失败时重启或替换 WebLogic 服务器，以及从编程的角度考虑如何在服务器失败的情况下正常地结束应用。

重启或替换 WebLogic 服务器

当 WebLogic 服务器失败时，有三种途径恢复系统：

重启失败的服务器

用与失败服务器相同的 IP 地址来启动一台新的服务器。

用与失败服务器不同的 IP 地址来启动一台新的服务器。

要重启失败的服务器，或者使用与失败服务器相同的 IP 地址来启动一台新的服务器，必须先重启服务器，然后启动服务器进程。有关重启 WebLogic 的内容可以参见“启动与结束 WebLogic 服务器”

要使用与失败服务器不同的 IP 地址来启动一台新的服务器，使用以下步骤：

1．修改域名服务（Domain Name Service，简称为 DNS），使服务器别名指向一个新的 IP 地址。

2. 重启服务器与服务器进程。可以参见“启动与结束 WebLogic 服务器”中的内容。

3．然后，你可以选择执行以下任务：

应用所采用的技术	执行的任务
消息持久化——使用 JDBC 存储库	1．如过 JDBC 数据库位于失败的服务器上，那么应该把数据库迁移到新的服务器上并将 JDBC 连接池的 URL 属性设置为数据库的新位置。 2．如果 JDBC 数据库不在失败的服务器上，那么对数据库的访问不受到影响，因此不需要做任何改变。
消息持久化——使用文件存储库	将文件迁移到新服务器上，注意主目录（home directory）的路径名应该与原来服务器的一样。
事务	如果你用其它 IP 地址启动 WebLogic 服务器，那么 把所有名字为<servername>.tlog 的事务日志文件迁移到新服务器。这可以通过一个双口硬盘，或手工复制文件来实现。 如果迁移后的日志文件所在目录与原来服务器的不同，那么在启动服务器之前，你应该更改服务器的 **TransactionLogFilePrefic** 属性。 注意，如果因为系统崩溃而需要进行迁移，那么在新位置重启服务器之前，必须保证日志文件可用，否则在系统崩溃时正在提交的事务将不能正确解析，从而引起数据不一致。 所有未提交的事务都将被回滚。

注意：随着 JMS 备份库中的消息的增多，初始化 WebLogic 服务器所使用的内存也随之增多。因此如果服务器重启时，因为内存不足而引起的服务器初始化失败，可以通过增加虚拟机的

堆大小来解决，所增加的比例取决于当前 JMS 备份库中的消息数量。

编程考虑

通过编程你可以在 WebLogic 服务器失败时正常地结束应用。例如：

如果服务器失败并且...	那么...
你连接到一个失败了的 WebLogic 服务器	JMSException 异常将会被传递到连接异常监听器中。重启或替换了失败服务器后，应重启应用。
你没有连接到一个失败的服务器	在服务器重启或被取代时，必须重新建立所有组件
JMS 服务器位于一个失败了的 WebLogic 服务器上	ConsumerClosedException 将传递到会话异常监听器。你可以重新创建任何丢失了的消息使用者。

第十八章 管理 JNDI

本章介绍以下内容：

如何将对象装载到 JNDI 树中

查看 JNDI 树

JNDI 管理概述

你可以通过管理控制台管理 JNDI.JNDI API 允许应用以名字的方式寻找对象- 包括数据源、EJB、JMS 和 Mail 对话。JNDI 树表示在管理控制台左侧的窗格中。

JNDI 管理任务包括：

将对象加载到 JNDI 树中

查看 JNDI 树

将对象装载到 JNDI 树

用 WebLogic JNDI 访问对象之前，应该把对象装载到 WebLogic 服务器的 JNDI 树中。你可以用 WebLogic 服务器的管理控制台将 WebLogic 服务器的 J2EE 服务装载到 JNDI 树中。EJB, RMI, JMS, JDBC 以及 Mail 对象也都可以放在 JNDI 树中。这些对象均含有 JNDI 属性域，你只需要填入名字。

在把对象装载到 JNDI 树之前，应该先确定对象在 JNDI 名字，然后创建对象时，在 JNDI Name 字段输入对象的 JNDI 名。

查看 JNDI 树

有时，查看 WebLogic 服务器 JNDI 树中的对象会对你有所帮助。要查看服务器上的 JNDI 树：

1. 在左侧窗格中右键点击服务器节点，显示一个弹出式菜单。
2. 选择 JNDI 树，该服务器的 JNDI 树即显示在右侧的窗格中。

第十九章 管理 WebLogic J2EE 连接器构架

WebLogic J2EE 连接器构架基于 Sun Microsystems J2EE 连接器规范, Version1 1.0, Proposed Final Draft2, 它可以将 J2EE 平台与一或多个异种企业信息系统（Enterprise Information Systems,简写为 EIS）。接下来的内容将解释如何管理 WebLogic J2EE 连接器构架：

WebLogic J2EE 连接器构架概述

安装资源适配器（Resource Adapter

配置与部署资源适配器

监控连接池与参数

删除一个资源适配器

编辑资源适配器分发描述符（Resource Adapter Deployment Descriptors

有关 BEA WebLogic J2EE 连接器体系结构的信息，可以参考“WebLogic J2EE 体系结构编程”

WebLogic J2EE 连接器构架概述

BEA WebLoic 服务器继续建立在实施 Sun MicrosSystems J2EE Platform Specification,version 1.3 规范上。J2EE 连接器构架的作用是将简化的企业信息系统集成到 J2EE 平台，目的是为了借助于 J2EE 平台的优势——包括组件模型，事务与安全设施来解决 EIS 集成中出现的问题。

J2EE 连接器构架提供了多个应用服务器与 EIS 集成的 Java 解决方案。借助于 J2EE 连接器构架，EIS 供应商不再需要为每个应用服务器定制它们的产品，同时遵循 J2EE 连接器规范的应用服务器供应商（如 BEA WebLogic 服务器）如果想扩展它的应用服务器以支持对新 EIS 的连接也不需要添加定制代码。

J2EE 连接器构架使 EIS 供应商为它们的 EIS 产品提供标准的资源适配器。资源适配器作为应用服务器（如 WebLogic 服务器）的插件提供了 EIS 与应用服务器集成的基础设施。

应用服务器供应商（例如 BEA WebLogic Server）只需通过对 J2EE 连接器架构的支持，即可扩展其系统功能，就可以保证与各种 EISes 系统的连接了。另一方面，EIS 供应商应该提供一个标准的资源适配器，该适配器可以插入到任何支持 J2EE 连接器构架的应用服务器中。

安装资源适配器

本节将讨论如何用管理控制台将新的资源适配器安装到 WebLogic 服务器中。

1．启动 WebLogic 服务器

2．打开管理控制台

3．打开要处理的域

4．在 Deployments 下，右点左窗格中的 Connectors 将显示一个弹出菜单。

5．选择 Install a New Connector Component

6．在文本字段中输入资源适配器（.rar）的路径，或点击 Browse 按钮浏览你的文件系统并选择所要安装的资源适配器

7．点 Upload 按钮安装资源适配器。新安装的资源适配器被加到左窗格中的 Connectors 节点下。

配置与部署资源适配器

本节将介绍如何在管理控制台中配置与部署一个新的资源适配器。

与部署相关的信息，可以参考“WebLogic J2EE 连接器构架编程”的第 3 章“资源适配器”。

配置与部署资源适配器

以下是在管理控制台中配置与部署资源适配器的步骤：

1．启动 WebLogic 服务器

2．打开管理控制台

3．打开你要进行工作的那个域

4．在左窗格的 Deployments 下，选择 Connectors Connector Deployments 表格显示在由窗格中。该表格列出了所有已经部署了的连接器（资源适配器）

5．选择 Configure a new Connector Component

6．输入以下信息：

Name-连接器组件的名字，你可以不使用连接器的缺省名而另给一个名字。

URI Uniform Resource Identifier,统一资源定位符）——输入连接器组件的 URI

Path——输入资源适配器.rar 文件的路径

Deployed——指出是否需要在创建完后就部署资源适配器.rar 文件。

7．点 Create 按钮

8．新的资源适配器显示在右窗格中的 Deployments 表格中

查看已部署的资源适配器

以下是在管理控制台中查看已经部署了的资源适配器：

1．在管理控制台左窗格的 Deployments 节点下，选择 Connectors

2 Connector Deployments 表格显示在右窗格中。该表格列出了所有已经部署的连接器。

卸载已部署的资源适配器

在管理控制台卸载一个已部署的资源适配器的步骤如下：

1．在管理控制台左窗格的 Deployments 节点下，选择 Connectors

2．在 Connector Deployments 表格中，选择一个要卸载的连接器

3．在 Configuration 标签页中，不要选 Deployed 复选框

4．点 Apply

卸载一个资源适配器并未将该适配器的名字从 WebLogic 服务器中删除。一旦资源适配器卸载了并且你未做任何变动，该资源适配器在服务器整个会话期间一直保持卸载状态，直到服务器重新启动前，你都不可以利用该部署名和有关参数来部署新的资源适配器，你可以重用部

署名来更新部署，就象“更新已部署的资源适配器”中所描述的那样。

更新已部署的资源适配器

当你更改了一个已经部署到 Weblogic 服务器的资源适配器.jar 文件或部署目录的内容时，只有完成了以下任务，所作的更新才会反映到 WebLogic 服务器中：

重启服务器（如果.rar 或目录会被自动部署）或

在管理控制台中更新资源适配器的部署

如果从管理控制台中更新资源适配器的部署，应该完成以下步骤：

1．在管理控制台的 Deployments 下，选择左窗格中的 Connectors(Resource Adapters)

2．在 Connector Deployments 表格中，选择要更新的连接器

3．更改连接器的名字或部署状态

4．点 Apply 按钮

监视资源适配器

监视一个连接器运行时所用的所有连接池，采用如下步骤：

1. 在控制台左侧窗格中选择所要监视的连接器。

2. 点击鼠标右键，从弹出式菜单中选择“Connector Connection Pool Runtimes”

所选的连接池的运行信息会显示在右侧的窗格中。

注：你也可以通过管理控制台右窗格来访问这些信息。在右侧的窗格的连接器列表中选择所有监视的连接器，然后选择 Monitoring 标签页中的”Monitoring All Connector…”

删除一个资源适配器

在管理控制台中删除资源适配器有两种方式：

在左侧窗格中的 Deployments→Connectors→Connector 选择要删除的连接器名

在右侧窗格中连接器列表中，选择垃圾筒图标，右侧窗格中会显示：

Are you sure you want to permanently delete <Connector Name>

from the domain configuration?

点击 yes 来删除连接器

编辑资源适配器分发描述符

本节将说明如何用管理控制台的分发符描述符来编辑以下资源适配器分发描述符：

ra.xml

weblogic-ra.xml

有关资源适配器的分发描述符元素的详细信息，可以参考“WebLogic J2EE 连接器构架编程”。

以下是编辑资源适配器分发描述符的步骤：

1．在浏览器中用以下 URL 打开管理控制台：

http://host:port/console

其中 host 是运行 WebLogic 服务器的机器的计算机名，port 是 WebLogic 服务器的监听端口号。

2．展开左窗格的 Deployments 节点

3．展开 Deployments 节点下的 Connectors 节点

4. 右点需要编辑分发描述符的资源适配器的名字，然后从下拉菜单中选择 Edit Connector Descriptor 菜单。

管理控制台窗口显示在一个新的浏览器窗口中。左窗格的树结构列出了两个资源适配器分发描述符的所有元素。右窗格的表单中列出了 ra.xml 文件的描述符元素。

5．如果要编辑、删除或添加资源适配器分发描述符中的元素，那么在左窗格中展开需要编辑的分发描述符文件对应的节点。

RA 节点包含了 ra.xml 分发描述符中的元素

WebLogic RA 节点包含了 weblogic-ra.xm;分发描述符中的元素

6．要编辑分发描述符文件中的元素：

a. 在左窗格的树中，依次展开各父元素直到找到需要编辑的元素为止。

b. 点击该元素。显示在右窗格中的表单列出了该元素的属性或该元素的子元素

c. 编辑表单中的文本文字

d. 点 Apply 按钮。

7．在资源适配器的分发描述符中添加元素的步骤如下：

a. 在左窗格的树中，依次展开各父元素直到找到需要创建的元素名为止。

b. 右键点击元素，并从下拉菜单中选择 Configure a New Element 选项

c. 在右窗格的表单中输入元素的信息

d. 点 Create 按钮

8．以下步骤可以用来删除资源适配器描述符文件中的元素：

a. 在左窗格的树中，依次展开各父元素直到找到需要删除的元素为止。

b. 右键点击该元素，并从下拉菜单中选择 Delete Element 菜单项。

C. 点 Yes 确认删除

9．在修改完资源描述符后，点击左窗格中的根元素。根元素可能是资源适配器*.rar 包文件的名字也可能是资源适配器的名字

10．点 Validate 以保证资源适配器描述符中的条目是有效的。

11．点 Persist 将你对分发描述符文件所作的编辑写到硬盘以及 WebLogic Server 的内存中

第二十章 管理 WebLogic 服务器许可证

运行 WebLogic 需要一个有效的许可证。下面的内容将介绍如何安装及更新 WebLogic 许可证:

安装 WebLogic 许可证

更新许可证

安装 WebLogic 许可证

WebLogic 服务器的评估版可以使用 30 天。此后，如果希望继续使用 WebLogic 服务器，你应该与销售人员商讨继续使用评估版还是分别对每一个 IP 地址购买 WebLogic 服务器许可证。所有 WebLogic 服务器的评估版产品只允许在一台服务器上使用并最多可以接受三个具有不同 IP 地址的客户端的访问。

从 BEA 网站下载的 WebLogic 服务器中包含了评估许可证。安装时，WebLogic 服务器的安装程序会要求指定 BEA 主目录的位置，BEA 的许可证文件 license.bea 将安装在这个目录下。

更新许可证

如果出现以下情况，那么你需要更新 BEA 许可证文件

购买了其它 BEA 的软件

获得包含新产品的分发

已经申请并收到可以延长使用评估版的许可证。

在上述情况下，新的许可证文件将以邮件附件的形式分发给你。然后你可以按照以下步骤来更新你的 BEA 许可证文件。

1．将许可证更新文件保存在 BEA 的主目录下，注意不要用 license.bea 作为该文件的名字

2 Java(Java 2)应该包含在路径中，以下命令可以把 JDK 加入到路径中：

set PATH=.\jdk130\bin:%PATH% (Windows 系统)

set PATH=./jdk130/bin:$PATH (UNIX 系统)

3．在命令行方式下，进入 BEA 的主目录，执行以下命令：

UpdateLicense license_update_file

其中 license_update_file 就是你通过邮件获得并保存在 home 目录下的许可证更新文件。运行该命令将会更新 license.bea 文件。

4．备份 license.bea，并将它保存在一个安全的地方。即使别人不会使用你的许可证文件，也应该保护这些信息，避免遭受恶意或因无知而造成的破坏。

A. 使用 WebLogic Java 工具

WebLogic 提供了几个 Java 应用程序用以简化安装与配置任务、提供服务以及提供方便的快捷方式。本章描述了 WebLogic 所提供的这些工具。本章给出了所有工具的命令行语法，有些还提供了示例。本章描述了以下工具：

AppletArchiver

Conversion

der2pem

dbping

deploy

getProperty

logToZip

MulticastTest

myip

pem2der

Schema

showLicenses

system

t3dbping

verboseToZip

version

writeLicense

在使用这些工具之前必须先正确地设置 CLASSPATH 。详细信息，请参见“设置类路径选项”。

AppletArchiver

AppletArchiver 工具在一个单独的框架中运行 applet，并记录所有下被 applet 下载的类与 applet 使用的资源，然后将它们打包在一个.jar 文件或.cab 文件中（cabarc 工具可以从 Microsoft 得到）

语法

$ java utils.applet.archiver.AppletArchiver *URL filename*

参数	定义
URL	Applet 的 URL
filename	打包后的本地.jar/.cab 文件名

Conversion

如果你使用过以前版本的 WebLogic，那么必须要转换 weblogic.properties 文件。在管理控制台在线帮助的“Conversion”部分中介绍了如何使用转换脚本来转换文件。

Der2pem

Der2pem 工具用来把 DER 格式的 X509 证书转换为 PRM 格式。转换所得到的.pem 格式的文件与原.der 文件位于同一个目录中。

格式：

$ java utils.der2pem derFile [headerFile][footerFile]

参数	描述
derFile	要进行转换的文件名。该文件的扩展名必须为.der 扩展，并且.der 文件中必须有一个有效的证书。
headerFile	放在 PEM 文件中的文件头。缺省的文件头为“------BEGIN CERTIFICATE-------” 如果要进行转换的 DER 文件是一个私钥文件，那么应该创建一个头文件，该头文件包含以下内容之一： 如果是未加密的私钥文件，那么为“-----BEGIN RSA PRIVATE KEY-----” 如果是加密的私钥文件，那么为“-----BEGIN ENCRYPTED PRIVATE KEY” 注意：文件头所在行的结尾应该另起一行。
footerFile	放在 PEM 文件中的尾。缺省的文件尾为“-----END CERTIFICATE-----”。 如果所转换的 DER 文件是一个私钥文件，那么应该使用一个脚注文件。所创建的脚注文件应该包含以下内容之一 如果是一个未加密的私钥，那么为“-----END RSA PRIVATE KEY-----” 如果是加密的私钥文件，那么应该为“-----END ENCRYPTED PRIVATE KEY-----”。 注意：在文件尾所在行应该另起一行。

例子

$ java utils.der2pem graceland_org.der

Decoding

.....

.....

.....

.....

.....

dbping

dbping 命令行工具两层 WebLogic jDriver 来测试 DBMS 与客户端之间的连接。

语法

$ java utils.dbping *DBMS user password DB*

参数	定义
DBMS	该参数为 MSSQLSERVER4, ORACLE, 或 INFORMIX4
user	登录用的有效用户名。可以用 isql 或 sqlplus 中所使用的用户名
password	用户的口令。可以用 isql 或 sqlplus 所使用的口令。
DB	数据库的文件名，可以用 isql 或 sqlplus 所使用的值。

Deploy

Deploy 工具从一个包文件（.jar、.war 或 .ear）中得到 J2EE 应用，并把该应用部署到正在运行着的 WebLogic 服务器中。更多的信息可以参见“组装并配置 Web 应用”以及编程指南中的“开发 WebLogic 服务器应用”中的内容。

参数

$ java weblogic.deploy *[option*s] [action] password {application name} {source}

Actions(从下表中选一个)

Action	描述
delete	按给出的应用名删除一个应用
deploy	将一个 J2EE 应用.jar、.war、.rar 或.ear 文件部署到指定的服务器中
list	列出指定 WebLogic 服务器中的所有应用
undeploy	删除指定服务器上的一个应用
update	在指定服务器上重新部署一个应用

其它参数

参数	描述
口令	指定 WebLogic 服务器的系统口令
应用名	指定应用的名字。应用的名字可以在分发时指定，也可以使用分发工具或控制台工具指定。
源	指出应用包文件（.jar,.war 或.ear）的实际位置，或者是应用目录所在的顶层路径。

选项

选项	定义
-component componentname: target1, target2	部署到各种目标上的组件，必须指定为： componentname:target1, target2 其中 componentname 为.jar 、.rar 或 .war 文件的文件名（不包含扩展名），可以为任意多个组件（.jar 、.rar

	或 .war 文件）指定该选项，且可以指定任意多次。不能用于部署.ear 文件。采用此选项，每个组件必需分开部署。
-debug	将部署过程中的详细调试信息输出到 stdout
-help	打印出 deploy 工具的所有选项
-jspRefreshComponentName	指明更新了的文件要复制到哪一个 webapp 组件中，使用此选项和-jsoRefreshFiles 选项一起，可以对静态文件进行刷新。详细信息参见“部署 Web 应用”中的“刷新静态组件”。
-jsoRefreshFiles	更新静态文件，例如：JSP 文件、HTML 文件、图形文件（.gif 和.jpg）以及文本文件。类文件不能被刷新。若需要刷新文件，需要更新标志以重新部署应用。详细信息参见“部署 Web 应用”中的“刷新静态组件”。
-host *host*	指定部署 J2EE 应用（.jar, .war, .ear）的 WebLogic 服务器的主机名。若未指定此选项，deploy 工具使用 localhost 作为主机名。
-port *port*	指定 WebLogic 服务器用来部署 J2EE 应用.jar,.war, 或.ear 文件的端口号。若未指定，deploy 工具采用缺省的端口号 7001.
-url *url*	指定 WebLogic 服务器的 URL，缺省为 localhost:7001
，-username *username*	连接所使用的用户名，缺省为 sytem
-version	打印出 deploy 工具的版本

例子

部署工具可用于多种目的：

查看一个 J2EE 应用

部署一个新 J2EE 应用

删除一个 J2EE 应用

更新一个 J2EE 应用

查看一个 J2EE 应用

以下命令可以查看部署在本地服务器上的一个应用

% java weblogic.deploy list *password*

Password 的值为 WebLogic 服务器系统帐号的口令。

如果要列出远程服务器上所部署的应用，那么应该指定 port 与 host 选项，如下所示：

java.weblogic.deploy –port *port_number* –host *host_name* list *password*

部署 J2EE 应用

部署一个还不曾分发到 WebLogic 的 J2EE 应用文件（.jar, .war, .ear）或应用目录，使用如下命令：

% java weblogic.deploy -port port_number -host host_name deploy password application source

application 为分配给所部署应用的字符串

source 为所要部署的 J2EE 应用文件的全路径（.jar,.war, .ear），或者是应用目录的全路径。

例如：

% java weblogic.deploy -port 7001 -host localhost deploy weblogicpwd Basic_example c:\mysamples\ejb\basic\BasicStatefulTraderBean.jar

注意：当 J2EE 文件（.jar, .war, .ear）复制到管理服务器的应用目录时，将以应用的名字重新命名该文件。因此，上面这个例子中，应用包的名字在 . . ./config/mydomain/applications 目录中从 BasicStatefulTraderBean.jar 改变为 Basic_example.jar

删除一个 J2EE 应用

要删除一个已部署的 J2EE 应用，只需要引用所分配的应用名，如下图所示：

% java weblogic.deploy -port 7001 -host localhost undeploy weblogicpwd Basic_example

注意：删除一个 J2EE 应用并没有把该应用从 WebLogic 服务器中删去。你不能在 deploy 工具中重用这个应用的名字。你只能在更新所部署的应用时重用应用的名字。具体内容见下节。

更新一个部署了的 J2EE 应用

要更新 J2EE 应用，应该使用 update 参数并指定活动 J2EE 应用的名字，如下所示：

% java weblogic.deploy -port 7001 -host localhost update weblogicpwd Basic_example c:\updatesample\ejb\basic\BasicStatefulTraderBean.jar

以下命令用于更新一或多个服务器上的指定组件。

% java weblogic.deploy -port 7001 -host localhost –component

BasicStatefulTraderBean.jar:sampleserver,exampleserver update weblogicpwd Basic_example c:\updatesample\ejb\basic\BasicStatefulTraderBean.jar

getProperty

你可以用 getProperty 工具获得系统与 Java 设置的详细信息。它不需要参数。

语法

$ java utils.getProperty

例子

$ java utils.getProperty

-- listing properties --

user.language=en

java.home=c:\java11\bin\..

awt.toolkit=sun.awt.windows.WToolkit

file.encoding.pkg=sun.io

java.version=1.1_Final

file.separator=\

line.separator=

user.region=US

file.encoding=8859_1

java.vendor=Sun Microsystems Inc.

user.timezone=PST

user.name=mary

os.arch=x86

os.name=Windows NT

java.vendor.url=http://www.sun.com/

user.dir=C:\weblogic

java.class.path=c:\weblogic\classes;c:\java\lib\cla...

java.class.version=45.3

os.version=4.0

path.separator=;

user.home=C:\

logToZip

logToZip 工具搜索以使用通用日志格式并保存在 HTTP 服务器下的日志文件，搜索该服务器加载的 java 类，并创建一个包含这些 java 类的未解压的.zip 文件。该工具需要在 HTTP 服务器根路径下执行。

要使用该工具，你必须有权访问由 HTTP 服务器创建的日志文件。

语法

$ java utils.logToZip *logfile codebase zipfile*

参数	定义
logfile	该参数是必需的。为日志文件的全路径名
codebase	该参数是必需的。Code base for the applet, or "" ifthere is no code base.By concatenating the code base with the full package name of the applet, you get the full pathname of the applet (relative to the HTTP document root).
zipfile	该参数是必需的，zipfile 是所创建的.zip 文件的文件名，该 zip 文件位于你运行程序的那个目录。该文件的路径可以是相对路径也可以是绝对路径。下面的例子给出了一个相对路径，因此.zip 文件被创建在当前目录下。

例子

下面这个例子说明了如何为文档根中（即没有用 code base）的一个 applet 创建一个.zip 文件：

$ cd /HTTP/Serv/docs

$ java utils.logToZip /HTTP/Serv/logs/access "" app2.zip

下面这个例子说明了如何为文档根的子目录中的一个 applet 创建一个.zip 文件：

C:\>cd \HTTP\Serv

C:\HTTP\Serv>java utils.logToZip \logs\applets\classes app3.zip

MulticastTest

你可以在配置 WebLogic 集群时，用该工具调试有关广播的问题。该工具发送广播包并返回广播在网络中的工作效率的信息。特别地，MulticastTest 将以下类型的信息显示在标准输出上。

1. 服务器所发送的每个消息的确认与序列号。
2. 集群中的服务器所收到的每个消息的序列号与发送方 ID
3. 当一个消息没有按顺序接收到时的“顺序错误”的警告。
4. 当一个预期的消息没有受到时的“消息丢失”的警告。

在使用 MulticastTest 时，应该在每个需要测试广播消息流量的节点上启动该工具。

警告：在指定 MulticastTest 工具的广播地址时（由-a 参数指定），注意不要设置为正在运行的 WebLogic 集群所使用的广播地址。在启动集群服务器之前用该工具验证广播是否能正常工作。

有关设置广播地址的详细信息，可以参见 WebLogic 服务器所在主机的操作系统/硬件的配置文档。更多信息，请参见“使用 WebLogic 服务器集群”。

语法

$ java utils.MulticastTest -n *name* -a *address* [-p *portnumber*] [-t *timeout*] [-s *send*]

参数	定义
-n *name*	必需的参数。用以表示顺序消息的发送方。应该为每个启动的测试进程使用不同的名字
-a *address*	必需的参数。是一个广播地址用于：（a）广播顺序消息；（b）集群中的服务器与其它服务器通信。（集群的广播地址的缺省值不应设为 237.0.0.1
-p *portnumber*	可选参数。集群中的服务器进行通信的广播端口。（广播端口与 WebLogic 的监听端口相同，缺省为 7001
-t *timeout*	可选参数。以秒为单位，若未收到广播消息时的空闲超时。缺省时间为 600 秒（10 分钟）。如果过了超时时间，stdout 会显示一个为正数的超时确认信息。
-s *send*	可选参数。发送消息的时间间隔，缺省值为 2 秒。对每个发送出去的消息，stdout 会显示一个为正数的确认。

例子

```
$ java utils.MulticastTest -N server100 -A 237.155.155.1
Set up to send and receive on Multicast on Address 237.155.155.1 on
port 7001
Will send a sequenced message under the name server100 every 2
seconds.
Received message 506 from server100
Received message 533 from server200
I (server100) sent message num 507
Received message 507 from server100
Received message 534 from server200
I (server100) sent message num 508
Received message 508 from server100
Received message 535 from server200
I (server100) sent message num 509
Received message 509 from server100
Received message 536 from server200
I (server100) sent message num 510
Received message 510 from server100
Received message 537 from server200
I (server100) sent message num 511
Received message 511 from server100
Received message 538 from server200
I (server100) sent message num 512
Received message 512 from server100
Received message 539 from server200
I (server100) sent message num 513
Received message 513 from server100
```

myip

返回主机的 IP 地址。

语法

$ java utils.myip

例子

$ java utils.myip

Host toyboat.toybox.com is assigned IP address: 192.0.0.1

Pem2der

Pem2der 工具用以将 PEM 格式的 X509 证书转换为 DER 格式，转换所得的.der 文件与原.pem 文件位于同一目录中。

语法

$ java utils.pem2der *pemFile*

参数	描述
pemFile	要进行转换的证书的文件名。该文件的扩展名必须为.pem，且必须包含一个有效的.pem 格式的证书。

例子

$ java utils.pem2der graceland_org.pem

Decoding

.....

.....

.....

.....

.....

Schema

Schema 工具使用 WebLogic JDBC 驱动程序将 SQL 语句上载到一个数据库中。有关数据库连接的详细信息可以参见“WebLogic JDBC 编程”中的信息。

语法

$ java utils.Schema driverURL driverClass [-u username] [-p password][-verbose SQLfile]

参数	定义
driverURL	该参数是必需的。JDBC 驱动的 URL

driverClass	该参数是必需的。JDBC 驱动类的路径名
-u *username*	可选参数。一个有效的用户名
-p *password*	可选参数。一个有效的用户名
-verbose	可选参数。打印出 SQL 语句与数据库的信息
SQLfile	如果用了-verbose 参数，那么就应该设置该参数。包含 SQL 语句的文本文件

例子

下面是一个 Schema 命令行的例子：

$ java utils.Schema "jdbc:cloudscape:demo;create=true" COM.cloudscape.core.JDBCDriver -verbose examples/utils/ddl/demo.ddl

下面是一个.ddl 文件中的内容

```
DROP TABLE ejbAccounts;
CREATE TABLE ejbAccounts
(id varchar(15),
bal float,
type varchar(15));
DROP TABLE idGenerator;
CREATE TABLE idGenerator
(tablename varchar(32),
maxkey int);
```

showLicenses

showLicenses 工具显示了所安装的 BEA 产品的许可证信息。

语法

$ java utils.showLicenses

system

system 工具显示操作系统的基本信息，包括所使用的 JDK 的制造商与版本号、classpath 以及操作系统的详细信息。

语法

S$ java utils.system

例子

$ java utils.system

* * * * * * * java.version * * * * * * *

1.1.6

* * * * * * * java.vendor * * * * * * *

Sun Microsystems Inc.

* * * * * * * java.class.path * * * * * * *

\java\lib\classes.zip;\weblogic\classes;

\weblogic\lib\weblogicaux.jar;\weblogic\license

...

* * * * * * * os.name * * * * * * *

Windows NT

* * * * * * * os.arch * * * * * * *

x86

* * * * * * * os.version * * * * * * *

4.0

t3dbping

该工具通过任何可用的两层 JDBC 驱动程序，对 WebLogic JDBC 连接到 DBMS 之间的物理连接进行测试。使用该工具时，你必须有访问 WebLogc 与 DBMS 的权限。

语法

$ java utils.t3dbping WebLogicURL username password DBMS driverClass driverURL

参数	定义
WebLogicURL	该参数是必需的。WebLogic 服务器的 URL
username	该参数是必需的。一个 DBMS 用户的用户名
password	该参数是必需的。DBMS 用户口令
DBMS	该参数是必需的。数据库的名字
driverClass	该参数是必须的。WebLogic 两层驱动程序的完整包名
driverURL	该参数是必需的。WebLogic 两层驱动程序的 URL

verboseToZip

当从 HTTP 服务器的文档根目录中执行该工具时，verboseToZip 获得以在 verbose 模式下运行的 java 应用所使用的标准输出，它找到所引用的 java 类，并创建一个包含这些 java 类的未解压的.zip 文件。

语法

$ java utils.verboseToZip *inputFile* *zipFileToCreate*

参数	定义
inputFile	该参数是必须的。它设置了一个临时文件保存以 verbose 模式运行的应用的输出。
zipFileToCreate	该参数是必须的。要创建的.zip 文件的名字。该文件保存在你运行工具所在的目录。

UNIX 上的例子

$ java -verbose myapplication > & classList.tmp

$ java utils.verboseToZip classList.tmp app2.zip

NT 上的例子

$ java -verbose myapplication > classList.tmp

$ java utils.verboseToZip classList.tmp app3.zip

version

version 工具将所安装的 WebLogic 的版本信息输出到 stdout

语法

$ java weblogic.Admin -url host:port -username username –password password VERSION

例子

$ java weblogic.Admin

-url localhost:7001 -username *system* -password *foo* VERSION

writeLiense

writeLiense 工具将 WebLogic 的许可证信息写到当前目录中的一个名字为 writeLicense.txt 文件中，然后可以将这个文件通过 email 发送给其它人，如 WebLogic 的技术支持。

语法

$ java utils.writeLicense -nowrite -Dweblogic.system.home=*path*

参数	定义
-nowrite	该参数是必须的。输出到 stdout 而不是 writeLicense.txt
-Dweblogic.system.home	该参数是必须的。该参数设置了 WebLogic system home(WebLogic 所在的根目录) **注意：**该参数是必须的，但如果从 WebLogic system home 运行 writeLicense 工具，那么可以不设置该参数

例子

```
$ java utils.writeLicense -nowrite
```

UNIX 上的输出示例：

```
* * * * * * System properties * * * * * *
* * * * * * * java.version * * * * * * *
1.1.7
* * * * * * * java.vendor * * * * * * *
Sun Microsystems Inc.
* * * * * * * java.class.path * * * * * * *
c:\weblogic\classes;c:\weblogic\lib\weblogicaux.jar;
c:\java117\lib\classes.zip;c:\weblogic\license
...
```

Windows NT 上的输出示例：

```
* * * * * * * os.name * * * * * * *
Windows NT
* * * * * * * os.arch * * * * * * *
x86
* * * * * * * os.version * * * * * * *
4.0
******IP******
Host myserver is assigned IP address: 192.1.1.0
* * * * * * Location of WebLogic license files * * * * * *
No WebLogicLicense.class found
No license.bea license found in
weblogic.system.home or current directory
Found in the classpath: c:/weblogic/license/license.bea
Last Modified: 06/02/1999 at 12:32:12
* * * * * * Valid license keys * * * * * *
Contents:
Product Name : WebLogic
IP Address : 192.1.1.0-255
Expiration Date: never
```

Units : unlimited

key : b2fcf3a8b8d6839d4a252b1781513b9

...

* * * * * * All license keys * * * * * *

Contents:

Product Name : WebLogic

IP Address : 192.1.1.0-255

Expiration Date: never

Units : unlimited

key : b2fcf3a8b8d6839d4a252b1781513b9

...

* * * * * * WebLogic version * * * * * *

WebLogic Build: 4.0.x xx/xx/1999 10:34:35 #xxxxx

B. WebLogic 服务器命令行接口参考

本附录将介绍以下内容：

命令行接口简介

使用 WebLogic 服务器命令

WebLogic 服务器管理命令参考

配置及管理连接池的命令

Mbean 管理命令参考

命令行接口简介

作为管理控制台的另一种选择， WebLogic 服务器提供了命令行接口作为管理工具，同时也可以设定多数的 configuration 以及 run-time Mbean 的属性。

以下情况可以使用命令行界面：

如果要用脚本进行有效的管理

如果不能使用浏览器访问管理控制台

相比于 GUI，你更愿意使用命令行界面

开始前的准备工作

本文档中的例子是基于以下假设：

WebLogic 服务器安装在 c:\weblogic 目录

JDK 位于 c:\java 目录

WebLogic 服务器是从它的安装目录启动的

运行 WebLogic Server 命令之前，必须先完成以下工作：

1．安装并配置 WebLogic 服务器软件。参见以下页面的“WebLogic 服务器安装指南”中的部分：

位于http://e-docs.bea.com/wls/docs61/install/index.html

2．正确地配置 CLASSPATH 环境变量。参见 2-7 页的“设置类路径选项”

3．执行以下步骤中的一个来启用命令行界面

从服务器的安装目录启动服务器

如果不是从服务器的安装目录启动服务器，那么应该使用以下命令，把命令中的 c:\weblogic 用你的 webLogic 服务器所在的目录名替代。

-Dweblogic.system.home=c:/*weblogic*

使用 WebLogic 服务器命令

本章介绍了 WebLogic Server 的命令与所需参数。WebLogic Server 的命令不区分大小写。

语法

java weblogic.Admin [-url *URL*] [-username *username*] [-password *password*] *COMMAMD arguments*

参数

许多 WebLogic 服务器命令都需要以下参数：。

参数	定义
URL	WebLogic 服务器主机的 URL，其中包括 WebLogic 服务器监听客户端请求的 TCP 端口。格式为 hostname:port。缺省为 localhost:7001. 注意：服务器命令中的 URL 参数引用的是 WebLogic 服务器，而 run-time Mbea 与 configuration Mbean 命令总是指向一个特定的管理服务器。
username	可选参数。要验证的用户名用以确定是否可以执行命令。缺省为 guest
password	可选参数。对口令进行验证以确定命令是否可以执行

注：若管理员不能连接到服务器上，则所有命令的退出代码为 1

管理员必须具有相应的访问权限，以执行管理 run-time Mbeans 的命令。

见以下章节：

B-4 页的“WebLogic 服务器管理命令参考”

B-18 页的“WebLogic 服务器连接池管理命令参考”

B-28 页的“Mbean 管理命令参考”

WebLogic 服务器管理命令参考

表 B-1 概要介绍了 WebLogic 服务器的管理命令，接下来的内容将描述命令的语法与参数，并给出了每个命令的例子。

表 T-1 WebLogic 服务器管理命令概述

任务	命令	描述
取消关闭 WebLogic 服务器命令	CANCEL-SHUTDOWN	取消执行对指定 URL 上的 WebLogic 服务器关闭命令。 见 B-6 页 的“CANCEL-SHUTDOWN”
连接到 WebLogic 服务器	CONNECT	对 WebLogic 服务器进行指定次数的连接，并返回每个回合连接的总时间（用两位数表示）以及每次连接的平均持续时间 见 B-7 页的“CONNECT”
获得命令的帮助信息	HELP	给出了所有 WebLogic 服务器的语法与使用信息。如果 HELP 命令中指定了命令值，那么只给出这一命令的帮助信息。 参见 B-8 页的“HELP”部分
启动或停止 WebLogic 服务器	java	用 Java 命令启动或停止 WebLogic 服务器，这与启动或停止其它 Java 应用的方式是一样的。 见管理指南的“启与停止 WebLogic 服务器”中的部分，位于： http://e-docs.bea.com/wls/docs61/adminguide/startstop.html .
查看 WebLogic 服务器的许可证	LICENSES	列出一服务器上所有 WebLogic 服务器实例的许可证。见 B-9 页的“LICENSES”

列出 JNDI 名字树节点所绑定的内容	LIST	列出 JNDI 名字树上某一节点所绑定的内容 见 B-10 页的“LIST”
锁住 WebLogic 服务器	LOCK	对非特权用户登录，锁定 WebLoigc 服务器。以后的任何的登录企图都将引发一个包含可选字符串信息的安全异常。 见 B-11 页的“LOCK”
验证 WebLogic 服务器的监听端口	PING	发送消息验证 WebLogic 服务器是否在端口进行监听的消息、是否可以接收 WebLogic 客户端的请求。 见 B-12 页的“PING”。
查看服务器的日志文件	SERVERLOG	显示指定服务器所产生的服务器日志文件。 见 B-13 页的“SERVERLOG”
关闭一个 WebLogic 服务器	SHUTDOWN	关闭 URL 所指定的 WebLogic 服务器。 见 B-14 页的“SHUTDOWN”。
查看线程	THREAD_DUMP	提供了当前正在运行的 WebLogic 服务器线程的一个实时快照。 见 B-15 页的“THREAD_DUMP”
解锁 WebLogic 服务器	UNLOCK	解开用 LOCK 命令对指定 WebLogic 服务器添加的锁。 见 B-16 页的“UNLOCK”。
查看 WebLogic 服务器版本	VERSION	显示 URL 所指定机器上的 WebLogic 服务器软件的版本。 见 B-17 页的“VERSION”。

注：若管理员不能连接到服务器上，则所有命令的退出代码为 1

CONNECT

CANCEL_SHUTDOWN 命令取消对指定的 URL 上的 WebLogic 服务器执行的关闭命令。见 B-14 页上的“SHUTDOWN”。

当使用 SHUTDOWN 命令时，你可以以秒指定一个延时，管理员可以在这段延时的时间内取消关闭命令。但是 SHUTDOWN 命令禁止用户登录，即使在取消关闭命令执行后，依旧保持禁止状态。可以通过 UNLOCK 命令重新允许用户登录，参见 B-16 页的“UNLOCK”。

语法：

示例：

CONNECT

对 WebLogic 服务器进行指定次数的连接，并返回每个回合连接的总时间（用两位数表示）以及每次连接的平均持续时间（以微秒为单位）。

语法：

java weblogic.Admin [-url *URL*] [-username *username*] [-password *password*] CONNECT *count*

参数	定义
count	连接的次数

示例：

在下面的例子中，一个名字为 adminuser 口令为 gumby1234 的用户运行 CONNECT 命令对名字

为 localhost 的服务器建立了 25 次连接并返回这些连接的信息：

java weblogic.Admin -url localhost:7001 -username adminuser -password gumby1234 CONNECT 25

HELP

给出所有 WebLogic 服务器命令的语法与使用信息（缺省的），如果 HELP 命令行给出了一个命令值，那么 HELP 命令将显示该命令的帮助信息。

语法：

java weblogic.Admin HELP *[COMMAND]*

示例：

下面的例子使用 HELP 命令获得 PING 命令的帮助信息。

java weblogic.Admin HELP PING

HELP 命令将以下信息输出到标准输出上：

Usage: weblogic.Admin [-url *url*] [-username *username*]

[-password *password*]<*COMMAND*><*ARGUMENTS*>

PING <*count*><*bytes*>

LICENSES

列出安装在某一服务器上的所有 WebLogic 服务器实例的许可证

语法：

java weblogic.Admin [-url *URL*] [-username *username*] [-password *password*] LICENSES

示例：

在下面的例子中，管理远使用缺省的用户名（guest）与口令（guest）获得 localhost 机器中在 7001 端口运行的 WebLogic 服务器的许可证信息。

java weblogic.Admin -url localhost:7001 -username guest -password guest LICENSES

LIST

列出 JNDI 名字树上某一节点的绑定信息。

语法：

java weblogic.Admin [-username *username*] [-password *password*] LIST *context*

参数	定义
context	必需的参数。进行 JNDI 查找的上下文。例如，weblogic, weblogic.ejb.javax

示例：

在本例中，用户 adminuser（口令为 gumby1234）的用户请求列出 weblogic.ejb 中节点的绑定信息。

java weblogic.Admin -username adminuser -password gumby1234 LIST weblogic.ejb

LOCK

对非特权用户的登录，锁定 WebLogic 服务器。以后的登录请求都将引发一个包含可选字符串信息的安全异常。

注意：运行该命令必须有相应的权限。该命令要求使用 WebLogic 服务器管理用户的口令

语法：

java weblogic.Admin [-url *URL*] [-username *username*] [-password *password*] LOCK "*string_message* "

参数	定义
"string_message"	可选参数。当 WebLogic 服务器被锁住后，没有权限的用户进行登录将引发一个安全异常，该异常包含了双引号中的消息字符串。

示例：

下例中，WebLogic 服务器已被锁住。

java weblogic.Admin -url localhost:7001 -username adminuser -password gumby1234 LOCK "Sorry, WebLogic Server is temporarily out of service."

当应用使用一个非特权用户的登录名与口令登录被锁住的服务器时，将收到以下消息：sorry, webLogic Server is temporarily out of service

PING

发送消息以验证 WebLogic 服务器是否在端口上进行监听以及是否可以接收 WebLogic 客户端请求。

语法：

java weblogic.Admin [-url *URL*] [-username *username*] [-password *password*]PING *[round_trips*][*message_length*]

参数	定义
round_trips	可选参数。Ping 的次数
message_length	可选参数。每个 ping 发送的包的大小。当 ping 的包大于 10MB 将引发异常

示例：

下例中，PING 命令对运行于 localhost 机器 7001 端口中的 WebLogic 服务器检查了 10 次。

java weblogic.Admin -url localhost:7001 -username adminuser -password gumby1234 PING 10

SERVERLOG

显示指定服务器所生成的日志文件。

如果没有指定 URL，那么将显示管理服务器上的服务器日志。

如果指定服务器的 URL，那么可以取得非管理服务器中的日志。

如果省略了 startime 与 endtime 参数，会启动运行显示整个服务器日志。

语法：

java.weblogic.Admin [-url *URL*] [-username *username*] [-password *password*] SERVERLOG [[*starttime*]][*endtime*]]

参数	定义
starttime	可选参数。显示的日志从该时间的消息开始。如果没有指定这一参数，缺省情况下将显示执行 SERVERLOG 命令后的消息。日期的格式为 yyyy/mm/dd。时间的格式使用 24 小时制。起始日期与时间应该用双引号括起来，即使用以下格式："yyyy/mm/dd hh:mm"
endtime	可选参数。显示消息的截止时间。如果没有指定这一参数，缺省为 SERVERLOG 命令执行时的时间。日期的格式为 yyyy/mm/dd。时间的格式使用 24 小时制。结束日期与时间应该用双引号括起来，即使用以下格式："yyyy/mm/dd hh:mm"

示例：

下面的例子使用 SERVERLOG 命令显示运行在 localhost 机器上 7001 端口的 WebLogic 服务器的日志。

java weblogic.Admin -url localhost:7001 SERVERLOG "2001/12/01 14:00""2001/12/01 16:00"

该命令显示 2001/12/1 2:00p.m. 与 2001/12/1 4:00p.m.之间的日志。

SHUTDOWN

关闭指定 URL 上的 WebLogic 服务器

语法：

java weblogic.Admin [-url *URL*] [-username *username*] [-password *password*] SHUTDOWN *[seconds*]["*lockMessage*"]

参数	定义
seconds	可选参数。指定命令执行后多长时间关闭服务器。
"lockMessage"	可选参数。当用户登录一个被锁住的 webLogic 服务器时，系统将发送双引号中的消息。

示例：

下面的例子中，adminuser 用户（口令为 gumby1234）关闭 localhost 机器中在 7001 端口监听的 WebLogic 服务器：

java weblogic.Admin -url localhost:7001 -username adminuser -password gumby1234 SHUTDOWN 300 "Server localhost is shutting down."

当命令执行后，过 30 秒，所指定的服务器被关闭并将以下消息发送到标准输出：

Server localhost is shutting down.

THREAD_DUMP

提供当前正在运行的 WebLogic 服务器线程的实时快照。

语法：

java weblogic.Admin [-url *URL*] [-username *username*] [-password *password*] THREAD_DUMP

unlock

将一个被锁住的 WebLogic 服务器解锁。

语法：

java weblogic.Admin [-url *URL*] [-username *username*] [-password *password*] UNLOCK

参数	定义
username	必需的参数。一个有效的管理员用户的用户名
password	必需的参数。管理员用户的口令

示例：

本例中，管理员用户 adminuser（口令为 gumby1234）请求对运行在 loalhost 机器 7001 端口的 WebLogic 服务器解锁。

java weblogic.Admin -url localhost:7001 -username adminuser -password gumby1234 UNLOCK

VERSION

显示运行在 URL 所指定的机器上的 WebLogic 服务器软件的版本。

语法：

java weblogic.Admin [-url *URL*] [-username *username*] [-password *password*] VERSION

示例：

下面的例子中，一个用户要求显示运行在 localhost 机器 7001 端口的 WebLogic 服务器的版本

java weblogic.Admin -url localhost:7001 -username guest -password guest VERSION

注意：本例使用了缺省的 usename 与 password 参数, 即使用的是 guest 用户，其口令也是 guest

WebLogic 服务器连接池管理命令参考

表 B-2 是关于 WebLogic 服务器连接池管理命令的概述，本节将介绍命令的语法、参数并对每一个命令提供了示例

关于连接池的详细信息，请参考在http://e-docs.bea.com/wls/docs61/jdbc/index.html 中的 WebLogic JDBC 编程指南。有关管理 JDBC 连接，见管理参考中的章节 http://e-docs.bea.com/wls/docs61/adminguide/jdbc.html

表 B-1 WebLogic 服务器连接池管理命令概述

任务	命令	描述
创建一个动态的连接池	CREATE__POOL	在 WebLogic 运行时，允许创建连接池。注：动态连接池不能用于数据源和 Tx 数据源。 见 B-20 页的“CREATE__POOL”
删除一个连接池	DESTROY_POOL	关闭和从连接池中删除连接，当连接池中没有任何连接时，连接池就不存在了。只有"System"用户和拥有“Admin”权限并定义了关于连接池的 ACL 的的用户才可以删除连接池

		参见 B-23 页的“DESTROY__POOL”部分
禁用连接池	DISABLE__POOL	你可以暂时禁用一个连接池，阻止其他用户从该池中获取连接，只有"System"用户和拥有“Admin”权限并定义了关于连接池的 ACL 的的用户才可以禁用和启用连接池。 参见 B-23 页的“DISABLE__POOL”部分
允许使用连接池	ENABLE__POOL	在一个连接池被禁用后启用它，每个在使用的连接的 JDBC 连接状态与连接池被禁用时的一样，客户端可以继续当其被停滞时的操作。 参见 B-25 页的“ENABLE__POOL”部分
判断连接池是否存在	EXISTS__POOL	测试 WebLogic 服务器中指定名字的连接池是否存在，你可以使用该命令判断连接池是否已经建立或者在创建动态连接池前确认其名字是否唯一。 参见 B-26 页的“EXISTS__POOL”部分
重置连接池	RESET__POOL	关闭并重新打开连接池中的连接。用于 DBMS 重起后，例如，当连接池中的一个连接失败后时，连接池中的所有连接均不可用了。 参见 B-27 页的“RESET__POOL”部分

CREATE__POOL

在WebLogic运行时，允许创建连接池。详细信息参见在 http://e-docs.bea.com/wls/docs61/jdbc/programming.html#programming004 上的*WebLogicJDBC编程* 中的“动态创建连接池”

语法

java weblogic.Admin [-url *URL*] [-username *username*]

[-password *password*] CREATE_POOL poolName aclName=*aclX*,

props=*myProps*,initialCapacity=1,maxCapacity=1,

capacityIncrement=1,allowShrinking=*true*,shrinkPeriodMins=*15*,

driver=*myDriver*,url=*myURL*

参数	定义
poolName	必需的。连接池的唯一名称。
aclName	必需的。表明不同的访问权限，定义于服务器中的 config 目录下的 fileRealm.propertiose 文件中。名字必需是 dynaPool
props	与数据库连接的属性；通常以“数据库登录名；口令；服务器网络 ID”来表示。
initialCapacity	连接池的初试连接数。若该属性定义为一个大于 0 的正数，WebLogic 服务器在启动时创建该数目的连接。缺省值为 1；它不能超过 maxCapacity 的值。
maxCapacity	连接池所允许的最大连接数目。缺省为 1，若定义了，该值一定大于等于 1
capacityIncrement	每次连接增加的数目。缺省值为 1.
allowShrinking	当发现连接不再使用时，是否允许连接池缩减连接数目。缺省值为

	true.
shrinkPeriodMins	必需的。缩减的间隔时间。单位为分钟，最小为 1.如果 allowShrinking 为 true，那么缺省的时间为 15 分钟。
driver	必需的。JDBC 驱动程序名，即本地（非 XA）的驱动
url	必需的。JDBC 驱动程序的 URL
testConnsOnReserve	表明保留测试连接，缺省为 false
testConnsOnRelease	表明在释放时，测试连接，缺省为 false
testTableName	用于测试连接的数据库表名；用以表明测试成功。当 testConnsOnReserve 或 testConnsOnRelease 定义时，该选项为必需的。
refreshPeriod	设定连接刷新间隔。每个不再使用的连接将用 testTableName 测试，如果没有通过测试，则关闭连接，并重新建立一个和 DB 的物理连接。如果 testTableName 没有设定，则不进行测试过程。
loginDelaySecs	在建立每个实际的与 DB 的连接前的延时，以秒为单位。当一个物理连接建立时，该延时在初始化和连接池生命周期内一直有效。一些数据库不能处理多个快速的连续的连接请求，设置该属性可以使数据库可以响应处理。它在一个物理连接建立时，初始化和连接池生命周期内一直有效。

示例

在下例中，一个名为 adminuser 的用户，其口令为 gumby1234，运行 CREATE_POOL 命令来创建一个动态连接池。

```
java weblogic.Admin -url localhost:7001 -username adminuser
    -password gumby1234 CREATE_POOL myPool
java weblogic.Admin -url t3://forest:7901 -username system
    -password gumby1234 CREATE_POOL dynapool6 "aclName=someAcl,
    allowShrinking=true,shrinkPeriodMins=10,
    url=jdbc:weblogic:oracle,driver=weblogic.jdbc.oci.Driver,
    initialCapacity=2,maxCapacity=8,
    props=user=SCOTT;password=tiger;server=bay816"
```

DESTROY__POOL

关闭和从连接池中删除连接，当连接池中没有任何连接时，连接池就不存在了。只有"System"用户和拥有“Admin”权限并定义了关于连接池的ACL的的用户才可以删除连接池。

语法

java weblogic.Admin [-url *URL*] [-username *username*]

[-password *password*] DESTROY_POOL *poolName [true|false]*

参数	定义
poolName	必需的。连接池的唯一名称。
false (软关闭)	表示只有连接返回到连接池中才关闭。
true (缺省的，强制关闭)	立即关闭连接。客户使用已经关闭了的连接池中的连接会得到异常信息

示例

在下例中，为运行于 xyz.com 主机，并侦听 7001 端口的 WebLogic 服务器，刷新一个名为"eng"的连接池，

java weblogic.Admin t3://xyz.com:7001 RESET_POOL eng system gumby

Mbean 管理命令参考

表 B-3 概要介绍了 Mbean 的管理命令，接下来的内容将详细介绍各命令的语法与参数，并提供了相应的例子。

表 T-2 Mbean 管理命令概览

任务	命令	描述
创建 configuration Mbeans	CREATE	创建 configuration Mbean 的一个实例。如果成功则将 OK 返回 stdout。不能对 run-time Mbeans 使用该命令。 参见 B-29 页的“CREATE”。
删除 configuration Mbeans	DELETE	删除一个 configration Mbean。如果删除成功则将 OK 返回给 stdout 该命令不能应用于 run-time Mbeans 见 B-30 页的“DELETE”。
查看 run-time Mbean 的属性	GET	显示 run-time Mbean 的属性 见 B-31 页的“GET”。
调用 run-time Mbeans	INVOKE	调用不是用来 get 或 set 属性的方法。只能对 run-time Mbeans 执行该命令。 见 B-33 页的“INVOKE”
查看 Mbeans 的运行节奏和统计	INVOKE GET	运行 INVOKE 与 GET 命令来查看 Mbeans 的运行节奏和统计。只能对 run-time Mbeans 执行该命令。 见 B-31 页的“INVOKE”以及 B-33 页的“GET”命令
设置 configration Mbeans 的属性	SET	设置给指定名字的 configuration Mbean 的属性。如果设置成功则将 OK 返回给 stdout。该命令只能用于 run-time Mbeans 见 B-34 页的“SET”操作。

CREATE 命令

用 CREATE 命令创建一个 configration Mbean 实例。如果创建成功则将 OK 返回给 stdout。该命令不适用于 run-time Mbeans Mbean 实例可以保存在 config.xml 文件中，也可以保存在安全域中，这取决于你在什么地方做的更改。

注：在创建 Mbeans 时，同时创建了配置对象（configration objects

有关创建 Mbeans 的详细信息，可以参见以下页面的“WebLogic 服务器应用开发”中部分，位于：http://e-docs.bea.com/wls/docs61/programming/index.html

语法：

java weblogic.Admin [–url *URL*] [-username *username*] [-password *password*] CREATE –name *name* –type *mbean_type* [–domain *domain_name*]

java weblogic.Admin [–url *URL*] [-username *username*] [-password *password*] CREATE –mbean *mbean_name*

参数	定义
name	必需的参数。要创建的 Mbean 的名字。
mbean_type	必需的参数。用于为同一类型的多个对象创建属性时。
mbean_name	必需的参数。Mbean 的完整名字。格式如下： “domain:Type=*type*, Name=*name*” 其中 Type 指明对象类型，Name 指明 Mbean 的名字
domain_name	可选参数。域的名字，例如，mydomain。如果没有指定 domain_name，那么使用缺省域的名字。

示例：

java weblogic.Admin -url localhost:7001 -username adminuser -password gumby1234 CREATE –mbean “mydomain:Type=Server,Name=acctServer”

DELETE 命令

删除一个 configuration Mbean。如删除成果则把 OK 返回给 stdout。该命令不能用于 run-time Mbeans

注：在删除 Mbeans 时，同时会删除配置对象（configuration objects

有关删除 Mbeans 的更多信息，可以参见以下页面的“WebLogic 服务器应用开发”中部分，位于：http://e-docs.bea.com/wls/docs61/programming/index.html

语法：

java weblogic.Admin [–url *URL*] [-username *username*] [-password *password*] DELETE {–type *mbean_type*|–mbean *mbean_name*}

参数	定义
mbean_type	必需的参数。用于为同一类型的多个对象创建属性时。
mbean_name	必需的参数。Mbean 的完整名字。格式如下： *“domain*:Type=*type*,Name=*name*” Type 指定了对象组的类型，Name 是 Mbean 的名字。

示例：

java weblogic.Admin -url localhost:7001 -username adminuser -password gumby1234 DELETE –mbean “mydomain:Type:Server,Name=AcctServer”

GET 命令

显示 run-time Mbean 的属性。你可以利用该命令查看同一类型的多个对象的一组属性，查看的步骤如下：

属于同一类型的所有 Mbeans

GET {-pretty} -type *mbean_type*

一个指定的 Mbean

GET {-pretty} -mbean *mbean_name*

所指定的每一个 Mbean 的名字都包括在输出中。如果指定了 –pretty，那么每个属性的名字-属性值对将显示在新的一行中。

GET 命令只能调用 run-time Mbeans

每个属性的名字-属性值对都指定在花括号中。这种格式可以简化解析输出信息，因此有利于脚本的编写。

输出中的 Mbean 的名字如下所示：

{mbeanname mbean_name {property1 value}{property2 value}. . .}

{mbeanname mbean_name {property1 value}{property2 value}...}

...

如果指定了-pretty 参数，那么每个属性的名字-属性值对显示在新的一行中。所指定的每个 Mbean 的名字都包括在输出中，如下所示：

mbeanname: mbean_name

property1: value

property2: value

.

.

.

mbeanname: mbean_name

property1: value

property2: value

语法：

java weblogic.Admin [–url *URL*] [-username *username*] [-password *password*] GET {-pretty} {–type *mbean_type*|–mbean *mbean_name*} [–property *property*1][–property *property*2]...

参数	定义
mbean_type	必需的参数。当要得到同一类型的多个对象的属性时，输出中将包括 Mbean 的名字。
mbean_name	Mbean 的完整名字，格式如下： *"domain*:Type=*type*,Location:*location*,Name=*name"* Type 指定了对象组的类型，Location 指定了 Mbean 的位置，Name 指定了 Mbean 的名字。
pretty	可选参数。产生格式输出。
property	可选参数。列出 Mbean 的属性名。 注意：如果没有用这个参数指定属性名，那么将显示所有属性。

示例：

在本例中，用户请求显示运行在 localhost 的 7001 端口上的服务器中的 Mbean 的属性：

java weblogic.Admin -url localhost:7001 GET -pretty -type Server

INVOKE 命令

调用某一 Mbean 的指定方法（带参数）。该命令只适用于 run-time Mbeans。用该命令调用 Mbean 的方法（非 get 或 set Mbean 属性的方法）

语法：

java weblogic.Admin [–url *URL*] [-username *username*] [-password *password*] INVOKE {–type *mbean_type*|–mbean *mbean_name*} –method *methodname [argument ...]*

参数	定义
mbean_type	当调用同一类型的多个对象的属性时，该参数是必须的。参数中应该包含 Mbean 的完整包名，如下： “domain:Name:name,Type=*type*,Application=*application*”
mbean_name	必需的参数。该参数设置为 Mbean 的完整名字。如下所示： “domain:Type=*type*,Location=*location*,Name=*name*” 其中： ■ Type 指出了对象组的类型。 ■ Location 指对象的位置。 ■ Name 指 Mbean 的名字。 当参数为一个 String 数组时，必须以以下格式传递参数： “*String1*;*String2*;. . . ”
methodname	必需的参数。所调用的方法名。在方法名后，用户可以指定要传递的参数，如下所示： “*domain*:Name=*name*,Type=*type*”

示例：

下面的例子调用了一个名字为 admin_one 的管理 Mbean 的 getAttributeStringValue 方法：

java weblogic.Admin -username system -password gumby1234 INVOKE -mbean mydomain:Name = admin_one, Type=Administrator -method getAttributeStringValue PhoneNumber

SET 命令

该命令用于设置 configuration Mbean 的属性值，设置成功则将 OK 返回到 stdout。该命令不能用于 run-time Mbeans

新设置的属性值保存在 config.xml 文件或安全域中，这取决于新的属性值是在哪里定义的。

语法

java weblogic.Admin [–url *URL*] [-username *username*] [-password *password*]SET{–type *mbean_type*|–mbean *mbean_name*} –property *property1 property1_value* [-property *property2 property2_value*]...

参数	定义
mbean_type	如果是设置同一类型的多个对象的属性时，必须设置该参数，并且必须包括 Mbeans 的完整名字，如下所示： “domain:Name:name,Type=*type*,Application=*application*”

mbean_name	必需的参数。必须是 Mbean 的完整修饰名，格式如下： *“domain*:Name=*name*,Location:*location*,Type=*type*” 其中： Name 设置了 Mbean 的名字 Location 指定 Mbean 的位置 Type 指定了对象组的类型
property	必需的参数。要设置的属性的名字
property _value	必需的参数。属性的值。 ■ 如果参数是一个 Mbean 数组，那么应该用以下格式传递参数 *“domain*:Name=*name*,Type=*type*;*domain*:Name=*name*,Type=*type*” ■ 如果参数是一个 String 数组，那么应该用以下格式传递参数 *“String*1;*String*2;. . . ” ■ 如果设置 JDBC 连接池属性，应该用以下格式传递参数： “user:*username*;password:*password*;server:*servername*”

C. Web 服务器插件参数

以下内容将描述 Apache, Netscape 与 Microsoft IIS Web 服务器插件的配置参数。主要有以下内容：

概述

Web server 插件的一般参数

Web server 插件的 SSL 参数

概述

每一类型的 web 服务器的插件参数在特定的配置文件中设置。各服务器插件的配置文件都有自己的名字与文件格式。详细内容，参见各插件的以下部分的内容：

11-1 页的“安装及配置 Apache HTTP Server 插件”

12-1 页的“安装及配置 Microsoft Internat Information Server(ISAPI)插件”

13-1 页的“安装及配置 Netscape Enterprise Server 插件（NSAPI）”

在 web server 插件的配置文件中设置下表中所描述的参数。

Web 服务器插件的通用参数

注意：所有参数都区分大小写。

参数	默认值	描述
WebLogicHost （如果代理到单个 WebLogic 服务器，那么应该设置该参数）	none	HTTP 请求被递交到该 WebLogic Server 主机（或者是定义在 WebLogic Server 中运行的类似于 Web server 的虚拟主机名） 如果使用集群，那么应该使用 WebLogicCluster 参数而不是 WebLogicHost 参数。
WebLogicPort （如果代理到单个 WebLogic 服务器，那么应该设置该参数）	none	WebLogic 服务器监听 WebLogic 连接请求的端口。（如果在插件与 WebLogic 服务器之间使用 SSL，那么应该把该参数设置为 SSL 监听端口（见 8-4 页的“配置监听端口”）并将 SecureProxy 参数设置为 ON 如果使用 WebLogic 集群，那么应该设置 WebLogicCluster 参数而不是 WebLogicPort 参数。
WebLogicCluster （如果代理到一个 WebLogic 服务器集群，那么必须设置该参数）	none	集群中的 WebLogic 服务器列表，用于负载平衡目的。该列表由逗号分隔开的 host:port 组成。例如： WebLogicCluster myweblogic.com:7001, yourweblogic.com:7001,theirweblogic.com:7001 如果插件与 WebLogic 服务器之间使用 SSL 协议，那么将端口号设置为 SSL 监听端口（见 8-3 页的“配置监听端口”）并将 SecureProxy 参数设置为 ON 应该用该参数取代 WebLogicHost 与 WebLogicPort 参数，WebLogic Server 首先查找 WebLogicCluster 参数，如果没有找到该参数，它将寻找并使用 WebLogicHost 与 WebLogicPort 参数。 插件对所有可用的集群成员进行轮询。该参数所指定的集群列表是服务器与插件共同维护的动态列表的初始值。WebLogic 服务器与插件将根据新加入的、失败的以

		及恢复的集群成员的情况动态地更新集群列表。 如果将 DynamicServerList 参数设置为 OFF（只适用于 Microsoft Internet Information Server），那么集群列表的动态更新被禁用。 插件将请求导向集群中最初创建 cookie 的那个服务器上（请求包含 cookie、URL-encoded 会话或存于 POST 数据中的会话信息）。
PathTrim	none	在请求被转交到 WebLogic 服务器之前，被插件从原始 URL 中裁剪掉的字符串。例如：如果原始 URL 为 http://myWeb.server.com/weblogic/foo 被传递到插件进行解析且 PathTrim 参数被设置为 /weblogic，那么传递到 WebLogic 服务器的 URL 变为： http://myweblogic.server.com:7001/foo
PathPrepend	null	加在原始 URL 前的前缀字符串，该动作发生在 PathTrim 被裁剪后，请求转向 WebLogic 服务器之前。
ConnectTimeoutSecs	10	插件进行 WebLogic 服务器主机连接尝试的时间上限。该值应该大于 ConnectRetrySecs 参数。如果超过 ConnectTimeoutSecs 还没能连接成功，即使进行了适当次数的连接重试（见 ConnectRetrySecs 参数），也将把 HTTP 503/Service Unavailable 响应返回给客户端。 可以使用 ErrorPage 参数定制错误响应。
ConnectRetrySecs	2	该参数以秒为单位，设置了两次 WebLogic Server 主机（或集群中的所有服务器）连接尝试之间，插件的休眠时间。该参数的值应该小于 ConnectTimeoutSecs。插件在返回 HTTP 503/Service Unavailable 响应之前，它将进行的连接次数为 ConnectTimeoutSecs 除以 ConnectRetrySecs 所得的值。 如果不希望重试连接，那么应该将 ConnectRetrySecs 值应该与 ConnectTimeoutSecs 相等。不过，插件会至少进行两次连接尝试。 可以用 ErrorPage 参数定制错误响应。
Debug	OFF	设置调试操作时的日志类型。在生产系统中不建议你开启这些调试选项。 在 UNIX 系统中，调试信息被写到/tmp/wlproxy.log 文件中；在 Windows NT 系统，调试信息被写到 c:\temp\wlproxy.log 文件中，通过 WLLogFile 参数，你可以使用其他路径下的其他文件覆盖文件名、路径参数。你可以设置以下日志选项（其中 HFC, HTW, HFW, HTC 可以联合使用，它们之间用逗号隔开，如“HFC, HTW”）； ON 插件只记录报告性消息与错误消息 OFF 不记录调试信息 HFC 记录来自客户端消息、报告性消息以及错误消息的消息头。 HTW 记录从 weblogic 发送来的消息头，报告性消息与错误消息 HFW

		记录来自 weblogic 服务器消息的消息头，报告性消息与错误消息 HTC 记录发送到客户端消息的消息头，报告性消息与错误消息 ALL 记录发送到客户端以及客户端发送的消息的头，发送到 WebLogic 服务器以及 WebLogic 服务器发送的消息头，报告性消息，错误消息
WLLogFile	参见 Debug 参数	指明当 Debug 参数为 ON 时，产生日志文件的路径和文件名，在设置该参数前，必须创建相应目录。
DebugConfigInfo	OFF	启用特殊查询参数“_WebLogicBridgeConfig”。该参数可以被用来了解插件的配置参数的细节。 例如，如果把 DebugConfigInfo 设置为 ON，那么“_WebLogicBridgeConfig”被启用。发送一个包含查询字符串?_WebLogicBridgeConfig 的请求，插件将收集配置信息有运行时的统计信息并将这些信息返回给浏览器。在处理该请求时，插件没有连接到 WebLogic 服务器。 该参数只应严格用于调试目的。消息的输出格式随版本的变化而不同。为了安全起见，在生产环境中应该将该参数设置为 OFF
StatPath Microsoft Internet Information Server 插件没有这个参数。）	false	如果把该参数设置为真，插件在把请求传递到 WebLogic 服务器之前检查被转换的路径是否存在或及其访问权限 ”Proxy-Path-Translated”）。如果文件不存在，将把 HTTP 404 File Not Found 响应返回给客户端。如果文件存在，但它的权限不是 world-readable，那么将返回 HTTP 403/Forbidden 响应。这两种情况下 Web 服务器处理这些响应的缺省机制是执行响应的体内容。如果 WebLogic 服务器的 Web 应用与 Web 服务器具有相同的文档根，那么该选项非常有用。 可以使用 ErrorPage 参数定制错误响应。
ErrorPage	none	可以制作自己的错误响应页面，在 Web 服务器不能将请求代理到 WebLogic 服务器时使用。 设置该参数的方式有两种： ■ 作为相对 URI（文件名）。插件自动将返回错误的 Web 应用的上下文路径加到 URI 中。对错误页面的请求是否回代理到 WebLogic 服务器取决于你对代理的配置（是 MIME 类型式代理还是路径式代理）。 ■ 作为绝对 URI（建议）。使用错误页面的绝对路径能够使请求总是被代理到 WebLogic 服务器中的正确资源上。例如： http://host:port/myWebApp/ErrorPage.html
HungServerRecoverSecs	300	定义了插件等待 WebLogic 服务器响应请求的时间。在等待了 HungServerRecoverSecs 时间后，插件还没有得到服务器的响应，那么它将宣布该服务器已经死机并失败转移到下一个服务器。应该把该参数设置为一个较大的值。如果所设置的值小于 servlets 进行处理的时间，那么会得到意想不到的后果。 最小值为：10

		最大值为：600
Idempotent	ON	如果该参数设置为 ON，那么当服务器在指定的 HungServerRecoverSecs 时间没有响应，那么插件将进行容错处理。如果设置为 OFF，插件将不进行失败转移。如果所使用的是 Netscape Enterprise Server 插件或 Apache HTTP Server 插件，不同的 URL 与 MIME 类型可以有不同的 Idempotent 参数设置。
CookieName	JSESSIONID	如果改变了 WebLogic 服务器 Web 应用中的 WebLogic 服务器会话 cookie 的名字，那么相应地应该将插件的 CookieName 参数设置为相同的值。WebLogic 会话 cookie 的名字在特定于 WebLogic 的分发描述符的 <session-descriptor>元素中定义（见 http://e-docs.bea.com/wls/docs61/webapp/weblogic_xml.html#session-descriptor
DefaultFileName	none	如果 URI 为“/”，插件将执行以下步骤： 1．裁剪掉 PathTrim 参数所指定的路径 2．在后面加上 DefaultFileName 所指定的文件名 3．在前面加上 PathPrepend 参数所指定的值 这样处理可以防止 WebLogic 服务器的重定向。将 DefaultFileName 设置为代理 WebLogic 服务器的 Web 应用的缺省欢迎页面。例如，如果 DefaultFileName 被设置为 welcome.html，那么下面这个 HTTP 请求： http://somehost/weblogic 变为：http://somehost/weblogic/welcome.html。只有当所有被重定向的 web 应用指定相同的欢迎页面，该参数才起作用。可以参见以下页面的“Configuring Welcome Pages”部分： http://e-docs.bea.com/wls/docs61/webapp/components 对于 Apache 用户，所使用的是 Stronghold 或 Raven 的版本，在 Location 块中定义参数，而非 IfModule 块中
MaxPostSize	-1	POST 数据的允许的最大长度。如果内容的长度超过 MaxPostSize，插件将返回一个错误消息。如果设置为-1，将不检查 POST 数据的长度。设置该参数可以防止通过发送大量数据使服务器过载的“拒绝服务攻击”。
MatchExpression （该参数只适用于 Apache HTTP 服务器）	none	如果采用 MIME 类型方式代理，应该在一个 IfModule 块中使用 MatchExpression 参数设置文件名模式。 下面的一个例子说明了使用 MIME 类型方式的代理： <IfModule mod_weblogic.c> MatchExpression *.jsp WebLogicHost=myHost|paramName=value </IfModule> 下面的一个例子说明了使用路径方式的代理： <IfModule mod_weblogic.c> MatchExpression /weblogic WebLogicHost=myHost|paramName=value </IfModule>
FileCaching	ON	当该参数设置为 ON，如果请求中的 POST 数据大于 2084 个字节，那么 POST 数据保存在硬盘的一个临时文件中，

		然后以 8192 字节为单位传给 WebLogic 服务器。但将 FileCaching 设置为 ON，可能引起的问题是浏览器上将显示一个进展条表明正在进行下载。即使文件还在传输，浏览器也会显示下在已经完成。 如果该参数设置为 OFF，那么当 POST 数据大于 2084 字节时，数据保存在内存中并以 8192 字节为单位发送到 WebLogic 服务器。将参数设置为 OFF 可能会引起问题。因为插件不能进行失败转移，因此如果请求被处理时 WebLogic 服务器宕机了，那么数据将被丢失。
WlForwardPath （只有 Microsoft IIS 才需要定义该单数）	null	如果 WLForwardPath 设置为“/”，那么所有请求都被代理到 WebLogic 服务器。如果只想代理以特定字符串开头的请求，那么应该将 WlForwardPath 参数设置为这个字符串。例如，将 WlForwardPath 设置为/webogic，那么所有以/weblogic 开始的请求都将被代理到 WebLogic 服务器。 如果采用路径方式的代理，那么必须设置该参数。可以为该参数设置多个字符串，字符串之间用逗号隔开，例如：WlForwardPath=/weblogic,/bea
KeepAliveSecs 不要为 Apache HTTP 服务器版本 1.3.x 定义该参数）	30	该参数定义了隔多长时间后，插件与 WebLogic 服务器之间的非活动连接将被关闭。要使该参数生效，KeepAliveEnabled 参数应该设置为 true 该参数的值应该小于或等于在管理控制台的 Server/HTTP 标签页中 Duration 字段的值，或者是服务器 Mbean 中的 keepAliveSecs 属性的值。
KeepAliveEnabled	True	启用插件与 WebLogic 服务器之间的连接池。
QueryFromRequest 只适用于 Apache HTTP 服务器）	OFF	如果该参数设置为 ON，那么 Apache 插件使用（request_rec * r->the request 将查询字符串传递到 WebLogic 服务器。（详细信息，请参见 Apache 文档。）这种行为在以下场合非常有用： 当 Netscape 版本 4.x 浏览器发出的请求的查询字符串中包含空格。 如果你在 HP 上使用 Raven Apache 1.5.2 如果该参数设置为 OFF，那么 Apache 插件使用 request_rec * r->args 将查询字符串传递到 WebLogic 服务器。
MaxSkips（在 Apache 1.3 中不可用）	10	只有当 DynamicServerList 设置为 OFF 时，该参数的设置才生效。如果 WebLogicCluster 参数所设置的列表或由 WebLogic 服务器返回的动态集群列表中的 WebLogic 服务器失败了，那么该失败的服务器被标记为“坏的”，同时插件将连接到列表中的下一个服务器中。 MaxSkip 设置了插件重试“坏”服务器的次数。每当插件接收到一个唯一请求（即不包含 cookie 的请求）时，它会连接到列表中的一个新服务器上
DynamicServerList （只能在 Microsfot IIS 中设置该参数）	ON	如果该参数设置为 OFF，在对由插件所代理的请求进行负载平衡时，不使用动态集群列表，而是使用 WebLogicCluster 参数指定的静态列表。通常情况下，该参数应该设置为 ON 将该参数设置为 ON，可能会产生以下影响： ■ 如果静态列表中的一或多个服务器失败了，那么插件可能会因为重试连接到失效服务器而导致性能的降低。 ■ 当你在集群中新增了一个服务器，如果不重新定义

		这个参数，插件就不能将请求代理到这个新服务器上。WebLogic 服务器会自动地将新服务器加到动态服务器列表，从而使新服务器成为集群的一部分。

Web 服务器插件的 SSL 参数

注意：以下所有参数都区分大小写。

参数	默认值	描述
SecureProxy	OFF	如果该参数设置为 ON，那么 WebLogic 服务器插件与 WebLogic 服务器之间的通信将采用 SSL 协议。在设置该参数之前，不要忘了先定义 WebLogic 服务器监听 SSL 请求的端口。 该参数可以在两个层次上定义：在主服务器配置或虚拟主机配置中定义（若定义了虚拟主机）。如果没有在虚拟主机的配置中指定这个参数，那么它将从主服务器的配置中继承 SSL 配置。
TrustedCAFile	none	包含数字证书的文件的名字。该数字证书由可靠的证书认证机构发放并被 WebLogic 服务器插件使用。如果 SecurityProxy 参数设置为 ON，那么必须设置这个参数。 文件名必须包括这个文件的全路径。
RequireSSLHostMatch	true	该参数决定 WebLogic 服务器代理插件所连接的 WebLogic 服务器主机的名字是否必须与所连接的 WebLogic 服务器所使用的数字证书中的 Subject Distinguished Name 字段匹配。
SSLHostMatchOID	22	ASN.1 对象ID (OID)对双数字证书中的Subject Distinguished Name中的哪一个字段，它用于执行和主机名的匹配。 该参数的缺省值为Subject Distinguished Name中的 Common OID 值包括： ■ Sur Name—23 ■ Common Name—22 ■ Email—13 ■ Organizational Unit—30 ■ Organization—29 ■ Locality—26