1 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
- ✓ 运行时对象与配置对象
- ✓ 对日志消息的集中访问

BEA WebLogic Server™ 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，资源配置的监控，诊断并修改问题，监控并评估系统性能，分发 Web 应用、EJB 以及其它资源。

WebLogic 服务器提供了一个健壮的基于 Web 的工具——管理控制台，它是执行上述任务的主要工具。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 管理扩展标准（JMX），它是 WebLogic 资源管理的基础。

你可以用管理控制台来配置资源的属性，分发应用及组件，监控资源的使用情况（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，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。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，该应用的各组成部分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。

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——EJBs 或 Web 应用，服务器或集群，JDBC 连接池等等都定义在一个域配置中。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看待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控受管服务器。管理服务器的作用是配置与监控受管服务器。如果管理服务器宕机了，部署在受管服务器上的应用不受影响，可以继续处理客户端的请求；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散到一组受管服务器上能带来一些好处。将 EJB 以及其它组件分布到一组服务器上可以保证主应用的可用性。如果不同的组件完成不同的功能可以提高系统的性能，例如将数据库访问与帐单事务分在不同的受管服务器上处理。象 EJB 这种可以实现各种功能的组件或应用是可以被隔离开的，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。在管理服务器运行期间，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控或配置整个域的资源。

配置存储库（configuration repository）还保存了其它非活动的配置文件，你可以通过管理终端来编辑这些文件。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 Configuation 链接来访问非活动的配置文件。

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 为管理服务器终端的名字或 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树形结构，你可以通过这棵树来浏览数据表、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。树的最左边的节点是一些文件夹，如果文件夹里包含有叶子节点，那么你可以点击其左边的加号来展开树并访问叶子节点。

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创建一组具有管理权限的用户，此后这些用户就可以通过管理控制台来执行管理工作了。

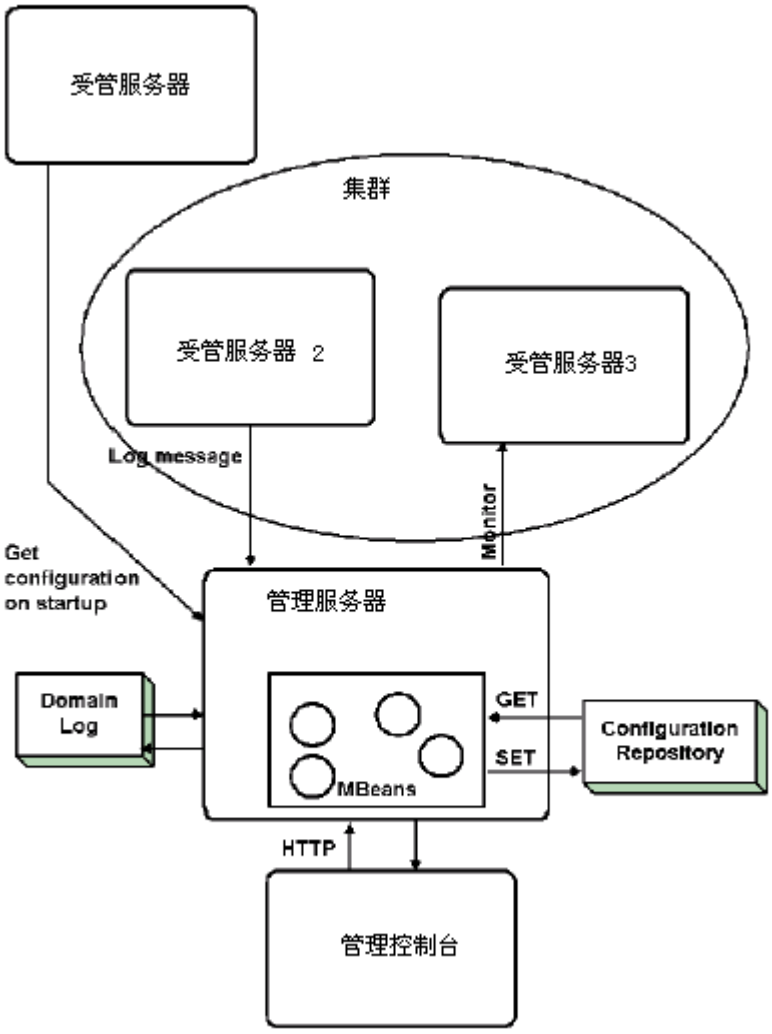
运行时对象与配置对象

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。

管理服务器包含了配置 Mbeans 与运行时 Mbeans。管理 Mbeans 提供了配置属性的 SET(写)与 GET（读）访问。

运行时 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，管理控制台执行 GET 操作获取当前的属性值。



管理服务使域资源的属性可以被动态修改，即使 WebLogic 服务器正在运行，也可以修改属性。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控属性。可以用命令行工具创建 script，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。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”）

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
- ✓ 将一个 WebLogic 受管服务器加到域中
- ✓ 启动 WebLogic 受管服务器
- ✓ 从老版本 WebLogic 服务器升级
- ✓ 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

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，其中必须有一个管理服务器，该域中的其它 WebLogic 服务器被称为受管服务器。你可以将服务器启动为 WebLogic 管理服务器或 WebLogic 受管服务器。

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只需要两个步骤：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。

启动时的错误消息

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 以及一个特殊的启动日志文件——servername-startup.log 文件中（其中 servername 是服务器的名字）。如果启动成功，那么该日志中的最后一条消息会指向本地服务器的日志文件所

在的位置。有关 WebLogic 服务器的日志子系统的更多信息，请参见“利用日志消息管理 WebLogic 服务器”。

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 Start 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，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，又没有 password.ini 文件，那么在启动时系统会提示你输入口令。

从 Start 菜单启动 WebLogic 管理服务器

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

调用 Start 菜单中的 WebLogic Server 快捷方式实际上就是启动了 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。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 应用，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。

  上述参数值会影响 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。

- ✓ 提供用户名与口令

  缺省用户为 system，口令为安装时所输入的口令。要将 WebLogic 管理服务器启动为某一特定用户，那么命令行应该使用以下参数：

  -DwebLogic.management.username=username

  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动态值。

- ✓ 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。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。

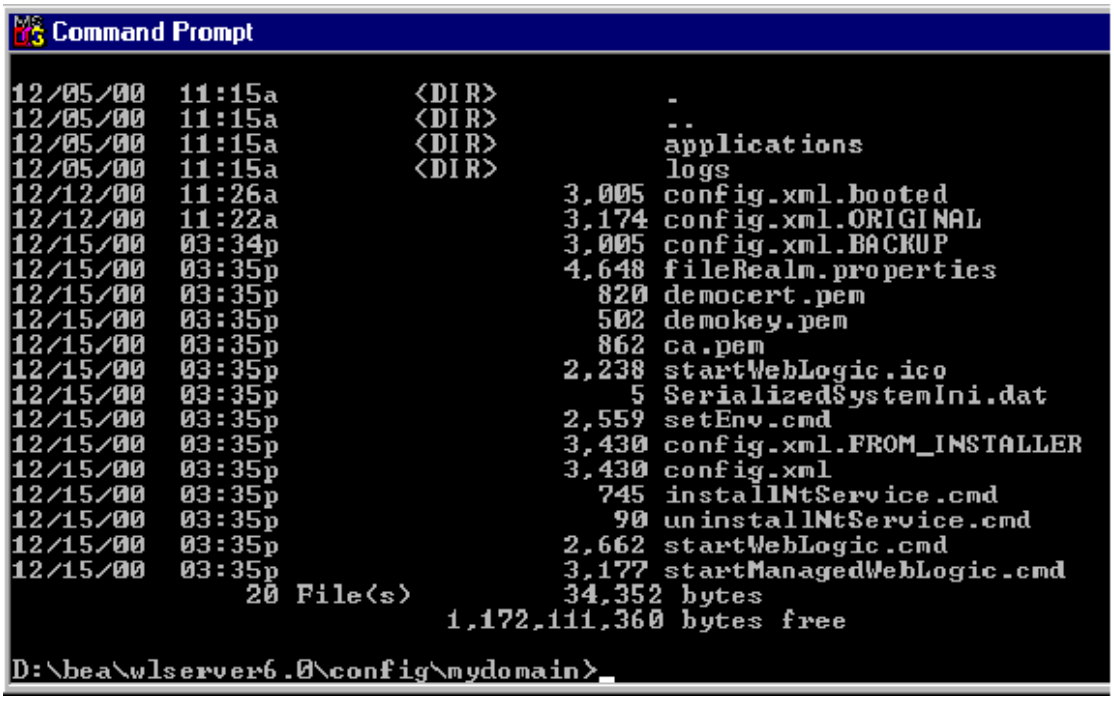
- ✓ 在命令行中使用以下参数可以在启动管理服务器时指定域配置文件的名字：

  -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。

图1.1 - 配置存储库中域目录中的内容



启动管理服务器所使用的域配置使这个域成为活动域。

- ✓ 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.*attribute*=*value*

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 软件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，应注意以下问题：

- ✓ 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分发在受管服务器中。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。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，它可以帮助你从被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 false。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

列表来检查运行着的受管服务器。

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。管理服务器的发现操作不会使管理服务器监控受管服务器或者是更改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 受管服务器加入到域

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

1. 启动域中的管理服务器
2. 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）。
3. 在管理控制台中为服务器所在的机器创建一个条目（Machines->Create a new machine， Servers->Create a new server）（如果该服务器与管理服务器在不同的机器上）

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

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。

以下是启动 WebLogic 受管服务器所必须的参数：

- ✓ 指定 Java 堆的最大内存与最小内存

  例如，可以为 WebLogic 服务器分配 64M Java 堆内存，这是一个默认值。相应的选项为：java -ms64 与 -mx64。

  这两个参数的值对 WebLogic 服务器的性能有很大影响。在生成环境中，你应该谨慎考虑适合于应用与环境的值。

- ✓ 设置 java -classpath 选项

  该选项的最基本设置请参见“设置类路径选项”中的内容

- ✓ 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*

- ✓ 提供用户名与口令

  缺省的用户为 System，System 用户的口令是在安装时指定的。要用其它用户来启动 WebLogic 受管服务器，那么在命令行中使用以下参数：

  -Dweblogic.management.password=*password*

  有关如何使用口令的内容，请参见“在启动 WebLogic 服务器时使用口令”。

- ✓ 如果要使用 SSL 协议，那么在启动时需要传入私有密钥口令以便服务器能够对 SSL 私钥文件进行解密。在启动服务器的命令中使用以下参数可以将 SSL 私有密钥传递给服务器。

  -Dweblogic.pkpassword=*pkpassword*

  其中 pkppassword 指的是私有密钥口令

- ✓ 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”。

注意：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。

注意：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。

- ✓ 你可以在命令行中指定 WebLogic 配置属性的值。由这种方式设置的值将成为该属性的运行时值，保存在永久配置中的值将被忽略。在命令行中指定 WebLogic 属性值使用以下形式：

  -Dweblogic.*attribute*=*value*

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。

使用脚本示例时应注意以下事项：

- ✓ 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。

从老版本 WebLogic 服务器升级

如果你想用老版本的启动脚本来启动 WebLogic 6.0 服务器软件，那么你需要对该脚本作一些修改。以下是 WebLogic6.0 与以前版本的主要不同点：

- ✓ 动态类装载

  在启动以前版本的WebLogic服务器时，需要设置两种类路径(Class Path)

  1. Java 系统类路径
  2. WebLogic 类路径

  WebLogic 类路径属性用于动态类装载的配置。6.0 版本的 WebLogic 不再使用 WebLogic 类路径属性，同时对 Java 系统路径也做了修改。因此，启动老版本 WebLogic 软件的脚本需要做相应的修改才能用来启动 WebLogic 6.0 服务器。在这个版本中，需要动态装载的类由应用负责设置，这可以通过应用的 XML 描述符文件指定类的位置来实现。

  有关 Java 类路径设置的信息，请参见“设置 Classpath 选项”。
- ✓ 命令中不需要指定许可证文件或 Policy 文件所在的位置
- ✓ 管理服务器与受管服务器的区别是在这个版本才出现的。如果你想将 WebLogic 服务器启动为受管服务器，那么在启动时，需要有指向一个正在运行着的管理服务器的 URL。
- ✓ 启动 WebLogic 受管服务器使用新的启动脚本——startManagedWebLogic.cmd（Windows）与 startManagedWebLogic.sh（UNIX），而原来的 startWebLogic.sh 与 startWebLogic.cmd 则用来启动管理服务器。

从管理控制台终止 WebLogic 服务器

要终止一个 WebLogic 服务器：

- ✓ 在管理控制台的域树上（位于左边窗格），选择要停止的服务器
- ✓ 选择 Monitoring->General 标签页，点“Shutdown this server”链接

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 WebLogic 服务器管理员权限的用户，缺省为 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。
- ✓ 在 Monitoring->General 标签页上，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。
- ✓ 在 Monitoring->General 标签页上，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

2 节点管理器

本章将介绍以下内容：

- ✓ 节点管理器（Node Manager）概述
- ✓ 配置与启动节点管理器
- ✓ 受管服务器的远程启动与 Killing

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 kill 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。

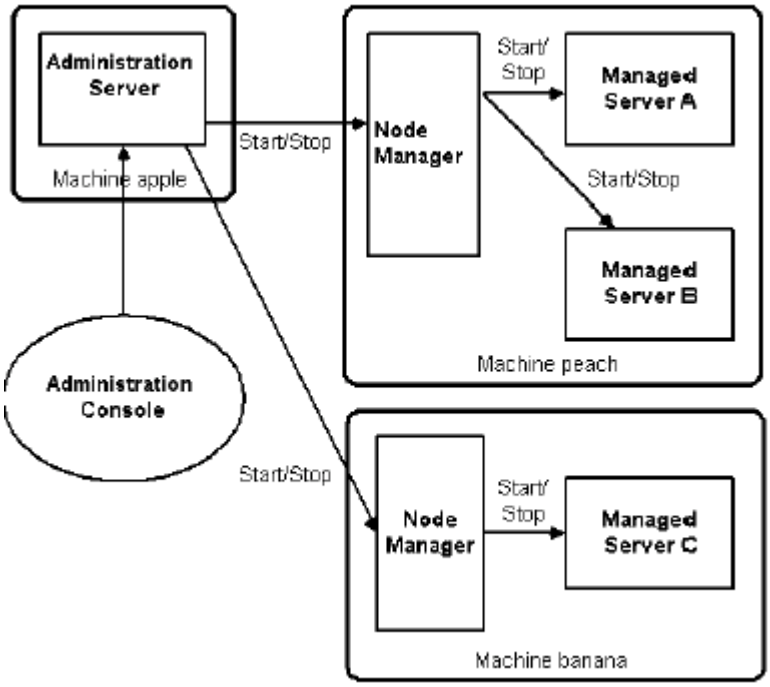
你可以通过管理控制台来结束受管服务器，另一种方式是用节点管理器 kill 远程受管服务器。当远程服务器被 hung 或没有响应时，就需要杀掉远程服务器进程。

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 killing。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。这保证了机器上的管理服务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，可以启动或杀掉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）。

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。节点管理器将这些信息保存在节点管理器日志目录中的文件里，这些消息同时会显示在管理控制台的右窗格中。

缺省情况下，节点管理器的日志文件保存在节点管理器被启动时所在的那个目录中。节点管理器为每个由它启动的受管服务器创建一个日志文件子目录。



配置与启动节点管理器

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
```

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，应该编辑该可靠主机文件，使它只包含可以启动或 killing 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。可靠主机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。

有关数字证书与安全套接层的更多信息，请参见“安全管理”中的内容。

启动节点管理器

启动节点管理器之前，首先应该保证 JAVA_HOME 环境变量指向了 JDK 的根目

录。节点管理器需要用到 JDK。以下是一个 Window NT 中的例子：

```
set JAVA_HOME=D:\bea\jdk130
```

节点管理器 JDK 版本的要求与 WebLogic 服务器的 JDK 版本要求一样。

类路径必须作为 Java 命令行的选项或者作为一个环境变量来设置。下面（在 Window NT 中）是一个将类路径设置为一个环境变量的例子。

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

启动节点管理器的命令为：

```
java weblogic.nodemanager.NodeManager
```

命令行参数

应该用以下参数指定节点管理器所在机器的 IP 地址或主机名：

-Dweblogic.nodemanager.listenAddress=*address*

节点管理器监听管理服务器请求的缺省端口号为 5555，你可以用以下启动参数设置监听端口号：

-Dweblogic.nodemanager.listenPort=*port*

节点管理器为它所负责的每个受管服务器创建日志文件。缺省情况下，这些子目录位于 NodeManagerLogs 目录中，你可以用以下启动参数改变目录的位置：

-Dweblogic.nodemanager.saveFilesDirectory=*path*

节点管理器使用安全套接层与管理服务器进行通信。因此启动节点管理器时，必须指定一个数字证书。应该用以下参数指定数字证书所在的位置：

-Dweblogic.nodemanager.certificateFile=*path_to_cert*

如果在可靠主机文件中使用的是主机名而不是 IP 地址，那么应该包括以下启动参数：

-Dweblogic.nodemanager.reverseDnsEnabled=true

缺省情况下，DNS 是禁用的。

可以用以下参数指定可靠主机文件所在的位置：

-Dweblogic.nodemanager.trustedHosts=path

其中 path 是指文件 nodemanager.hosts 所在的位置。

类路径选项

节点管理器要用到一些 WebLogic 服务器同样也要使用的 Java 类。在启动管理服务器时，-classpath 选项中包含以下值：

- ✓ */weblogic*/lib/weblogic_sp.jar
- ✓ */weblogic*/lib/weblogic.jar

启动管理服务器

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，调用管理控制台（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：

1. 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。
2. 填写 Node Manager 标签页的以下字段：

   Listen Address: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：该字段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。默认证书为 democert。在生成环境中，建议你换一个证书。

3. 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. 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。
2. 输入受管服务器启动所需要的类路径

   Classpath 选项中至少应该指定以下内容：

   /weblogic/lib/weblogic_sp.jar

   /weblogic/lib/weblogic.jar

   该选项可能还需包含启动受管服务器所用到的 JDK 的根目录。有关设置类路径的更多信息，可以参见“启动与终止 WebLogic 服务器”中的内容。
3. 在 Arguments 字段中输入启动命令所需要的参数。

例如，你想设置 Java 堆内存的最大最小值。可以使用-ms64m 与-mx64m 选项将默认的 65M java 堆内存指定给 WebLogic 服务器。

不需要指定服务器的名字，也不需要指定用户名与口令

远程启动与 Killing 受管服务器

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. 右点左窗格中受管服务器的名字
2. 选 kill this server **...**

✓

3 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
- ✓ 服务器的配置任务列表
- ✓ 集群的配置任务列表
- ✓ 创建一个新的域

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 目录中装载配置文件。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，这个服务器将是管理服务器。

在生产环境中，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。

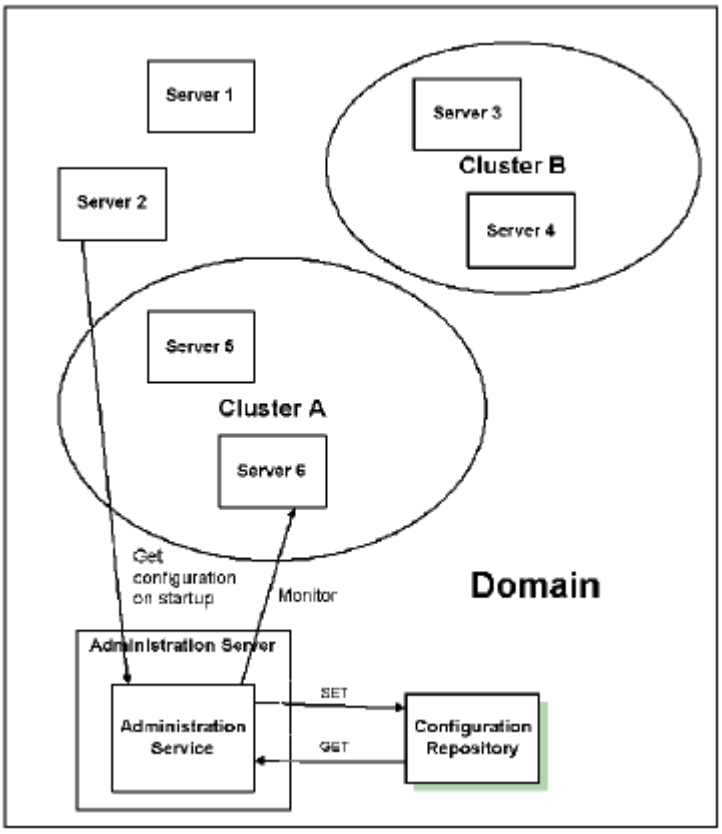


图3.1 - WebLogic 服务器的配置

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：

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）。

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前面，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，所有分发了某 EJB 的服务器必须具有相同的分发与持久化配置。也就是说所有服务器都应该有相同的 JDBC 配置。
- ✓ 所有分发了 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 View JNDI Tree 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 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 节点监控服务器的安全。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 节点来分发 EJB。点击需要分发 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 节点来监控分发在某一服务器上的所有 EJB。点击需要监控的服务器，点击 Monitor All EJB Deployments 连接来显示 EJB 分发表。
- ✓ 通过管理控制台的 Server 节点将 web 应用组件分发在某一服务器上。选择要分发 web 应用的服务器。选择需要分发的 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 Deployments 表。
- ✓ 通过管理控制台的 Server 节点在服务器上分发启动与终止类。点击需要分发启动类的服务器，然后点击需要分发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 Shutdown Class 控件来分发终止类的过程与此相同。
- ✓ 通过管理控制台的 Server 节点来分配 web 服务器。点击一个用来分发 web 应用的服务器。显示在右窗格的对话框包含了与该实例相关的标签页。在 Availble 列中点击一到多个需要分发到该服务器的 web 应用，然后使用移动控件将它们移动到 Chosen 列中。点击 Apply 来保存上述选择。
- ✓ 通过管理控制台的 Server 节点为服务器分配 JDBC 连接池。Click a server for web-server assignment。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 节点监控某一服务器的 JDBC 连接池。点及一个需要监控的连接池所在的服务器。点 Monitor All JDBC Connection Pools on This Server 链接。分配给该服务器的所有连接池都显示在右窗格中的 JDBC 连接池表格中。
- ✓ 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 节点监控某一服务器上的所有 WLEC 连接池。选择一个需要监控连接池的服务器。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显示在右窗格中的 WLEC Connection Pools 表格中。
- ✓ 通过管理控制台的 Server 节点为一台服务器分配 JMS 服务器，连接工厂，以及消息收信方。点击一个需要分配 JMS 的服务器，从 Available 列中选择需要分配给这个服务器的 JMS 服务器，然后点移动组件把所选择的 JMS 服务器移到 Chosen 列。使用 JMS Connection Factories 组件以及 JMS Destination 组件按照分配 JMS 服务器的步骤为服务器分配连接工

厂与消息收信方。

- ✓ 通过管理控制台的 Server 节点为一台服务器分配 XML registries。选择要分配 XML registry 的服务器，从 XML Registry 下拉列表中选择一个 registry。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。

新建一个域

本节说明如何创建一个新的域。所有 WebLogic 域的配置信息都保存在 /config 目录的配置存储库中。每个域在/config 目录下都有一个同名子目录。

安装 WebLogic 服务器时，建议你对缺省的/mydomain 配置目录进行备份保存在一个 zip 文件中。并保存该 zip 文件的一个备份。/mydomain 目录包含了域配置的基本组件，例如 fileRealm.properties 文件与配置文件。

以下是新建一个域的步骤：

1. 启动一个已存在的域（例如缺省的 mydomain 域）中的管理服务器。
2. 使用以下 URI 调用管理控制台。

http://*hostname*:*port*/console

其中 hostname 是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）。

3. 选择 mydomain->Create 或编辑其它域

   这将显示 domains 表

4. 选择 Default->Create a new Domain

   输入新域的名字并点 Create

5. 从左窗格的域列表中选择一个新域，使它成为当前域。

6. 为新建的域创建一个 Administration Server 条目。

   选择 Servers->Create a new Server

   输入新建管理服务器的名字，然后点 Create

7. 管理控制台会在\configure 目录下创建一个与新建域同名的子目录，并在这个子目录中创建一个配置文件 config.xml。然后使用 shell 命令或使用 Windows 的浏览器在域目录下创建一个\application 子目录。

8. 将管理控制台应用复制到新建的\applications 目录。即把位于 mydomain 域的\applications 目录中的 console.war 文件复制到新建的\applications 目录。

9. 缺省的 mydomain 目录中包含启动 WebLogic 服务器的脚本。在 windows 中，启动 WebLogic 服务器的脚本为 startWebLogic 与 startManagedWeblogic.cmd，在 UNIX 中为 startWebLogic.sh 与 startManagedWebLogic.sh。把这些启动脚本复制到新的域目录中。

10. 你可以在文本编辑器中编辑上述启动脚本。缺省情况下，域的名字设置为：

    -Dweblogic.Domain=mydomain

    用新建域的名字代替 mydomain

    管理服务器的名字缺省设置为：

    -Dweblogic.Name=MyServer

    用新建管理服务器的名字代替上面的 Myserver。

11. 启动脚本的最后是一个 cd 命令：

    cd config\mydomain

    使用新建域的子目录名替换上面的 mydomain。启动脚本还包含下面一行：

    echo startWebLogic.cmd must be run from the config\mydomain directory

    用新建域的名字替换上面的 mydomain

12. 把缺省域目录中的 SerializedSystemIni.dat 文件复制到新建的域目录中。如果还没有复制这个文件，千万别启动新建的管理服务器。

13. 如果在安装时创建了 password.ini 文件，还必须把 mydomain 目录中的 password.ini 文件复制到新建域的子目录中。

完成上述步骤后，就可以启动新建域的管理服务器了。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
- ✓ 管理控制台的 Monitoring 页面一览表

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：配置 Mbeans 与运行时 Mbeans.配置 Mbeans 用于域的配置，运行时 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

你可以通过服务器表格以及单个服务器的监控标签页来监控 WebLogic 服务器。服务器表格提供了域中所有服务器的状态摘要信息。如果服务器只将其部

分日志消息上传到域日志中，那么通过本地服务器的日志来修复故障或研究事件将是非常有用的。

有关日志文件以及日志子系统的更多内容，请参见“使用日志消息管理 WebLogic 服务器”。

每个 WebLogic 服务器的监控标签页都包含了该服务器的监控数据。通过 Logging 标签页可以访问服务器的本地日志（即保存在运行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。从这个标签页，你还可以访问 JNDI 树、Execute Queues 表、Active Sockets 表以及 Connections 表。Execute Queues 表包含了有关性能的信息，如挂起时间最长的请求以及当前被起的请求数量。

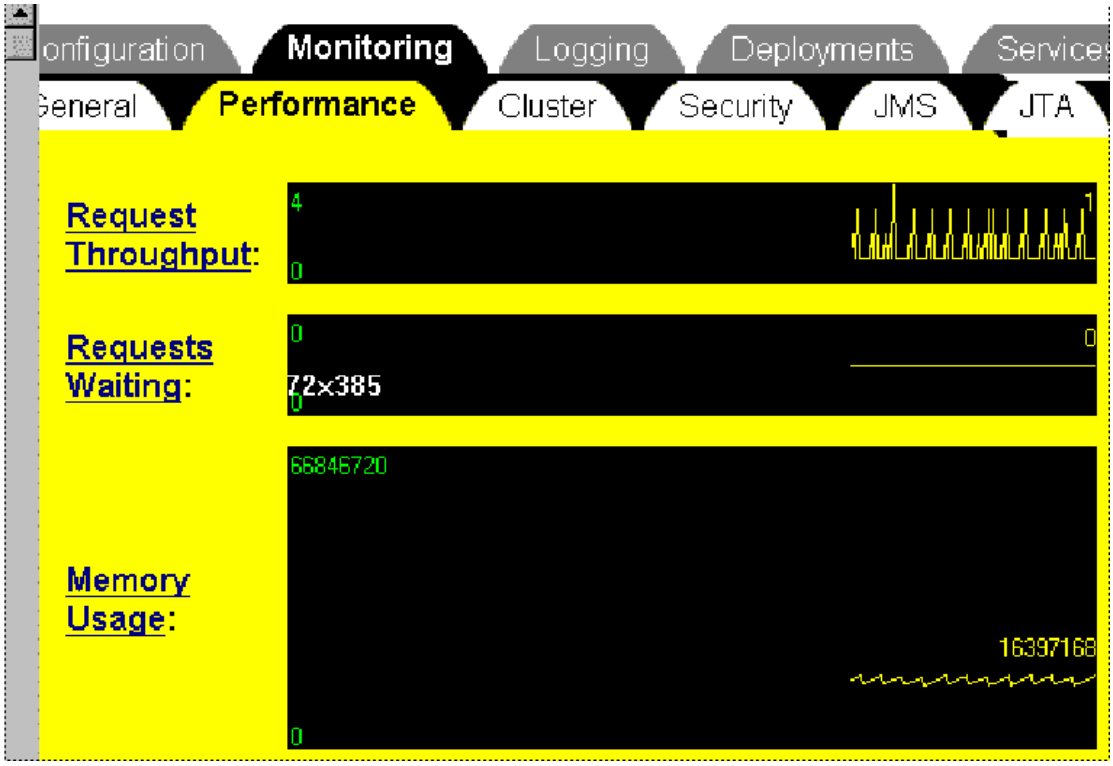
终止或暂停服务器

你可以通过 Monitoring->General 标签页来终止或暂停服务器。暂停的服务器只能接收管理服务器的请求，客户的请求被忽略。

性能

Monitoring->Performance 标签页以图形的方式描述了有关 JVM 内存堆使用状况、请求对象与等待请求的实时数据。通过该页面，你还可以强制 JVM 对内存堆执行垃圾收集。

图 4-1 服务性能图



Java 堆中存放了 Java 对象（可以是活动对象也可以是死亡对象）。正常情况下，因为 JVM 会自动进行垃圾收集，因此不需要你手工处理。当 JVM 的内存要用光时，它会停止所有执行应并使用垃圾收集算法来释放不在被应用使用的空间。

另一方面，程序员在调试应用时可能会执行强制性垃圾收集。手工执行垃圾收集非常有用，例如，可以用来测试 JVM 内存泄漏。

集群数据

Monitoring->Cluster 标签页提供了集群的信息(例如集群中有多少服务器处于活动状态）。

服务器安全

Monitoring->Security 标签页包含了以下信息：无效登录的统计信息，被锁用户的统计信息与开启用户的统计信息。

JMS

Monitoring->JMS 标签页提供了有关 JMS 服务器与连接的统计信息。在该页，你可以链接到活动 JMS 连接表格以及活动 JMS 服务器表格。通过这些表格，你

可以监控诸如当前总会话数等属性。

JTA

Monitoring->JTA 标签页提供了 Java 事务子系统的统计信息，如有关事务与所有回滚的统计信息。从该页面，你还可以链接到按资源或名字所选择的事务列表以及 in-flight 事务表格。

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 字段的值近似于 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，那么应该增加连接池的大小。

Shrink Period 字段用来指定 JDBC 子系统在连接池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。

管理控制台的 Monitoring 页面一览表

下表列出了管理控制台中所有的表格以及监控标签页面。

表 4-1 Monitoring 页面一览表

页面	页面的路径	监控的数据
Monitoring 标签页面		
一般的服务器信息	服务器名字-> Monitoring -> General	服务器的状态以及 activation time
服务器性能	服务器名字 -> Monitoring -> Performance	请求吞吐量、JVM 的内存使用以及等候请求的实时图表

集群的统计信息	服务器名字-> Monitoring -> Cluster	有关集群的统计信息，如集群中活动服务器的数量、sent and received fragments
服务器的安全信息	服务器名字-> Monitoring -> Security	无效的登录数量以及所有被锁定的用户数与未锁定的用户数
服务器的版本信息	Servername -> Monitoring -> Version	JDK, WebLogic 以及操作系统的版本信息
集群	Clusters -> clustername -> Monitoring	集群的相关统计信息，如 the number of alive servers and sent and received fragments
实体表		
服务器	Servers	服务器的相关数据，例如内存使用、启动时间、状态、cluster participation，无效登录、堆的状态、套接字的数量 and total restarts
执行队列	Servername -> Monitoring -> General -> Monitor Execute Queues on this server	请求吞吐量、JVM 内存使用状况以及等待请求等信息的实时显示
执行套接字	Servername -> Monitoring -> General -> Monitor Active Sockets on this server	活动套接字的协议及其它属性
连接	Servername -> Monitoring -> General -> Monitor Connections on this server	连接的相关数据，如连接时间、远程地址、所发送的字节数与接收到的字节数
集群	Clusters	缺省的负载算法以及多点传送地址的相关数据
Transactions By Name	Servername -> Monitoring -> JTA -> Monitor Transactions by Name on this server	按名字所组织的事务的相关数据
Transaction By Resource	Servername -> Monitoring -> JTA -> Monitor Transactions by Name on this server	按资源所组织的事务的相关数据
活动事务	Servername -> Monitoring -> JTA -> Monitor In-flight Transactions on this server	该服务器中有关活动事务的数据
机器	Machines	机器的地址以及其它属性
应用	Applications	应用列表
EJB 的分发	Deployments -> EJB	各个 EJB 的 URL、应用名字及其它属性
Web 应用	Deployments -> Web Applications	Web 应用的 URL 以及它的缺省 Servlet 等属性
活动的 Web 应用	Deployments -> Web Applications -> appname -> Monitoring -> Monitor all instances of appname	Data about deployed copies of this Web application
Web 应用的 Servlets	Deployments -> Web Applications -> appname -> Monitoring -> Monitor all servlets for this Web Application	有关 Web 应用的统计数据，如最大池容量以及最长的执行时间
启动类与终止类	Deployments -> Startup & Shutdown	经过注册的启动类与终止类列表
JDBC 连接池	Services -> JDBC -> Connection Pools	JDBC 连接池的初始容量、容量增量及其它属性
JDBC 多池	Services -> JDBC -> Multipools	JDBC 多池的负载平衡及其它属性
JDBC 数据源	Services -> JDBC -> Data Sources	JDBC 数据源的 Pool 名、JNDI 名字以及其它属性
JDBC Tx 数据源	Services -> JDBC -> Tx Data Sources	JDBC Tx 数据源的 Pool 名字、JNDI 名字以及其它属性
JMS Connection Factories	Services -> JMS -> Connection Factories	JMS connection factories 的 JNDI 名字，客户号，缺省优先级及其它属性
JMS 模板	Services -> JMS -> Templates	JMS 模板的有关数据
JMS 收信方关键字	Services -> JMS -> Desination Keys	JMS 收信方的关键字类型与其它属性
JMS 存储器	Services -> JMS -> Stores	JMS 存储器的描述信息

JMS 服务器	Services -> JMS -> Servers	JMS 服务器的有关信息
活动 JMS 服务	Services -> JMS -> Servers -> Monitor all Active JMS Services	连接的上限及其它活动 JMS 服务的有关数据
活动 JMS 服务器	Services -> JMS ->Servers -> Monitor all instances	有关会话与消息挂起的统计信息以及其它数据。
活动 JMS 收信方	Services -> JMS ->Servers -> Monitor all Active JMS Destinations	活动 JMS 收信方的使用者、所收到的消息及其它属性。
活动 JMS 会话池	Services -> JMS -> Servers -> Monitor all Active JMS Session Pools	使用者数量的上限及其它属性
JMS 收信方	Services -> JMS -> jmsservername -> Destinations	JNDI 名字及其它数据
JMS 会话池	Services -> JMS -> jmsservername -> Session Pools	JMS 会话池的肯定应答模式、会话的最大量及其它属性
XML 注册记录	Services -> XML -> XML Registries	DocumentBuilderFactories 以及 SAXPArserFactories 列表
WLEC 连接池	Services -> WLEC -> WLEC Connection Pools	WebLogic 企业域的名字、失败转移地址、最大最小容量以及其它信息
Jolt 连接池	Services -> Jolt	Jolt 连接池的失败转移地址、连接池的最大与最小容量以及其它信息
活动 Jolt 连接池	Service-> Jolt->*joltconnectionpool name*->Monitoring -> Monitor all active pools	Jolt 连接池的最大实例数、当前连接及其它实例数据
虚拟主机	ServiceVirtual Hosts	虚拟主机的格式、日志文件及其它属性
邮件会话	Service -> Mail	邮件会话的名字与虚拟列表
文件 T3	Services -> File T3	文件的名字与路径
用户	Securit ->Users	用户列表
组	Security ->Groups	组列表
访问控制列表	Security->ACLs	ACL 列表
缓存域	Security->Caching Realms	缓存域列表
域	Security->Realms	对域的描述
域日志过滤器	Domain Log Filters	Servers on which the filter is registered 以及用于过滤消息的属性

4 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。分发在 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。

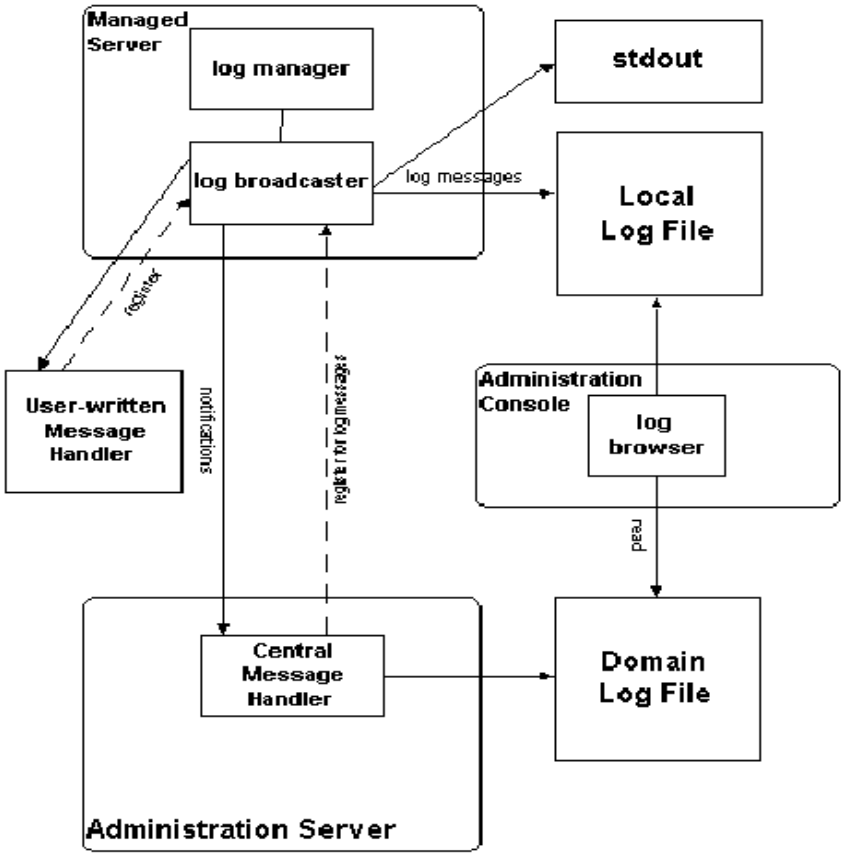
嵌在 WebLogic 服务器的 JMX（Java 管理扩展）程序，会把 WebLogic 服务器产生的日志消息传送到管理服务器。如果本地 WebLogic 服务器主动将消息发送到其它实体，用 JMX 的术语来说，这就是“布告”。

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，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



本地服务器的日志文件

老版本的 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 6.0 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。

启动日志

在 WebLogic 服务器启动时，如果还未完成初始化就发生了错误，那么这些错误将输出到标准输出以及名字为 weblogic-startup.log 的本地服务器启动日志文件中。如果启动成功，那么启动日志文件的最后一条消息将指向记录普通日志信息的本地服务器日志文件。

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。

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形迹。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：ThreadID, UserID, Transaction ID,

Message ID, 以及 Message Text。

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

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。
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）产生的消息另外还包含了其它预定义的信息，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“严重级别”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	否	用于报告普通操作
警告	否	发生了可疑的操作或配置，但对普通操作没有影响
错误	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	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	是	发生了系统或服务错误。系统可以恢复但是可能会造成瞬间或永久性的服务性能降低
警报	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	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。

5 分发应用

本章将介绍如何在 WebLogic 服务器上分发应用程序以及应用组件，内容如下

- ✓ 分发格式
- ✓ 通过管理控制台分发应用
- ✓ 动态分发

分发格式

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，我们建议所有应用组件都采用包的形式。

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。

用管理控制台分发应用

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

步骤 1：配置与分发应用

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. 填写以下字段
   应用的名字
   应用（EAR 文件）所在的路径

应用是否已经被部署到 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，管理服务器会在为应用在域配置文件中（/config/domain_name/config.xml）加上与被分发的应用或应用组件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 服务器中分发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）分发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分发的目标。

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

   选择分发顺序。该顺序指定了服务器启动时 Web 应用的分发顺序。（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

   选择分发顺序。它决定了服务器启动时，该 EJB 被分发的顺序。

   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 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 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 EJBs 部署到受管服务器上的步骤：

1. 选择 Deployments->Connectors 调用 EJB Deployment 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

   选择分发顺序。它决定了服务器启动时，该资源连接器被分发的顺序。（见“分发顺序”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 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 EJB 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，那么管理控制台将调用一个文件分发 servlet 将文件复制到目标服务器的/config/*domain_nam*e/applications/wlnotdelete 目录中。

分发顺序

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，例如各 EJB 的部署顺序。分发顺序为 0 的组件是同类型组件中最先部署的组件。

但是，各种组件类型之间的分发顺序不受用户所定义的同类型分发顺序所影响。WebLogic 在启动时，按以下顺序启动各类型的组件：

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

自动分发

自动分发（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分发方式来测试应用。不建议你在生产环境或受管服务器上使用使用自动分发。

如果目标 WebLogic 服务器域启用了自动分发，当应用被复制到 WebLogic 管理服务器的*/config/domain_name/applications* 目录时，管理服务器会自动检测到新应用并自动部署该应用（如果管理服务器是运行着的）。（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 服务器启动时将分发这个应用。

启用或禁用自动部署

缺省情况下，自动部署是启用的。

自动分发选项是否开启决定了应用的分发有两种方式：

- ✓ 如果启用了自动分发选项，那么在把应用复制到管理服务器的 /config/domain_name/applications 目录下后，应用会自动分发到管理服务器上。
- ✓ 如果禁用了自动分发选项，你必须在这个应用的 Configuration 标签页上指明要分发该应用，那么所安装的这个应用才会被分发。

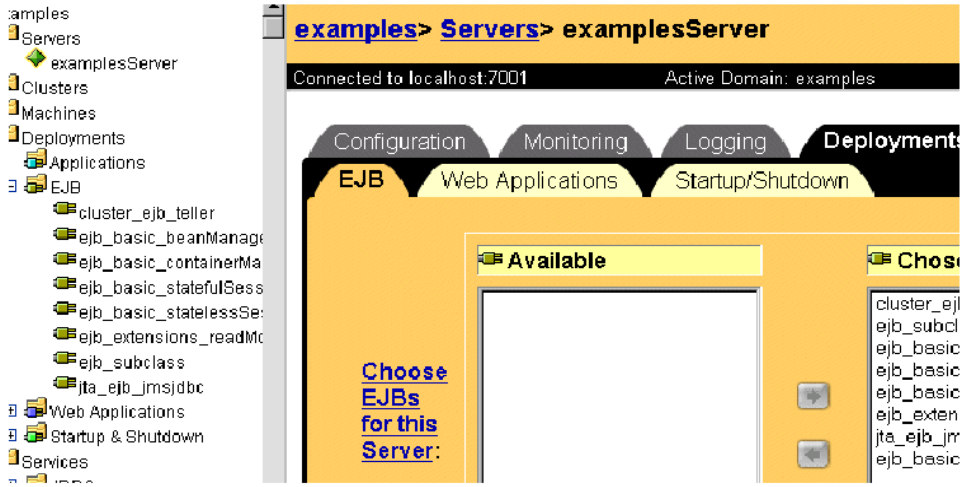
3、将应用组件（或者是 Web 应用组件与 EJB）的实例分发在受管服务器上

当应用被安装到管理服务器上（/config/domain_name/applications 目录）之后，就可以将应用的组件分发到 WebLogic 受管服务器上。

在管理控制台中，打开目标服务器的 Deployments->EJB（见图 6-3）或 Deployments->Web Applications 标签页，选择需要分发的应用组件。

或者通过组件的 Targets 标签页面来选择要分发应用组件的目标服务器。

图5.1 - 某一服务器的 Deployment->EJB 标签页



如果要把应用组件（例如 EJBs 或 WAR 文件）分发到集群中的受管服务器中，那么就必须保证集群中的所有受管服务器都分发了该应用组件，选择集群作为分发目标就可以确保这一点。

动态分发

当你将应用拷贝到 WebLogic 管理服务器的 /config/domain_name/applications 目录时（子目录 domain_name 是指要分发应用的 WebLogic 域的名字），如果目标 WebLogic 服务器域启用了自动分发并且运行了管理服务器，那么管理服务器将会检测到新应用并自动分发该应用。这种分发应用的方式被称为动态分发。如果在把应用复制到 /applications 目录下时没有运行 WebLogic 服务器，那么应用将在 WebLogic 服务器重新启动时分发。

缺省情况下，自动分发是启用的。如果禁用了自动分发，那么仍然可以通过管理控制台来手工分发应用组件，这种分发应用的方式被称为静态分发。

启用与禁用自动分发

缺省情况下，自动分发是启用的。

要确定是否启用了自动分发选项，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分发是否启用以及设置 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 秒种检查 \applications 目录的变化。

自动分发展开目录格式的应用

自动分发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。

以下是自动分发展开式应用的步骤：

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。如果启用了自动分发，那么该应用被自动部署。

卸载或重新分发被自动分发的应用

被自动分发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分发。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

你可以动态地重新分发分发展开式应用或应用组件。如果所分发的应用是展开格式，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分发这个应用。

如果要更新展开式应用目录中的文件，按以下步骤：

1. 第一次分发展开式应用时，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，它会自动分发展开式应用目录下的内容。

6 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“POST 拒绝服务”攻击
- ✓ 设置 WebLogic 服务器的 HTTP Tunneling
- ✓ 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，提供 HTML 文件、图像文件、servlets 以及 JSP 等静态页面服务。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。

- ✓ Default Web Application

  缺省应用处理不能被其它 Web 应用解析的请求。访问缺省应用的 URI 不需要包含上下文路径（Web 应用的上下文通常是 Web 应用的名字）。

- ✓ Post Timeout Seconds

  WebLogic 服务器在相邻两次接收 HTTP POST 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。

- ✓ Max Post Time

  限制了 WebLogic 服务器用于接收 POST 数据的时间

- ✓ 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）。如果超出了该限制，将引发 MaxPostSizeExceeded 异常

- ✓ Enable Keep Alive

  启用或禁用持久的 HTTP 连接。如果浏览器使用 HTTP 1.1 协议，那么该参数一般是开启的。

- ✓ Connection timeout

  WebLogic 服务器在关闭非活动的 HTTP 连接之前所等待的时间

- ✓ HTTPS Duration

  WebLogic 服务器在关闭非活动的 HTTPS（安全套接层或 SSL）连接之前所等待的时间。

- ✓ HTTP Access Logging

  你可以禁用或启用 HTTP 访问日志，并且可以设置访问日志的回旋参数，如日志旋回的方式及时间。详细内容，请参见本章“设置 HTTP 访问日志”中的内容。

有关设置上述参数的详细信息，请参见以下网页的“虚拟主机”部分：

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

配置监听端口

你可以指定 WebLogic 服务器监听 HTTP 请求的端口。尽管可以指定任何有效的监听端口，但是如果把 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 服务的部署遵照 Sun MicroSystems 的 Servlet2.2 标准。该标准定义 *Web 应用*作为组合各种基于 Web 应用的组件的标准。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。一个服务器可以分发任意多个 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，以下页面的 Assembling and Configuring Web Applications：

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，参见以下页面的“Using WebLogic Server Clusters”部分：

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。

如果没有声明缺省的 Web 应用，那么 WebLogic 服务器启动时时会自动创建一个缺省 Web 应用。所创建的缺省 Web 应用的名字为 *DefaultWebApp_servername*，其中 servername 是服务器的名字；如果是虚拟主机，那么将被命名为 *DefaultWebApp_VirtualHostName*。

如果没有正确分发所声明的缺省 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
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“Using WebLogic Server Clusters”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，请参见以下页面的“Deploying WebApplications as Part of an Enterprise Application”：

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，请参见以下页面的“Configuring Servlets”：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

详细内容参见以下页面的“Configuring Welcome Pages”

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）

日志的回旋可以基于文件大小也可以基于时间间隔。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）。请参见以下页面的“W3C Technical Reports and Publications”：

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。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。

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<type>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 7-18 页的“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 7-12 页的“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 7-11 页“使用扩展日志格式”。

注意：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。

注意：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，请参见以下网页的“扩展日志格式”部分

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 7-1 HttpAccountingInfo 的 Getter 方法

HttpAccountingInfo	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();	javax.servlet.http.Http.ServletRequest
HttpAccountingInfo Methods	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 7-1 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“POST 拒绝服务”攻击

拒绝服务工具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，请参见以下页面的“Writing Web Application Deployment Descriptors”：

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

7 代理对另一个 HTTP 服务器的请求

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。详细信息可以参见以下资源中的“分发与配置 Web 应用”

   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，把文件扩展名映射到 ProxyServlet，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。如果你只希望代理特定扩展名的文件，那么应该映射以下扩展：*.jsp， .html 以及*.htm。

代理 Servlet 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。

表 8-4 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>
```

8 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 示例

概述

HttpClusterServlet 负责把请求从一个 WebLogic 服务器代理到 WebLogic 集群中的服务器成员，同时 HttpClusterServlet 还为代理的 HTTP 请求提供负载平衡与失败转移。有关 servlets 与 WEbLogic 集群的更多内容，可以参见以下页面的“理解 HTTP 会话状态复制”部分：

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。
   a. 新建一个域。
   b. 在新建的域中新建一个应用
   c. 在新建的域中新建一个服务器，或使用缺省的服务器
   d. 把在步骤 b 中创建的 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，映射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。如果你只希望代理特定扩展名的文件，那么应该映射以下扩展名文件：*.jsp， .html 以及*.htm。

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>
```

9

配置 Web 应用的安全性

通过验证，限制对 Web 应用中某些资源的访问或者在 servlet 中使用安全调用，你可以保护你的应用。WebLogic 定义了几种安全域，有关安全域的讨论见以下资源的“安全基础知识”：

http://e-docs.bea.com/wls/docs60/security/concepts.html

注意，安全域为多个虚拟主机所共享。

设置对 Web 应用的授权

Web 应用的验证在 web.xml 的<longin-config>元素中定义。用该元素定义安全域。安全域包含用户证书、验证方式与需要进行验证的资源所在的位置。有关设置安全域的更多内容，请参见以下资源中的“安全基础”部分。

要设置 Web 应用的安全验证：

1. 选择一种验证方式。有以下几种选项

   BASIC

   BASIC 验证通过 Web 浏览器显示一个用户名/口令对话框。然后依照安全域对用户名及口令进行验证。

   FORM

   基于表单的验证需要一个包含用户名与口令的 HTML 表单。该表单所返回的字段必须是 j_username 与 j_password，表单的 action 属性必须设为 j_security_check，以下是一个用 FORM 授权的 HTML 示例：

```
<form method="POST" action="j_security_check">
<input type="text" name="j_username">
<input type="password" name="j_password">
  </form>
```

   产生该表单的资源可能是 HTML 页面、JSP 或者是 servlet。你必须用<form-login-page>元素定义产生该表单的资源。

   会话对象在返回登录页面时创建。因此在验证通过后，从被服务的页面中调用 session.isNew()方法，该方法将返回 FALSE。

   CLIENT-CERT

   使用客户端验证对请求进行验证。详细内容，参见以下资源的“配置 SSL 协议”部分：

http://e-docs.bea.com/wls/docs60/adminguide/cnfgsec.html#cnfgsec015

2. 如果选用 FORM 验证，那么还需要定义生成 HTML 页面的资源所在的位置与

响应验证失败的资源。有关配置表单验证的详细信息，请参见以下网页中的“<login-config>”的内容：

http://e-docs.bea.com/wls/docs60/programming/web_xml.html#login –config.

3. 定义用来验证的域。如果没有指定域，将使用缺省的 WebLogic FileRealm 域。详细内容，参见以下网页发“<login-config>”部分：

http://e-docs.bea.com/wls/docs60/programming/web_xml.html#login-config

多 Web 应用、Cookies 与身份验证

缺省情况下，所有 Web 应用都使用一样的 cookie 名字（JSESSIONID）。无论你用何种验证方式，所有使用相同 cookie 名的应用对授权都采用单点签到的方式。即一旦一个用户验证通过后，那么该验证对于其它使用同样 cookie 名字的 Web 应用的请求也是有效的，用户不会被要求重新验证。

如果要对一个应用单独授权，那么应该为这个应用指定一个不同的 cookie 名字。该名字由 weblogic.xml 的<session-descriptor>元素中的 CookieName 参数定义。详细内容，参见以下网页“session-descriptor 元素”中的内容：

http://e-docs.bea.com/wls/docs60/programming/weblogic_xml.html#session-descriptor.

限制对 Web 应用资源的访问

你可以对 Web 应用中的资源（servlets, JSPs, 或者 HTML 页面）应用安全约束。要使用安全约束，你应该：

1、定义一个角色，该角色对应于安全域里的一到多个准则（principal）。角色在 web.xml中的<security-role>元素中定义（见http://e-docs.bea.com/wls/docs60/programming/web_xml.html#security-role ）。然后在 weblolgic.xml 中，用<security-role-assignment>元素把这些角色映射到域中的准则(见 http://e-docs.bea.com/wls/docs60 /programming/weblogic_xml.html#security-role-assignment)。

2. 用<web-resource-collection>元素的<url-pattern>子元素定义 Web 应用中需要实施安全限制的资源。<url-pattern>可以是一个目录、文件名或者是一个<servlet-mapping>

要对整个应用实施安全限制，那么应该用以下<url-pattern>:

```
<url-pattern>/*</url-pattern>
```

3、使用<web-resource-collection>元素中的<http-method>子元素定义需要应用安全限制的 HTTP 方法（Get 或 Post）

4、使用<user-data-constraint>元素中的<transport-guarantee>子元素设置客户端与服务器端的通信是否使用 SSL。

列表 8-6 资源限制示例

web.xml entries:

<security-constraint>
<web-resource-collection>
<web-resource-name>SecureOrdersEast</web-resource-name>
<description>
Security constraint for resources in the orders/east directory
</description>
<url-pattern>/orders/east/*</url-pattern>
<http-method>POST</http-method>
<http-method>GET</http-method>
</web-resource-collection>
<auth-constraint>
<description>constraint for east coast sales</description>
<role-name>**eas**t</role-name>
<role-name>**manage**r</role-name>
</auth-constraint>
<user-data-constraint>
<description>SSL not required</description>
<transport-guarantee>NONE</transport-guarantee>
</user-data-constraint>
</security-constraint>
...
<security-role>
<description>east coast sales</description>
<role-name>**eas**t</role-name>
</security-role>
<security-role>
<description>managers</description>
<role-name>**manage**r</role-name>
</security-role>

weblogic.xml 中的条目:

<security-role-assignment>
<role-name>**eas**t</role-name>
<principal-name>tom</principal-name>
<principal-name>jane</principal-name>
<principal-name>javier</principal-name>
<principal-name>maria</principal-name>

</security-role-assignment>
<security-role-assignment>
<role-name> **manager** </role-name>
<principal-name>peter</principal-name>
<principal-name>georgia</principal-name>
</security-role-assignment>

在 servlet 中使用用户和角色

你可以在 servlet 中用 javax.servlet.http.HttpServletRequest.isUserInRole (String role)方法访问用户与角色。字符串 role 对应于 Web 应用分发描述符的<security-role-ref>元素的<role-name>子元素所定义的名字，<secutity-role-ref>元素位于<servlet>声明中。<role-link>元素映射到 Web 应用分发描述符的<security-role>元素所定义的<role-name>元素。

例如

列表 8-7 安全角色映射示例

Servlet 代码：

isUserInRole("**manager**");
web.xml 中的条目:
<servlet>
. . .
<role-name>**manager**</role-name>
<role-link>**mgr**</role-link>
...
</servlet>
<security-role>
<role-name>**mgb**</role-name>
</security-role>
weblogic.xml 中的条目:
<security-role-assignment>
<role-name>**mgr**</role-name>
<principal-name>al</principal-name>

<principal-name>george</principal-name>

<principal-name>ralph</principal-name>

</security-role-ref>

配置 Web 应用的外部资源

在访问外部资源时，例如通过 JNDI 从 Web 应用访问数据源，你可以让代码中使用的 JNDI 名字映射到绑定在 JNDI 树中的实际名字。这种映射是通过 web.xml 以及 Weblogic.xml 分发描述符来实现的。这样当这些资源发生变化时，就不需要更新应用程序的代码。你应该为这些资源提供一个由代码使用的名字，即资源在 JNDI 树中所绑定的名字，以及资源的 Java 类型；并且还应该指出是否应该由 servlet 以编程的方式来处理该资源的安全性还是通过 HTTP 请求加密来保证资源的安全。

要配置外部资源：

1. 在分发描述符中定义资源的名字、Java 类型以及安全验证类型。有关编写分发描述符的内容请参见以下网页中的“引用外部资源”部分：

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

2. 将资源名映射到 JNDI 名。有关编写分发描述符的内容请参见以下网页的“Map external resources ”部分:

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

下面的例子假定你定义了一个名字为 accountDataSource 的资源。详细内容，参见以下页面的“JDBC 数据源”部分：

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

表 8-8 Example of Using a DataSource

Servlet 代码：

javax.sql.DataSource ds = (javax.sql.DataSource) ctx.lookup("myDataSource");

web.xml 中的设置：

<resource-ref>

...

<res-ref-name>**myDataSourc**e</res-ref-name>

<res-type>javax.sql.DataSource</res-type>

<res-auth>CONTAINER</res-auth>

...

</resource-ref>

weblogic.xml 中的设置：

<resource-description>

<res-ref-name>**myDataSourc**e</res-ref-name>

<jndi-name>accountDataSource</jndi-name>

</resource-description>

在 Web 应用中引用 EJBs

当 Web 应用要引用 EJB 时，你应该在 web 应用分发描述符中为所引用 EJB 的定义一个名字，这个名字应该与 webloigc-ejb-jar.xml 文件中所定义的 EJB 的 JNDI 名相同，然后你就可以在程序中用所定义的名字来引用 EJB。

要在 Web 应用中使用 EJB，需要进行以下配置：

1. 在 Web 应用分发描述符的<ejb-ref>元素中定义 EJB 的引用名（程序用这个名字来查找 EJB）、Java 类名以及 home 接口与 remote 接口的类名。有关如何编写分发描述符的<ejb-ref>条目，参见以下页面的“引用 EJB 资源”部分：

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

2. 将在 weblogic.xml 的<ejb-reference-description>元素中所定义的引用名映射到 weblogic-ejb-jar.xml 文件中定义的 JNDI 名。关于如何编写分发描述符中的条目，你可以参见以下页面的“Map EJB resources”部分：

http://e-docs.bea.com/wls/docs60/programming/webappdeployment.html#ejb-reference-description.

如果 Web 应用是企业应用的组成部分，你可以用<ejb-link>元素通过名称来引用.ear 文件中的 EJB。

配置会话管理

WebLogic 服务器被缺省设置为处理会话跟踪，因此要使用会话跟踪，不需要配置任何以下将要提到的属性。但 WebLogic 服务器如何管理会话对于提升应用的性能非常关键。性能提升与以下因素有关：

- ✓ 预期会有多少个用户访问一个 servlet。
- ✓ 会有多少个并发用户访问一个 servlet。
- ✓ 每个会话持续的时间
- ✓ 为每个用户存储的数据量

HTTP 会话属性

你可以在特定于 WebLogic 的分发描述符中配置 HTTP 会话属性，配置 WebLogic 服务器的会话跟踪。详细内容，参见以下网页的“定义会话参数”部分：

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

会话属性的完整列表可在以下网页获得：

http://e-docs.bea.com/wls/docs60/programming/weblogic_xml.html#session-descriptor.

会话超时

你可以指定 HTTP 会话隔多长时间失效。在会话失效后，所有保存在会话中的数据都将丢失。以下两种方式可以配置会话失效的时间间隔。

TimeoutSecs 属性在特定于 WebLogic 的分发描述符——weblogic.xml 文件的<session-descriptor>元素中定义，属性值以秒为单位。你可以参见以下资源：

http://e-docs.bea.com/wls/ docs60/programming/ weblogic_xml.html #session-descriptor

也可以用 Web 应用分发描述符——web.xml 文件中的<session-timeout>元素定义会话超时，以分钟为单位。该属性会覆盖在 weblogic.xml 的<session-descriptor>元素中所设置的<TimeoutSecs>属性。有关在 web.xml 中定义超时属性的内容可以参见以下资源：

http://e-docs.bea.com/wls/docs60/programming/web_xml.html#web_xml_session-config）

配置会话 Cookies

如果客户的浏览器支持 cookies，那么 WebLogic 服务器将用 Cookies 来管理会话。

缺省情况下，WebLogic 服务器把跟踪会话的 cookies 设置为暂态的，即浏览器关闭时，cookie 丢失并且会话的生命期便被看作是结束了。这种行为是使用会话的精髓所在，推荐你以这种方式使用会话。

通过定义 weblogic.xml 中的有关属性，你可以配置 cookies 的方方面面。有关会话与 cookie 的属性的完整列表可以参见：

http://e-docs.bea.com/wls/docs60/programming/weblogic_xml.html#session-descriptor

有关如何编辑 weblogic.xml 的内容，可以参见以下资源的“定义会话参数”部分：

http://e-docs.bea.com/wls/docs60/programming/webappdeployment.htm l#session-descriptor.

使用长效 cookies

对于长效的客户端数据，应用程序应该通过 HTTP servlet API 创建自己的 cookies，并且尽量让这个 cookie 与 HTTP 会话存在什么关联。应用程序可能要用 cookie 来实现用户从某一机器上的自动登录功能，这种情况下就需要创建一个长效的 cookie。记住，该 cookie 只能由该用户的机器发送。如果该用户要从多台机器上访问，那么必须把数据保存在服务器端。

注意，浏览器 cookie 的生命周期与会话的生命周期没有直接的联系。如果，一个 cookie 先于它所关联的会话失效，那么该会话就成了孤儿。如果会话先于它所关联的 cookie 失效，那么 servlet 就找不到会话。此时，当 getSession()方法被调用时，会创建一个新的会话。

配置持久化会话

持久会话有以下四种实现：

- ✓ 内存（单服务器，不存在复制）
- ✓ 文件系统
- ✓ JDBC
- ✓ 内存复制（在集群中的服务器之间）

这里只介绍前面三种实现；有关内存复制的方式参见以下资源的“HTTP 会话状态的复制”部分：

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

要通过文件系统、JDBC 或者内存复制来实现持久会话，那么需要设置包括 PersistentStoreType 在内的一些属性。每种实现方式都有一组自己的属性。

通用属性

在 weblogic.xml 中，以下属性设置保存在内存中的会话数量。只有在使用持久会话时，这些属性才是有效的

CacheSize

该属性限制了缓存在内存中的活动会话的数量。如果可能会有大量并发的活动会话，而又不希望活动会话占用太多的内存使得内存页面交换频繁发生而导致性能问题，那么可以通过这个属性来控制。当缓冲区满了时，最少使用的那些会话将被保存在持久存储设备中，当需要时，会话被自动地装载到缓冲区中。如果不使用持久会话，那么该属性被忽略，并且不限制内存中的会话数量。缺省情况下，缓存会话的数量为 1024 个，最小为 16 个，最大为 Integer.MAX_Value 个。一个空会话所使用的字节数少于 100 个字节，但

随着数据的加入，它所使用的内存数量也虽之增加

SwapIntervalSecs

当内存中会话的数量达到 cacheEntries 界限时，服务器每隔多长时间把新近最少使用的会话保存到持久存储器中。

该属性缺省为 10 秒，最小为 1 秒，最大为 604800 秒（一个星期）。

InvalidationIntervalSecs

该属性设置了 WebLogic 服务器检查超时与失效会话并删除这些会话以释放内存空间的时间间隔。该属性值应该大于<session-timeout>元素中设置的值。该参数可以用来对 WebLogic 服务器的性能进行优化。

最小设置为 1 秒一次，最大设置为一星期一次（604800 秒），缺省为 60 秒。

<session-timeout>在 web.xml 的<session-config>元素中设置，详细内容见以下资源：

http://e-docs.bea.com/wls/docs60/programming/web_xml.html#web_xml_session-config

使用基于内存的、单服务器的、非复制的持久存储

如果要用基于内存的、单服务器的、非复制的方式来实现持久会话，那么应该把 PersistentStoreType 属性设置为 memory。对于这种实现方式，所有的会话数据都保存在内存中。因此当你终止并重启 WebLogic 服务器时，所有的会话数据都将丢失。

使用基于文件的持久存储

要以基于文件存储的方式实现持久会话：

1. 将 PersistentStoreType 属性设置为 file
2. 设置存储会话的目录。有关如何设置该目录的详细内容，请参见以下资源中的“PersistentStoreDir”部分：

http://e-docs.bea.com/wls/docs60/programming/weblogic_xml.html#persistentStroeDir

如果没有设置该属性，WebLogic 服务器会创建一个临时目录保存会话。

如果在集群中，持久会话采用基于文件存储的实现方式，那么必须明确设置一个能被集群中所有成员访问的共享目录，而且你必须自己创建该目录。

使用基于数据库的持久存储

要以基于 JDBC 的方式实现持久会话：

1. 将 PersistentStoreType 属性设置为 jdbc
2. 设置 PersistentStorePool 属性，该属性被设置为一个在控制台中定义的连接

池。通过设置该属性，持久会话将使用 JDBC 连接池。

所使用的连接池必须是仅被用于 JTS 驱动器的 JDBC 连接。也就是说，一个被非 JTS 驱动器所使用的 JDBC 连接池不能用于会话的持久化。为保险起见，你应该为持久会话单独创建一个 JDBC 连接池。不过，持久会话可以与 servlet 持久会话以及 EJB 共用一个 JDBC 连接池，因为后两者都要用到 JTS 驱动器

有关设置数据库连接池的详细内容请参见以下资源的“管理 JDBC 连接”部分：

http://e-docs.bea.com/wls/docs60/adminguide/jdbc.html

总结一下：尽管 servlet 可以用非-JTS 连接访问数据库，但是你不能在持久会话中使用该连接。

3. 设置连接的 ACL。只有那些有权限的用户才能使用该连接。有关设置数据库连接的详细内容，请参见以下资源的“管理数据库连接”部分：

http://e-docs.bea.com/wls/docs60/adminguide/jdbc.html.

4. 创建一个名字为 wl_servlet_sessions 的数据库表。基于 JDBC 实现的持久会话要用到这个表。数据库连接池应该具有该表的读/写权限。下表列出了 wl_servlet_sessions 表的字段及其类型。

表 8-4 wl_servlet_sessions 表

列名	类型
Wl_id	长度不超过 100 的可变字符串型。例如 Oracle 的 VARCHAR2(100)。主键必须设置为 wl_id + wl_context_path
Wl_context_path	长度不超过 100 的可变字符串型。例如 Oracle 的 VARCHAR2(100)。该列是主键的组成部分
Wl_is_new	单字符型。例如，Oracle 的 CHAR(1)
Wl_create_time	长度为 20 的数值型。例如，Oracle 的 NUMBER(20)
Wl_is_valid	单字符型。例如，Oracle 的 CHAR(1)
Wl_session_values	Large binary column。例如，Oracle 的 Long RAW 类型
Wl_access_time	长度为 20 的数值型。例如，Oracle 的 NUMBER(20)
Wl_max_inactive_interval	整数型。例如 Oracle 的 Integer 类型。该列指的是两次客户端请求的最大时间间隔（单位为秒），超过该时间间隔会话就变为无效。如果该列的值为负数，表明会话永远不过期。

如果使用 Oracle 数据库管理系统，那么可以用以下 SQL 语句创建 wl_servlet_session 表。

```
create table wl_servlet_sessions
( wl_id VARCHAR2(100) NOT NULL,
wl_context_path VARCHAR2(100) NOT NULL,
wl_is_new CHAR(1),
wl_create_time NUMBER(20),
wl_is_valid CHAR(1),
wl_session_values LONG RAW,
wl_access_time NUMBER(20),
wl_max_inactive_interval INTEGER,
PRIMARY KEY (wl_id, wl_context_path) );
```

对上述 SQL 语句作一些改动，就可以用于其它数据库管理系统上。

注意：JDBCConnectionTimeoutSecs 属性设置了 JDBC 持久会话在连接池中的连接装载数据失败前的最长等待时间。详细内容，参见以下网页：

http://e-docs.bea.com/wls/docs60/programming/weblogic_xml.html#JDBCConnectionTimeoutSecs)

使用 URL 重写

某些情况下，因为浏览器不支持 cookies，从而不能用 cookie 来跟踪会话。在这种情况下，WebLogic 服务器检测到浏览器不支持 cookie 时，它会自动起用 URL 重写来进行会话跟踪。URL 重写将会话号编写在页面的超级链接中，你的 servlet 把这个页面返回到浏览器。当用户点击页面上的链接时，WebLogic 服务器从 URL 地址中抽取出会话号并在 servlet 调用 getSession()方法时找到合适的 HttpSession。

要在 WebLogic 中启用 URL 重写，需要在特定于 WebLogic 的分发描述符——weblogic.xml 中把 <session-descriptor> 元素的 URLRewritingEnabled 属性设置为 true（该属性的缺省值为 true）。有关 URLRewritingEnabled 属性的信息，请参见以下网页中的内容：

http://e-docs.bea.com/wls/docs60/programming/weblogic_xml.html#URLRewritingEnabled

URL 重写的编码指南

以下是在使用 URL 重写时如何在代码中处理 URL 的指导性规则：

✓ 不要将 URL 直接输出到输出流中，就象下面这样：

```
out.println("<a href=\"/myshop/catalog.jsp\">catalog</a>");
```

而是应该使用 HTTPServletResponse.encodeURL()方法。例如：

```
out.println("<a href=\""+ response.encodeURL("myshop/catalog.jsp")+ "\">catalog</a>");
```

encodeURL()方法会确定是否要重写 URL，如果需要重写，该方法会在 URL 中加上会话号。会话号附加在 URL 后并且以分号开头

✓ 除了作为响应返回到 Weblogic 的 URL 外，还要对需要 redirect 的 URL 重编码。例如 ：

```
if (session.isNew())
        response.sendRedirect(response.encodeRedirectUrl(welcomeURL));
```

对于新建的会话，即使浏览器支持 cookies，WebLogic 服务器也会使用 URL 重写，因为服务器在第一次访问一个会话时，不能区分浏览器是否支持 cookies。

✓ 通过检测 HttpServletRequest.isRequestedSessionIdFromCookie() 方法返回的值，Servlet 可以确定所会话号是否是来自于 cookie。应用程序可以对此作出合适的响应，或者只是使用 WebLogic 服务器的 URL 重写。

URL 重写与无线访问协议(WAP)

如果要编写 WAP 应用，那么就必须使用 URL 重写，因为 WAP 协议不支持 cookies。而且某些 WAP 设备把 URL 的长度限制在 128 个字符（包括参数在内），从而限制了 URL 重写所能传送的数据量。要使参数能有更多的空间，你可以设置 IDLength 属性来限制 WebLogic 服务器随机产生的会话号的长度。有关该属性的详细内容，请参见以下网页中的内容

http://e-docs.bea.com/wls/docs60/programming/weblogic_xml.html#IDLength

使用字符集与 POST 数据

对于一个使用 Post 方法的表单，你可以设置处理该表单数据的字符集。通过在处理表单数据的 URL 中加上特定的“信号”字符（在 <form>标签的 action 属性中指定），应用程序就知道如何处理使用了特殊字符集的表单数据并且会将这些字符映射到 Web 应用分发描述符-web.xml 中指定的编码方式。通常 POST 数据被看作是 ASCII 字符。如果要处理非 ASCII 码的 POST 数据，应该使用以下步骤：

1. 在 Web 应用分发描述符（web.xml）的<distributable>元素与<servlet>元素之间增加<content-param>元素。在这个元素中，<param-name>中的内容是类名 weblogic.httpd.inputCharse，后跟句点号与信号字符传。<param-value>中的内容是 HTTP 字符集的名字。在下面的这个例子中，字符串/rus/jo*映射到 windows-1251 字符集。

```
<context-param>
<param-name>weblogic.httpd.inputCharset./rus/jo*</param-name>
<param-value>windows-1251</param-value>
</context-param>
```

2. 在需要发送数据的表单中使用信号字符串。例如：

```
<form action="http://some.host.com/myWebApp/rus/jo/index.html">
...
</form>
```

信号字符串应该在 Web 应用名（也被称为上下文路径，上面的例子中为 myWebApp）与 URL 的其它部分之间。

有关 Web 应用分发描述符的详细内容，请参见以下网页的“上下文参数”部分：

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

10 配置 Apache-Weblogic 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
- ✓ 安装库
- ✓ 配置 httpd.conf 文件
- ✓ Apache-WebLogic Server 插件的配置参数
- ✓ 使用 SSL
- ✓ 与 SSL-Apache 配置有关的问题
- ✓ httpd.conf 文件示例
- ✓ 分发 Apache-WebLogic Server 插件

概述

本章描述如何安装与配置 Apache-Weblogic Server 插件。该插件可以把 Weblogic 服务器的 HTTP 功能平滑地集成到 Apache 服务器中。因为可以透明地访问 Weblogic Server 的 servlet 引擎，从而增强了原有 Apache 服务器的功能。

插件用于这样一种环境中：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 请求是来自相同的源；后端的 WebLogic Server 是不可见的。因为 WebLogic 服务器的客户-服务器协议的 HTTP-隧道部件可以在插件中运行，因此插件可以访问包括 HTTP servlets 在内的所有 WebLogic Server 服务。

Apache-WebLogic Server 插件在 Aparhe 服务器中作为一个 Apache 模块运行。Apache Server 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(Java) servlets，只是 Apache 模块是与平台相关的。Apache 插件已经过测试，并且可以与 Apache Server1.3.9 一同使用。

平台支持

Linux，Solaris 与 HPUX11 等平台都支持 Apache-WebLogic Server 插件。该插件可以与 Apache Server 1.3.9 以及 Apache Server 1.3.12 一起使用。

安装库

在 Solaris、Linux 以及 HPUX11 平台上，Apache 插件是一个共享对象（.so）。这些文件位于 weblogic/lib 目录中。

在 Solaris 上，Apache 插件由 weblogic/lib/solaris 目录中的以下共享对象文件组成：

mod_wl.so

mod_wl_ssl.so

mod_wl_ssl_raven.so

在 Linux 上，Apache 插件由 weblogic/lib/linux 目录中的以下共享对象文件组成：

mod_wl.so

mod_wl_ssl.so

mod_wl_ssl_raven.so

在 HPUX11 上，Apache 插件由 weblogic/lib/hpux11 目录中的以下共享对象文件组成：

mod_wl.so

mod_wl_ssl.so

mod_wl.so 是标准 Apache(没有 EAPI)共享对象。Mod_wl_ssl.so 用于 Apache+SSL/EAPI 安装（Stronghole, modssl 等）。Mod_wl_ssl_raven.so 用在 Apache+Raven 安装。此时，需要 mod_wl_ssl_raven.so，因为 Raven 使用了 frontpage 的补丁，使之与 mod_wl_ssl.so 不兼容。

上述二进制文件通过了 C2Net 的 Stronghold/3.0 (Apache/1.3.12)以及 Covalen 的 Raven/1.4.3 (Apache/1.3.12)认证。

安装后的 Apache 插件是一个 Apache DSO（动态共享对象）。Apache 对 DSO 的支持依赖于一个名为 mod_so.c 的模块，你必须在装载 mod_wl.so 之前先启用该模块。如果你用所提供的脚本在 Solaris 上安装 Apache，那么 mod_so.c 已经启用。要确定是否启用了 mod_so.c，运行以下命令：

APACHE_HOME/conf/http –l

该命令列出所有已启用的模块，如果其中包含了 mod_so.c，那么安照本节末列出的六个步骤来完成 Apache 插件的安装。如果该命令没有列出 mod-so.c，那么请按照相关说明用以下参数手工配置 Apache：

./configure –prefix=<destination directory>

--enable-module=so

--enable-rule=SHARED_CORE

然后，根据 Apache 安装指南安装该产品。

安装完 Apache 并启用了 mod_so.c 后，你可以用一个名为 apxs（Apache eXtenSion）的支持程序来安装 Apache 插件，该程序会在 Apache 源文件树之外构建 DSO，并在 httpd.conf 文件中加入这样一个条目：addModule mod_so.c。以下是安装 Apache_Weblogic Server 插件的步骤：

1. 进入 weblogic/lib/PLATFORM 目录，用以下命令激活 weblogic_module：

```
APACHE_HOME/bin/apxs –i –a –n weblogic mod_wl.so
```

注意这是一个 perl 脚本，因此在运行该脚本前，先要安装 Perl：

```
perl APACHE_HOME/bin/apxs –i –a –n weblogic mod_wl.so
```

2. 该命令将把 mod-wl.so 文件复制到 APACHE_HOME/libexec 目录，并在 httpd.conf 文件的 weblogic_module 块添加了两条语句，同时激活了该模块。验证 APACHE_HOME/conf/httpd.conf 文件是否加入了以下行：

```
LoadModule weblogic_module
AddModule mod_weblogic.c
```

3. 用以下命令验证 APACHE_HOME/conf/httpd.conf 文件的语法是否正确：

```
APACHE——HOME/bin/apachect1 start
```

4. 配置 Apache 的 httpd.conf 配置文件（下节将介绍该文件）
5. 启动 Weblogic 服务器
6. 启动 Apache

配置 httpd.conf 文件

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。该配置文件位于 APACHE_HOME/conf/目录中（APACHE_HOMES 是安装 Apache 的根目录）。

在 httpd.conf 文件中添加设置 WebLogic Apache 插件的以下信息：

- ✓ weblogic_module 的配置信息（是由 apxs 工具所添加的）：

```
LoadModule weblogic_module      libexec/mod_wl.so
AddModule mod_weblogic.c（该行不总是被加上）
```

- ✓ 确定由 Apache 插件交给 WebLogic 处理的 HTTP 请求类型。你可以通过 URL 或 MIME 文件扩展类型来转交该类型的请求。下面将讨论这些选项。

通过 URL 代理

如果用 URL 的形式转交 HTTP 请求，那么要使用位置块与 SetHandler。SetHandler 指出了 Weblogic 模块的处理器，如下所示：

```
<location URL>
setHandler weblogic_handler
</Location>
```

例如：

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

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

通过这种方式，某些形式的 URL 被传递到 WebLogic Server 中进行语义分析。因为 PathTrim 设置为从 URL 中剥离/weblogic，因此转发到 WebLogic 服务器的 URL 为：

http://myenterprise.server.com:7001

注意，Apache-WebLogic 服务器插件不能将请求转发到 multiple WebLogic Cluseter。但你可以通过多虚拟主机来实现相同的功能。与位置块中所指定的任何 URL 匹配的请求将被转发到同一集群中。

通过 MIME 文件类型代理

如果以 MIMI 文件扩展类型的方式转发 HTTP 请求，那么应该在 IfModule 块中加上 MatchExpression 语句。例如：

```
<IfModule>
MatchExpression *.jsp
...
</IfModule>
```

在 MatchExpression 语句可以包含多个表达式，表达式之间用逗号隔开。例如：

```
example:
<IfModule>
MatchExpression *.jsp,*.xyz
...
</IfModule>
```

Apache-WebLogic Server 插件的参数

Apache 的 Weblogic 模块识别以下参数。要改变 Apache-WebLogic Server 插件的属性，你应该把这些参数放在在 IfModule 中或包含在 IfModule 中的 Weblogic.conf 文件。

```
<IfModule mod_weblogic.c>
# Config file for Weblogic which defines the parameters
Include conf/weblogic.conf
</IfModule>
```

或者

```
<IfModule mod_weblogic.c>
# define your parameters here.
</IfModule>
```

注意一行只能包含一个参数，在参数与值列表之间不能使用‘=’。例如：

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

WebLogicHost *domain name*

必须使用该参数（如果是 WebLogic 集群，则应该用 WebLogicCluster 参数）。HTTP 请求将递交到该 WebLogic Server 主机。

WebLogicPort *port*

必须使用该参数（如果是 WebLogic 集群，则应该使用 WebLogicCluster 参数）。WebLogic Server 主机将在该端口监听 WebLogic 连接请求。

WebLogicCluster *cluster list*

必须使用该参数（如果你正在使用一个 WebLogic 集群），该参数列出了在集群中进行负载平衡的 WebLogic 服务器。集群列表是由冒号分割的 host:port 条目的一个列表。例如，

WebLogicCluster myweblogic.com:7001,

yourweblogic.com:6999,theirweblogic.com:6001

应该用该参数取代 WebLogicHost 与 WebLogicPort 参数。WebLogic Server 将首先查找 WebLogicCluster 参数，如果没有找到该参数，它将寻找并使用 WebLogicHost 与 WebLogicPort 参数。

Aparche 插件在所有可利用的集群成员中进行简单的负载平衡计算，然后将包含 cookie 的 HTTP 请求递交到集群中创建该 cookie 的服务器。

PathTrim *string*

要裁减掉的字符串，缺省为“weblogic”。

PathPrepend *string*

在去除了 PathTrim 字符串后并在请求被递交到 WebLogic 之前，将该字符串加到原始 URL 的前面。缺省为“”。

ConnectTimeoutSecs *seconds*

Weblogic 模块进行 WebLogic Server 主机连接尝试的最大时间间隔，缺省为 10 秒，应该大于 ConnectRetrysecs。如果超过了 ConnectTimeoutSecs 所指定的时间还未能连接上，那么将把 HTTP 501/Service Unavailable 回复发送到客户端。

ConnectRetrySecs *seconds*

该参数确定了在进行连接 WebLogic Server 主机尝试之间模块应该休眠的时间。缺省为 2 秒，该参数的值应该小于 connectTimeoutSecs 参数的值。在将 HTTP503/Service Unavailable 回复返回到客户端之前，Apache 插件每 ConnectRetruSecs 会进行一次连接尝试。如果 ConnectRetrySecs 参数的值等于 ConnectTimeoutSecs 参数的值，那么只会进行一次连接尝试。

StatPath *boolean*

如果该参数设置为真，那么 Apache-Weblogic Server 插件会在把请求传递到 WebLogic 服务器之前检查翻译路径是否存在及其可读权限。

缺省值为 false。如果文件不存在，那么把 HTTP 404/Not found 回复返回到客户端。如果存在该文件，但是它的读权限不是设置为 world-readable，那么将返回 HTTP 403/Forbidden 回复。在这两种情况下，Apache 服务器处理这些回复的缺省机制是完成回复中提的内容。如果 WebLogic Server servlet 引擎以及 Apache Server 具有相同的文档根的话，将使用这个选项。

ErrorPage *URL*

你也可以创建自己的错误页面，当 Apache 服务器无法将请求递交给 WebLogic 服务器时，将显示这个页面。把 URL 设置为错误页面所在的位置。

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 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 truested principal。

WebLogic 服务器代理插件不会用传输协议（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。要使用 SSL 协议，必须把使用代理插件的 WebLogic 服务器配置为使用 SSL 协议。WebLogic 服务器插件代理使用 WebLogic 服务器监听 SSL 通信的端口与 Apache 服务器通信。

在 WebLogic 服务器代理插件使用 SSL 协议，必须在 httpd.conf 文件中指定以下参数：

SecureProxy ON/OFF

如果 WebLogic 服务器代理插件与 Apache 之间的通信采用 SSL 协议，那么这个参数应该设置为 ON，该参数缺省为 OFF。在定义这个参数之前，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：在主服务器配置或在虚拟主机配置中定义。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=*filename*

数字证书所在的文件，该数字证书由可靠的证书管理机构发放并被 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。该参数没有缺省值。

文件名必须是数字证书文件的全路径文件名。

RequireSSLHostMatch TRUE/FALSE

该参数决定 WebLogic 服务器代理插件连接的主机名是否必须与连接代理插件的 WebLogic 服务器所使用的数字证书中的 Subject Distinguished Name 字段匹配。参数缺省为 TRUE。

SSLHostMatchOID *Integer*

This parameter is the ASN.1 Object ID (OID) that identifies which field in theSubject Distinguished Name of the peer digital certificate is to be used toperform the host match comparison.缺省为 22，对应于 Subject Distinguished Name 中的 CommonName 字段。Common OID 有以下值：Sur Name—23，Common Name—22，Email—13，Organizational Unit—30，Organization—29，and Locality—26。

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>
```

✓ 指令对 apache SSL 不起作用。避免在 apache SSL 中使用以下设置：

```
<IfModule mod_weblogic.c>
MatchExpression *.jsp
Include weblogic.conf
</IfModule>
```

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 Weblogic 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 集群的例子

```
# 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>
#######################################################
```

不使用 WebLogic 集群的例子

```
# 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>
```

分发 Apache-WebLogic Server 插件

在你安装并配置了 Apache-WebLogic Server 插件后，按照以步骤进行分发与测试：

1. 启动 WebLogic 服务器
2. 如果 Apache 服务器已经运行，为了使新配置生效，必须重启 Apache 服务器
3. 测试 Apache 插件：打开浏览器，并将 URL 设置为 Apache 服务器+ “/weblogic”，这将使用缺省的 WebLogic HTTP servlet，如这个例子：

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

11 配置 Microsoft-IIS 插件

本章将介绍如何在 Microsoft Internet Information Server 中安装与配置 WebLogic Server 插件。通过插件，可以把 WebLogic 服务器的 HTTP 功能平滑地集成到 Microsoft IIS 中。通过 WebLogic-to-IIS 插件，客户端可以透明地访问 WebLogic 中的资源，从而增强了 IIS 的功能。

主要介绍以下内容：

- ✓ 概述
- ✓ 安装库
- ✓ 更新 IIS 设置使代理可以代理到 WebLogic
- ✓ 创建 iisproxy.ini 文件
- ✓ 使用 SSL 协议
- ✓ 将 servlets 从 IIS 代理到 WebLogic Server
- ✓ 安装测试

概述

如果要让 IIS 处理静态页面而把动态页面如 HTTP Servlets 与 JSP 交给 WebLogic 服务器处理，那么就需要用到 IIS 插件。WebLogic 服务器可能运行在另一个进程，也可能是在另一主机上。但对于终端即浏览器用户而言，转交给 WebLogic 处理的 HTTP 请求与由 IIS 处理的请求都来自相同的服务器。运行在后端的 WebLogic 服务器是不可见的。WebLogic 客户-服务器协议的 HTTP 隧道功能可以在插件中使用，因而也就提供了对所有 WebLogic 服务的访问，而不仅仅是 HTTP servlets

连接池以及保持活动状态

通过把插件和 WebLogic 服务器的连接放在一个可重用的连接池中，可以使得 WebLogic Server ISAPI 插件具有极高的性能。ISAPI 插件会自动保持插件与 WebLogic 服务器连接的活动状态。如果一个连接处于非活动状态的时间超过 30 秒，那么该连接将自动关闭。

有关兼容 WebLogicISAPI 插件的平台的最新信息，请见以下网站中的“平台支持”页面：

http://www.weblogic.com/platforms/index.html

安装库

在 NT 上，WebLogic ISAPI 是作为一个动态链接库“iisproxy.dll”来分发的。该文件位于 weblogic/bin 目录下。

要安装 ISAPI 插件：

1. 启动 WebLogic 服务器

2. 启动 IIS

3.按照下面的说明修改 IIS 设置

4. 创建初始化文件

5. 测试安装

更新 IIS 设置使请求转给 WebLogic

要使用 IIS 能与 webLogic 服务器 ISAPI 一同工作，你需要做以下配置：

1. 将 iisproxy.dll 文件复制到一个可以被 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。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。
6. 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。

   在对话框中，找到“iisproxy.dll”文件，缺省情况下，该文件位于 WebLogic 服务器产品的 weblogic/bin 目录下
7. 设置要委托给 WebLogic 服务器处理的文件类型的扩展名。

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

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

   c. 设定”Method exclusions”符合安装的安全惯例.

   e. 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。

注意与 NSAPI 插件不同的是，NSAPI 可以基于目录结构提供文件服务，而

ISAPI 插件可以基于文件类型（文件扩展名）或者路径来提供文件服务。通常的配置是使 IIS 通过 JSP 扩展名来提供 JSP 页面服务。详细内容，你可以参见 10-8 页的“通过路径代理页面请求”

在配置完所有要代理的文件类型后，点 OK 按钮关闭 Properties 窗格

创建 iisProxy.ini 文件

必须为 iisproxy.dll 创建一个.ini 文件。WebLogic 服务器会按照以下顺序来寻找 iisproxy.ini 文件。

1. iisproxy.dll 所在的目录
2. 被 Windows 注册表所引用的 WebLogic 服务器最新版本所在的目录。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。
3. C:\weblogic

我们建议你把该初始化文件放在与 iisproxy.dll 相同的目录中。

初始化文件包含了以下变量的设置，其形式为:名字=值

WebLogicHost=域名或者虚拟主机名

HTTP 请求被转交到的 WebLogic 主机（或者是运行在 WebLogic 服务器中的 Web 服务器中所定义的虚拟主机名）。例如:WebLogicHost=localhost。

WebLogicPort=端口号

WebLogic 主机监听 WebLogic 连接请求的端口。例如：WebLogicPort=7001。

WebLogiCluster=集群列表

一个集群中的所有 WebLogic 服务器列表。该列表是一组由逗号分隔开的 host:port 条目。例如：myweblogic.com:7001,yourweblogic.com:7001。

插件可以在集群的所有成员之间通过轮询算法实现简单的负载平衡。该列表是服务器以及插件维护动态集群列表的起始点。WebLogic 服务器同插件一起根据新加入的、失败的与恢复了的集群成员来自动地更新集群列表。

此外，插件会把包含 cookie 的 HTTP 请求引导到最初创建该 cookie 的集群成员。

ConnectTimeoutSecs=秒数

插件对 WebLogic 服务器主机进行连接尝试的最大时间间隔。缺省为 10 秒。该属性值应该大于 ConnectRetrySecs 属性值。如果超过了 ConnecTimeoutSecs 属性所指定的时间，即使进行了适当次数连接尝试，还未能成功（见 ConnectRetrySecs），那么将返回 HTTP 503/Service

Unavaiable 给客户端。例如，ConnectTimeSecs=20。

ConnectRetrySecs=秒数

在两次连接尝试之间的时间间隔。缺省为 2 秒，该参数值应该不大于 ConnectTimeoutSecs 参数值。在将 HTTO 503/Service UnavailableSecs 返回给客户端之前，插件会做 ConnectTimeoutSecs/ConnectRetrySecs 次连接尝试。如果不想让插件重试，那么把 ConnectRetrySecs 设为与 ConnectTimeoutSecs 相等。例如，ConnectRetrySecs=5。

ErrorPage=url

你可以创建一个错误页面，当 IIS 不能把请求转交给 WebLogic 服务器时，它就用这个页面来响应请求。将 url 设为错误页面所在的位置。

CookieName=cname

如果你在 WebLogic 服务器中改变了会话 cookie 的名字，相应地要在 iisproxy.ini 文件中加上这个属性。WebLogic 会话 cookie 在特定于 WebLogic 的分发描述符文件，weblogic.xml 的 <session-descriptor> 元素中用 CookieName 属性定义（参见 http://e-docs.bea.com/wls/docs60/programming/weblogic_xml.html#session-descriptor）

Debug ON/OFF/HFC/HTW/HFW/HTC/ALL

该属性设置了进行调试时的日志记录类型，不建议在生产系统中使用这些调试选项。

调试信息保存在 c:/temp/wlproxy.log 文件中

该参数的设置有以下选项（HFC, HTW, HFW 以及 HTC 选项可以联合使用，选项之间用逗号隔开，例如“HFC, HTW”）：

ON

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

记录发送到 WebLogic 服务器消息的消息头，报告性消息以及错误消息

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 服务器发送的消息的消息头，报告性消息，错误消息。

缺省值为“OFF”

DebugConfigInfo=ON/OFF

该参数的设置决定是否启用特殊查询参数“__WebLogicBridgeConfig”。用这个查询参数获得插件的配置参数。

例如，如果把 DebugConfigInfo 设置为 ON，那么“__WebLogicBridgeConfig”查询参数被起用。发送一个如下的请求：

http://www.mywebserver.com:8000/weblogic/foo.jsp?__WebLogicBridgeConfig

插件会把配置信息返回到浏览器。注意这种情况下，不能连接到 WebLogic 服务器。

该参数只用在调试过程中，消息的输出格式随版本的变化而不同。处于安全性的考虑，建议你在生产系统中把该参数设置为 OFF

DefaultFileName=filename

如果 URI 是“/”，那么缺省的文件名被附加在 URL 后面，这样可以防止 WebLogic 服务器的重定向。DefaultFileName 应该设置为 WebLogic 服务器的缺省欢迎也面。如果 DefaultFileName 设置为 welcom.html，那么下面的这个 HTTP 请求：“http://somehost/weblogic” 变为：

http://somehost/weblogic/welcome.html” 。只有当所有被重定向到的 Web 应用指定相同的欢迎页面，该参数才起作用。例如，把欢迎页面都设置为 welcome.html

HungServerRecoverSecs =seconds

该参数定义了插件等待 WebLogic 服务器响应的时间。如果过了 HungServerRecoverSecs 时间，WebLogic 服务器没有响应请求，插件就认为该服务器死了，并失败转移(fail over)到另一台服务器。该参数的最小设置为 10，最大设置为 600，缺省为 300。

Idempotent =ON/OFF

这是一个 URL 级别的标志。缺省值为”ON”，这意味着如果服务器没有在 HungServerRecoverSecs 时间内响应，那么插件会进行失败转移。如果设置为”OFF”，插件不会进行失败转移。根据不同的 URL 或 Mime 类性，该参数可以有不同的设置。该参数应该设为一个比较大的值。如果是设置的值小于 servlet 的处理时间，就会得到意想不到的结果。

PathPrepend=string

在 PathTrim 被裁减(trim)之后，请求转交给 WebLogic 之前，加在原始 URL 之前的字符串。该参数的缺省值为””。

WlForwardPath=string

如果 wlForwardPath 设置为空，那么 iis 把所有请求交给 iis.proxy 处理。如果只转交某一特定字符串开头的请求，那么应该设置 wlForwardPath 参数。例如，把 wlForwardPath 设置为/weblogic，那么所有以/weblogic 开头的请求都被转交给 WebLogic 服务器。该参数的缺省值为””。如果使用路径方式的请求代理，那么必须设置该参数。

MaxPostSize

如果内容的长度超过 MaxPostSize，插件会返回一个错误消息。如果设置为 -1，不会检查那么 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超载的拒绝服务攻击。该参数的缺省值为-1。

文件扩展名方式的请求代理

ISAPI 模块能够把那些在 Internet Service Manager 注册的代理文件类型的文件转交给 WebLogic 服务器处理。例如，你注册了".jsp"类型代理，那么所有扩展名为".jsp"的请求都将转交给 WebLogic 处理。

位于 URL 的服务器与端口部分后面的路径信息会直接传递给 WebLogic。例如从 IIS 请求以下 URL："http://myiis.com:80/jspfiles/myfile.jsp"，转交给 WebLogic 的 URL 变为：

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

路径方式的请求代理

如果以路径的方式代理请求：

1. 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 *"Properties*" -"ISAPI Filters" tab -> "Add" the iisforward.dll）。
2. 把.wlforward 注册为由 iisproxy.dll 处理的一种特殊文件类型。
3. 在 iisproxy.ini 文件中添加 " wlForwardPath " 属性。例如：WlForwardPath=/weblogic。
4. 如果需要，在 iisproxy.ini 文件加入 PathTrim 参数。例如使用 WlForwardPath=/weblogic, PathTrim=/weblogic 对要转发到 WebLogic 服务器的请求 URL 进行裁减，因此/weblogic/session 转变为/session。
5. 如果用户希望直接用 http://someroot.com/ 把请求转交给 WebLogic 服务器处理，那么还应该在 iisproxy.ini 文件中设置 DefaultFileName 参数。
6. 如果要改变参数设置，特别是 DefaultFileName, Debug, PathTrim, PathPrepend 以及 WlForwardPath 参数，那么应该启动"IIS 管理服务"（在控制窗格的 services 下）。
7. 如果在 iisproxy.ini 文件中 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。

.ini 文件示例

这是一个不使用 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 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
```

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 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靠委托人(truested principal)。

WebLogic 服务器代理插件不会用传输协议（http 或 https）来确定代理插件与 IIS 插件之间的连接是否使用了 SSL 协议保护。要使用 SSL 协议，必须把使用代理插件的 WebLogic 服务器配置为使用 SSL 协议。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。

在 WebLogic 服务器代理插件使用 SSL 协议，必须在 iisproxy.ini 文件中指定以下参数：

SecureProxy ON/OFF

如果 WebLogic 服务器代理插件与 IIS 之间的通信采用 SSL 协议，那么这个参数应该设置为 ON，该参数缺省为 OFF。在定义这个参数之前，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：在主服务器配置或在虚拟主机配置中定义。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=*filename*

数字证书所在的文件，该数字证书由可靠的证书管理机构发放并被 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。该参数没有缺省值。

文件名必须是数字证书文件的全路径文件名。

RequireSSLHostMatch TRUE/FALSE

该参数决定 WebLogic 服务器代理插件连接的主机名是否必须与连接代理插件的 WebLogic 服务器所使用的数字证书中的 Subject Distinguished

Name 字段匹配。参数缺省为 TRUE。

SSLHostMatchOID *Integer*

This parameter is the ASN.1 Object ID (OID) that identifies which field in theSubject Distinguished Name of the peer digital certificate is to be used toperform the host match comparison.缺省为 22，对应于 Subject Distinguished Name 中的 CommonName 字段。Common OID 有以下值：Sur Name—23，Common Name—22，Email—13，Organizational Unit—30，Organization—29，and Locality—26。

将 servlets 请求转交给 WebLogic 服务器处理

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:\\weblogic:7001\weblogic\snoop

如果没有注册 iisforward.dll，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）。详细内容，可以参见“更新 IIS 设置使请求转给 WebLogic”。
2. 在 Web 应用中注册 servlet。有关注册 servlets 的详细内容，可以参见“配置 Servlets”中的内容。
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。

注意：

- ✓ servlet 调用的所有图形连接也必须在 IIS 中注册相应类型(例如:.gif 和.jpg)以能够代理。
- ✓ 如果被调用的 servlet 调用了其它 servlets，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 ISAPI 插件后，按以下步骤分发并测试 ISAPI 插件：

1. 运行 WebLogic 与 IIS

2. 在 Web 应用的文档根下保存一个用来测试 ISAPI 插件的 JSP 文件。打开浏览器，URL 设置为 IIS+filename.jsp。例如：

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

12 配置 Netscape 插件

本章将介绍如何配置 Netscape Enterprise Server(NES)代理插件。通过插件，可以把 WebLogic 的 HTTP 功能平滑地集成到 NES 中，客户端可以透明地访问 webLogic 中的资源，从而增强了 NES 的功能。

主要介绍以下内容：

- ✓ 概述
- ✓ 配置插件
- ✓ 使用 SSL
- ✓ 使用 NASAPI 时，有关 WebLogic 服务器集群失败转移的注意事项
- ✓ obj.con 文件示例（不使用 WebLogic 集群）
- ✓ obj.con 文件示例（使用 WebLogic 集群）

概述

插件用于这样的环境之中：NES 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。也就是说，后端的 WebLogic 是不可见的。而且 WebLogic 客户-服务器协议的 HTTP-隧道功能可以作为插件运行，因此，通过它可以访问所有 WebLogic 服务（而不仅仅是动态网页）。

WebLogic 插件在 Enterprise 服务器中被当作一个 NSAPI 模块运行。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 模块是与平台有关的。

连接池和保持激活

通过使用插件到 WebLogic 服务器的可重用连接池，WebLogic Server NSAPI 插件提供了高效率的性能。NSAPI 插件自动实现了在插件和 WebLogic 服务器之间“保持激活”的连接。如果连接超过 30 秒不活动，连接将被关闭。

插件的配置

配置 NSAPI 插件有以下四个步骤：

1. 把相关的 NSAPI 库文件复制到 NES 的安装目录中。
2. 修改 obj.conf 文件
3. 更改 NES 配置目录中的 mimes.types 文件
4. 分发并测试 NSAPI 插件

步骤 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。这些文件分别位于 WebLogic 分发的 /lib 于 /bin 目录中。从平台支持表（http://e-docs.bea.com/wls/docs60/../platforms/index.html#plugin）中选择适合于您的环境的库文件。然后将所选择的库文件复制到 NES 所在的文件系统中。

步骤 2：设置 obj.conf 文件

为了使用 NSAPI 插件，需要修改 NES 的 obj.conf 文件。包括以下改动：

- ✓ 指示 NES 将本地库装载为 NSAPI 模块的代码
- ✓ 应该交由 NSAPI 插件处理的请求类型的对象定义
- ✓ 允许通过 NSAPI 插件使用 WebLogic 协议的 HTTP-协议隧道功能的可选代码。

以下是配置 obj.conf 文件的步骤

a. **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 goldengate 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-goldengate/config/obj.conf
```

b. 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"

这里的 SHARD_LIBRARY 是你在第一步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。

c. 声明交由 NSAPI 插件处理的请求

所有经过 NSAPI 插件传递给 WebLogic 服务器的请求都必须在位于 obj.conf 文件中的对象定义中声明。有两种方式识别这些请求：通过 MIME 文件扩展，或者通过请求的 URL 中的特定字符串（被称为 ppath）

无论是使用 MIME 方式还是 ppath 方式来代理请求，你都应该在<Object>标签中以及在开启与闭合<Object>标签中添加参数。NES 参数（位于<与>）包括名字或者 ppath（可选的）。在标记之间，Service 语句设置模块所识别的 WebLogic 参数：WebLogicHost（必须的），WebLogicPort（必须的），PathTrim，PathPrepend，ConnectTimeoutSecs，ConnectRetrySecs 以及 StatPath。如果你没有设置必需的参数，那么在调用时，对象会产生 HTML 错误指出配置中缺少参数。

Netscape 的 obj.conf 文件对于问被的位置有严格要求。要避免出现问题，请务必遵照以下指导：

去除无关的前导与尾部空格

如果所输入的字符必须跨越多行，那么在行的末尾加上后斜杠“\”，然后继续在下一行输入。后斜杠会直接把前后两行连接在一起。如果要在第一行的最后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（在后斜杠之前）或者是在第二行的开始。

属性不能跨越多行。（例如，你必须在 webLogicCluster 后面列出一个集群中的所有服务器）

d. 配置 ppath 代理方式

要通过ppath来代理请求，输入<Object name= “insert_name”>开始一个新对象的定义。按照所代理的服务命名该对象，并在<Object>标签中设置ppath。Ppath的一个字符串，用它来标识要递交给weblogic服务器的请求。当你使用了ppath，所有包含该路径的请求都将转交给WebLogic服务器来处理。例如，一个 “/weblogic*” 的 ppath 将会把所有以 http://enterprise.com/weblogic 开始的请求转交给NSAPI插件，然后该插件会把请求传送给指定的Weblogic主机/端口或集群。

每一个对象定义（位于标签之间的代码）都是以 Service fn=wl-proxy 开头。对于不是位于集群中的 WebLogic 服务器，你必须设置这两个参数：WebLogicHost 与 WebLogicPort。对于位于集群中的 webLogic 服务器，只需要设置一个参数：WebLogicCluste。

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。

服务器实例处理的

<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>
```

注意，不是必须配置的参数，如 PathTrim，可以用来进一步配置经过 NSAPI 插件传送 ppath 的方式。本文档在“参数”这一节中定义了所有 WebLogic NSAPI 插件的参数。

e. 配置文件扩展名方式的代理

无论请求的 URL 是怎样，总之所有包含指定 MIME 文件类型扩展（如.jsp 或.jhtml）的请求都会被挑出来转交给 NSAPI 插件。为了把某种文件类型的请求都转送到 WebLogic 服务器中，必须在已原来“缺省的”对象定义中添加 Service 语句。

为了把所有 JSP 代理到 WebLgic 服务器，必须在最后一个以 NameTrans fn=…开头的行后面以及以 PathCheck 开始的行之前加入以下 Service 语句。

```
Service method="(GET|HEAD|POST|PUT)" type=text/jsp
fn=wl-proxy\
WebLogicHost=192.1.1.4 WebLogicPort=7001
PathPrepend=/jspfiles
```

该语句将把所有以.jsp 作为扩展名的文件代理到所指定的 WebLogic 服务器，因此请求的 URL 类似于：

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

pathPrepend 的值应该对应于 WebLogic 的文档根(document root)，JSPServlet 在文档根中搜索 JSP 文件并对文件进行编译。（有关 JSP 以自动页面编译的更多信息，请阅读开发者指南中的使用 WebLogic JSP 中的内容）

在增加了上述语句后，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>
```

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，在缺省对象中添加对应的 Service 语句。

f. 启用 HTTP 隧道功能

如果你想通过 NSAPI 插件使用 WebLogic 协议的 HTTP 隧道功能，你应该在 obj.conf 文件中添加以下对象定义，替代希望使用 WebLogic 主机和端口，或者 WebLogic 集群。

```
<Object name="tunnel" ppath="*/HTTPClnt*">
Service       fn=wl-proxy       WebLogicHost=192.192.1.4
WebLogicPort=7001
</Object>
```

步骤 3：更改 MIME.types 文件

如果按照以上方式把 NSAPI 插件设置为通过 MIME 类型代理，必须在 NES 配置目录中添加要代理的 MIME 类型的标识符及其扩展，这可以通过两种方式来实现：通过管理终端修改，或者直接编辑 MIME.types 文件。

要直接编辑 MIME.types 文件，打开要编辑的文件并输入以下行：

```
type=text/jsp exts=jsp
```

注意，对于 NES 4.0（iPlanet），不要加入用于 JSP 的 MIME 类型，而是应该将原来的 MIME 类型/magnus-internal/jsp 改为 text/jsp。

对于所有 NES 版本，你需要手工编辑 MIME.types 文件，或者使用管理终端来编辑。点击 Manage->Preferences->Mime Types 打开管理终端，然后编辑 Mime Types 文件

步骤 4：分发与测试 NSAPI 插件

在安装并配置了 NSAPI 插件后，按以下步骤来进行分发与测试：

a. 启动 WebLogic

b. 启动 NetScapte Enteprise Server。如果已经运行了 NES，为了使新的设置生效，必须重新启动它，或者从管理终端应用新的设置。

c. 要测试 NSAPI 插件，打开浏览器并把 URL 设置为 Enterprise 服务器+“/weblogic”，这将会启用缺省的 WebLogc HTTP servlet，如下所示：

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

参数

NSAPI 可以识别以下参数：

注意在配置 WebLogic 集群时：必须使用 WebLogicCluster 参数而不是 WebLogicHost 与 WebLogicPort 参数。

WebLogicHost=*domain name*

必需的参数（如果使用 WebLogic 集群，则应该使用 WebLogicCluster 参数。）。HTTP 请求将被转交到该参数定义的 WebLogic 主机中。

WebLogicPort=*port* （相当于早期的 TengahPort）

必需的参数（如果使用 WebLogic 集群，则应该使用 WebLogicCluster 参数。）。WebLogic 主机将在该参数所设置的端口监听 WebLogic 连接请求。

WebLogicCluster=*cluster list*

必需的参数(如果是使用 WebLogic 集群)。该参数列出了集群中的 WebLogic 服务器。列表中的每个条目必需遵照 host:port 的格式。例如：

```
WebLogicCluster="myweblogic.com:7001,
yourweblogic.com:6999,theirweblogic.com:6001"
```

如果使用 WebLogic 集群，那么应该用这个参数，而不是 WebLogicHost 与 WebLogicPort 参数。WebLogic 服务器会首先查找 WebLogicCluster 参数，如果没有找到该参数，它将继续查找并使用 WebLogicHost 与 WebLogicPort 参数。（如果所有这些参数都找不到，它将寻找并使用 TengahHost 与 TengahPort 参数）

插件会在可使用的集群成员中进行简单的负载计算，如果一个成员不能被连接，那么该成员将被标记为“坏的”，然后在试着用其它的成员，在此成员被跳过 10 次之后，会再次尝试使用该成员，如果成功，那么将改变它的状态。

插件会把包含 cookie 的 HTTP 请求递交到集群中创建该 cookie 的服务器。

PathTrim=*string*

在把请求转交给 WebLogic 之前，将会从原始的 WebLogicURL 中裁减掉该参数所设置的字符串。缺省为“”

path 可能会作为 webLogic URL 的一部分传递到 WebLogic 中，也可能会被剥离，这取决于 Object 标签中 PathTrim 的设置。例如，一个典型的将请求代理到 WebLogic 的 Object 标签看上去应该如下面所示：

```
<Object name="weblogic" ppath="*/weblogic/*">
Service fn=wl-proxy WebLogicHost=myweblogic.server.com\
WebLogicPort=7001\
PathTrim="/weblogic"
</Object>
```

ppath 设置了将 URL 递交到 Enterprise 服务器的方式，例如：

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

将被转交给 WebLogic 服务器进行语义分析。因为 PathTrim 被设置为从 URL 中剥离“/weblogic”，这一步是在 URL 被交给 WebLogic 服务器之前进行，因而转交到 WebLogic 的 URL 为

```
http://myweblogic.server.com:7001/
```

PathPrepend=*string*

该参数设置了要添加在原始 URL 中的前缀字符串。添加字符串在裁减 PathTrim 字符串之后、把请求转交给 WebLogic 之前进行发生。缺省为“”

ConnectTimeoutSecs=*seconds*

NSAPI 模块尝试连接 WebLogic 主机的最大时间间隔（以秒为单位）。缺省为 10 秒，该参数的值应该大于 ConnectRetrySecs 参数。如果经过 ConnectTimeoutSecs 所设定的时间还未能成功连接，那么将返回一个 HTTP 503/Service Unavailable 回复。

ConnectRetrySecs=*seconds*

该参数定义了 NSAPI 模块在两次连接尝试之间的间隔时间。缺省为 2 秒，这个参数的值应该小于 ConectTimeoutSecs 参数。在把 HTTP 503/Service Unavailable 回复返回到客户端之前，NES 插件会尝试 ConnectTimeoutSects/ConnectRetrySecs 次连接。如果只想进行一次连接尝试，那么把 ConnectRetrySecs 参数与 ConnectTimeoutSecs 参数设置为相同的值。

StatPath=*boolean*

如果该参数设置为 true，那么在把请求转交给 WebLogic 之前，NSE 插件首先检查请求中的转换路径是否存在以及它的可读权限。该参数的缺省设置为 false。如果文件不存在，那么会把 HTTP 403/Forbidden 返回到客户端。如果文件存在，但是它的权限不是设置为 world-readable，那么将返回 HTTP 403/Forbidden 回复。在这两种情况下，Enterprise 服务器处理这些回复的缺省机制完成回复内容。该选项只能在 WebLogic servlet 引擎与 Enterprise 服务器具有相同的文档根（document root）时使用。

ErrorPage=*URL*

用于创建错误页，当 Netscapte Enterprise Server 无法把请求转交到 WebLogic 服务器时，会显示该页面。该参数设置为错误页面所在的位置。

CookieName=*cname*

如果要改变 WebLogic 服务器的 WebLogic 会话 cookie 名，那么应该在 obj.conf 文件中设置该属性。WebLogic 会话 cookie 的名字由 Weblogic.properties 文件的 weblogic.httpd.session.cookie.name 属性设置。详细内容，请参见管理员指南中“把 WebLogic 设置为 HTTP 服务器”中的内容。

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 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目标（trusted principal）。

WebLogic 服务器代理插件不会用传输协议（http 或 https）来确定代理插件与 NSAPI 插件之间的连接是否使用了 SSL 协议保护。要使用 SSL 协议，必须把使用代理插件的 WebLogic 服务器配置为使用 SSL 协议。WebLogic 服务器插件代理使用 WebLogic 服务器监听 SSL 通信的端口与 NSAPI 服务器通信。

在 WebLogic 服务器代理插件使用 SSL 协议，必须在 httpd.conf 文件中指定以下参数：

SecureProxy ON/OFF

如果 WebLogic 服务器代理插件与 NSAPI 之间的通信采用 SSL 协议，那么这个参数应该设置为 ON，该参数缺省为 OFF。在定义这个参数之前，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：在主服务器配置中定义或者在虚拟主机配置中定义。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=*filename*

数字证书所在的文件，该数字证书由可靠的证书管理机构发放并被 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。该参数没有缺省值。

文件名必须是数字证书文件的全路径文件名。

RequireSSLHostMatch TRUE/FALSE

该参数决定 WebLogic 服务器代理插件连接的主机名是否必须与连接代理插件的 WebLogic 服务器所使用的数字证书中的 Subject Distinguished Name 字段匹配。参数缺省为 TRUE。

SSLHostMatchOID *Integer*

此参数是 ASN.1 Object ID (OID)，标识端数字认证的 Subject Distinguished Name 中哪个字段是用于执行主机匹配比较的.缺省为 22，对应于 Subject Distinguished Name 中的 CommonName 字段。Common OID 有以下值： Sur Name—23， Common Name—22， Email—13，Organizational Unit—30，Organization—29，以及 Locality—26。

有关 WebLogic 服务器集群失败转移的注意事项

在大多数配置中，如果 NSAPI 插件把请求发送到集群的主要实例，而该实例不可用时，那么请求会转移到次要实例。然而，有一些配置联合使用了防火墙与负载控制器，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（防火墙或负载控制器）都可能会接受这个请求（并返回一个成功的连接）。在试着把请求转交到 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（包括或不包括负载控制器）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 服务器的从实例。因为在这些配置中“连接重设”响应可以进行失败转移转移，因此 servlets 必须是等幂的否则将导致事务的重复处理。

例如，如果在处理交易时，WebLogic 服务器的主实例出现故障，那么会将“连接重设”信息返回到 NSAPI 插件中。NSAPI 插件再把请求转交给 WebLogic 服务器的从实例再次执行请求。如果 servlets 不是等幂的话将会导致事务的重复处理。

在安装了服务补丁程序 6 或更低版本补丁程序的 WebLogic Server 4.5.1 的配置中，返回 “连接重设”回复的 NSAPI 请求不会进行失败转移，这样就不会在不等幂 servlets 中产生复制事务。但是在联合防火墙与/或负载指导的情况下该配置不能提供失败转移。

0bj.conf 文件示例（不使用 WebLogic 集群的情况）

下面这个例子演示了在不使用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。

```
## ------------ 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 集群的情况）

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。

```
## ------------ 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 ---------
```

13 安全管理

本章将介绍以下内容：

- ✓ 安全配置概述
- ✓ 设置 Java 安全管理器
- ✓ 改变系统口令
- ✓ 指定一个安全域
- ✓ 定义用户
- ✓ 定义组
- ✓ 为虚拟机定义一个组
- ✓ 定义 ACL
- ✓ 配置 SSL 协议
- ✓ 配置双向验证
- ✓ 口令的保护
- ✓ 安装审计提供者
- ✓ 安装连接过滤器
- ✓ 配置安全上下文传播

安全配置概述

在 WebLogic 服务器中，安全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，安全配置是一个递归的过程。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
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
3. 定义安全域的用户。你可以在安全域中用组来组织用户
4. 定义 ACL 以及对资源的访问权限
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管理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
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，客户端会对 WebLogic 服务器进行验证，然后 WebLogic 服务器在对客户端进行验证

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”。

有关如何在管理控制台中设置安全属性的详细说明，可以参见管理控制台在线帮助中的内容。

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 \wlserver6.0\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。

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。

指定一个安全域

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
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。

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。

注意：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。

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：设置了缓存域，但不启用缓存。你可以用管理控制台启用缓存。如果使用的安全域不是 File 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，缓存区调用从安全域（seconddary realm）的次数越少。减少调用的频率会提高新能，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。

缓存域的配置涉及到启用各种类型的缓存（如 ACL，验证，组，以及权限）以及定义各缓存的操作方式。所有这些工作都是通过设置 Caching Realm Configuration 窗口的 General 标签页上的属性来完成的。要保存所作的设置，点 Apply 按钮。定义完上述属性后，你应该重启 WebLogic 服务器使配置生效。

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代用安全域或定制安全域的类名。
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。

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。要保存所作的设置，点 Apply 按钮。当完成配置后，重启 WebLogic 服务器。

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

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。要保存所做的设置，点 Apply 按钮，重启 WebLogic 服务器。

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

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。要保存所作的设置，点 Apply 按钮，重启 WebLogic 服务器。

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

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。要保存所作的设置，点 Apply 按钮，重启 WebLogic 服务器。

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

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。要保存所作的设置，点 Apply 按钮，重启 WebLogic 服务器。

下表描述了 Permission 标签页上的各个属性。

表 12-7 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

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使用 LDAP realm V1，所以 BEA 建议用户升级到 LDAP realm V2。
- ✓ LDAP realm V2——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 安全域时，可以通过管理控制台查看 LDAP 目录服务器中

的用户与用户组，但是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）仍然需要使用 LDAP 服务器所提供的管理工具。

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。

在使用 LDAP 安全域之前，首先应该启用缓存域并在缓存域配置窗口的 General 标签页的 Basic Realm 属性中设置 LDAP 安全域的类名。

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。如果存在同名的用户组，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

  LDAP realm V2 中删除了 authProtocol 与 userAuthentication。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，那么进入管理控制台左窗格中的 Security->Realms 节点。然后在团粒控制台的右窗格中，点击 Configure a New LDAP Realm V1 链接。

在 LDAP Realm Create 窗口的 General 标签页中指定 LDAP 安全域的名字以及 LDAP 安全域的类名。点 Apply 按钮保存设置。完成属性配置后，重启服务器。

下表描述了 General 标签页上的各个属性。

表 14-8 General 标签页上的 LDAP 安全域配置属性

属性	描述
Name	LDAP 安全域的名字，例如 AccountingRealm
Realm Class Name	LDAP 安全域的 Java 类名。该 Java 类应该包含在 WebLogic 服务器的 CLASSPATH 中。

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。点 Apply 按钮保存设置，重启 WebLogic 服务器使设置生效。

下表描述了 LDAP Realm V1 标签页上的各个属性。

表 14-9 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。
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. 如果 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 Directory server 支持 CRAM-MD5 验证。Microsoft Site Server Novell NDS 支持 Simple 验证

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。点 Apply 按钮保存设置，重启 WebLogic 服务器使设置生效。

下表描述了 Users 标签页上的各个属性。

表 14-10 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 服务器的一组属性进行核对，从而实现对用户的验证。
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，但常常使用简写字符串，例如 User ID。

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。点 Apply 按钮保存设置，重启 WebLogic 服务器使设置生效。

下表描述了 Groups 标签页上的各个属性。

表 14-11 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。

如果启用了缓存，那么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。Users 缓冲区与 Groups 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。例如调用 getUser() 方法，那么对象只有等到程序释放了它才能被域返回，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。

为了使用 LDAP Security realm V2，进入管理控制台左窗格中的 Security->Realms 节点。选择要使用的 LDAP 服务器。有以下选项：

- ✓ defaultLDAPRealmforOpenLDAPDirectoryServices
- ✓ defaultLDAPRealmforNovellDirectoryServices
- ✓ defaultLDAPRealmforMicrosoftSiteServer
- ✓ defaultLDAPRealmforNetscapeDirectoryServer

右窗格显示了 LDAP 服务器的配置窗。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。

点 Apply 按钮保存设置，设置完所有属性后重启服务器。

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。

WindowNT 安全域提供身份验(用户与组)证但不进行授权。定义在 WebLogic 服务器中的 System 用户同时必须在 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 WindowsNT 安全域，WebLogic 服务器必须作为 NT 的一个服务运行。不要在域控制器中运行 WebLogic 服务器。

因为 WebLogic 在启动时，需要从 fileRealm.properties 文件中读 ACL。改变 ACL 后，应该重启 WebLogic 服务器。如果 ACL 中使用组，可以减少重启 WebLogic 服务器的频率，因为通过更改 Windows NT 组的成员，你就可以动态地管理单个用户对 WebLogic 资源的访问权限。

在使用 Windows NT 安全域之前，需要先启用缓存域并在 Basiec Realm 属性中输入 Windows NT 安全域的类名。

要使用 Window NT 安全域，先进到管理控制台的 Security->Realm 节点，然后在右窗格中，选 Create a New NT Realm 链接。

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，然后点 Apply 按钮保存设置。重启服务器是设置生效。

下表描述了 NT Realm Configuration 窗口中的各属性。

表 14-12 Windows NT 安全域的配置属性

属性	描述
Name	Window NT 安全域的名字，例如 AccountingRealm
Realm Class Name	实现 Window NT 安全域的 Java 类名。应该把这个类放在 WebLogic 服务器的 CLASSPATH 中。
Primary Domain	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。

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\wlserver6.0\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，例如浅口令（shadow password）。

WebLogic 在启动时需要从 fileRealm.Properties 文件中读入 ACL，因此如果你改变了 ACL，那么需要重启服务器。如果在 ACL 中使用组，那么可以减少重启服务器的频率。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。要更改 wlauth 程序的文件属性与文件的所有权以及为 wlauths 设置 PAM 需要有根用户权限。

以下是配置 UNIX 安全域的步骤：

1. 如果 WebLogic 服务器安装在一个网络驱动上，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 是你所使用的平台的名字。
2. 用根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：

```
# 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，创建一个包含以下内容的文件，名字为/etc/pam.d/wlauth

```
#%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：如果不使用浅口令，那么省略上文中的 shadow。

要在 WebLogic 服务器中使用 UNIX 安全域，那么先进入管理控制台左窗格中的 Security->Realms 节点，然后在右窗格中，点 Create a New UNIX Realm 链接。

在使用 UNIX 安全域之前，必须先启用缓存域，然后在 Basic Realm 属性中输入 UNIX 安全域的类名。

UNIX 安全域的配置涉及以下属性：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。点 Apply 按钮可以保存设置，然后重启 WebLogic 服务器。

下表描述了 UNIX Realm Create 窗口上的各属性。

表 14-13 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.

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。

要使用 RDBMS 安全域，先进入管理控制台左窗格的 Security->Realms 节点，然后在右窗格点 Create a New RDBMS Realm 链接。

在使用 RDBMD 安全域之前，需要先启用缓存域并在 Basic Realm 属性中输入 RDBMS 安全域的类名。

需要对 RDBMS 安全域进行以下配置：定义连接数据库的 JDBC 驱动程序以及定义一个数据库模式以保存用户、组以及 ACLs。

与 RDBMS 安全域配置有关的属性位于 RDBMS Realm Create 窗口上的 Gerneral 标签页、Databases 标签页以及 Schema 标签页上。

下表描述了 General 标签页上的各个属性。

表 12-14 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。

下表描述了 Database 标签页上的各个属性。

表 14-15 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

有关保存用户、组以及 ACL 的数据库模式的属性位于在 Schema 标签页上，当对上述三个标签页中的属性进行了必要的设置后，点 Apply 按钮保存设置，然后重启 WebLogic 服务器。

安装一个定制安全域

你可以利用已有的用户存储库（例如网上的目录服务器）创建一个定制安全域。要使用定制安全域，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，先进入管理控制台左窗格 Security->Realms 节点，然后点击左窗格中的 Create a New Custom Realm 链接。

在使用定制的安全域之前，必须先启用缓存域并在 Basic Realm 属性中输入定制安全域的类名。

需要对定制安全域进行以下配置：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。点窗口中的 Apply 按钮保存设置，然后重启服务器。

下表列出了 Custom Security Realm Create 窗口中必须设置的属性。

表 14-16 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

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。

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://*localhos*t:*portnumbe*r/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 HelloWorld 的信息。

迁移安全域

WebLogic Server 6.0 为安全域提供了一个新的管理体系结构。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 属性把安全域转换到新的体系结构中。不过，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，那么需要转换源以使用 Mbeans。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的信息。

定义用户

注意：本节将说明如何在文件域中添加用户，如果使用代用安全域，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 客户端软件。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。

下表描述了 Users 窗口中的属性

表 14-17 用户属性

属性	描述
Name	指用户的名字，区分大小写。用户是一个可以访问 WebLogic 服务器资源的实体。
Password	用户的口令。口令的长度至少要不少于 8 个字符。口令区分大小写

File 安全域有两个特殊的用户：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，如启动与结束服务器，锁定与解锁资源。system 用户在安装 WebLogic 服务器时定义。
- ✓ 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

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）

为了提供 WebLogic 服务器的安全性，建议你禁用 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 用户并未把它删除掉，只是没有人可以以 guest 用户的身份登录到 WebLogic 服务器。

在 Remove These Users 列表框中输入一个用户的名字，点 Remove 按钮就可以删除一个用户。

有关用户以及 WebLogic 服务器的访问控制模型，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，如果代用后备安全域，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。

按以下步骤在 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
3. 在管理控制台的 Group 窗口中设置 Name 属性。
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。

BEA 推荐在给用户组命名时使用复数。例如，使用 administrators 而不是 administrator。

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结束服务器、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。

在 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。

有关组以及 WebLogic 服务器的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。

定义虚拟机的用户组

在 WebLogic 服务器中，如果虚拟主机要求身份验证，那么可以在安全域中定义一个组代表这个虚拟主机。虚拟主机中的用户首先在 WebLogic 服务器的安全域中定义，然后在把它们加入到代表虚拟机的组中。

定义 ACL

用户是否具有访问 WebLogic 安全域中的资源取决于该资源的访问控制列表 ACL。资源的 ACL 定义了某个用户是否有访问该资源的权限。要定义一个资源的 ACL，首先创建该资源的一个 ACL，然后指定资源的权限，然后将权限分配给指定的用户与组。

每个 WebLogic 资源都具有一到多个权限，下表总结了各资源要求验证的功能项。

表 14-18 WebLogic 服务器资源的 ACL

资源	ACL	要求验证的功能
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 servlets	weblogic.servlet.*servletName*	执行
WebLogic JDBC 连接池	weblogic.jdbc.connectionPool.*poolname*	Reserve Reset Shink
WebLogic 的口令	weblogic.passwordpolicy	对用户解锁
WebLogic JMS 收信方	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

以下是在管理控制台中创建 WebLogic 服务器资源的 ACL 的步骤：

1. 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

2. 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。

3. 指定哪些用户或用户组具有该资源的哪些权限。

在创建 WebLogic 服务器资源的 ACL 时，应该使用表 14-18 表中所列出的语法。例如，如果你要指定 JDBC 连接池 demopool 的 ACL，那么该 ACL 的格式应该是 weblogic.jdbc.connectionPool.demopool。

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 Server 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用户访问 Mbean 的信息。

配置 SSL 协议

采用安全套接层协议（SSL），两个通过网络连接的应用可以相互进行身份验证，可以对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。

以下是配置 SSL 协议的步骤：

1. 获得一个 WebLogic 服务器使用的私钥与数字证书。每个使用 SSL 协议的 WebLogic 服务器都需要有各自的私钥与数字证书。
2. 保存 WebLogic 服务器的私钥与数字证书。
3. 通过管理控制台配置与 SSL 协议、私钥、数字证书及 WebLogic 服务器所信任的证书发放机构等相关的属性。这些字段的定义是服务器级的，因此，你必须为每个要使用 SSL 协议的 WebLogic 服务器定义上述字段。

以下各小节将详细介绍上述步骤。

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

要从证书管理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 提交给一个证书管理机构，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。

以下是生成 CSR 的步骤：

1．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。在表单中填写下表所描述的信息。

表 14-19 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	你所在部门的名字
Organization name	你所在机构的名字。证书管理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。 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，那么证书生成器会给出提示信息，按浏览器的回褪按钮更正错误。

如果所有的字段都填写正确，证书生成器在 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—提交到证书管理机构的 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. 选择一家证书管理机构，按以下步骤从证书管理机构的 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 browser 提供 40 位加密选项。
- ✓ Entruest.net 数字证书为 domestic browser versions 提供 128 位加密，为 export browser versions 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. 在收到证书管理机构所提供的数字证书后，把它保存在 \wlserver6.0\config\mydomain 目录中。

注意：如果私钥文件不是由证书生成器生成的，那么应该验证该私钥文件是否符合 PKCS#5/PKCS#8 PEM 格式。

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目录与文件名。
- ✓ 在 Trusted CA File Name 字段中输入证书管理机构发放的数字证书所在的完整目录名。
- ✓ 在 Server Key File Name 字段中，输入 WebLogic 服务器的私钥文件所在的完整目录名。

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 是在请求数字证书时输入的口令

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 \wlserver6.0\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。不同的格式采用不同的文件扩展名。

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，有时，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。典型地，WebLogic 服务器的数字证书文件与证书链分别位于不同的文件，数字证书文件的后缀名可以是.pem 或.der。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。如果你用多个证书管理机构发放的数字证书创建一个文件，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。

定义可靠的证书管理机构

WebLogic 服务器在建立 SSL 连接时，根据一组可靠的证书管理机构列表检查证书管理机构的身份，从而保证当前所使用的证书管理机构是可靠的。

把数字管理机构的根证书复制到 WebLogic 服务器的\wlserver6.0\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管理机构所发放的数字证书后面。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，那么应该在可靠 CA 文件（trusted CA file）中包含证书管理机构的根证书。

定义 SSL 协议的配置字段

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

下表描述了 Server Configuration 窗口 SSL 标签页上的属性字段。

注意：如果使用 PKCS-8 保护的私钥，那么在启动 WebLogic 服务器的命令行中应该指定私钥的口令。

表 14-20 SSL 协议的配置属性

属性	描述
Enabled	如果要使用 SSL 协议，那么选中这个复选框。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文件名。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	WebLogic 服务器数字证书所在的目录。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。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签名 WebLogic 服务器数字证书的数字证书，第二个成员应该是签名第一个数字证书的文件，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管理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管理机构发放的数字证书。该文件包含证书管理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，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”。
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 服务器提供了 SSL 协议的纯

	Java 实现；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 字段所指定的证书管理机构发放的。 缺省情况下，SSL Handler 允许 WebLogic 服务器把 SSL 连接引向另一个 WebLogic 服务器。例如，WebLogic 服务器中的一个 EJB 可以打开另一个 WebLogic 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 连接的能力不受 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。详细内容，可以参见“在 Applets 中使用双向验证”中内容。
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 erifier 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。

在以前的 WebLogic 服务器版本中，Server Certificate File Name 属性所定义的数字证书可能是一个自签名的无效数字证书。这不是一个很好的安全策略，因此目前的 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 服务器根据一个可靠的证书管理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 服务器使用的数字管理机构根证书复制到 \wlserver6.0\config\mydomain 目录中。在双向认证中，客户端需要出示由这些可靠的证书管理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。

Configuring RMI over IIOP over SSL（TBD）

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保护连接。要用 SSL 协议保护 RMI 上的 IIOP 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。
3. 用 host2ior 工具把 WebLogic 服务器 IOR 打印到控制台中。Host2ior 工具打印两个版本的 IOR：一个是 SSL 连接的，一个是非 SSL 连接的。IOR 的头指定了该 IOR 是否可以用于 SSL 连接。
4. 使用 SSL IOR 获得访问 CosNaming 服务的初始引用。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（TBD）中的内容。

口令的保护

应该保护用来访问 WebLogic 服务器资源的口令。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 \wlserver6.0\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步骤：

- ✓ 备份 SerializedSystemIni.dat 文件。
- ✓ 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。

Config.xml 文件中的口令不是明文的，而是经过加密并被散列化的口令。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。管理控制台在下一次写文件时将加密并散列化口令。

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. 打开 Security Configuration 窗口
3. 选择 Passwords 标签页。定义该标签页中需要配置的属性（详细信息，参见下表）。
4. 点 Apply 按钮保存所做的设置
5. 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 14-21 口令保护属性

属性名	描述
Minimum Password Length	口令的长度，至少为 8 个字符。缺省为 8 个字符
Lockout Enabled	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 不正确）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 属性所定义期限内。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，多长时间内该用户帐号不能被使用。默认为 30 分钟。要解开一个被锁住的用户帐号，你必须拥有 weblogic.passwordpolicy 的 unlockuser 权限。
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。 当某一帐号在该属性定义的时间范围内，被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 Security Configuration 窗口的 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，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 General 标签页的 Connect Filter 字段，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 目录中的内容。

设置 Java 安全管理器

当你在 Java 2(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。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。

注意：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不同。

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。你可以在该文件的基础上创建自己的安全策略文件。

1. 编辑 webLogic.policy 文件的以下行，用你的 WebLogic 服务器所在的目录替换相应的部分

```
grant codebase "file:./c:/weblogic/-"{
permission                          java.io.FilePermission
"c:${/}weblogic${/}-",...
```

注意：该变更假定 WebLogic 服务器的安装目录结构与 BEA WebLogic Server Installation Guide 中的 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${/}wlserver600${/}weblogic${/}management${/}co
nsole${
/}-", "read";
permission java.io.FilePermission
"D:{/}BEA${/}wlserver600${/}config${/}mydomain${/}applic
ations$
{/}.wl_temp_do_not_delete${/}weblogic${/}management${/}c
onsole$
{/}-", "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。

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。

在启动 WebLogic 服务器的 Java 命令行上设置 java.security.manager 与 java.security.policy 属性。这两个属性执行以下功能：

- ✓ java.security.manager 属性指定 Java 虚拟机使用安全策略。不需要为这个属性指定任何参数。
- ✓ java.security.policy 属性指定了被 JVM 使用的安全策略文件所在的位置。该属性的参数应该设置为 weblogi.policy 文件的路径名加文件

名。

如果要在 WebLogic 服务器中使用安全管理器与 weblogic.policy 文件，在启动时使用以下命令

```
$ 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。

可以用 RecordingSecurityManager 工具发现 WebLogic 服务器在启动与运行过程中所出现的权限问题。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 RecordingSecurityManager。

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 WebLogic 企业（WebLogic Enterprise，简称为 WLE）域中的资源与对象。WebLogic 服务器的 BEA WLEC（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 服务器安全域中的用户安全标识将作为 IIOP 请求的服务环境上下文的一部分被传播，这里的 IIOP 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 WLE 域创建 WLEC 连接池。WLEC 连接池由 IIOP 连接组成。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 WLE 域中的对象、激活 WLE 域中的操作。

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 *WLE_HOM*E/lib/wleorb.jar 和 *WLE_HOM*E/lib/wlepool.jar 。

更多资料，请参照“使用 WLEC”。

要实现安全上下文传播的步骤如下：

1. 为安全上下文传播创建一个新的 WLEC 连接池。要创建 WLEC 连接池，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 14-22 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 WLE 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*:*por*t.

Failover Addresses	不能与 Primary Addressed 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 WLE 域的名字。一个 WLE 域只能有一个 WLEC 连接池。域的名字必须与 WLE 域的 UBBCONFIG 文件中的 RESOURCES 部分的 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。

2. 点 Create 按钮

3. 定义 Connection Pool Configuration 窗口的 Security 标签页上的属性，这样就可以把一个用户的安全上下文传播到 WLE 域。下表描述了 Security 标签页中的这些字段。

表 14-23 Security 标签页上的 WLEC 连接池属性

属性	描述
User Name	一个 WLE 用户的用户名。只有当 WLE 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	WLE 用户的角色。只有当 WLE 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	WLE 应用的口令。只有当 WLE 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	WLE 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 WLE 之间的 SSL 连接将失败。
Maximum Encryption Level	WLE 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 WLE 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 WLE 域，那么应该启用该选项。 该属性缺省为禁用的。

4. 点 Apply 按钮保存所做的设置，然后重启 WebLogic 服务器。

5. 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

6. 设置 ISL 命令的-E 选项。该命令的作用是使 IIOP 监听器/处理器可以监听并利用来自 WebLogic 服务器的安全上下文。在-E 选项中，需要指定一个 principal name。Principal name（TBD）定义了 WLEC 连接池使用什么样的 principal 来登录到 WebLogic Enterpise 域。Principal name 应该与 WLEC 连接池的 User Name 属性匹配。

当 WebLogic 服务器环境与 WebLogic Enterprise 环境之间的通信使用安全验证时，这意味着在 WebLogic 服务器与 WebLogic Enterprise 环境的 CORBA 对象或 RMI 对象或者是 EJB 建立连接时，会重新执行 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，那么必须使用证书验证。以下是设置证书验证的步骤：

1. 获得 principal(TBD)的数字证书。把私钥文件保存在 WebLogic Enteroprise 的 TUXDIR/udataobj/security/keys 目录中。
2. 运行 tpusradd 命令。该命令的作用是把 principal（TBD）定义为一个 WebLogic Enterprise 用户。
3. 定义 UBBCONFIG 文件的 IIOP 监听器/处理器，使用-E 选项表明该 principal 是用于验证的。
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 principal（TBD）名字。
5. 获得 IIOP 监听器/处理器的数字证书。
6. 在 ISL 命令中用 SEC_PRINCIPAL_NAME 选项指定数字证书。使用-s 选项表明 WebLogic Enterprise 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 WLE 文档的“创建配置文件”部分。

有关 corbalocs 前缀的更多信息，可以参见 WLE 文档的“理解 Bootstrap 对象的地址格式”部分。

有关 WLE 安全级别的更多内容，可以参见 WLE 文档的“定义安全级别”部分。

14 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以前配置事务属性；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 实行。有关设置日志属性的更多内容，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 any transaction with heuristic outcome（TBD）

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。

15 管理 JDBC 连接

WebLogic 服务器通过连接池与 JDBC 驱动程序相连。通过管理控制台，你可以设置或管理连接池。本章节介绍以下与 JDBC 连接相关的内容：

- ✓ 用管理控制台管理 JDBC
- ✓ JDBC 配置指南
- ✓ 连接池
- ✓ 多池
- ✓ 数据源

用管理控制台管理 JDBC

你可以用管理控制台完成以下工作：

- ✓ 定义 JDBC 连接的监控属性
- ✓ 管理已建立的连接

有关如何打开管理控制台的信息，可以参见“WebLogic 服务器与集群”中的内容

有关 JDBC 与 JDBC 驱动程序的详细信息，请参见“WebLogic JDBC 编程指南”中的“WebLogic JDBC 概述”

有关 JDBC 属性的完整列表，可以参见以下网页的“WebLogic 管理控制台在线帮助”部分。

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

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。

JDBC 配置概述

在配置 JDBC 连接前，先要创建一个连接池与一个数据源对象（建议你使用这种方式，但某些情况下这是可选的），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 14-1 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 驱动器
功能	本地事务	两阶段提交	单一资源管理器与单一数据库示例
两层 JDBC 驱动程序	WebLogic jDriver for Oracle, Microsoft SQL Server, and Informix 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 JTS 驱动程序
多层 JDBC 驱动程序	WebLogic JDBC JTS, Pool, 以及 RMI 驱动程序		
数据源	建议使用 DataSource 对象	必须使用 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

注意：分布式使用 WebLogic jDriver for Oracle/XA，它是 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，你可以创建建立数据连接、进行查询、修改数据源以及处理结果集所要的对象。

支持分布式事务的驱动程序

- ✓ 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 驱动程序有 WebLogic jDriver for Oracle/XA。

只支持 JDBC 2.0 core API 但不支持 JDBC 2.0 分布式事务标准扩展接口的 JDBC 驱动程序不能参与分布式事务， 但如果把 TxDataSource 的两阶段提交属性设置为 true 的话，那么这类驱动程序也可以参与分布式事务。

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

要配置本地事务驱动程序，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。这些属性将被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：“装及使用 WebLogic jDriver for Oracle”“ 安装及使用 WebLogic jDriver for Informix”与 “安装并使用 WebLogic jDriver for Microsoft SQL Server”。如果使用第三方驱动程序，请参见供应商所提供的文档。

下表是一个使用 WebLogic jDriver for Oracle 的连接池配置示例。

表 14-2 WebLogic jDriver for Oracle 的 JDBC 连接池配置

属性类型	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;password=tiger;server=localdb

下表是一个使用 WebLogic jDriver for Oracle 的数据源配置示例。

表 14-3 WebLogic jDriver for Oracle：数据源配置

属性类型	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 XA 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 14-4 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 14-5 WebLogic jDriver for Oracle/XA：TxDataSource 配置

属性类型	属性值
名字	FundsXferAppPool
目标	Myserver
JNDIName	Myapp.fundsXfer
连接池的名字	Myapp.fundsXfer

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 实例的自省（reflection）来配置的。也就是说，对于一个 abc 属性，XADataSource 实例必须支持名字分别为 getAbc 与 segAbct 的 set 与 get 方法。

下表是一个使用 Oracle Thin 驱动程序的 JDBC 连接池配置。

表 14-6 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 14-7 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 14-8 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 14-9 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 14-10 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 字符串中的字段。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 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 14-10 WebLogic jDriver for Oracle/XA 的数据源属性

属性名	类型	描述	JDBC2.0	Optional	缺省值
databaseName (不使用)	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 （不使用）	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 （不使用）	int	服务器监听请求的端口号	Y	Y	None
roleName （不用）	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	Y*	None
openString	String	Oracle's XA 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有一个形式为 xa_poolnamedate.trc 文件	N	Y	true

表 14-11 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 14-11 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	Y

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 Non-XA JDBC 驱动程序

在配置连接池时，如果要使 non-XA JDBC 驱动程序参与到分布式事务中，那么必须指定 JDBC Tx 数据源的 enableTwoPhaseCommit 属性（支持 XAResource 接口的资源会忽略这个参数。）

如果 enableTwoPhaseCommit 属性设置为 true，那么在 XAResouc.prepare()方法调用时，non-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 heruristic 错误结果，从而应用数据可能会因为 heruristic 失败而出现不一致。

如果 enableTwoPhaseCommit 设置为 false,那么 non-XA 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,那么一阶段优化会绕过 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 14-12 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 14-13 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

连接池

连接池中存放了一组 JDBC 连接，这些连接在连接池注册时（通常是在启动 WebLogic 服务器的时候）创建的。应用程序从连接池中取出一个连接，然后用它连接数据库，当应用使用完这个连接时，应用将关闭这个连接，并将该连接返回到连接池中。使用连接池对性能及应用设计可以带来以下好处：

- ✓ 使用连接池比每次在客户端有连接数据库请求时再新建一个连接有效的多。
- ✓ 不需要在应用中写硬代码，如 DBMS 口令
- ✓ 可以限制 DBMS 的连接个数，这样可以有效管理 DBMS 连接的 licens 所带来的限制。
- ✓ 可以更换数据库管理系统而不需要更改应用代码。

设置连接池

可以用管理控制台来创建连接池并将所创建的连接池分配给某个服务器或某个域。要建立一个连接池，需要做如下操作：创建一个连接池，将它分配给一个服务器或一个域，然后使它与一个数据源关联。在启动时，WebLogic 将打开与数据库的连接，并将这些连接加入到连接池中。连接池配置属性的完整列表，请参考以下网址中 WebLogic 管理控制台在线帮助中 JDBC 属性列表中的配置表。

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

管理控制台的以下标题中解释了进行连接池配置所要完成的步骤：

- ✓ 创建一个 JDBC 数据源，也可以参见“数据源”中的内容
- ✓ 创建 JDBC 连接池
- ✓ 克隆连接池
- ✓ 分配连接池

你也可以使用 API，在运行的 WebLogic 服务器中通过程序来创建连接池，有关这方面的内容，请参见 WebLogic JDBC 编程指南中的“创建动态连接池”中的内容。

管理连接池

在建立了 JDBC 连接后，可以使用管理控制台来管理连接池。你可以在管理控制台的以下标题中找到如何管理连接池的内容。

- ✓ 监控一个连接池的所有实例
- ✓ 删除一个连接池

多池

多连接池为单个服务器的 WebLogic 服务器配置提供负载平衡以及备份连接池的支持。一个连接池中的所有连接都是相同的，也就说它们都连接到同一个数据库。而多池中的连接池则可以连接到不同的数据库管理系统。

创建多池

在使用多池之前，必须先定义或命名一个多池。然后在决定要将哪些先前定义的连接放入到元池中。你可以在管理控制台在线帮助中的以下标题找到有关这些任务的内容。

- ✓ 创建一个 JDBC 多池
- ✓ 分配 JDBC 多池
- ✓ 复制一个 JDBC 多池

管理多池

多池的管理包括对多池的监控以及在必要时删除多池。

监控 JDBC 连接池

有关监控 JDBC 连接池的更多内容，请参见管理员指南中的“监控 WebLogic 域”中的内容。在管理控制台在线帮助中的以下标题中解释了如何完成这些任务。

- ✓ 监控一个多池的所有实例
- ✓ 删除一个 JDBC 多池

删除连接池

你可以在管理控制台在线帮助中的以下标题中找到有关如何完成这些任务的内容：

数据源

JDBC 客户端通过数据源对象获得数据库连接。在使用数据源对象之前，你必须先通过管理控制台创建一个数据源对象，然后将该数据源对象指向一个连接池。

你可以在定义数据源的同时定义用来支持分布式事务的 JTA。有关如何在定义

数据源对象的 JTA，请参见以下网址中的事务管理部分：

http://bernal.beasys.com/stage/docshal/adminguide/managetx.html.

创建数据源

你可以在管理控制台在线帮助的以下标题中，找到有关如何创建数据源的内容：

创建 JDBC 数据源

复制 JDBC 数据源

分配一个 JDBC 数据源

数据源管理

你可以使用管理控制台来监控数据源的状态，甚至在必要时删除数据源。你可以在管理控制台的以下主题中找到如何完成上述任务的内容：

管理一个 JDBC 数据源的所有实例

删除一个数据源

删除一个 JDBC 数据源

16 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、收信方(队列与主题)、收信方模板、收信方主键、备份库、会话池、连接使用者。
- ✓ 建立客户 JMS 应用。
- ✓ 定义极限和配额（TBD）
- ✓ 启用需要的 JMS 特性，如服务器集群（见下节），并发消息处理，收信方的排序，以及消息的持久化。

有一些 WebLogic JMS 属性有缺省值，而其它属性需要配置。如果没有正确配置这些属性，那么重启时，WebLogic 服务器不会启动 JMS。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，你可以参照这个表来规划 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。

用管理控制台配置连接工厂的步骤如下：

- ✓ 需要配置的属性包括：

  1．连接工厂的名字

  2．连接工厂的 JNDI 名字。

  3．长期订阅方客户端标识（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（即：优先级、存活期及模式）

  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，广播会话将采取的行动)。

  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收信方（destination）。使用模板有以下好处：

- ✓ 在定义新的收信方时，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。

收信方模板的属性在管理控制台的“Templates”节点设置。收信方模板的属性与收信方相同。通常收信方会继承所使用的收信方模板的属性配置，但不包括以下情况：

- ✓ 如果使用了模板的收信方重新设置了某个属性，那么新值有效。
- ✓ 收信方不会继承收信方模板的 Name 属性，因为 Name 属性只对收信方模板有效。每个收信方必须明确定义一个唯一的名字。
- ✓ 收信方模板没有定义 JNDI Name，Enable Store 以及 Template 属性。
- ✓ 收信方模板没有定义 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收信方属性使用缺省值。如果一个属性没有缺省值，那么必须为收信方模板或收信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”。

配置收信方主键

收信方的的排序方式用收信方主键（Destination Key）来定义。

在管理控制台的 Destination Keys 节点中，配置收信方主键的以下属性：

- ✓ 收信方主键的名字
- ✓ 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收信方主键的详细信息，可以参考管理控制台在线帮助中的“JMS 收信方主键”。

配置备份库

备份库（Backing Store）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数据库访问持久化的消息。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。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备份库，需要在管理控制台的 Store 节点下设置以下属性：

- ✓ 备份库的名字
- ✓ 文件备份库所在的目录（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 数据库备份库而言）。

有关如何创建及配置文件备份库与数据库备份库的信息，请分别参见管理控制台在线帮助中的“JMS 文件存储器”以及“JMS JDBC 存储器”中的内容。

注意：随着存储的消息增多，使用 JMS 备份库会增加 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启动失败，这时就需要根据 JMS 备份库中的消息数量来增加虚拟机中堆的大小，然后重新启动 WebLogic 服务器。

配置 JMS 服务器

JMS 服务器管理连接以及用户的消息请求。

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. 用来创建临时收信方的模板，临时收信方可以是临时队列也可以是临时主题。

  4. 消息数量和尺寸的极限和配额 (最大数量，上下限)（TBD）

- ✓ JMS 服务器所关联的目标服务器（WebLogic 服务器）用以支持集群。通过目标服务器可以限制要部署 JMS 服务器的服务器集、组或/以及集群。

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。

配置收信方

一个收信方（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收信方。一个 JMS 服务器可以配置多个收信方。

可以直接定义收信方，也可以通过模板生成收信方。前面讲过，收信方模板用来定义具有相似配置的收信方。

你可以在管理控制台的 Destiation 节点直接配置收信方。以下是需要配置的属性：

- ✓ 收信方的名字与类型（队列或主题）
- ✓ 收信方的 JNDI 名字
- ✓ 是否用备份库保存持久化消息
- ✓ 用来创建收信方的模板
- ✓ 定义收信方排序的主键
- ✓ Thresholds and quotas for messages and bytes (maximum number, and high and low thresholds)（TBD）
- ✓ 可以被重载的属性（如优先级，存活期（time-to-live），以及传递模式）
- ✓ 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收信方的内容，可以参见管理控制台在线帮助的“JMS Destiantion”。

某些收信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收信方

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，收信方，消息生产者，消息使用者以及服务器会话池。

服务器在运行期间会不断地产生上述对象的统计信息，只有重启服务器才能重置统计值。

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。

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 备份库	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——使用文件备份库	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。
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	CoonsumerClosedException 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。

17 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

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。

在把对象装载到 JNDI 树之前，应该先确定对象的 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 JNDI 树，点击 Monitering 标签页上的“view JNDI Tree”链接。

18 管理 WebLogic J2EE 连接器构架

WebLogic J2EE 连接器构架基于 Sun Microsystems J2EE Connector Specification, 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
- ✓ 删除一个资源适配器
- ✓ 监控连接池与参数
- ✓ 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 Sun MicrosSystems J2EE Platform Specification,version 1.3 的实现上。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）只需将对 J2EE 连接器构架的支持扩展到它们的产品中，就能保证与各种 EISes 的连接。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。The resource adapter remains undeployed for the duration of the server session, as long as you do not change it once it has been undeployed。You cannot re-use the deployment name with the deploy argument until you reboot the server。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

注意：可以直接编辑 Weblogic-ra.xml 文件来配置资源适配器。详细信息可以参考“WebLogic J2EE 连接器构架”的第 3 章“资源适配器”。

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：

- ✓ 使用弹出菜单

  a．在左窗格的 Deployments 下，选择要删除的连接器

  b．右点鼠标，然后选 Delete 菜单项

  c．确认删除

- ✓ 使用 Connector Deployments 表格

  a．在左窗格的 Deployments 下，选要删除的连接器名

  b．一个已部署的连接器列表显示在右窗格中的 Connector Deployments 表格中。

  c．点垃圾箱图标删除连接器

  e．确认删除

查看元素与属性

在当前的 Beta 版中，还不能用 WebLogic 服务器的管理控制台查看或编辑配置 J2EE 连接器构架资源适配器的分发描述符元素与属性。

但是，可以用 weblogic.Admin 来查看 ConnectorComponent Mbean 的信息。命令如下：

```
java weblogic.Admin -url t3://localhost:port -username xxx
-password yyy  GET -pretty -type ConnectorComponent
```

有关如何使用 weblogic.Admin 工具的说明，可以参考“管理指南”中的“使用命令行查看或更改 Mbean 属性以管理 WebLogic 域”中的内容。

监控连接池与参数

当前的 Beta 版本中，还不能用管理控制台来查看 J2EE 连接器构架资源适配器的监控信息。

可以用 weblogic.Admin 查看连接器组件的运行时信息与配置信息。所使用的命令如下：

```
java weblogic.Admin -url t3://localhost:port -username xxx
-password yyy
```

```
GET    -pretty    -type    (ConnectorServiceRuntime
ConnectorConnectionPoolRuntime
ConnectorConnectionRuntime)
```

有关如何使用 weblogic.Admin 的说明，，可以参考“管理指南”中的“使用命令行查看或更改 Mbean 属性以管理 WebLogic 域”中的内容，以及 weblogic.management.runtime 的 Javadocs，它详细描述了 runtime MBean 的属性值。

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. 以下是对分发描述符文件中的元素进行编辑的步骤：

   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。

   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

   编辑表单中的文本文字

   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：

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。

右点元素，并从下拉菜单中选择 Configure a New Element 选项

在右窗格的表单中输入元素的信息

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：

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。

右点该元素，并从下拉菜单中选择 Delete Element 菜单项。

点 Yes 确认删除

9．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。

19 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

使用评估许可证

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 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.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 文件中的头。缺省的文件头为“-----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：在文件头所在行的结尾处应该另起一行。

例子

$ java utils.der2pem graceland_org.der
Decoding
.....
.....
.....
.....
.....

dbping

dbping 命令行工具用来测试 DBMS 与客户端之间使用两层 WebLogic jDriver 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
List	列出指定 WebLogic 服务器中的所有应用
deploy	将一个 J2EE 应用.jar,.war 或.ear 文件部署到指定的服务器中
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
delete	按给出的应用名删除一个应用

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，也可以使用分发或控制台工具指定。
源	指出应用包文件（.jar,.war 或.ear）的 exact 位置，或者是应用目录所在的路径。

选项

选项	定义
-help	打印出 deploy 工具的所有选项
-version	打印出 deploy 工具的版本

-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。
-host *host*	指定部署 J2EE 应用（.jar, .war, .ear）的 WebLogic 服务器的主机名。
-url *url*	指定 WebLogic 服务器的 URL，缺省为 localhost:7001
-component componentname: target1, target2	部署到各种目标上的组件，必须指定为： componentname:target1, target2 可以任意多个组件指定该选项，且可以指定任意多次。
，-username *username*	连接所使用的用户名，缺省为 sytem。
-debug	将部署过程中的详细调试信息输出到 stdout。

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，

将以应用的名字重命名该文件。因此，上面这个例子中，应用包的名字 . . ./config/mydomain/applications 目录从 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 阴功，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

The logToZip utility searches an HTTP server log file in common log format, finds the Java classes loaded into it by the server, and creates an uncompressed .zip file that contains those Java classes. It is executed from the document root directory of your HTTP server.

要使用该工具，你必须有访问由 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，用该工具测试有关广播的问题。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错误顺序警告。
4. 当一个预期的消息没有受到时的消息丢失警告。

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），注意不要设置为集群的广播地址。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。Idle timeout, in seconds, if no multicast messages are received. 缺省时间为 600 秒（10 分钟）。如果过了超时时间，a positive confirmation of the timeout is sent to sdout。
-s *send*	可选参数。发送消息的时间间隔，缺省值为 2 秒。A positive confirmation of each message sent out is sent to stdout。

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、类路径以及操作系统的详细信息。

语法

```
$ 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

该工具用来测试使用两层 JDBC 驱动程序的 WebLogic JDBC 连接。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

如果从 HTTP 服务器的文档根目录中执行该工具，verboseToZip 获得运行在 verbose 模式的标准输出，找到所引用的 Java 类，并创建一个包含这些 Java 类的 uncompressed .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.version
```

例子

$ java weblogic.version WebLogic Build: 4.0.1 04/05/1999 22:02:11 #41864

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 服务器的命令行接口参考

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
- ✓ Mbean 管理命令参考
- ✓ 配置及管理连接池的命令

命令行接口简介

As an alternative to the Administration Console, WebLogic 服务器提供了管理工具与多数配置工作以及许多运行时的 Mbean 属的性了命令行界面

以下情况可以使用命令行界面：

- ✓ 如果要用脚本进行管理
- ✓ 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

Before You Begin

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 Server Installation Guide”部分：

   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

   a. 从服务器的安装目录启动服务器

   b. 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*

本节中的例子基于以下假设：

- ✓ 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

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

运行管理 run-time Mbeans 的命令的管理员不许具有相应的访问控制权限。

见以下章节：

- ✓ B-3 页的“WebLogic Server 管理命令参考”
- ✓ B-17 页的“Mbean 管理命令参考”

WebLogic 服务器管理命令参考

表 T-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
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 each round trip 的总时间（用两位数表示）以及每次连接的平均持续时间 见 8-6 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 8-7 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“starting and Stopping WebLogic Servers”部分，见以下页面： 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-8 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
锁住 WebLogic 服务器	LOCK	Locks a WebLoigc Server against non-privileged logins。以后的任何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 8-10 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-11 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-12 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-13 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-14 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-15 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-16 页的“VERSION”。

CONNECT

对 WebLogic 服务器进行指定次数的连接，并返回 each round trip 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

例子

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 *[COMMAN*D]

例子

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 *ur*l] [-username *usernam*e]
[-password *passwor*d]<*COMMAN*D><*ARGUMENT*S>
PING <*coun*t><*byte*s>

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 *UR*L] [-username *usernam*e] [-password *passwor*d] LICENSES

例子

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 *usernam*e] [-password *passwor*d] 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。

例子

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

Locks a WebLogic Server against non-privileged logins。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。

例子

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."

当应用使用一个没有权限的用户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

例子

下例中，PING 命令对 localhost 机器中的 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。
- ✓ 如果没有指定服务器的 URL，那么可以取得非管理服务器中的日志。
- ✓ 如果省略了 starttime 与 enttime 参数，a running display of the entire server log is started。

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。Latest time at which messages are to be displayed。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"

例子

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。

例子

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.
```

THR EAD_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

例子

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

例子

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。

Mbean 管理命令参考

表 T-2 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-18 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-19 页的“DELETE”。
查看 Run-Time Mbean 的属性	GET	显示 run-time Mbean 的属性 见 B-22 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-22 页的“INVOKE”
Viewing Run-Time Metrics and Statistics	INVOKE GET	运行 INVOKE 与 GET 命令来查看 run-time metrics ans statistics。只能对 run-time Mbeans 执行该命令。 见 B-22 页的“INVOKE”以及 B-20 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-23 页的“SET”操作。

CREATE 命令

用 CREATE 命令创建一个 configration Mbean。如果创建成功则将 OK 返回给 stdout。该命令只适用于 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。

注意：在创建 Mbeans 时，同时创建了配置对象（configrationobjects）。

有关创建 Mbeans 的详细信息，可以参见以下页面的“Developing WebLogic Server Application”部分：

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。When creating attibutes for multiple objects of the same type。
mbean_name	必需的参数。Mbean 的完整名字。格式如下： “domain:Type=*type*, Name=*name*”
domain_name	可选参数。域的名字，例如，mudomain。如果没有指定 domain_name，那么使用缺省域的名字。

例子

java weblogic.Admin -url localhost:7001 -username adminuser -password gumby1234 CREATE –mbean “mydomain:Type=Server,Name=acctServer”

DELETE 命令

删除一个配置 Mbean。如删除成果则把 OK 返回给 stdout。该命令不能用于 run-time Mbeans。

注意：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“Developing WebLogic Server Applications”部分

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。When deleting attibutes for multiple objects of the same type
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。

例子

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。You can request a list of attributes for multiple objects of the same type by requesting attributes for the following:

- ✓ 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。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。
property	

例子

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*" 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： "*String*1;*String*2;. . . "
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*"

例子

下面的例子调用了一个名字为 dmin_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*"

配置及管理连接池的命令

以下命令可以用来配置或管理连接池：

- ✓ CREATE_POOL
- ✓ RESET_POOL
- ✓ DESTROY_POOL
- ✓ ENABLE_POOL
- ✓ DISABLE_POOL
- ✓ EXISTS_POOL

命令的名字

正在开发之中。

语法

```
java weblogic.Admin …
```

参数	定义

Web 服务器插件的参数

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

web 服务器插件的参数在配置文件中设置。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 server 插件的一般参数

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 Web server 中的虚拟主机名） 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 8-3 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创建 cookie 的那个服务器上（请求包含 cookie、URL-encoded 会话或 POST 数据中的会话）。
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，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 记录发送到 weblogic 消息的头，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
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。 该参数只应该用于调试目的。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。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完成响应的体内容。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，那么插件将进行失败转移。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。更多信息，可以参见以下页面的“Configuring Welcome Pages”部分：http://e-docs.bea.com/wls/docs61/webapp/components
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。

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 WebLogi C 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 server mbean 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	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 OFF，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 server 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机配置中定义。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管理机构发放并被 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主机的名字是否必须与代理所连接的 WebLogic 服务器所使用的数字证书中的 Subject Distinguished Name 字段匹配。
SSLHostMatchOID	22	The ASN.1 Object ID (OID) that identifies which field in the Subject Distinguished Name of the peer digital certificate is to be used to perform the host match comparison. The default for this parameter corresponds to the CommonName field of the Subject Distinguished

		Name. Common OID values are: _ Sur Name—23 _ Common Name—22 _ Email—13 _ Organizational Unit—30 _ Organization—29 _ Locality—26