BEA WebLogic:
Guía de Administración

Índice

Introducción a la Administración de WebLogic Server

Esta sección describe las herramientas disponibles para administrar WebLogic Server.
Nuestra implementación de BEA WebLogic Server proporciona a los usuarios un conjunto de recursos inter-relacionados. El manejo de estos recursos incluye tareas como arrancar y parar servidores, balancear las cargas de servidores o los almacenes de conexiones, seleccionar y monitorizar la configuración de recursos, detectar y corregir problemas, monitorizar y evaluar el rendimiento del sistema, y desplegar aplicaciones Web, Enterprise Javabeans (EJBs) u otros recursos.
La herramienta principal que WebLogic proporciona para realizar estas tareas es una robusta Consola de Administración basada en Web. La Consola de Administración es nuestra ventana al interior del Servicio de Administración de WebLogic. El Servicio de Administración, una implementación del estándar "Java Management Extension" (JMX) de Sun, proporciona facilidades para el manejo de recursos de WebLogic.
A través de la Consola de Administración podemos configurar atributos de recursos, desplegar aplicaciones o componentes, monitorizar el uso de recursos (como la carga del servidor o el uso de la memoria de la Máquina Virtual Java o la carga del almacén de conexiones de base de datos), ver los mensajes de log, apagar el servidor, o realizar otras acciones de control.

Dominios, el Servidor de Administración y los Servidores Controlados

Un conjunto de recursos de WebLogic Server inter-relacionados manejados como una unidad se llama un **dominio**. Un dominio incluye uno o más WebLogic Servers, y podría incluir clusters WebLogic Server.
La configuración para un dominio se define en Extensible Markup Language (XML). El almacenamiento persistente de la configuración del dominio lo proporciona un único fichero de configuración XML `install_dir/config/domain_name/config.xml` (donde `install_dir` es el directorio bajo el que instalamos WebLogic Server).
Un dominio es una unidad administrativa auto-contenida. Si una aplicación se despliega en un dominio, los componentes de esa aplicación no pueden desplegarse en servidores que no sean parte de ese dominio. Cuando un cluster está contenido en un dominio, todos sus servidores deben ser parte de ese dominio también.
A un WebLogic Server que ejecuta el Servicio de Administración se le llama Servidor de Administración. Este servidor proporciona un punto central de control para la configuración y monitorización de todo el dominio. El Servidor de Administración debe estar ejecutándose para poder realizar cualquier operación de control sobre el dominio. En una configuración con varios WebLogic Servers, sólo uno es el Administration Server; a los otros servidores se les llama servidores controlados. Todo servidor WebLogic controlado obtiene su configuración de arranque desde el servidor de administración. La misma clase, `weblogic.Server`, podría arrancarse en el Servidor de Administración de un dominio o como un WebLogic Server controlado. Un WebLogic Server no arrancado como servidor controlado es un Servidor de Administración.
En una configuración típica de un sistema de producción, las aplicaciones y componentes con nuestra lógica de negocio podrían desplegarse sobre Servidores Controlados y el servidor de administración sería para la configuración y monitorización de los servidores controlados.
Un dominio está activo si el Servidor de Administración se ha arrancado usando esa configuración. Mientras el Servidor de Administración se está ejecutando, sólo él puede modificar el fichero de configuración. La Consola de Administración y la utilidad de administración de la línea de comandos proporcionan ventanas al interior del Servidor de Administración que nos permiten modificar la configuración del dominio.
Configuraciones adicionales no-activas podrían residir en el repositorio de configuración, y podemos editarlas usando la Consola de Administración. El repositorio de configuración consta de una serie de subdirectorios (al menos uno) bajo el directorio `/config`. Todo dominio está definido en un fichero **config.xml** distinto que reside en un subdirectorio con el mismo nombre que el dominio. Para acceder a las configuraciones no-activas, seguimos el enlace **Domain Configurations** en la página de bienvenida de la Consola de Administración cuando arrancamos la consola.

Arrancar la Consola de Administración

La Consola de Administración es una aplicación Web que usa Java Server Pages (JSPs) para acceder a recursos de control en el Servidor de Administración. Después de arrancar el Servidor de Administración (ver Arrancar y Parar Servidores WebLogic), podemos arrancar la Consola de Administración dirigiendo nuestro navegador a la siguiente URL:

```
http://hostname:port/console
```

El valor de `hostname` es el nombre o dirección IP del Servidor de Administración y `port` es la dirección del puerto en el que está escuchando peticiones el Servidor de Administración (7001 por defecto). Si arrancamos el Servidor de Administración usando Secure Socket Layer (SSL), deberíamos añadir una **s** después de **http** de esta forma:

```
https://hostname:port/console
```

Si tenemos nuestro navegador configurado para enviar peticiones HTTP a un servidor proxy, podríamos tener que configurar nuestro navegador para no enviar las peticiones HTTP del Servidor de Administración hacia el proxy. Si el Servidor de Administración está en la misma máquina que el navegador, podríamos asegurarnos de que la peticiones enviadas a `localhost` o `127.0.0.1` no se envían al proxy.

El panel izquierdo de la Consola de Administración contiene un árbol para navegar por tablas de datos, páginas de configuración, y logs de acceso. Seleccionando (es decir, haciendo click con el botón izquierdo) sobre un ítem en el árbol de dominio, podemos mostrar una tabla de datos con los recursos de un tipo particular (como WebLogic Servers) o páginas de configuración y monitorización de un recurso seleccionado. Los nodos de más alto nivel en el árbol son contenedores. Si estos contenedores tienen nodos hojas, podemos pulsar sobre el signo (+) de la izquierda para expandir el árbol y acceder a los nodos hojas.

Las tablas de entidades -- tablas de datos sobre recursos de un tipo particular -- pueden optimizarse añadiendo o eliminando columnas que muestren valores de atributos. Podemos personalizar una tabla siguiendo el link `Customize this table` que hay en la parte superior de la tabla. Cada columna de la tabla corresponde con un atributo que ha sido seleccionado para ser incluido en la tabla.

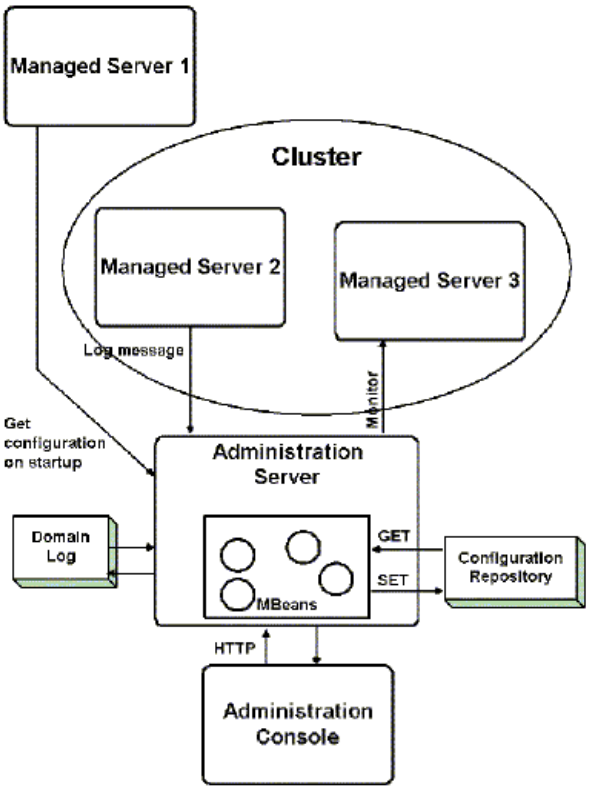
Cuando se arranca, la Consola de Administración pide una password. La primera vez que se arranca, podemos usar el nombre de usuario y la password bajo las que se arrancó el Servidor de Administración. Podemos usar la Consola de Administración para crear una lista de usuarios con privilegios de administración. Una vez designados, estos usuarios pueden también realizar tareas administrativas mediante la Consola de Administración.

Objetos de Configuración de Tiempo de Ejecución

El Servidor de Administración está lleno con objetos como los JavaBeans llamados **Management Beans** (**MBeans**), que están basados en el estándar Java Management Extension (JMX) de Sun. Estos objetos proporcionan control de acceso a recursos de dominio. El Servidor de Administración contiene **MBeans** de configuración y **MBeans** de tiempo de ejecución. Los **MBeans** de administración proporcionan métodos de acceso `SET` (escribir) y `GET` (leer) a los atributos de configuración.

Los **MBeans** de tiempo de ejecución proporcionan un poco de información sobre los recursos del dominio, como su sesión HTTP actual o la carga de un almacén de conexiones JDBC. Cuando un recurso particular del dominio (como una aplicación Web) es ejemplarizado, se crea un ejemplar `MBean` que recoge información sobre ese recurso.

Cuando accedemos a la página de monitorización de un recurso particular en la Consola de Administración, el Servidor de Administración realiza una operación `GET` para recuperar los valores del atributo actual



El Servidor de Administración nos permite cambiar dinámicamente los atributos de configuración de los recursos de dominio, es decir, mientras se está ejecutando WebLogic Server. Para muchos atributos, no necesitamos re-arrancar los servidores para que nuestros cambios tengan efecto. En este caso, un cambio en la configuración se refleja tanto en el valor de tiempo de ejecución actual del atributo, como en el valor persistentemente almacenado en el fichero de configuración.
Además de la Consola de Administración basada en Web, WebLogic Server proporciona una utilidad de la línea de comandos para acceder a la configuración y monitorizar los atributos de recursos de dominio. Esta herramienta se proporciona para aquellos que quieren crear scripts para automatizar el control del sistema.

Punto Central para Acceder a los Mensajes Log

El Servidor de Administración también soporta acceso central para los mensajes críticos del sistema de todos los servidores mediante el **log** de dominio. JMX proporciona una facilidad para reenviar los mensajes a entidades que se subscriben para mensajes específicos. Las entidades subscriptoras especifican qué menajes reenviar proporcionando un filtro que selecciona los mensajes de su interés. Un mensaje reenviado a otras entidades de la red sobre la iniciativa de un WebLogic Server local se llama una **notificación**. Las notificaciones JMX se usan para reenviar mensajes críticos de log desde todos los servidores WebLogic Servers del dominio hacia el Servidor de Administración. Cuando arranca un Servidor WebLogic Controlado, el Servidor de Administración lo registra en el log de mensajes críticos. Dichos mensajes son almacenados en el **log** de dominio. El Servidor de Administración registra un simple filtro log con cada servidor WebLogic para seleccionar los mensajes a reenviar. Podemos cambiar el filtro de log del dominio, la vista del log de dominio, y la vista del log del servidor local usando la Consola de Administración.

Arrancar y Parar WebLogic Server

El Servidor de Administración y los Servidores Controlados de WebLogic

Un **dominio** WebLogic podría consistir en uno o más Servidores WebLogic. Uno (y no más de uno) de ellos debe ser el Servidor de Administración del dominio. Los servidores WebLogic adicionales del dominio son Servidores **Controlados**. El mismo ejecutable podría arrancar un Servidor de Administración WebLogic y un Servidor Controlado WebLogic.
Siendo el de Servidor de Administración el papel por defecto para un Servidor WebLogic. Por lo tanto, si sólo hay un Servidor WebLogic en un dominio, este servidor es el Servidor de Administración. En un dominio multi-servidor, un Servidor WebLogic se convierte en un Servidor Controlado sólo si se le instruyó para obtener su configuración desde un Servidor de Administración que se está ejecutando en el momento de su arrancada.
El Servidor de Administración controla los accesos a la configuración de un dominio WebLogic y proporciona otros servicios de control como la monitorización y la navegación por los mensajes de log. El Servidor de Administración sirve a la Consola de Administración que proporciona acceso al usuario para manejar los servicios ofrecidos por el Servidor de Administración. Cuando se arranca un Servidor WebLogic Controlador, obtiene su configuración desde el Servidor de Administración. Por esta razón, arrancar un domino WebLogic multi-servidor es un proceso de dos pasos: Primero arrancamos el Servidor de Administración, y luego arrancamos los Servidores Controlados.

Nota:
Los Servidores Controlados deben ser de la misma versión WebLogic que el Servidor de Administración

Mensajes de Error en Arrancada

Cuando un Servidor WebLogic está arrancando, el sistema de log normal todavía no está disponible. Según esto, cualquier error encontrado durante la arrancada se muestra por **stdout** y un fichero log especial de arrancada llamado `servername-startup.log` (donde `servername` es el nombre del servidor). Si la arrancada tiene éxito, el último mensaje de este log es la localización del fichero log del servidor donde ocurren los logs normales.

Arrancar el Servidor de Administración WebLogic

Hay varias formas en las que poder arrancar el Servidor de Administración WebLogic:

- Desde la línea de comandos: El comando para arrancar el Servidor WebLogic puede teclearse en un shell de comandos manualmente, o puede colocarse en un script para evitar tener que teclear el comando cada vez que se arranca el servidor.
- Desde el menú Inicio (sólo Windows)
- Un servidor WebLogic instalado como un servicio Windows arrancará automáticamente cuando se rebote el ordenador.

Uso de Password para Arrancar el Servidor de Administración

Durante la instalación se nos pidió que especificáramos una password que sería requerida cuando se arrancara el servidor. Si usamos los scripts **start** para arrancar el Servidor de Administración o un Servidor Controlado, podemos incluir la password en la línea de comandos (ver Arrancar el Servidor de Administración WebLogic desde la Línea de Comandos. Si arrancamos el servidor usando un script sin haber especificado la password como un argumento de la línea de comandos, se nos pedirá que la introduzcamos si no hay un fichero **password.ini**.

Arrancar el Servidor de Administración WebLogic desde el Menú Inicio

Si instalamos el servidor WebLogic sobre Windows con el programa de instalación de Bea, podemos usar el acceso directo al Servidor WebLogic del menú de Inicio para arrancar el Servidor de Administración WebLogic. Seleccionamos:
`Start --> Programs --> BEA WebLogic E-Business Platform --> Weblogic Server Version --> Start Default Server`
donde `version` es el número de versión del software WebLogic Server.
Llamar al Servidor WebLogic desde el Menú de Inicio ejecuta el script de arrancada **startWeblogic.cmd** (que está en `install_dir/config/domain_name` donde `domain_name` es el nombre del dominio e `install_dir` es el directorio donde instalamos el software WebLogic Server.

Arrancar y Parar el Servidor de Administración WebLogic como un Servicio Windows

Cuando lo instalamos como un servicio Windows, el Servidor WebLogic arranca automáticamente cuando arranca el ordenador Windows. El servidor WebLogic se arranca ejecutando el script **startWeblogic.cmd**. Un Servidor WebLogic arrancado de esta forma es un Servidor de Administración.
Para ejecutar el Servidor WebLogic como un servicio windows, debemos haberlo instalado como tal. Para información sobre cómo instalar y eliminar el Servidor WebLogic como un servicio Windows, puedes ver Configurar el Servidor WebLogic como un Servicio Windows. También podemos parar y arrancar fácilmente el Servidor WebLogic desde el **Service Control Panel**:

1. Seleccionamos `Start --> Settings --> Control Panel`.
2. Hacemos doble click en el **Services Control Panel** para abrirlo.
3. Dentro de él, buscamos WebLogic Server. Si está arrancado, tenemos la opción de pararlo, cuando lo seleccionamos, pulsamos el botón **Stop** de la derecha. Si WebLogic está parado, se activará el botón **Start**.

Podemos hacer el servicio de Windows como Automático, Manual, o Desactivado pulsando el botón **Startup** y seleccionando un modo.

Arrancar el Servidor de Administración WebLogic desde la Línea de Comandos

El Servidor WebLogic es un fichero class Java, y como cualquier aplicación Java, puede arrancarse con el comando **java**. Los argumentos necesarios para arrancar el Servidor WebLogic desde la línea de comandos pueden ser bastante largos y teclearlos cada vez que necesitamos arrancar el servidor puede ser tedioso. Para asegurarnos de que los comandos de arrancada son seguros, BEA System recomienda que incorporemos los comandos en un script que podremos utilizar siempre que queramos arrancar un Servidor WebLogic.
Se necesitan los siguientes argumentos cuando se arranca el Servidor de Administración WebLogic desde la línea de comandos java:

- Especificamos unos valores máximo y mínimo para la pila de memoria Java. Por ejemplo, podríamos querer arrancar el Servidor WebLogic con una asignación por defecto de 64 megabytes de pila de memoria Java para el Servidor WebLogic. Para hacer eso, podemos arrancar el servidor con las opciones java `-ms64m` y `-mx64m`.
  Los valores asignados a estos parámetros puedes afectar dramáticamente al rendimiento de nuestro Servidor WebLogic y sólo se proporcionan aquí como valores por defecto generales. En un entorno de producción deberíamos

considerar cuidadosamente el tamaño correcto de la pila de memoria a utilizar por nuestras aplicaciones y entorno.

- Seleccionar la opción `-classpath`.
  El contenido mínimo para esta opción se describe bajo la sección Seleccionar el Classpath
- Especificar el nombre del servidor.
  La configuración de dominio especifica la configuración por nombre de servidor. Para especificar el nombre del servidor en la línea de comandos, usamos el siguiente argumento:
- `-Dweblogic.Name=servername`

  Por defecto el valor es `myserver`.

- Proporcionamos la password de usuario:
  El usuario por defecto es **system** y la password requerida es la password que especificamos durante la instalación. Para introducir la password, incluimos el siguiente argumento:
- `-Dweblogic.management.password=password`
- Especificamos el directorio raíz de WebLogic si no estamos arrancando el Servidor WebLogic desde su directorio raíz.
  El directorio raíz contiene los recursos de seguridad para el dominio y el Repositorio de Configuración (un directorio llamado `\config`). Podemos especificar la localización del directorio raíz en la línea de comandos con el argumento:
- `-Dweblogic.RootDirectory=path`

  Donde `path` es el path al directorio raíz. Si no especificamos este atributo en la línea de comandos, se usará el directorio actual para seleccionar el valor de este atributo en tiempo de ejecución.

- Especificar la localización del directorio `bea.home`:
- `-Dbea.home=root_install_dir`
  Donde `root_install_dir` es el directorio raíz bajo el que instalamos el software del BEA WebLogic Server.

- Si generamos una clave privada protegida por password, necesitamos pasar al servidor la password de la clave privada en la arrancada para que el servidor pueda desencriptar el fichero de la clave privada PKCS. Para pasar la password de la clave privada al servidor en la arrancada incluimos el siguiente argumento en la línea de comandos:
- `-Dweblogic.pkpassword=pkpassword`
  Donde `pkpassword` es la password de la clave privada.
  Las claves privadas protegidas por password se generan cuando se especifica el campo **Private Key Password** en el servlet **Certificate Request Generator**. Para más información puedes ver Manejar la Seguridad

- Podemos especificar el nombre del dominio de configuración cuando arrancamos el Servidor de Administración usando el siguiente argumento de la línea de comandos:

- `-Dweblogic.Domain=domain_name`

  Donde `domain_name` es el nombre del dominio. También puede ser el subdirectorio donde está el fichero de configuración que se utilizará para arrancar el dominio.

El repositorio de configuración consta de los dominios bajo el directorio `/config`. El repositorio de configuración podría contener una gran variedad de posibles configuraciones de dominio. Cada uno de esos dominios está localizado bajo un subdirectorio separado. Cuando especificamos `domain_name` estamos especificando el nombre del subdirectorio. El subdirectorio que especifica el fichero de configuración XML (**config.xml**) y los recursos de seguridad para ese dominio (ver ejemplo abajo). El fichero **config.xml** especifica la configuración para ese dominio.
La configuración de domino con la que se arrancó el Servidor de Administración se convierte en el dominio activo. Sólo un dominio puede estar activo.

- También podemos especificar valores para los atributos de configuración WebLogic en la línea de comandos. Estos valores se convierten en los valores de tiempo de ejecución para ese atributo, y cualquier valor almacenado en la configuración persistente es ignorado. El formato para seleccionar un valor en tiempo de ejecución para un atributo en la línea de comandos es:
- `-Dweblogic.attribute=value`

Seleccionar la Opción Classpath

Se debe incluir lo siguiente como argumentos de la opción `-classpath` de la línea de comandos:

- `/weblogic/lib/weblogic_sp.jar`
- `/weblogic/lib/weblogic.jar`
- WebLogic Server viene con una versión de prueba para todos los sistemas de control de bases de datos Java (DBMS) llamado Cloudscape. Si vamos a usar este DBMS, necesitaremos incluir en el classpath:
- `/weblogic/samples/eval/cloudscape/lib/cloudscape.jar`
- Si vamos a usar **WebLogic Enterprise Connectivity**, necesitaremos incluir lo siguiente:
- `/weblogic/lib/poolorb.jar`

  Donde `weblogic` es el directorio donde instalamos WebLogic Server.

Arrancar el Servidor de Administración WebLogic usando un Script

Con la distribución WebLogic se proporcionan algunos scripts de ejemplo que podemos usar para arrancar servidores WebLogic. Necesitaremos modificar estos scripts para ajustarlos a nuestro entorno y nuestras aplicaciones. Se proporcionan scripts separados para arrancar el Servidor de Administración y Servidores Controlados. Los scripts para arrancar el Servidor de Administración se llama **startWebLogic.sh** (UNIX) y **startWeblogic.cmd** (Windows). Estos scripts están localizados en el subdirectorio de configuración de nuestro dominio.
Para usar los scripts suministrados debemos:

- Prestar atención a las selecciones del classpath y los nombres de los directorios.
- Cambiar el valor de la variable JAVA_HOME a la localización de nuestro JDK.
- Los usuarios de UNIX deben cambiar los permisos del script de ejemplo UNIX para hacerlo ejecutable. Por ejemplo:
- `chmod +x startAdminWebLogic.sh`

Reiniciar el Servidor de Administración cuando se están ejecutando Servidores Controlados

Para un sistema de producción típico se recomienda que las aplicaciones que contienen la lógica de negocio crítica se desplieguen sobre Servidores Controlados. En dicho escenario, el rol del Servidor de Administración es el de configuración y monitorización de los Servidores Controlados. Si el Servidor de Administración deja de estar disponible en dicha configuración, las aplicaciones que se están ejecutando en los Servidores Controlados pueden continuar procesando las peticiones de los clientes.
Cuando se arranca el Servidor de Administración, hace una copia del fichero de configuración que se utilizó al arrancar el dominio activo. Esto se graba en el fichero `install_dir/config/domain_name/config.xml.booted` donde `install_dir` es el directorio donde instalamos el software WebLogic Server y `domain_name` es el nombre del dominio. El Servidor de Administración crea el fichero **config.xml.booted** sólo después de que se haya completado satisfactoriamente su secuencia de arrancada y ya esté listo para procesar peticiones.
Deberíamos hacer una copia de este fichero para tener un fichero de configuración que funciona para poder revertir si necesitamos deshacer los cambios realizados en la configuración activa desde la Consola de Administración.
Si el Servidor de Administración se cae mientras los Servidores Controlados continúan en ejecución, no necesitamos re-arrancar los Servidores Controlados para recuperar el control del dominio. EL procedimiento para recuperar el control del dominio depende de si podemos o no re-arrancar el Servidor de Administración desde la misma máquina en la que se arrancó el dominio.

Reiniciar el Servidor de Administración desde la misma Máquina

Si re-arrancamos el Servidor de Administración WebLogic mientras los Servidores Controlados continúan su ejecución, el Servidor de Administración puede detectar la presencia de los Servidores Controlados que se están ejecutando si lo instruimos para que realice la búsqueda. Para hacer esto, introducimos el siguiente argumento en la línea de comandos cuando arrancamos el Servidor de Administración:

```
-Dweblogic.management.discover=true
```

El valor por defecto para este atributo es false. El fichero de configuración del dominio contiene un fichero **running-managed-servers.xml** que es una lista de los Servidores Controlados que conoce el Servidor de Administración. Cuando se le instruye al Servidor de Administración para que realice la búsqueda en la arrancada, usa esta lista para comprobar la presencia de los Servidores Controlados que están en ejecución.
Reiniciar el Servidor de Administración no actualiza la configuración en tiempo de ejecución de los Servidores Controlados para tener en cuenta cualquier cambio hecho en los atributos que sólo pueden configurarse estáticamente. Los Servidores WebLogic pueden re-arrancarse para que tengan en cuenta los cambios de los atributos de configuración estáticos. La búsqueda de Servidores Controlados permite al Servidor de Administración monitorizar los Servidores Controlados para hacer los cambios en tiempo de ejecución en los atributos que pueden ser configurados dinámicamente.

Reiniciar el Servidor de Administración desde otra Máquina

Si un crash de una máquina no nos deja re-arrancar el Servidor de Administración en la misma máquina, podemos recuperar el control de los Servidores Manejados en ejecución de esta forma:

1. Asignando a otro ordenador el mismo nombre de host que la máquina del Servidor de Administración averiada.
2. Instalando el software WebLogic Server sobre la nueva máquina de administración (si no está hecho ya).
3. El directorio `/config` (el repositorio de configuración) usado para arrancar el Servidor de Administración de la máquina que está siendo reemplazada necesita estar disponible para la nueva máquina. Podemos copiarlo desde un medio de backup o montando NFS, por ejemplo. Esto incluye el fichero de configuración (**config.xml**) usado para arrancar el dominio activo así como las aplicaciones y componentes instalados en el directorio `/applications` para ese dominio.

4. Re-arrancando el Servidor de Administración desde la nueva máquina con la adicción del siguiente argumento de la línea de comandos:
5. `-Dweblogic.management.discover=true`
   Este argumento forzará al Servidor de Administración a detectar la presencia de los Servidores Controlados que están en ejecución.

Añadir un Servidor Controlado a un Dominio

Antes de poder ejecutar un Servidor WebLogic como un Servidor Controlado, primero debemos crear una entrada para ese servidor en la configuración del dominio. Para hacer esto, realizamos lo siguiente:

1. Arrancamos el Servidor de Administración del dominio.
2. Llamamos a la Consola de Administración apuntando nuestro navegador a **http://hostname:port/console**, donde `hostname` es el nombre de la máquina donde se está ejecutando el Servidor de Administración y `port` es el número de puerto por el que está escuchando (por defecto es 7001)
3. Creamos una entrada para la máquina del servidor (`Machines --> Create a new machine`) (si es diferente de la máquina del Servidor de Administración) y el nuevo servidor (`Servers --> Create a new server`).

Para más información puedes ver Configurar Servidores y Clusters WebLogic.

Arrancar un Servidor WebLogic Controlado

Una vez que hemos añadido un Servidor WebLogic Controlado a nuestra Configuración (ver Añadir un Servidor Controlado a un Dominio), podemos arrancar el Servidor Controlado desde la línea de comandos **java**. El comando para arrancar el Servidor WebLogic puede teclearse en un shell de línea de comandos o puede situarse en un script para evitar tener que re-teclear la línea de comandos cada vez que arrancamos el servidor. Para información sobre los scripts de ejemplo puedes ver Arrancar un Servidor WebLogic Controlados usando Scripts.
La principal diferencia entre los parámetros de arrancada de un Servidor Manejado y de un Servidor de Administración es que necesitamos proporcionar un argumento que identifica la localización del Servidor de Administración al que el Servidor Controlador le pedirá su configuración. Un Servidor WebLogic arrancado sin ese parámetro se ejecuta como un Servidor de Administración.
Se requiere lo siguiente cuando arrancamos un Servidor WebLogic Controlado:

- Especificamos unos valores máximo y mínimo para la pila de memoria Java. Por ejemplo, podríamos querer arrancar el Servidor WebLogic con un asignación por defecto de 64 mega bites de pila de memoria Java para el Servidor WebLogic. Para hacer eso, podemos arrancar el servidor con las opciones java `-ms64m` y `-mx64m`.
  Los valores asignados a estos parámetros puedes afectar dramáticamente al rendimiento de nuestro Servidor WebLogic y sólo se proporcionan aquí como valores por defecto generales. En un entorno de producción deberíamos considerar cuidadosamente el tamaño correcto de la pila de memoria a utilizar por nuestras aplicaciones y entorno.
- Seleccionar la opción `-classpath`.
  El contenido mínimo para esta opción se describe bajo la sección Seleccionar el Classpath
- Especificar el nombre del servidor.
  Cuando un Servidor WebLogic Controlado pide su información de configuración al Servidor de Administración, se identifica mediante su nombre

de servidor. Esto permite al Servidor de Administración responder con la configuración apropiada. Por esta razón, también debemos seleccionar el nombre del servidor cuando se arranca el Servidor Controlado. Esto puede especificarse añadiendo el siguiente argumento a la línea de comandos:

- `-Dweblogic.Name=servername`
- Proporcionar un nombre de usuario y una password.
  El usuario por defecto es **system** y la password requerida es la password que definimos durante la instalación. Para arrancar el Servidor WebLogic Controlado como un usuario particular, usamos el siguiente argumento de la línea de comandos:
- `-Dweblogic.management.username=username`

Para introducir la password para el usuario especificado, incluimos el siguiente argumento:

```
-Dweblogic.management.password=password
```

Para más información sobre el uso de passwords, puedes ver Uso de Passwords Cuando se Arranca el Servidor WebLogic.

- Especificar la localización del directorio `bea.home`:
- `-Dbea.home=root_install_dir`

Donde `root_install_dir` es el directorio raíz bajo el que instalamos el software del BEA WebLogic Server.

- Si queremos arrancar el servidor con el protocolo Secure Socket Layer (SSL), necesitamos pasarle al servidor la password de la clave privada en la arrancada para que el servidor pueda desencriptar el fichero de clave privada SSL. Para pasar la password de la clave privada SSL al servidor en la arrancada incluimos el siguiente argumento en la línea de comandos:
- `-Dweblogic.pkpassword=pkpassword`

Donde `pkpassword` es la password de la clave privada SSL.

- Especificar el nombre de host y el puerto de escucha del Servidor de Administración WebLogic.
  Cuando arrancamos un Servidor Controlado, es necesario especificar el nombre de host y el puerto de escucha del Servidor de Administración al cual solicitará su configuración. Esto puede especificarse añadiendo el siguiente argumento a la línea de comandos cuando arrancamos el servidor controlado:
- `-Dweblogic.management.server=host:port`

  ó
- `-Dweblogic.management.server=http://host:port`

Donde `host` es el nombre o dirección IP de la máquina donde se está ejecutando el Servidor de Administración y `port` es el puerto de escucha del Servidor de Administración (por defecto 7001).

Si estamos usando Secure Socket Layer (SSL) para comunicarnos con el Servidor de Administración, deberemos especificarlo como:

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

Para usar el protocolo SSL en comunicaciones entre los Servidores Controlados y el Servidor de Administración, necesitamos habilitar SSL en el Servidor de Administración.

Nota:
Cualquier Servidor WebLogic que sea arrancado sin especificar la localización del Servidor de Administración se arranca como un Servidor de Administración.

Nota:
Como el Servidor Controlado recibe su configuración desde el Servidor de Administración, el Servidor de Administración especificado debe estar en el mismo dominio que el Servidor Controlado.

- También podemos especificar valores para los atributos de configuración WebLogic en la línea de comandos. Estos valores se convierten en los valores de tiempo de ejecución para ese atributo, y cualquier valor almacenado en la configuración persistente es ignorado. El formato para seleccionar un valor en tiempo de ejecución para un atributo en la línea de comandos es:
- `-Dweblogic.attribute=value`

Arrancar un Servidor WebLogic Controlado usando Scripts

Con la distribución WebLogic se proporcionan algunos scripts de ejemplo que podemos usar para arrancar servidores WebLogic. Necesitaremos modificar estos scripts para ajustarlos a nuestro entorno y nuestras aplicaciones. Los scripts para arrancar un Servidor Controlado se llama **startManagedWebLogic.sh** (UNIX) y **startManagedWeblogic.cmd** (Windows). Estos scripts están localizados en el subdirectorio de configuración de nuestro dominio.
Para usar los scripts suministrados debemos:

- Prestar atención a las selecciones del classpath y los nombres de los directorios.
- Cambiar el valor de la variable JAVA_HOME a la localización de nuestro JDK.
- Los usuarios de UNIX deben cambiar los permisos del script de ejemplo UNIX para hacerlo ejecutable. Por ejemplo:
- `chmod +x startManagedWebLogic.sh`

Hay dos formas de arrancar el Servidor Controlado usando el script:

- Si seleccionamos el valor de las variables de entorno `SERVER_NAME` y `ADMIN_URL`, no necesitamos proporcionarlas como argumentos cuando llamemos al script de arrancada.

`SERVER_NAME` debería contener el nombre del Servidor WebLogic Controlado que deseamos arrancar. `ADMIN_URL` debería apuntar al host (nombre de host o dirección IP) y el número de puerto donde está escuchando el Servidor de Administración (por defecto es 7001). Por ejemplo:

```
set SERVER_NAME=bigguy
set ADMIN_URL=peach:7001
startManagedWebLogic
```

- Podemos invocar el script de arrancada y pasarle el nombre del Servidor Controlado y la URL del Servidor de Administración en la línea de comandos:
- `startManagedWebLogic server_name admin:url`

  Donde `server_name` es el nombre del Servidor Controlado que estamos arrancando y `admin_url` es `http://host:port` o `https://host:port` donde `host` es el nombre de host (o dirección IP) del Servidor de Administración y `port` es el número de puerto para el Servidor de Administración.

Migrar desde Versiones Anteriores de WebLogic Server

Si tenemos los scripts de arrancada que usamos en versiones anteriores de WebLogic Server, necesitaremos modificarlos para que funcionen con esta version. Aquí tenemos una lista de los cambios principales:

- La carga dinámica de clases ha cambiado.
  Las versiones anteriores de WebLogic Server tenían dos selecciones de classpath en la línea de comandos:
  - El classpath del sistema Java
  - Una classpath especial de WebLogic

  La propiedad del classpath WebLogic se usaba para facilitar la carga dinámica de clases. En esta versión, la propiedad classpath de WebLogic ha sido eliminada y ha cambiado la selección del classpath del sistema Java. Los scripts usados con versiones anteriores necesitan ser modificados de forma correcta. En esta versión, la carga dinámica de clases necesaria para las aplicaciones Java 2 es responsabilidad de dichas aplicaciones, y la especificación de las clases compiladas se consigue mediante descriptores de Extensible Markup Language (XML) en los ficheros comprimidos de la aplicación.

Puedes encontrar más información en Seleccionar el Classpath Java.

- Ya no es necesario especificar la localización del fichero de licencia ni del fichero de política en la línea de comandos.
- La distinción entre Servidor de Administración y Servidor Controlado es nueva en esta versión. Necesitaremos añadir la URL que apunta a un Servidor de Administración en ejecución si queremos arrancar un servidor como un Servidor Controlado.
- Puedes ver una lista completa de argumentos en Arrancar el Servidor de Administración desde la Línea de Comandos o Arrancar un Servidor Controlado
- Se proporcionan nuevos scripts de arrancada **startManagedWebLogic.cmd** (Windows) y **startManagedWebLogic.sh** (UNIX), para arrancar Servidores WebLogic como Servidores Controlados. Los scripts **startWebLogic.sh** (UNIX) y **startWebLogic.cmd** (Windows) ahora sólo se usan para arrancar el Servidor de Administración.

Parar un Servidor WebLogic desde la Consola de Administración

Para parar un Servidor WebLogic individual:

- En el árbol de dominios de la Consola de la Administración (en el panel izquierdo), seleccionamos el servidor que queremos parar.
- En la página de la pestaña `Monitoring --> General`, seleccionamos el enlace **Shutdown this server**.

Parar un Servidor desde la Línea de Comandos

También podemos parar un Servidor WebLogic desde la línea de comandos con el siguiente comando:

```
Java weblogic.Admin -url host:port SHUTDOWN -username adminname
  -password password
```

Donde:

- `host` es el nombre o dirección IP de la máquina donde se está ejecutando el Servidor WebLogic.
- `port` es el puerto de escucha del servidor WebLogic (por defecto es 7001).
- `adminname` designa al usuario que tiene privilegios de administrador para el servidor WebLogic destino (por defecto es **system**).
- `password` es la password para `adminname`.

Suspender y Resumir un Servidor Controlado

La Consola de Administración nos permite suspender un Servidor WebLogic Controlado. Cuando se suspende un Servidor Controlado, éste sólo acepta peticiones desde el Servidor de Administración. Un uso típico para esta característica sería en una situación en la que un Servidor WebLogic se está ejecutando como backup "en caliente" de otro servidor. El servidor backup se podría mantener en estado suspendido hasta que queramos que procese peticiones.

Nota:
Al suspender un Servidor WebLogic sólo se suspenden las respuestas a peticiones HTTP. Las aplicaciones Java o las invocaciones RMI no son suspendidas.

Para suspender un Servidor WebLogic Controlado:

- En el árbol de dominio de la Consola de Administración (en el panel izquierdo), seleccionamos el servidor que queremos suspender.
- Sobre la ventana de la pestaña `Monitoring --> General`, seleccionamos el enlace **Suspend this server**.

Para decirle al Servidor Controlador que vuelva a procesar peticiones de clientes:

- En el árbol de dominio de la Consola de Administración (en el panel izquierdo), seleccionamos el servidor que queremos re-activar.
- Sobre la ventana de la pestaña `Monitoring --> General`, seleccionamos el enlace **Resume this server**.

Configurar un Servidor WebLogic como un Servicio Windows

Podemos ejecutar el Servidor WebLogic como un servicio Windows. Cuando se instala como un servicio Windows, el Servidor WebLogic arranca automáticamente cuando lo hace el ordenador Windows. Un Servidor WebLogic se arranca de esta forma invocando al script de arrancada **startWeblogic.cmd**. Si el Servidor WebLogic se arranca como un Servidor de Administración o como un Servidor Controlado depende de los parámetros de la línea de comandos java que invoca al Servidor WebLogic. Puedes ver una lista completa de argumentos en Arrancar el Servidor de Administración desde la Línea de Comandos o Arrancar un Servidor Controlado.
Para configurar un Servidor WebLogic para que se ejecute como un servicio Windows o para que deje de hacerlo, debemos tener privilegios de administrador. Para instalar el Servidor WebLogic como un servicio Windows, hacemos lo siguiente:

1. Movernos al directorio `weblogic\config\mydomain` (donde `weblogic` es el directorio donde instalamos WebLogic Server y `mydomain` es el subdirectorio con nuestra configuración de dominio).
2. Ejecutamos el script **installNTService.cmd**.

Eliminar WebLogic Server como un Servicio Windows

Para eliminar el Servidor WebLogic como un servicio Windows, hacemos lo siguiente:

1. Movernos al directorio `weblogic\config\mydomain` (donde `weblogic` es el directorio donde instalamos WebLogic Server y `mydomain` es el subdirectorio con nuestra configuración de dominio).
2. Ejecutamos el script **uninstallNTService.cmd**.

También podeos desinstalar el Servidor WebLogic como un servicio Windows desde el menú Inicio de Windows.

Cambiar Passwords de un Servidor Instalado como un Servicio Windows

Si instalamos el servidor por defecto como un servicio Windows, la password de **system** que introdujimos durante la instalación del software WebLogic se usará en la creación del servicio. Si posteriormente es modificada esa password, debemos hacer lo siguiente:

1. Desinstalar el Servidor WebLogic como un servicio Windows usando el script **uninstallNTService.cmd** (localizado en `install_dir/config/domain_name`.
2. El comando **installNTservice.cmd** contiene el siguiente comando:

```
rem *** Install the service
"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%
```

   Debemos añadirle el siguiente comando:
   -password:"your_password"
   Donde `your_password` es la nueva password.

8. Ejecutamos el script **installNTservice.cmd** modificado. Esto creará un nuevo servicio con la password actualizada.

El Programa de Servicio Windows de WebLogic (beasvc.exe)

Los scripts para instalar y eliminar un Servidor WebLogic como un servicio Windows llaman al programa WebLogic Windows Service, **beasvc.exe**. Se pueden instalar o eliminar múltiples instancias de WebLogic Server como un servicio Windows usando **beasvc.exe**.
Todas las configuraciones para múltiples servicios son almacenadas en el Registro de Windows usando un nombre de servicio diferente y bajo un ítem específico del servidor en:
`HKEY_LOCAL_MACHINE\SYSTEM\Current\ControlSet\Services`
Cuando arrancamos el servicio, se recogen las entradas del registro de Windows y se inicializa y arranca la JVM. Como cada servicio se instala independientemente de los otros, podemos instalar varios ejemplares de WebLogic Server para ejecutarlos como servicios de Windows, siempre que cada servicio tenga un nombre único.
beasvc.exe tiene disponibles las siguientes opciones:

- **–install**
  Instala un servicio específico.
- **–remove**
  Elimina un servicio específico.
- **–svcname: service_name**
  El nombre del servicio especificado por el usuario para ser instalado o eliminado.
- **–cmdline: java_cmdline_parameters**
  Los parámetros de la línea de comandos java a ser usados por la arrancada del Servidor WebLogic durante la arrancada como un servicio Windows.
- **–javahome: java_directory**
  El directorio raíz de la instalación de Java. La línea de comandos de arrancada será formateada añadiendo `\bin\java` a `java_directory`.
- **–execdir: base_dir**
  Directorio donde se ejecutará este comando de arrancada.
- **–extrapath: additional_env_settings**
  Selecciones de paths adicionales que serán añadidas al path aplicable a la ejecución de este comando.
- **–help**
  Imprime el uso del comando **beasvc.exe**.

Los sistemas Win32 tienen una limitación de 2K en la longitud máxima de la línea de comandos. Si la selección del classpath para la arrancada de un servicio Windows es muy larga, se podría exceder el límite de 2K. Con las versiones 1.2 o posteriores de la JVM de Sun Microsystems, podemos especificar un fichero que contenga el classpath usando la opción **@**. Podríamos usar esta opción con **beasvc.exe** como en el siguiente ejemplo:

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

Registrar las Clases StartUp y ShutDown

WebLogic proporciona unos mecanismos para realizar tareas siempre que un Servidor WebLogic arranca o se para amigablemente. Una clase `startup` es un programa Java que se carga y se ejecuta automáticamente cuando se arranca o re-arranca un Servidor WebLogic. Las clases `Startup` se cargan y ejecutan sólo después de que se hayan completado las tareas de inicialización.
Las clases **Shutdown** funcionan de la misma forma que las clases `startup`. Una clase `shutdown` se carga y se ejecuta automáticamente cuando se para un Servidor WebLogic desde la Consola de Administración o usando el comando **weblogic.admin shutdown**.
Para que nuestro Servidor WebLogic pueda usar las clases `startup` o `shutdown`, es necesario registrarlas, lo que podemos hacer desde la Consola de Administración.
Podemos registrar estas clases de la siguiente forma:

1. Accediendo a la tabla **Startup & Shutdown** desde el árbol del dominio (en el panel izquierdo) de la Consola de Administración. Esta tabla proporciona opciones para crear entradas para clases `shutdown` o `startup` en la configuración del dominio.
2. Proporcionando el nombre de la clase y los argumentos necesarios, si existen, en la página de la pestaña **Configuration** para la clases `startup` o `shutdown` que estamos añadiendo.

Configurar Servidores y Clusters WebLogic

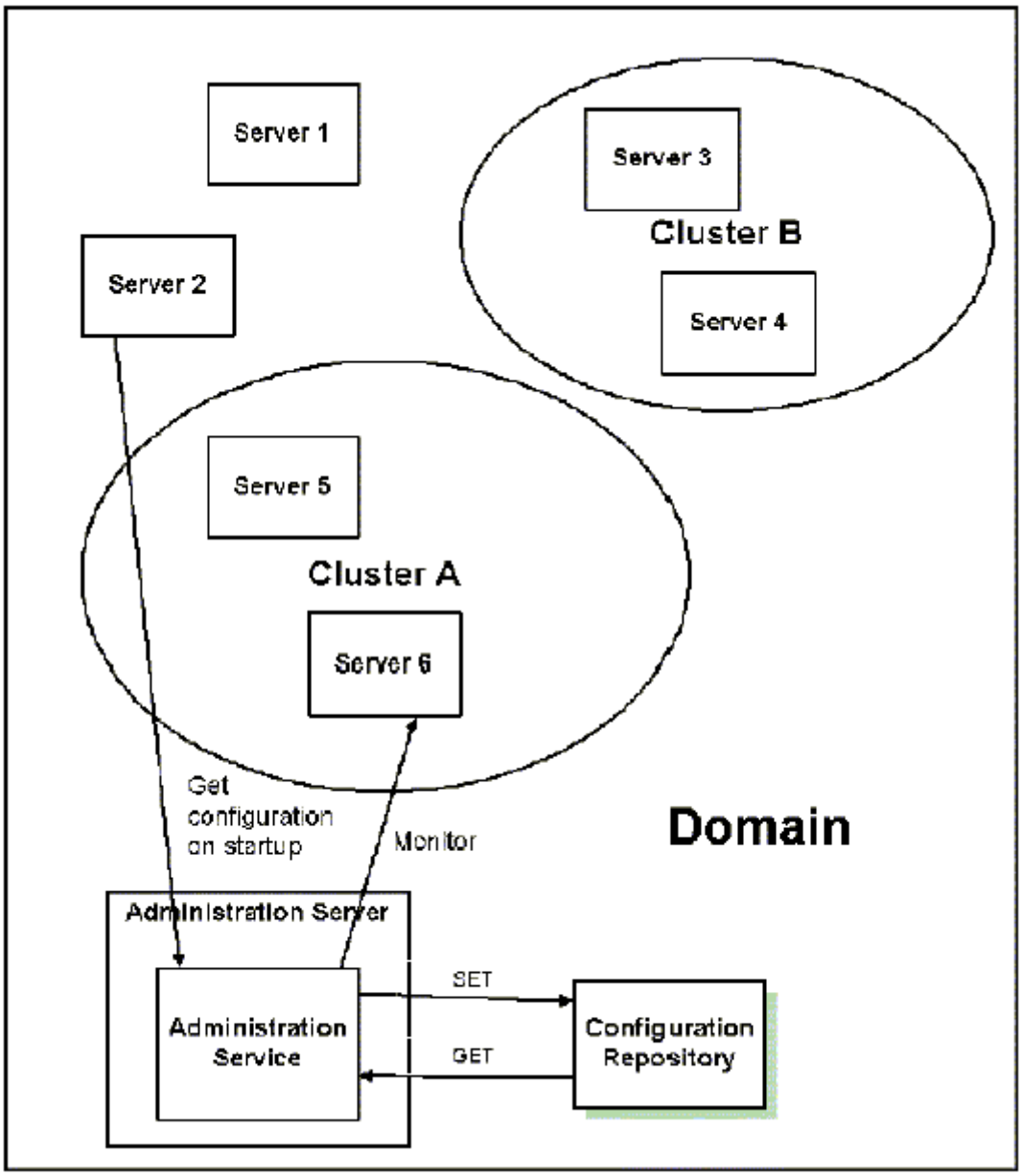
Introducción a la Configuración de Servidores y Clusters

La configuración persistente de un dominio de servidores y clusters WebLogic se almacena en un fichero de configuración XML. Podemos modificar este fichero de tres formas:

- A través de la Consola de Administración, el Interfase de Usuario de BEA para manejar y monitorizar la configuración de dominios. Está pensada como la forma principal para modificar o monitorizar la configuración de dominios.
- Escribiendo un programa que modifique los atributos de configuración, basado en el API proporcionado con WebLogic Server.
- Ejecutando la utilidad de la línea de comandos de WebLogic Server para acceder a los atributos de configuración de los recursos del dominio. Esto se proporciona para aquellos que quieren crear scripts para automatizar el control de los dominios.

Papel del Servidor de Administración

Con cualquier método que elijamos, el Servidor de Administración debe estar ejecutándose cuando modifiquemos la configuración del dominio.
El Servidor de Administración es el Servidor WebLogic en el que se ejecuta el Servicio de Administración. El Servicio de Administración proporciona la funcionalidad de un Servidor WebLogic, y controla la configuración de todo el dominio.
Por defecto, un ejemplar de servidor WebLogic es tratado como un Servidor de Administración. Cuando el Servidor de Administración arranca, carga los ficheros de configuración, que se almacenan, por defecto, en un directorio llamado `config` bajo el directorio **WEBLOGIC_HOME**. El directorio `config` tiene un subdirectorio por cada dominio que esté disponible en el Servidor de Administración. La configuración real reside en el directorio específico del dominio y se llama **config.xml**. Por defecto, cuando arranca un Servidor de Administración, busca el fichero de configuración (**config.xml**) bajo el directorio de dominio por defecto, que se llama `-mydomain`.
Cada vez que el Servidor de Administración arranca con éxito, se crea un fichero backup de la configuración llamado **config.xml.booted** en el directorio específico del dominio. En el caso de que el fichero **config.xml** se corrompiera durante el tiempo de vida del servidor, es posible volver a una buena configuración conocida anteriormente.
Un dominio podría consistir en un sólo Servidor WebLogic, que opera como Servidor de Administración.
Un entorno de producción típico contiene un Servidor de Administración y varios Servidores WebLogic. Cuando arrancamos los servidores de dicho dominio, primero se arranca el Servidor de Administración. Cuando se arranque cada servidor adicional, se le instruye para que contacte con el Servidor de Administración para obtener su información de configuración. De esta forma, el Servidor de Administración opera como una entidad de control central para la configuración de todo el dominio. No puede haber más de un Servidor de Administración activo en un dominio. Sólo el Servidor de Administración puede modificar los ficheros de configuración cuando se está ejecutando.



Arrancar la Consola de Administración

El punto principal de acceso al Servidor de Administración es a través de la Consola de Administración. Para abrir la Consola de Administración, completamos el siguiente procedimiento:

```
http://host:port/console
```

En esta URL, `host` es el nombre o dirección IP de la máquina en la que se está ejecutando el Servidor de Administración y `port` es el número de puerto por el que está escuchando el Servidor de Administración (por defecto 7001).

El sistema nos pedirá que introduzcamos un ID de usuario y una password. Introducimos nuestro ID y nuestra password. El sistema realiza la autentificación y chequea la autorización, verifica el ID de usuario y la password en una base de datos.

Si estamos autorizados a trabajar con la consola, se mostrará la consola en el modo de acceso que el administrador del sistema nos haya asignado originalmente: **ReadOnly** o **Read/Write**.

¿Cómo Funciona la Configuración Dinámica?

El Servidor WebLogic nos permite cambiar los atributos de configuración de los recursos de un domino dinámicamente, es decir, mientras los servidores se están ejecutando. En la mayoría de los casos no necesitamos re-arrancar el Servidor WebLogic para que los cambios tengan efecto. Cuando se reconfigura un atributo, el nuevo valor se ve inmediatamente reflejado en el valor actual en ejecución y en el valor almacenado del atributo en el fichero de configuración XML.
Sin embargo, hay excepciones. Si, por ejemplo, cambiamos el puerto de escucha de un Servidor WebLogic, la nueva dirección no se usará hasta la próxima vez que arranquemos el servidor afectado. En este caso, si modificamos el valor, estamos cambiando el valor almacenado en el fichero XML y el valor actual en ejecución del atributo podría ser diferente del valor almacenado. La Consola de Administración indica si los valores almacenado y actual de un atributo de configuración no son iguales usando un icono que cambia para alertarnos cuando el servidor necesita ser re-arrancado para que los cambios tengan efecto:

La consola hace un chequeo de validación sobre cada atributo que cambia el usuario. Los errores que son soportados son errores de fuera-de-rango y errores de tipo de datos erróneos. En ambos casos, se mostrará una ventana indicando al usuario que ha ocurrido un error.
Una vez que se ha arrancado la Consola de Administración, si otro proceso captura el puerto de escucha asignado al Servidor de Administración, deberíamos eliminar el proceso que ha capturado al servidor. Si no podemos eliminar ese proceso, debemos editar el fichero **config.XML** para cambiar el puerto de escucha asignado.

Planear una Configuración de Cluster

Cuando planeemos una configuración de cluster, debemos tener en mente las siguientes restricciones sobre el entorno de red y la configuración del cluster:

1. La(s) máquina(s) que usaremos como hosts WebLogic para el cluster deben tener asignadas direcciones IP estáticas. No podemos usar direcciones IP dinámicas en un entorno de clusters. Si los servidores están detrás de un firewall y los clientes están delante del firewall, cada servidor debe tener una dirección IP estática pública que pueda ser alcanzada por los clientes.
2. Todos los Servidores WebLogic de un cluster deben estar localizados en la misma red de área local (LAN) y deben poder ser alcanzados mediante IP multicast.
3. Todos los servidores de un cluster deben están ejecutando la misma versión de WebLogic Server.

Configuramos los Servidores de nuestro cluster para soportar la mezcla de servicios particulares que queremos ofrecer:

- Para los EJBs que usan conexiones JDBC, todos los servidores que desplieguen un EJB particular deben tener la misma configuración de despliegue y persistencia.
  Esto significa el mismo almacén de conexiones JDBC en cada servidor.
- Cada máquina que hospede servlets debe mantener la misma lista de servlets con idénticos ACLs (access control lists).
- Si nuestra aplicación cliente usa almacenes de conexiones JDBC directamente, debemos crear almacenes de conexiones idénticos (con idénticos ACLs) en cada

Servidor WebLogic. Esto significa que debe ser posible crear cualquier almacén de conexiones en uso en todas las máquinas del cluster. Si, por ejemplo, configuramos un almacén de conexiones para una base de datos Microsoft SQL Server sobre un servidor NT ejecutando WebLogic, no podemos usar ese almacén de conexiones en un cluster que tenga máquinas no-Windows (es decir, cualquier máquina que no pueda soportar una conexión Microsoft SQL Server).

- Otros detalles de configuración podrían diferir en varios miembros del cluster. Por ejemplo, podríamos configurar un servidor Solaris para procesar más peticiones login que una pequeña estación de trabajo NT. Dichas diferencias son aceptables. Así, en el ejemplo dado arriba, los atributos específicos del rendimiento de los miembros individuales del cluster podrían ser configurados con diferentes valores, mientras que la configuración de servicio para todos los miembros es idéntica. En la práctica, esto resulta frecuentemente en que los Servidores WebLogic de un cluster están configurados idénticamente en todas las áreas que tienen que ver con los servicios WebLogic, ficheros de clases, recursos externos (como bases de datos).

Tareas de Configuración de un Servidor

Las tareas de configuración de servidor que pueden realizarse desde la Consola de Administración incluyen:

- **Configurar un servidor individual** usando el nodo Server de la Consola de Administración. Los atributos que se pueden modificar usando este nodo incluyen el Nombre, el puerto de escucha y la dirección IP del servidor.
- **Clonar un servidor individual** usando el nodo Server de la Consola de Administración. El servidor individual es clonado, manteniendo los valores de atributos del servidor original y el nombre del nuevo servidor se selecciona en la parte `Configuration` del nodo Server.
- **Borrar un servidor** usando el nodo Server de la Consola de Administración. Pulsamos la confirmación de borrado del servidor.
- **Ver un Log de servidor** usando el nodo Server de la Consola de Administración. Seleccionamos el servidor que queremos monitorizar. Pulsamos la pestaña `Monitoring`. Pulsamos en el enlace `View Server Log` y monitorizaremos el servidor en el panel derecho de la Consola de Administración.
- **Ver un árbol JNDI de servidor** usando el nodo Server de la Consola de Administración. Pulsamos sobre el servidor que queremos monitorizar. Pulsamos la pestaña `Monitoring`. Pulsamos sobre el enlace `View JNDI Tree` y veremos el árbol en el panel derecho de la Consola de Administración.
- **Ver las Colas de Ejecución de servidor** usando el nodo Server de la Consola de Administración. Pulsamos sobre el servidor que queremos monitorizar. Pulsamos sobre el enlace `Execute Queues` y veremos la tabla en el panel derecho de la Consola de Administración.
- **Ver los threads que ejecuta un servidor** usando el nodo Server de la Consola de Administración. Pulsamos sobre el servidor que queremos monitorizar. Pulsamos sobre el enlace `Execute Threads` y veremos la tabla en el panel derecho de la Consola de Administración.

- **Ver los Sockets del servidor** usando el nodo Server de la Consola de Administración. Pulsamos sobre el servidor que queremos monitorizar. Pulsamos sobre el enlace `View Sockets` y veremos la tabla en el panel derecho de la Consola de Administración.
- **Ver las conexiones del servidor** usando el nodo Server de la Consola de Administración. Pulsamos sobre el servidor que queremos monitorizar. Pulsamos sobre el enlace `View Connections` y veremos la tabla en el panel derecho de la Consola de Administración.
- **Forzar la recolección de basura sobre un servidor** usando el nodo Server de la Consola de Administración. Pulsamos sobre el servidor que queremos monitorizar. Pulsamos sobre la pestaña `JVM`. Pulsamos sobre el enlace `Force Garbage Collection`. Aparecerá un diálogo para confirmar que ha tenido lugar la recolección de basura.
- **Monitorizar la seguridad del servidor** usando el nodo Server de la Consola de Administración. Pulsamos sobre el servidor que queremos monitorizar. Pulsamos sobre la pestaña `Monitoring`. Pulsamos la pestaña `Security`. Se mostrará la información de seguridad.
- **Ver la versión del servidor** usando el nodo Server de la Consola de Administración. Pulsamos sobre el servidor que queremos monitorizar. Pulsamos sobre la pestaña `Version`. Se mostrarán los datos de versión de este servidor.
- **Monitorizar los clusters del servidor** usando el nodo Server de la Consola de Administración. Pulsamos sobre el servidor que queremos monitorizar. Pulsamos la pestaña `Cluster`. Se mostrarán los datos de cluster de este servidor.
- **Desplegar EJBs en un servidor** usando el nodo Server de la Consola de Administración. Pulsamos sobre el servidor en el que queremos desplegar los EJBs. Pulsamos sobre el EJB que queremos desplegar y usamos el control `move` para moverlo a la columna `Chosen`. Pulsamos `Apply` para grabar las selecciones.
- **Monitorizar todos los EJBs desplegados en un servidor** usando el nodo Server de la Consola de Administración. Pulsamos sobre el servidor del que queremos monitorizar los EJBs. Pulsamos el enlace `Monitor All EJB Deployments` para mostrar la tabla de EJB desplegados.
- **Desplegar componentes de una aplicación web en un servidor** usando el nodo Server de la Consola de Administración. Pulsamos sobre el servidor en el que queremos desplegar la aplicación web. Seleccionamos la aplicación web que queremos desplegar y usamos el control `move` para moverlo a la columna `Chosen`. Pulsamos `Apply` para grabar nuestras selecciones.
- **Monitorizar todos los componentes de aplicación web en un servidor** usando el nodo Server de la Consola de Administración. Pulsamos sobre el servidor en el que queremos monitorizar las aplicaciones web. Pulsamos el enlace `Monitor All Web Applications` para mostrar la tabla `Web Application Deployments`.
- **Desplegar clases startup y shutdown en un servidor** usando el nodo Server de la Consola de Administración. Pulsamos sobre el servidor en el que queremos desplegar las clases `startup`. Seleccionamos la clase `startup` que queremos desplegar y usamos el control `move` para moverla a la columna `Chosen`. Pulsamos `Apply` para grabar nuestras selecciones. Usamos el mismo proceso para desplegar clases `shutdown` pero usando el control `Shutdown Class`.

- **Asignar servidores Web a un servidor** usando el nodo Server de la Consola de Administración. Seleccionamos un servidor para desplegar la aplicación web. Se muestra un panel en el lado derecho mostrando las pestañas asociadas con este ejemplar. Pulsamos una o más aplicaciones Web de la columna `Available` que queramos desplegar en el servidor y usamos el control `move` para mover la aplicación web selecciona a la columna `Chosen`. Pulsamos **Apply** para grabar nuestras asignaciones.
- **Asignar almacenes de conexiones JDBC a un servidor** usando el nodo Server de la Consola de Administración. Seleccionamos un servidor para asignar almacenes de conexiones JDBC. Seleccionamos uno de los almacenes de conexiones JDBC de la columna `Available` que queramos asignar al servidor y usamos el control `move` para moverla a la columna `Chosen`. Pulsamos **Apply** para grabar nuestras asignaciones.
- **Monitorizar todos los almacenes de conexiones JDBC en un servidor** usando el nodo Server de la Consola de Administración. Pulsamos sobre el servidor en el que queremos monitorizar. Pulsamos el enlace `Monitor All JDBC Connection Pools on This Server`. En el panel derecho de la Consola de Administración podremos ver la tabla de conexiones JDBC.
- **Asignar almacenes de conexiones WLEC a un servidor** usando el nodo Server de la Consola de Administración. Pulsamos sobre el servidor en el que queremos asignar los almacenes de conexiones WLEC. Seleccionamos uno o más almacenes de conexiones WLEC en la columna `Available` que queremos asignar al servidor y usamos el control `move` para moverlo a la columna `Chosen`. Pulsamos **Apply** para grabar nuestras asignaciones.
- **Monitorizar todos los almacenes de conexiones WLEC de un servidor** usando el nodo Server de la Consola de Administración. Pulsamos sobre el servidor en el que queremos monitorizar. Pulsamos el enlace `Monitor All WLEC Connection Pools on This Server`. En el panel derecho de la Consola de Administración podremos ver la tabla de conexiones WLEC de este servidor.
- **Asignar servidores, factorías de conexiones y destinos JMS a un servidor** usando el nodo Server de la Consola de Administración. Pulsamos sobre el servidor al que queremos asignar un JMS. Seleccionamos uno o más servidores JMS de la columna `Available` que queramos asignar al servidor. Usamos el control `move` para moverlo a la columna `Chosen`. Lo repetimos usando los controles `JMS Connection Factories` y `JMS Destinations` para asignarlos al servidor.
- **Asignar registros XML a un servidor** usando el nodo Server de la Consola de Administración. Pulsamos sobre el servidor al que queremos asignar un registro XML. Seleccionamos un registro de la lista desplegable `XML Registry`. Pulsamos **Apply** para grabar nuestras asignaciones.
- **Asignar una sesión de mail a un servidor** usando el nodo Server de la Consola de Administración. Pulsamos sobre el servidor al que queremos asignar una sesión de mail. Seleccionamos una o más sesiones de mail en la columna `Available` que queremos asignar al servidor y usamos el control `move` para moverlo a la columna `Chosen`. Pulsamos **Apply** para grabar nuestras asignaciones.

- **Asignar File T3s a un servidor** usando el nodo Server de la Consola de Administración. Pulsamos sobre el servidor al que queremos asignar ficheros T3. Seleccionamos uno o más ficheros T3s en la columna `Available` que queremos asignar al servidor y usamos el control `move` para moverlo a la columna `Chosen`. Pulsamos **Apply** para grabar nuestras asignaciones.

Tareas de Configuración de Clusters

Las tareas de configuración de cluster que pueden realizarse desde la Consola de Administración incluyen:

- **Configurar un Cluster de servidores** usando el nodo Cluster de la Consola de Administración. Los atributos se pueden cambiar usando este nodo incluido el Nombre del cluster, el puerto de escucha, y los nombres de los servidores del cluster.
- **Clonar un Cluster de servidores** usando el nodo Cluster de la Consola de Administración. El cluster se clona, manteniendo los valores de atributos y los servidores individuales del cluster original y el nombre del nuevo cluster se selecciona en la parte `Configuration` del nodo Server.
- **Monitorizar los servidores de un cluster** usando el nodo Cluster de la Consola de Administración. Seleccionamos el cluster a monitorizar. Pulsamos el enlace `Monitor Server Participation in This Cluster`. La tabla de servidores se mostrará en el panel derecho mostrando todos los servidores asignados a este cluster.
- **Asignar servidores a un cluster** usando el nodo Cluster de la Consola de Administración. Seleccionamos un cluster para asignarle servidores. Seleccionamos uno o más servidores en la columna `Available` que queramos asignar al cluster. Usamos el control `move` para mover los servidores seleccionados a la columna `Chosen`. Pulsamos `Apply` para grabar nuestras selecciones.
- **Borrar un cluster** usando el nodo Cluster de la Consola de Administración. Seleccionamos el icono `Delete` en la fila del cluster que queremos borrar. Se mostrará un diálogo en el panel derecho pidiéndonos que confirmemos la petición de borrado. Pulsamos `Yes` para confirmar nuestra decisión de borrar el cluster.

Crear un Nuevo Dominio

Esta sección describe como crear un nuevo dominio. La información de configuración para todos los dominios administrativos de WebLogic reside en el repositorio de configuración, que está localizado bajo el directorio `/config`. Cada dominio tiene un subdirectorio separado bajo `/config`. El nombre de un subdirectorio para un dominio debe ser el mismo que el nombre del dominio.
Cuanto instalamos WebLogic Server por primera vez, se recomienda que creemos un fichero zip que tenga una copia del directorio de configuración por defecto `/mydomain`. Deberíamos mantener una copia de ese fichero zip como backup para crear nuevos dominios. Este subdirectorio contiene los componentes necesarios para una configuración que funciona, como un fichero **fileRealm.properties** y un fichero de configuración.
Para crear un nuevo domino, hacemos lo siguiente:

1. Arrancar el Servidor de Administración bajo un dominio existente como el `mydomain` por defecto.

2. Llamar a la Consola de Administración apuntando nuestro navegador a:
3. `http://hostname:port/console`
   donde `hostname` es el nombre de la máquina donde arrancamos el Servidor de Administración y `port` es el puerto por el que éste está escuchando (por defecto es 7001)

4. Seleccionamos `mydomain --> Create or edit other domains`
   Esto muestra la tabla de dominios.
5. Seleccionamos `Default --> Create a new Domain`
   Introducimos el nombre del nuevo dominio y pulsamos `Create`.
6. Seleccionamos el nuevo dominio de la lista de dominios en el panel izquierdo para hacerlo el dominio actual.
7. Ahora necesitaremos crear una entrada en el Servidor de Administración para el nuevo domino:
   - Seleccionamos `Servers --> Create a new Server`.
   - Introducimos el nombre del nuevo Servidor de Administración y pulsamos `Create`.
8. La Consola de Administración creará un nuevo subdirectorio con el nombre de nuestro dominio y un fichero de configuración, **config.xml**, bajo ese subdirectorio.
   Ahora necesitamos crear un subdirectorio **\applications** bajo ese directorio de dominio. Podemos crear ese subdirectorio desde un shell de comandos, o usando el Explorador (en Windows).
9. Luego, copiamos la aplicación de la Consola de Administración al nuevo directorio `\applications` que acabamos de crear. Para hacer esto, copiamos el fichero **console.war** desde el directorio `\applications` bajo el directorio `mydomain` al nuevo directorio `\applications`.
10. El directorio `mydomain` por defecto contiene scripts de arrancada para arrancar el Servidor WebLogic. Para instalaciones Windows, estos son **startWebLogic.cmd** y **startManagedWebLogic.cmd**. Para instalaciones UNIX, estos son **startWebLogic.sh** y **startManagedWebLogic.sh**. Copiamos estos scripts al directorio del nuevo dominio.
11. Necesitaremos editar los scripts de arrancada en un editor de texto. Por defecto, el nombre del dominio se configura como:
12. `-Dweblogic.Domain=mydomain`
    Reemplazamos `mydomain` con el nombre del nuevo dominio.
    Por defecto el nombre del Servidor de Administración se configura como:
    -Dweblogic.Name=MyServer
    Reemplazamos `MyServer` con el nombre del nuevo Servidor de Administración.

13. Al final del script de arrancada hay un comando cd:
14. `cd config\mydomain`
    Reemplazamos `mydomain` con el nombre del subdirectorio del nuevo dominio.
    También hay una línea en el script de arrancada que pone:
    echo startWebLogic.cmd must be run from the config\mydomain
    directory.
    Reemplazamos `mydomain` aquí con el nombre del nuevo dominio.

15. Copiamos el fichero **SerializedSystemIni.dat** desde el directorio por defecto (`mydomain`) a nuestro nuevo directorio de dominio. No debemos intentar arrancar el Servidor de Administración antes de copiar este fichero.
16. Si creamos un fichero **password.ini** durante la instalación, también debemos copiar este fichero desde el directorio por defecto (`mydomain`) al directorio de nuestro nuevo dominio.

Una vez que hemos completado este procedimiento, podemos arrancar el Servidor de Administración para el nuevo dominio...

Monitorizar un Dominio WebLogic

Introducción a la Monitorización

La herramienta para monitorizar la salud y el rendimiento de nuestro dominio WebLogic es la Consola de Administración. Esta consola nos permite ver el estado y las estadísticas de los recursos WebLogic, como servidores, HTTP, el subsistema JTA, JNDI, la seguridad, los almacenes de conexiones CORBA, EJB, JDBC, y JMS.
La información monitorizada se presenta en el panel derecho de la Consola de Administración. Accedemos a una página seleccionando un contenedor o subsistema, o una entidad particular bajo un contenedor, o el árbol de dominio en el panel izquierdo.
La Consola de Administración proporciona tres tipo de páginas que contienen información de monitorización:

- Páginas de pestañas de monitorización para una entidad particular (como un ejemplar de JDBC Connection Pool o el rendimiento de un servidor particular.
- Tablas de datos sobre todas las entidades de un tipo particular (como una tabla de Servidores WebLogic).
- Vistas del log de dominio y los logs de los servidores locales.

La Consola de Administración obtiene información sobre los recursos del dominio desde el Servidor de Administración. Este servidor, a su vez, está lleno de `Management Beans` (MBeans), basados en el estándar Java Management Extension (JMX) de Sun, que proporcionan el esquema de control de acceso a los recursos de dominio.
El Servidor de Administración contiene `MBeans` de configuración, que controlan la configuración del dominio y `MBEans` de tiempo-de-ejecución. Éstos últimos proporcionan una "foto" de los recursos del dominio, como el uso de memoria de la JVM, o el estado de los servidores WebLogic. Cuando se ejemplariza un recurso particular del dominio (como una aplicación Web) , se crea un ejemplar `MBean` que recolecta información sobre ese recurso particular.
Cuando accedemos a la página de monitorización de recursos particulares en la Consola de Administración, ésta realiza una operación **GET** para recuperar los valores actuales de los atributos.
Las siguientes secciones describen algunas de las páginas de monitorización que son útiles para controlar un dominio WebLogic. Estas páginas se han seleccionado simplemente para ilustrar las facilidades proporcionadas por la Consola de Administración.

Monitorizar Servidores

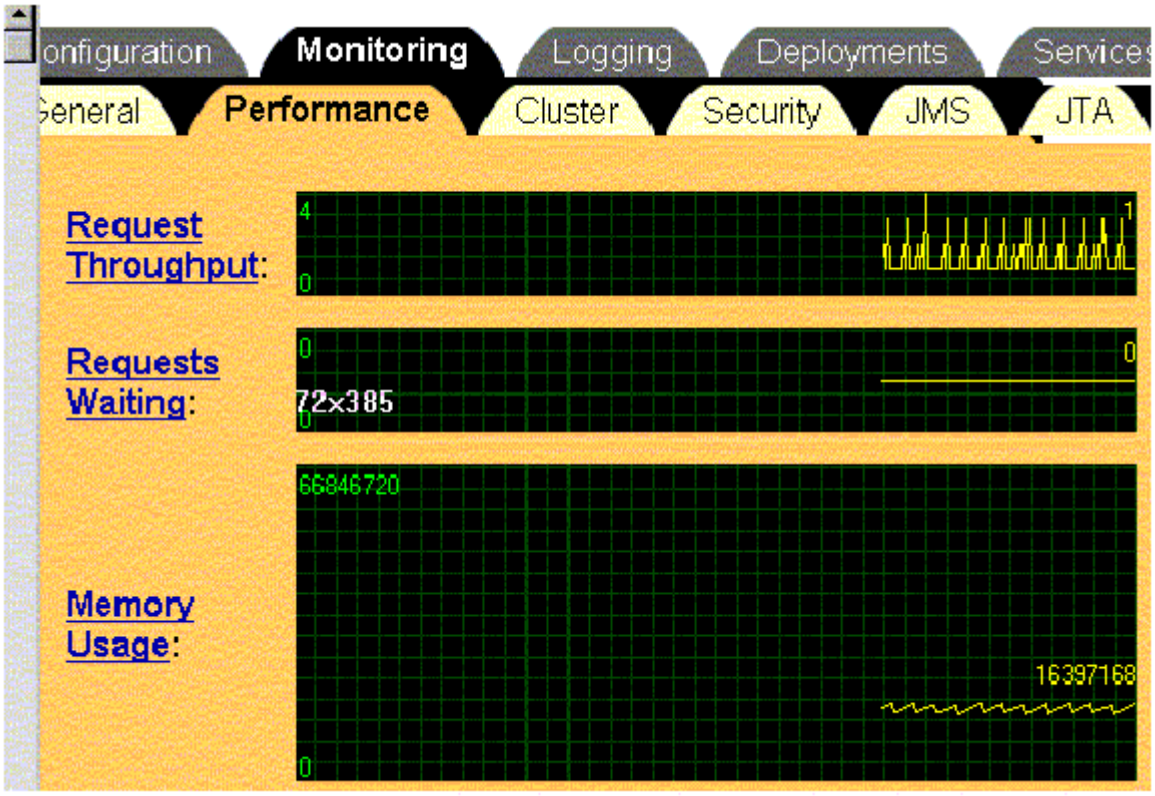
La tabla de servidores y las páginas de pestañas de monitorización para servidores individuales nos permiten monitorizar los Servidores WebLogic. La tabla de servidores proporciona un sumario del estado de todos los servidores de nuestro dominio. Si sólo se ha reenviado un pequeño conjunto de los mensajes de log desde el servidor hasta el log de dominio, podría ser útil acceder al servidor local para resolver problemas o buscar eventos.
Para más información sobre los ficheros de log y el subsistema de logging, puedes ver Usar los Mensajes de Log para Controlar Servidores WebLogic
Podemos acceder a los datos de monitorización de cada servidor WebLogic desde las pestañas de monitorización de ese servidor. La pestaña **Logging** proporciona acceso al log local de ese servidor (es decir, el log de la máquina donde se está ejecutando el servidor).
La página de la pestaña **Monitoring --> General** indica el estado actual y proporciona acceso al árbol JNDI, la tabla Execute Queues, la tabla Active Sockets, y la tabla Connections. La tabla Execute Queues proporciona información de rendimiento como la petición pendiente más vieja, y el número de peticiones pendientes en este momento.

Apagar o Suspender un Servidor

La pestaña **Monitoring --> General** también nos permite apagar o suspender un servidor. Si se suspende un servidor, éste sólo aceptará peticiones del Servidor de Administración. Las peticiones de clientes serán ignoradas.

Rendimiento

La pestaña **Monitoring --> Performance** nos muestra un gráfico en tiempo real del uso de la pila de memoria de la JVM, las peticiones servidas, y las peticiones en espera. Esta página también nos permite forzar a la JVM a que realice la recolección de basura sobre la pila de memoria.



La pila Java es un repositorio de objetos Java (vivos y muertos). Normalmente no necesitamos realizar la recolección de basura manualmente porque la JVM lo hace automáticamente. Cuando la JVM empieza a salirse de la memoria, para la ejecución y usa un algoritmo de recolección de basura para liberar el espacio que ya no utilizan las aplicaciones Java.
Por otro lado, los desarrolladores que están depurando aplicaciones podrían tener ocasión de forzar la recolección de basura manualmente. Esta recolección manual podría ser útil, por ejemplo, para comprobar picos de memoria que consume rápidamente la JVM.

Datos de Cluster

La pestaña **Monitoring --> Cluster** proporciona información sobre el cluster al que pertenece el servidor (como el número de servidores que están vivos actualmente).

Seguridad del Servidor

La pestaña **Monitoring --> Security** proporciona estadísticas sobre los intentos de login inválidos y los usuarios bloqueados y desbloqueados.

JMS

La pestaña **Monitoring --> JMS** proporciona estadísticas sobre los servidores y conexiones JMS. Esta página también proporciona enlaces a las tablas de conexiones y servidores JMS activos, que monitorizan dichos atributos como sesiones totales actuales.

JTA

La pestaña **Monitoring --> JTA** proporciona estadísticas sobre el subsistema de transacciones Java como el total de transacciones y las devueltas. La página proporciona enlaces a tablas que listan las transacciones por recurso y nombre, y una tabla con las transacciones en-vuelo.

Monitorizar Almacenes de Conexiones JDBC

Los recursos del subsistema Java Database Connectivity (JDBC) también pueden ser monitorizados mediante la Consola de Administración. La pestaña **Monitoring** para almacenes de conexiones JDBC nos permite acceder a una tabla que lista las estadísticas de los ejemplares de ese almacén. Al igual que en otras tablas de entidades de la Consola de Administración, podemos personalizar la tabla para seleccionar los atributos que queremos ver.
Muchos de estos atributos proporcionan información importante para el control de acceso de los clientes a la base de datos.
El campo **Waiters High** indica el mayor número de clientes que han esperado una conexión en un momento dado. El campo **Waiters** nos dice cuántos clientes hay esperando una conexión actualmente. El campo **Connections High** indica el mayor de número de conexiones que han ocurrido en un momento dado. El campo **Wait Seconds High** nos dice el tiempo máximo que un cliente ha estado esperando una conexión a la base de datos. Estos atributos nos permiten medir la eficiencia de nuestra configuración actual en respuesta a las peticiones de clientes.
Si el valor del campo **Connections High** está cerca del valor del campo **Maximum Capacity** (configurado en la pestaña **Configuration Connections**), podríamos considerar incrementar el valor de **Maximum Capacity** (el número máximo de conexiones concurrentes). Si el valor del campo **Waiters High** indica que los clientes están sujetos a largos tiempos de espera para acceder a la base de datos, podríamos querer incrementar el tamaño del almacén.
El valor del campo **Shrink Period** es el tiempo que el subsistema JDBC espera antes de reducir desde el máximo. Cuando el subsistema reduce el almacén, las conexiones a bases de datos son destruidas. Crear una conexión de base de datos consume recursos y puede consumir mucho tiempo. Si nuestro sistema tiene sobrecargas intermitentes de peticiones de clientes, un corto **Shrink Period** podría significar que las conexiones a la base de datos se están re-creando continuamente, lo que podría degradar el rendimiento.

Sumario de Páginas de Monitorización en la Consola de Administración

La siguiente tabla lista las tablas y páginas de pestañas de monitorización disponibles en la Consola de Administración.

Página	Camino	Datos mostrados
Páginas de pestañas de Monitorización		
Información general del Servidor	servername --> Monitoring --> General	Estado y tiempo de activación
Rendimiento del Servidor	servername --> Monitoring --> Performance	Gráfico en tiempo real de las peticiones servidas, uso de memoria de la JVM y peticiones en espera
Estadísticas de Cluster	servername --> Monitoring --> Cluster	Estadísticas sobre los clusters como el número de servidores vivos y los fragmentos enviados y recibidos.
Seguridad del Servidor	servername --> Monitoring --> Security	Número de intentos de login inválidos, total de usuarios bloqueados y desbloqueados, y otras estadísticas de seguridad
Información de Versión de Servidor	servername --> Monitoring --> Version	Versión del JDK, WebLogic, sistema operativo.
Cluster	Clusters --> clustername --> Monitoring	Información sobre los servidores participantes
Tablas de Entidades		
Servers	Servers	Datos específicos del servidor, como el uso de memoria, tiempo de arrancada, estado, participación del cluster, intentos de login fallidos, estado de la pila, contador de sockets y total de re-arrancadas.
Execute Queues	servername --> Monitoring --> General --> Monitor Execute Queues on this server	Información sobre las peticiones servidas y pendientes y otros atributos.
Execute Sockets	servername --> Monitoring --> General --> Monitor Active Sockets on this server	Protocolo y otros atributos de los sockets activos
Connections	servername --> Monitoring --> General --> Monitor Connections on this server	Tiempo de conexión, dirección remota, bytes enviados y recibidos y otros atributos de conexiones.
Clusters	Clusters	Datos como el algoritmo de carga por defecto y la dirección multicast.
Transactions By Name	servername --> Monitoring --> JTA --> Monitor Transactions by Name on this server	Datos sobre las transacciones organizados por nombre
Transactions By Resource	servername --> Monitoring --> JTA --> Monitor Transactions by Resource on this server	Datos sobre las transacciones organizados por recursos
Active Transactions	servername --> Monitoring --> JTA --> Monitor In-flight Transactions on this server	Datos sobre las transacciones en-vuelo en este servidor
Machines	Machines	Direcciones y otros atributos de las máquinas
Applications	Applications	Lista de aplicaciones
EJB Deployments	Deployments --> EJB	URL, nombre de aplicación y otros atributos sobre cada EJB.
Web Applications	Deployments --> Web Applications	Datos como la URL y servlet por defecto de cada aplicación Web
Active Web Applications	Deployments --> Web Applications --> appname --> Monitoring --> Monitor all instances of appname	Datos sobre las copias desplegadas de esta aplicación Web.
Web Application Servlets	Deployments --> Web Applications --> appname --> Monitoring --> Monitor all servlets for this Web Application	Estadísticas de la aplicación Web seleccionada, como la máxima capacidad del almacén y el tiempo de ejecución.
Startup and Shutdown Classes	Deployments --> Startup & Shutdown	Lista las clases startup y shutdown registradas.

JDBC Connection Pools	Services --> JDBC --> Connection Pools	Capacidad inicial, incremento de capacidad y otros atributos de los almacenes de conexiones JDBC.
JDBC Multipools	Services --> JDBC --> Multipools	Balance de carga y otros atributos de JDBC Multipools
JDBC Data Sources	Services --> JDBC --> Data Sources	Nombre del almacén, nombre JNDI y otros atributos de las fuentes de datos JDBC.
JDBC Tx Data Sources	Services --> JDBC --> Tx Data Sources	Nombre del almacén, nombre JNDI y otros atributos de las fuentes de datos JDBC Tx.
JMS Connection Factories	Services --> JMS --> Connection Factories	Nombre JNDI, ID del cliente, prioridad por defecto y otros atributos de las factorías de conexiones JMS.
JMS Templates	Services --> JMS --> Templates	Datos sobre las plantillas JMS
JMS Destination Keys	Services --> JMS --> Destination Keys	Tipo de clave y otros atributos de las claves destino JMS.
JMS Stores	Services --> JMS --> Stores	Descripción de los almacenes JMS
JMS Servers	Services --> JMS --> Servers	Datos sobre los servidores JMS
Active JMS Services	Services --> JMS --> Servers --> Monitor all Active JMS Services	Marca de agua más alta para las conexiones y otros datos, datos sobre servicios activos JMS.
Active JMS Servers	Services --> JMS --> Servers --> Monitor all instances	Estadísticas sobre sesiones, mensajes pendientes y otros datos.
Active JMS Destinations	Services --> JMS --> Servers --> Monitor all Active JMS Destinations	Clientes, mensajes recibidos y otros atributos de los destinos JMS.
Active JMS Session Pools	Services --> JMS --> Servers --> Monitoring --> Monitor all Active JMS Session Pools	Marca de agua más alta de clientes y otros datos monitorizados
JMS Destinations	Services --> JMS --> jmsservername --> Destinations	Nombre JNDI y otros datos
JMS Session Pools	Services --> JMS --> jmsservername --> Session Pools	Modo de reconocimiento, sesiones máximas y otros atributos sobre almacenes de conexiones JMS.
XML Registries	Services --> XML --> XML Registries	Lista de DocumentBuilderFactories y SAXParserFactories
WLEC Connection Pools	Services --> WLEC --> WLEC Connection Pools	Nombre de domino WebLogic Enterprise (WLE), direcciones de fallo, tamaño máximo y mínimo del almacén y otra información.
Jolt Connection Pools	Services --> Jolt	Dirección de fallo, tamaño máximo y mínimo del almacén y otros atributos de almacenes de conexiones Jolt.
Active Jolt Connection Pools	Services --> Jolt --> joltconnectionpoolname --> Monitoring --> Monitor all active pools	Capacidad máxima, conexiones actuales, y otros datos sobre los ejemplares de un almacén de conexiones Jolt.
Virtual Hosts	Services --> Virtual Hosts	Formato, nombre del fichero log y otros atributos de los host virtuales.
Mail Sessions	Services --> Mail	Nombre y propiedades de la sesión de mail.
File T3	Services --> File T3	Nombre y paths de los ficheros.
Users	Security --> Users	Lista de usuarios.
Groups	Security --> Groups	Lista de grupos.
Access Control Lists	Security --> ACLs	Lista de ACLs
Caching Realms	Security --> Caching Realms	Lista de los reinos en el caché
Realms	Security --> Realms	Describe los reinos.
Domain Log Filters	Domain Log Filters	Servidores en los que el filtro está registrado y atributos usados para filtrar mensajes de log.

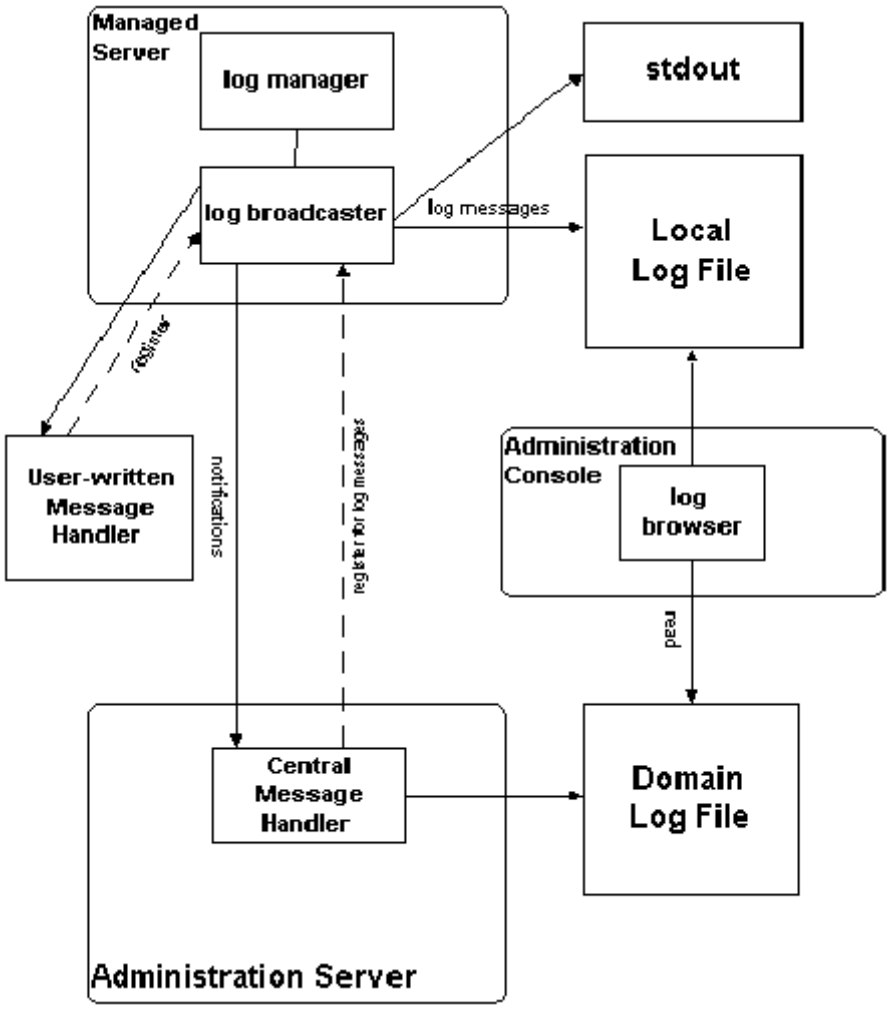
Usar los Mensajes de Log para Controlar Servidores WebLogic

Introducción al Subsistema de Logging

Los mensajes de Log son útiles herramientas para controlar sistemas. Estos nos permiten detectar problemas, descubrir la fuente de un fallo y seguir el rendimiento del sistema. Los mensajes de Log generados por el software WebLogic Server se almacenan en dos localizaciones:

- Los componentes subsistemas de un Servidor WebLogic generan mensajes que se guardan en un fichero local, es decir, un fichero que reside donde se está ejecutando el servidor. Si hay varios servidores en una máquina, cada servidor tiene su propio fichero log. Las aplicaciones desplegadas en nuestros servidores WebLogic también podrían generar mensajes de log al fichero de log local.
- Además, un subconjunto de los mensajes se guardan localmente en un fichero log de todo dominio mantenido por el Servidor de Administración.

Se utilizan las facilidades de Java Management Extension (JMX), embebidas en el Servidor WebLogic, para transmitir los mensajes de log desde los Servidores WebLogic hacia el Servidor de Administración. Un mensaje re-enviado a otras entidades en la iniciativa de un Servidor WebLogic se llama una **notificación** en terminología JMX.
Cuando arranca un Servidor WebLogic, el controlador de mensajes del Servidor de Administración registra al servidor para recibir mensajes de Log. En el momento del registro, se proporciona un filtro modificable-por-el-usuario que usa el servidor local para seleccionar los mensajes reenviados al Servidor de Administración. Estos mensajes son recolectados en el log de dominio.
Por defecto, sólo los mensajes de log más importantes (según lo determina la Severidad del Mensaje) son reenviados desde los servidores locales al log de dominio. El log de dominio nos da una visión general de todo el dominio ya que sólo se enfoca en los mensajes más críticos. Si queremos modificar el filtro, para recibir un subconjunto diferente de mensajes de log desde el servidor local, podemos hacerlo dinámicamente, usando la Consola de Administración. No necesitamos re-arrancar el servidor local para que los cambios tengan efecto. (Puedes ver Crear Filtros de Logs de Dominio.)
Los desarrolladores también pueden construir controladores de mensajes personalizados que pueden registrar con un Servidor WebLogic para recibir los mensajes log mediante notificaciones JMX.



Ficheros Log Locales del Servidor

En versiones anteriores del Servidor WebLogic, se creaba un nuevo fichero de log cuando éste alcanzaba el tamaño máximo. Este tipo de creación de ficheros de log automáticos se llama rotación larga. En la versión actual, (6.0) tenemos la opción de basar la rotación larga en el tamaño o en el tiempo. Para configurar la rotación, abrimos la Consola de Administración y hacemos lo siguiente:

1. En el panel izquierdo, seleccionamos un servidor.
2. En el panel derecho, seleccionamos `Configuration --> Logging`.
3. En el campo `Rotation Type`, seleccionamos `time` o `size`.

Si el valor de este campo es `none`, no ocurrirá rotación larga. Si basamos la rotación en el tiempo, se creará un fichero log cada intervalo de tiempo especificado en (`File Time Span`).
Por defecto, el fichero de log local del servidor se llama **servername.log** (donde `servername` es el nombre del servidor) y se crea en el directorio desde donde hayamos arrancado el Servidor WebLogic. También podemos seleccionar el nombre del fichero en la página `Configuration --> Logging` del servidor.
Podemos especificar el número máximo de ficheros de rotación que podemos acumular configurando apropiadamente el valor del campo `File Count`. Una vez que se ha alcanzado este número, se borra el fichero más viejo cada vez que ocurre una rotación de fichero. Los ficheros rotados se numeran en orden de creación **filenamennnnn**, donde `filename` es el nombre de fichero configurado. Por ejemplo **weblogic.log00007**.
Los logs locales del servidor tienen todos los mensajes que han ocurrido. La configuración de logging también nos permite especificar qué mensajes se sacan por `stdout`. Podemos excluir los mensajes de

menor severidad especificando que sean grabados. También podemos activar o desactivar los mensajes de depuración hacia `stdout`.

Log StartUp

Cuando está arrancando un Servidor WebLogic, si ocurre un error antes de que se haya completado la inicialización, estos errores se muestran por `stdout` y se graban en un fichero de log de arrancada llamado **weblogic-startup.log**. Si la arrancada tiene éxito, el último mensaje de este log apunta a la localización del fichero log local del servidor, donde ocurren los log normales.

Logs de Cliente

Los clientes Java que usan las facilidades de logging de WebLogic también podrían generar mensajes de log. Sin embargo, los mensajes enviados por clientes no son reenviados al log de dominio. Configuramos la propiedades de logging de un cliente introduciendo el argumento apropiado en la línea de comandos:
`-Dweblogic.log.attribute=value`
donde `attribute` es cualquier atributo `LogMBean`. Por defecto, el logging está desactivado para los clientes y los mensajes se sacan por `stdout`. Podemos activar el logging a un fichero y seleccionar el nombre del fichero usando el siguiente argumento de la línea de comandos:
`-Dweblogic.log.FileName=logfilename`
donde `logfilename` es el nombre del fichero log. También se pueden usar los siguientes argumentos de la línea de comandos para logging de clientes:
`-Dweblogic.StdoutEnabled=boolean`
`-Dweblogic.StdoutDebugEnabled=boolean`
`-Dweblogic.StdoutSeverityLevel = [64 | 32 | 16 | 8 | 4 | 2 | 1]`
Donde `boolean` es `true` o `false`.

Formato de Fichero Log

La primera línea de cada mensaje en un fichero log empieza con #### seguido por la cabecera del mensaje. La cabecera de mensaje proporciona el contexto en tiempo de ejecución del mensaje. Cada atributo del mensaje está contenido entre ángulos.
Las líneas que siguen al cuerpo del mensaje están sólo presentes para mensajes de excepciones y muestran el seguimiento de la pila de la excepción. Si un mensaje no se graba dentro del contexto de una transacción, los ángulos (separadores) para el `Transaction ID` están presentes aunque no haya ninguna transacción.
Aquí tenemos un mensaje de log de ejemplo:

```
####<Jun 2, 2000 10:23:02 AM PDT> <Info> <SSL> <bigbox> <myServer>
<SSLListenThread> <harry> <> <004500> <Using exportable strength SSL>
```

En este ejemplo, los atributos de mensajes son: `Timestamp`, `Severity`, `Subsystem`, `Machine Name`, `Server Name`, `Thread ID`, `User ID`, `Transaction ID`, `Message ID`, y `Message Text`.

> ***Nota:***
> *Los mensajes de los clientes no tienen los atributos `Server Name` o `Thread ID`.*

Nota:
La codificación de caracteres usada en la escritura de ficheros log es la codificación de caracteres por defecto del sistema local.

Atributos de Mensajes

En cada mensaje log grabado en un fichero log de servidor podrían estar definidos los atributos listados en la siguiente tabla. El `Message Id` también podría asociar el mensaje con un atributos adicionales (como `Probable Cause` y `Recommended Action`) contenido en el `Message Catalog`.

Atributo	Descripción
`Timestamp`	La hora y la fecha en que se originó el mensaje, en un formato especificado para la localidad.
`Severity`	Indica el grado de impacto o seriedad del evento reportado por el mensaje. Ver Severidad del Mensaje.
`Subsystem`	Este atributo denota el subsistema particular del Servidor WebLogic que fue la fuente del mensaje. Por ejemplo, EJB, RMI, JMS.

`Server Name`
`Machine Name`
`Thread ID`
`Transaction ID`
	Estos cuatro atributos identifican los orígenes del mensaje. `Transaction ID` sólo está presente para mensajes dentro del contexto de una transacción.
	Nota:
	`Server Name` y `Thread ID` no están presentes en mensajes log generados por un cliente Java y grabados en un log de cliente.

`User ID`	El usuario desde el contexto de seguridad cuando se generó el mensaje
`Message ID`	Un identificador único de seis-dígitos. Los mensajes superiores a 499999 están reservados para mensajes del sistema WebLogic Server.
`Message Text`	Para mensajes de Servidor WebLogic, esto contiene una corta descripción como se define en el catálogo de mensajes del sistema. (ver Catálogo de Mensajes). Para otros mensajes, este es el texto definido por el desarrollador del programa.

Catálogo de Mensajes

Además de la información contenida en un mensaje de log, los mensajes generados por los componentes de sistema WebLogic Server (o posiblemente por código de usuario) incluyen información adicional predefinida o que se almacena en un catálogo de mensajes. Los atributos adicionales almacenados en el catálogo de mensajes son los siguientes:

Atributo	Descripción
`Message Body`	Esta es una breve descripción textual de la condición que está siendo reportada. Es lo mismo que `Message Text` en el mensaje.
`Message Detail`	Una descripción más detallada del mensaje que se está reportando.
`Probable Cause`	Una explicación de la causa probable de la condición del mensaje que se está reportando.
Recommended Action	Una acción recomendada por el administrador para resolver o evitar la condición del mensaje que se está reportando.

Podemos acceder a los atributos adicionales del mensaje desde la vista **log** de la Consola de Administración.

Severidad del Mensaje

Los mensajes de log de WebLogic Server tienen un atributo llamado **severity** que refleja la importancia o impacto potencial sobre los usuarios del evento o condición reportada en el mensaje.
En la siguiente tabla podemos ver las severidades definidas, en orden ascendente, siendo `Emergency` la severidad más alta:

Atributo	¿Reenviado al de dominio por defecto?	Descripción
`Informational`	NO	Usado para reportar operaciones normales
`Warning`	NO	Ha ocurrido una operación o configuración sospechosa pero podría no tener impacto sobre la operación normal.
`Error`	SI	Ha ocurrido un error de usuario. El sistema o la aplicación pueden manejar el error sin interrupción, ni degradación limitada del servicio.
`Notice`	SI	Un mensaje de aviso: Ha ocurrido una operación o configuración sospechosa pero podría no tener impacto sobre la operación normal.
`Critical`	SI	Ha ocurrido un error de sistema o de servicio. El sistema puede recuperarse pero podría haber una pérdida momentánea, o una degradación permanente del servicio.
`Alert`	SI	Un servicio particular está en un estado inutilizable mientras que otras partes del sistema continúan su función. No es posible una recuperación automática; es necesaria la atención inmediata del administrador para resolver el problema.
`Emergency`	SI	El servidor está inutilizado. Esta severidad indica un fallo severo del sistema.

Mensajes de Depuración

Los mensajes con una severidad `debug` son un caso especial. Estos mensajes no son reenviados al log de dominio. Los mensajes de debug podrían contener información detallada sobre una aplicación o un servidor. Estos mensajes sólo deberían ocurrir cuando la aplicación se está ejecutando en modo depuración.

Navegar por los Ficheros de Log

Las capacidades de navegación por los ficheros log de la Consola de Administración nos permiten:

- Ver el fichero de log local de cualquier servidor
- Ver el fichero log de todo el dominio.

Cuando estamos viendo cualquiera de los ficheros de log, podemos:

- Seleccionar los mensajes de log a ver basándonos en el tiempo, la `user ID`, el subsistema, la severidad del mensaje, o la descripción breve del mensaje.
- Ver los mensajes según fueron cargados o buscar mensajes de log antiguos.
- Seleccionar los atributos del mensaje de log a mostrar en la Consola de Administración y el orden en que queremos verlos.

Crear Filtros de Log de Dominio

Los mensajes reenviados por los servidores WebLogic al log de dominio, son por defecto, un subconjunto de los mensajes generados localmente. Podemos configurar un filtro que selecciona los mensajes de log a reenviar basándonos en la severidad, en el subsistema o la user ID. (Los mensajes de depuración son un caso especial que no son reenviados al log de dominio). Podemos crear o modificar los filtros de logs de dominio desde la tabla **Domain Log Filters**. Esta tabla es accesible desde la página con pestañas de monitorización del dominio.

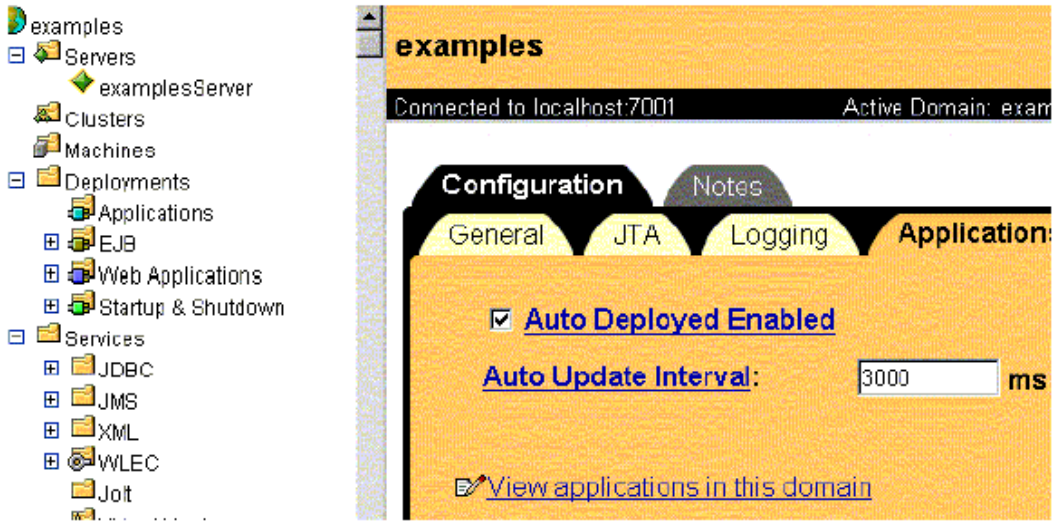
Desplegar Aplicaciones

Despliegue Dinámico

Si el despliegue automático está activado para el dominio WebLogic Server destino, cuando se copia una aplicación dentro del directorio `/config/domain_name/applications` del Servidor de Administración WebLogic, si el Servidor de Administración se está ejecutando, detecta la presencia de una nueva aplicación y la despliega automáticamente sobre el Servidor de Administración. (El subdirectorio `domain_name` es el nombre del dominio WebLogic Server en el que estamos desplegando la aplicación. Está técnica para desplegar aplicaciones se llama "despliegue automático" y sólo se recomienda cuando estamos desarrollando aplicaciones. No se recomienda el uso del despliegue automático en entornos de producción. Si WebLogic Server no se está ejecutando cuando copiamos la aplicación al directorio `/applications`, la aplicación se desplegará la próxima vez que se arranque el Servidor WebLogic.
Por defecto, el despliegue automático está activado. Si deshabilitamos el despliegue automático, todavía podemos desplegar aplicaciones o componentes manualmente mediante la Consola de Administración. Esta técnica se llama "despliegue estático".

Activar o Desactivar el Despliegue Automático

Para determinar si tenemos activado el despliegue automático, llamamos a la Consola de Administración y vamos a la página de selección de aplicaciones de dominio (`Configuration --> Applications`) del dominio. Esta página nos permite activar o desactivar el despliegue automático y seleccionar el intervalo (en milisegundos) en que el Servidor WebLogic chequeará para ver si hay nuevas aplicaciones en el subdirectorio `\applications`.
La siguiente figura muestra esta página para el servidor de demostración `examples`.



Por defecto, el Servidor de Administración chequea cada tres segundos los cambios en el directorio `\applications` cuando está activado el despliegue automático.

Nota:
El despliegue automático es un método para el despliegue rápido de una aplicación sobre el Servidor de Administración. Se recomienda que este método sólo se use en un entorno de desarrollo para probar aplicaciones. No se recomienda usar el despliegue automático o para desplegar componentes en Servidores Controlados.

Despliegue Automático de Aplicaciones en un Formato de Directorio Expandido

Una aplicación o un componente de una aplicación se pueden desplegar en un formato de directorio expandido o empaquetados en un fichero **Enterprise Application Archive** (EAR), en un fichero **Web Application Archive** (WAR), o en un fichero **Java Archive** (JAR).
Para desplegar dinámicamente una aplicación en formato expandido, hacemos lo siguiente:

1. Nos aseguramos de que nombre del directorio creado para expandir la aplicación es el mismo que el `Context Path` de la aplicación.
2. Copiamos este subdirectorio bajo `/config/domain_name/applications`, donde `domain_name` es el nombre del dominio objetivo donde se va a desplegar la aplicación. Esto desplegará automáticamente la aplicación si está activado el despliegue automático.

Eliminar o Re-desplegar Aplicaciones Dinámicamente

Una aplicación o un componente de aplicación se pueden re-desplegar dinámicamente mientras el servidor se está ejecutando. Esto podría ser útil si queremos actualizar la aplicación o el componente desplegados sin parar y re-arrancar el Servidor de Administración. Para re-desplegar dinámicamente un fichero JAR, WAR o EAR, simplemente copiamos la nueva versión sobre el fichero existente en el directorio `/applications`.
Esta característica es útil para desarrolladores que pueden simplemente añadir la copia al directorio `/applications` como el último paso de su **makefile**, y el servidor lo actualizará.
Re-despliegue Automático de Aplicaciones Expandidas
También podemos re-desplegar dinámicamente aplicaciones o componentes de aplicaciones que han sido desplegados en formato expandido. Cuando una aplicación se ha desplegado en formato expandido, el Servidor de Administración busca periódicamente un fichero llamado **REDEPLOY** en el directorio de la aplicación expandida. Si la fecha o la hora de este fichero cambian, el Servidor de Administración re-desplegará el directorio expandido.
Si queremos actualizar ficheros en un directorio de aplicación expandido, hacemos lo siguiente:

1. Cuando desplegamos por primera vez la aplicación expandida, creamos un fichero vacío llamado **REDEPLOY** en el directorio donde reside la aplicación.
2. Para actualizar la aplicación expandida, copiamos los ficheros actualizados sobre los ficheros existentes en ese directorio.
3. Después de copiar los nuevos ficheros, modificamos el fichero **REDEPLOY** para alterar su fecha u hora de modificación.

Cuando el Servidor de Administración detecta que ha cambiado la fecha o la hora de ese fichero, re-despliega los contenidos del directorio expandido.

Usar la Consola de Administración para Desplegar Aplicaciones

Podemos usar la Consola de Administración para instalar y desplegar una aplicación o componente de una aplicación (como ficheros EJB o JSP) y desplegar ejemplares de los componentes de la aplicación en los Servidores WebLogic destino. Hay varios pasos para llevar a cabo esta tarea:

1. Instalar la aplicación (o componente) en el directorio `/config/domain_name/applications` del Servidor de Administración (donde `domain_name` es el nombre del dominio).

Usamos la página **Install an Application** de la consola de administración para copiar los ficheros de aplicación J2EE (EAR), aplicación Web (WAR), o EJB o JSP (JAR) a el directorio `/config/domain_name/applications` del Servidor de Administración. Hay un enlace hacia la página **Install an Application** desde la tabla de aplicaciones (ver figura de abajo), **Web applications** y **EJB deployments**. Accedemos a estas tablas seleccionando el contenedor adecuado en el panel de la izquierda.

Instalar una aplicación (o componentes) mediante la consola de Administración también crea entradas para esa aplicación (o componentes) en el fichero de configuración del dominio (**/config/domain_name/config.xml**). El Servidor de Administración también genera JMX Management Beans (MBeans) que permiten la configuración y monitorización de la aplicación (o componentes).

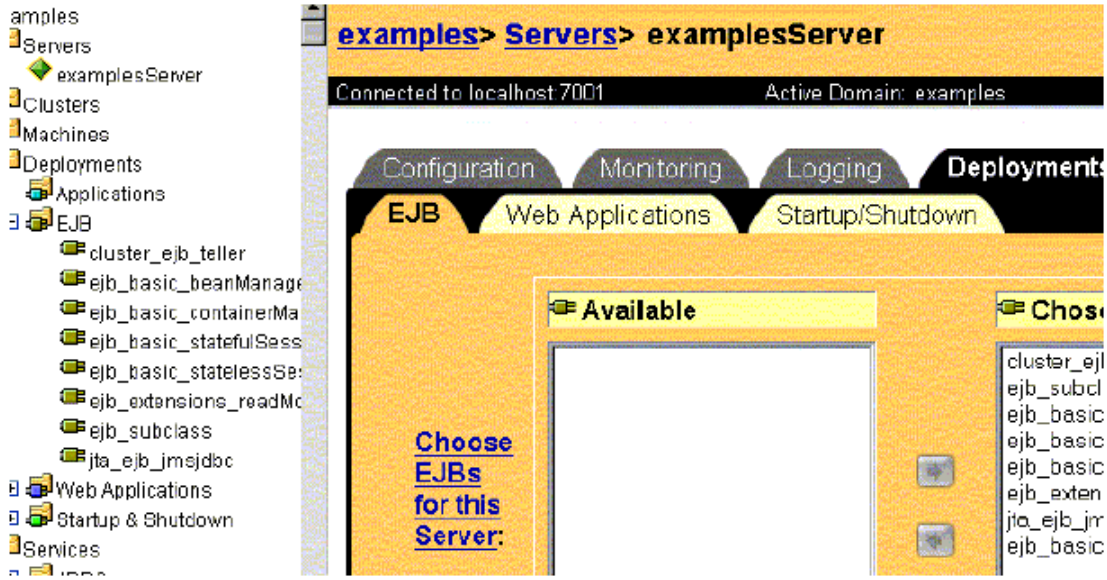
2. Desplegar la aplicación o componentes de aplicación.

Hay dos formas para desplegarlos, dependiendo de si tenemos activado o no el despliegue automático:

   - Si el despliegue automático está activado, el Servidor de Administración despliega la aplicación automáticamente una vez que la hemos copiado en el directorio `/config/domain_name/applications` del Servidor de Administración.
   - Si el despliegue automático está desactivado, una aplicación instalada sólo se despliega si la especificamos para ser desplegada en la página de **Configuration** para esa aplicación.

3. Desplegar ejemplares de los componentes de la aplicación (componentes de aplicación Web o EJBs) en los Servidores Controlados.

Una vez que hemos instalado nuestra aplicación en el Servidor de Administración (en el directorio `/config/domain_name/applications`), podemos desplegar los componentes de la aplicación en los Servidores Controlados.
Seleccionamos los componentes de la aplicación a desplegar sobre el servidor accediendo a las pestañas de las páginas `Deployments --> EJB` (ver siguiente figura) o `Deployments --> Web Applications` de ese servidor.
De forma alternativa, podemos seleccionar los servidores destino para desplegar una aplicación mediante la página `Targets` para ese componente.



Si desplegamos componentes de aplicación (como ficheros EJBs o WAR) para los Servidores Controlados en un cluster, debemos asegurarnos que los mismos componentes de la aplicación son desplegados en todos los servidores del cluster. Para hacer esto, seleccionamos el cluster como el destino para el despliegue.
Cuando una aplicación o componente de aplicación (como un fichero EAR o WAR, o fichero EJB o JAR) si desplegamos en un Servidor WebLogic particular, los ficheros son copiados al directorio `.wl_temp_do_not_delete_servername` bajo `/config/domain_name/applications` sobre el Servidor WebLogic destino. El Servidor de Administración WebLogic invoca a un fichero de distribución de servlet para copiar los ficheros al servidor destino.

Configurar Componentes Web en un Servidor WebLogic

Introducción

Además de su habilidad para hospedar dinámicamente aplicaciones distribuidas basadas en Java, WebLogic Server también es un Servidor Web totalmente funcional que puede manejar sitios Web de alto volumen, servir ficheros estáticos como ficheros HTML, y ficheros de imágenes igual que servlets y JavaServer Pages (JSP). WebLogic Server soporta el estándar HTTP 1.1.

Parámetros HTTP

Podemos configurar los siguientes parámetros de operación HTTP usando la Consola de Administración para cada ejemplar de WebLogic Server (o por cada host virtual):

- **Aplicación Web por Defecto**
  La aplicación Web por defecto intenta responder a las peticiones que no sean respondibles por cualquier otra Aplicación Web desplegada. A los recursos de la Aplicación Web se accede con una URI que no incluye el path de contexto (el path de contexto de una aplicación Web normalmente es el nombre de la aplicación Web).
- **Post Timeout Seconds**
  El tiempo (en segundos) que WebLogic Server espera entre bloques de datos enviados usando el método **HTTP POST**. Usado para evitar ataques de denegación-de-servicio que intentan sobrecargar el servidor usando el método **POST**.
- **Max Post Time**
  Límite de la cantidad total de tiempo que el WebLogic Server gasta en recibir datos **POST**.
- **Max Post Size**
  Limita el número de bytes de datos recibidos en un **POST** desde una sola petición. Si se sobrepasa este límite, se lanzará una excepción `MaxPostSizeExceeded`.
- **Enable Keep Alive**
  Activa o desactiva las conexiones HTTP persistentes. Si el navegador usa HTTP 1.1, siempre se usa **keep alive**.
- **Connection timeout**
  El número de segundos que espera WebLogic Server antes de cerrar una conexión HTTP inactiva.
- **HTTPS Duration**
  El número de segundos que espera WebLogic Server antes de cerrar una conexión HTTPs (Secure Socket Layer o SSL) inactiva.
- **HTTP Access Logging**
  Podemos activar o desactiva la generación de log de accesos HTTP. También podemos configurar los parámetros de cómo y cuándo se rota el fichero de logs de acceso. Para más información puedes ver Configurar los Logs de Acceso HTTP.

Configurar el Puerto de Escucha

Podemos especificar el puerto en el que escucha cada Servidor WebLogic las peticiones HTTP. Aunque podemos especificar cualquier número de puerto válido, si especificamos el puerto 80, podremos omitir el número de puerto en nuestras peticiones HTTP usadas para acceder a los recursos de nuestra aplicación Web. Por ejemplo, si definimos el puerto 80 como el puerto de escucha, podemos usar la forma `http://hostname/myfile.html` en lugar de `http://hostname:portnumber/myfile.html`.
Definimos un puerto de escucha diferente para las peticiones normales y seguras (usando SSL). Podemos definir el puerto de escucha regular sobre el nodo **Servers** en la Consola de Administración, bajo el pestaña **Configuration/General**, y podemos definir los puertos SSL bajo la pestaña **Configuration/SSL**.

Aplicaciones Web

HTTP y los servicios Web se despliegan de acuerdo a la especificación Servlet 2.2 de Sun Microsystems, que describe el uso de aplicaciones Web como una forma estandarizada de agrupar componentes de una aplicación basada en Web. Estos componentes incluyen páginas JSP, servlets HTTP, y recursos estáticos como páginas HTML o ficheros de imágenes. Cada servidor puede hospedar cualquier número de aplicaciones Web, Normalmente usamos el nombre de la Aplicación Web como parte del URI que usamos para solicitar los recursos de la aplicación Web.
Para más información, puedes ver Desplegar y Configurar Aplicaciones Web.

Las Aplicaciones Web y el Clustering

Las Aplicaciones Web se pueden desplegar en un cluster de Servidores WebLogic. Cuando un usuario solicita un recurso de una aplicación Web, la solicitud es enrutada a uno de los servidores del cluster que hospeda la aplicación. Si una aplicación usa un objeto de sesión, la sesión debe ser replicada en todos los nodos del cluster. Se proporcionan varios métodos para replicar sesiones.

Designar una Aplicación Web por Defecto

Cada servidor y host virtual de nuestro dominio tiene un tipo especial de aplicación Web, llamado "Aplicación Web por Defecto". Esta aplicación Web responde a cualquier solicitud HTTP que no pueda resolver cualquier otra aplicación Web. En contraste a todas las aplicaciones Web, la aplicación Web por defecto no usa el nombre de la aplicación Web como parte del URI. Cualquier aplicación Web destinada a un servidor o host virtual pueden ser declaradas como aplicación Web por defecto. (Destinar una aplicación Web se discute más adelante en esta página. Para más información sobre host virtual, puedes ver Configurar Host Virtuales.
Si no declaramos una aplicación Web por defecto, el Servidor WebLogic crea una por cada servidor o host virtual cuando lo arrancamos. La aplicación Web por defecto se llama `DefaultWebApp_servername`, donde `servername` es el nombre del servidor que hemos arrancado, o en el caso de un servidor virtual, `DefaultWebApp_virtualHostName`.
Si declaramos una aplicación Web que falla al desplegarse correctamente, se lanzará un error y el usuario recibirá un mensaje de error HTTP 400.
Por ejemplo, si nuestra aplicación Web se llama **shopping**, usaríamos la siguiente URI para acceder a un JSP llamado **cart.jsp** de la aplicación Web:
`http://host:port/shopping/cart.jsp`
Sin embargo, si declaráramos **shopping** como la aplicación Web por defecto, accederíamos a **cart.jsp** con la siguiente URI:
`http://host:port/cart.jsp`
(donde **host** es el nombre de host de la máquina que está ejecutando WebLogic Server y `port` es el número de puerto donde el servidor WebLogic está escuchando peticiones).
Para designar una aplicación Web por defecto para un servidor o host virtual, usamos la Consola de Administración:

1. En el panel izquierdo, en el nodo Web Application.
2. Seleccionamos nuestra aplicación Web.

3. En el panel derecho, pulsamos la pestaña `Targets`.
4. Seleccionamos la pestaña `Servers` y movemos el servidor (o host virtual) a la columna **Chosen**. (También podemos dirigir todos los servidores de un cluster seleccionando la etiqueta `Clusters` y mover el cluster a la columna **Chosen**.)
5. Pulsamos **Apply**.
6. Pulsamos el nodo Servers (o host virtual) en el panel izquierdo.
7. Seleccionamos el servidor o host virtual apropiado.
8. En el panel derecho, pulsamos la pestaña `General`.
9. Seleccionamos la pestaña `HTTP`.
10. Seleccionamos una aplicación Web de la lista desplegable etiquetada `Default Web Application`.
11. Pulsamos **Apply**.
12. Si estamos declarando una aplicación Web por defecto para uno o más servidores controlados, repetimos estos pasos por cada servidor controlado.

Configurar el Hosting Virtual

El Hosting virtual nos permite definir nombres de host a los que responden servidores o clusters. Cuando usamos hosting virtual usamos DNS para especificar uno o más nombres de host a los que mapean a direcciones IP de un servidor WebLogic o un cluster y especificamos que aplicaciones Web son servidas por el host virtual. Cuando se usa en un cluster, el balance de carga nos permite un uso más eficiente de nuestro hardware, incluso si uno de los nombres de hosts DNS procesa más solicitudes que otros.
Por ejemplo, podemos especificar que una aplicación Web llamada **books** responde a las peticiones para el nombre del host virtual `www.books.com.`, y que éstas solicitudes son dirigidas a los servidores WebLogic A,B y C, mientras que una aplicación Web llamada **cars** responde al nombre de host virtual `www.autos.com` y estas solicitudes son dirigidas a los servidores WebLogic D y E. Podemos configurar una gran variedad de combinaciones de host virtuales, Servidores WebLogic, clusters y aplicaciones Web, dependiendo de los requerimientos de nuestra aplicación y servidor Web.
Por cada host virtual que definimos también podemos definir separadamente parámetros HTTP y logs de acceso HTTP. Los parámetros HTTP y logs de acceso a un host virtual sobrescriben los del servidor. Podríamos especificar cualquier número de host virtuales. Podemos activar el hosting virtual dirigiendo el host virtual a un servidor o un cluster de servidores. Un host virtual dirigido a un cluster será aplicado a todos los servidores del cluster.

Hosting Virtual y la Aplicación Web por Defecto

También podemos designar una aplicación Web por Defecto para cada host virtual. La aplicación Web por defecto para un host virtual responde a todas las peticiones que no pueden resolver las otras aplicaciones desplegadas en el mismo servidor o cluster del servidor virtual. Al contrario que otras aplicaciones Web, una aplicación Web por defecto no usa el nombre de la aplicación Web (también llamado path de contexto) como parte del URI usado para acceder a los recursos de la aplicación web por defecto.
Por ejemplo, si definimos un host virtual llamado **www.mystore.com** y lo dirigimos a un servidor en el que tenemos desplegada una aplicación Web llamada **shopping**. Accederíamos a un JSP llamado **cart.jsp** desde la aplicación Web **shopping** con la siguiente URI:

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

Sin embargo, si declaramos **shopping** como la aplicación Web por defecto del host virtual **www.mystore.com**, accederíamos a **cart.jsp** con la siguiente URI:

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

Para más información, puedes ver Cómo Resuelve un Servidor WebLogic las peticiones HTTP.

Configurar un Host Virtual

Para definir un host virtual, usamos la consola de administración:

1. Crear un nuevo host virtual.
    - Pulsamos sobre **Services** en el panel izquierdo. El nodo se expande y muestra una lista de servicios.
    - Pulsamos sobre el nodo **virtual hosts**. Si hay definido algún host virtual, el nodo se expandirá y mostrará una lista de host virtuales.
    - Pulsamos sobre **Create a New Virtual Host** en el panel derecho.
    - Introducimos un nombre para representar este host virtual.
    - Introducimos los nombres de los host virtuales, uno por línea. Sólo las peticiones correspondientes a uno de estos nombres de host virtuales serán manejadas por el Servidor o Cluster WebLogic dirigido por este host virtual.
    - (opcional) Asignamos una aplicación Web por defecto para este virtual.
    - Pulsamos **Create**.
2. Definimos los parámetros de loggin y HTTP:
    - (opcional) Pulsamos sobre la pestaña **Logging** y rellanamos los atributos de **HTTP access log**.
    - Seleccionamos la pestaña HTTP y rellenamos los parámetros HTTP.
3. Definir los servidores que responderán a este host virtual.
    - Seleccionamos la pestaña **Targets**.
    - Seleccionamos la pestaña **Servers**. Veremos una lista de servidores disponibles.
    - Seleccionamos un servidor en la columna `available` y usamos el botón de la flecha derecha para moverlo a la columna **chosen**.
4. Definir el cluster que responderá a este host virtual (opcional). Previamente debemos haber definido un Cluster WebLogic.
    - Seleccionamos la pestaña **Targets**.
    - Seleccionamos la pestaña **Clusters**. Veremos una lista de servidores disponibles.
    - Seleccionamos un cluster en la columna **available** y usamos el botón de la flecha derecha para mover el cluster a la columna **Chosen**. El host virtual es dirigido a todos los servidores del cluster.
5. Dirigir aplicaciones Web al host virtual.
    - Pulsamos sobre el nodo **Web Applications** en el panel izquierdo.
    - Seleccionamos la aplicación Web que queremos dirigir.
    - Seleccionamos la pestaña **Targets** en el panel derecho.
    - Seleccionamos la pestaña **Virtual Hosts**.
    - Seleccionamos un host virtual en la columna **available** y usamos el botón de la flecha derecha para moverlo a la columna **chosen**.

Configurar Logs de Acceso HTTP

WebLogic Server puede mantener un log de todas las transacciones HTTP en un fichero de texto, en el formato de log normal o en el formato de log extendido. El formato de log extendido nos permita personalizar la siguiente convención estándar. Podemos configurar los atributos que definen el comportamiento de los logs de acceso HTTP por cada servidor o por cada host virtual que definimos.

Rotación de Log

También podemos elegir la rotación del fichero de log basándonos en el tamaño del fichero o en el tiempo. Cuando se cumplen uno de estos criterios, el fichero de log de accesos actual se cierra y se crea un nuevo fichero. Si no configuramos la rotación, el fichero de logs de accesos logs HTTP crecerá indefinidamente. El nombre de este fichero incluye una parte numérica que se incrementa en cada rotación. Se mantienen ficheros de logs de Accesos HTTP separados para cada Servidor Web que hemos definido.

Configurar el Log de Accesos HTTP Usando la Consola de Administración

Para configurar el log de accesos HTTP usamos la Consola de Administración:

1. Si tenemos configurado hosting virtual:
    - Seleccionamos el nodo **services** en el panel izquierdo.
    - Seleccionamos el nodo **virtual hosts**. El nodo se expande y muestra una lista de host virtuales.
    - Seleccionamos un host virtual.

Si no tenemos configurado el hosting virtual:

    - Seleccionamos el nodo **servers** en el panel izquierdo. El nodo se expande y muestra una lista de servidores.
    - Seleccionamos un servidor.
    - Seleccionamos la pestaña **Logging**.
    - Seleccionamos la pestaña **HTTP**.
2. Activamos la caja **Enable Logging**.
3. Introducimos los valores para nuestro **Log File Name**.
4. Seleccionamos **Common** o **Extended** de la lista desplegable etiquetada **Format**.
5. Seleccionamos el tipo de rotación a **By Size** o **By Date**.
    - **By Size**: Rota el fichero log cuando su tamaño excede el valor introducido en el parámetro **Log Buffer Size**.
    - **By Date**: Rota el fichero de Log después del número de minutos especificado en el parámetro **Rotation Period**.
6. Si hemos seleccionado **Size** como el tipo de rotación, seleccionamos el valor del buffer al valor máximo de bytes que queremos tener en el fichero.
7. Configuramos el parámetro **Flush Every** con el número de segundos después de los cuales el log de acceso escribe entradas de log.
8. Si hemos seleccionado **Date** como el tipo de rotación, seleccionamos el tiempo de rotación a la primera fecha en que queramos que se rote el fichero Log. (Efectivamente sólo si el tipo de rotación se configura como **Date**). Introducimos la fecha usando el formato de `java.text.SimpleDateFormat`, **MM-dd-yyyy-k:mm:ss**.

9. Si hemos seleccionado **Date** como el tipo de rotación, configuramos el periodo de rotación con el número de minutos después del que se rotará el fichero Log.

Formato de Log Común

El formato de log por defecto para la información HTTP es el formato común de http://www.w3.org/Daemon/User/Config/Logging.html#common-logfile-format. Este estándar sigue este patrón:

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

donde:

- **host**
  Es el nombre DNS o el número IP del cliente remoto.
- **RFC931**
  Cualquier información devuelta por **IDENTD** para el cliente remoto; WebLogic Server no soporta identificación de usuario.
- **auth_user**
  Si el usuario del cliente remoto usa un `userid` para autentificación, el nombre de usuario; de otro modo "-".
- **day/month/year:hour:minute:second UTC_offset**
  Día, mes año y hora de día (en formato de 24 horas) con la diferencia horaria entre la hora local y GMT, encerrado entre corchetes.
- **"request"**
  Primera línea de la petición HTTP enviada por el cliente remoto encerrada en dobles comillas.
- **status**
- Código de estado HTTP devuelto por el servidor, si está disponible; de otro modo "-".
- **bytes**
  Número de bytes listados como `content-length` en la cabecera HTTP, sin incluir la cabecera HTTP, si se conoce; sino "-".

Configurar el Log de Accesos HTTP usando el Formato Extendido

WebLogic Server también soporta un formato de fichero extendido, versión 1.0, según se define en la W3C. Este es un estándar emergente, y WebLogic Server sigue el borrador de la especificación de W3C en www.w3.org/TR/WD-logfile.html. La referencia definitiva puede encontrarse en la página www.w3.org/pub/WWW/TR.

El formato de fichero de log extendido nos permite especificar el tipo y orden de la información grabada sobre cada comunicación HTTP. Para activar el formato de información extendida, configuramos el atributo **Format** en la pestaña **HTTP** en la Consola de Administración como `Extended`. (Puedes ver el paso 4 en: Configurar el Log de Accesos HTTP Usando la Consola de Administración).

Podemos especificar qué información se debería grabar en el fichero Log con directivas, incluidas en el propio fichero log. Una directiva empieza una nueva línea y comienza con un signo **#**. Si el fichero log no existe, se crea uno nuevo con las directivas por defecto. Sin embargo, si el fichero de log ya existe cuando se arranca el servidor, debe contener directivas legales al principio del fichero.

Crear Directivas **`Fields`**

La primera línea de nuestro fichero log debe contener una directiva con el número de versión del formato del fichero. También debe contener una directiva `Fields` cerca del principio del fichero:

```
#Version: 1.0
#Fields: xxxx xxxx xxxx ...
```

Donde cada `xxxx` describe los campos de datos a grabar. Los tipos de campos se especifican como simples identificadores, o pueden tomar un formato prefijo de identificador, según se define en la especificación W3C. Aquí tenemos un ejemplo:

```
#Fields: date time cs-method cs-uri
```

Este identificador instruye al servidor a grabar la fecha y la hora de la transacción, el método de solicitud que usó el cliente, y la URI de la solicitud por cada acceso HTTP. Cada campo está separado por un espacio en blanco, y cada registro se añade en una nueva línea en el fichero log.

Nota:
La directiva `#Fields` debe ser seguida por una nueva línea en el fichero log, para que el primer mensaje de log no se añada en la misma línea.

Identificadores de Campos Soportados
Los siguientes identificadores no requieren prefijo:

- `date`
  La fecha en la que se completó la transacción, el campo tiene el tipo `<date>`, según define la especificación W3C.
- `time`
  La hora en la que se completó la transacción, el campo tiene el tipo `<time>`, según define la especificación W3C.
- **time-taken**
  El tiempo que tardó en completarse la transacción, en segundos, el campo tiene el tipo `<fixed>`, según define la especificación W3C. La fecha en la que se completó la transacción, el campo tiene el tipo `<date>`, según define la especificación W3C.
- **bytes**
  El número de bytes transferidos, tiene el tipo `<integer>`.

Observa que el campo **cached** definido en la especificación W3C no está soportado en WebLogic Server.
Los siguientes identificadores requieren prefijos, y no pueden usarse solos. Las combinaciones de prefijos soportadas se explican individualmente.

- Campos relacionados con la dirección IP:
  Estos campos dan la dirección IP y el puerto del cliente o del servidor que responde. Este campo tiene el tipo `<address>`, según lo define la especificación W3C. Los prefijos soportados son:
  - `c-ip`
    La dirección IP del cliente.
  - `s-ip`
    La dirección IP del servidor
- Campos relacionados con el DNS.
  Estos campos ofrecen el nombre de dominio del cliente o del servidor. Este campo tiene el tipo `<name>`, según lo define la especificación W3C. Los prefijos soportados son:
  - `c-dns`
    El nombre de dominio del cliente.
  - `s-dns`
    El nombre de domino del servidor solicitado.
  - `sc-status`
    El código de estado de la respuesta, por ejemplo (404) indicando un estado “File not found”. El tipo del campo es `<integer>`, según lo define la especificación W3C.

- `sc-comment`
  El comentario devuelto con el código de estado, por ejemplo “File not found”. Este campo tiene el tipo `<text>`.
- `cs-method`
  El método de solicitud, por ejemplo GET o POST. Este campo tiene el tipo `<name>`, según define la especificación W3C.
- `cs-uri`
  La URI solicitada completa. Este campo tiene el tipo <uri>, según lo define la especificación W3C.
- `cs-uri-stem`
  Sólo la primera parte de la URI (omitiendo la query). Este campo tiene el tipo `<uri>`, según lo define la especificación W3C.
- `cs-uri-query`
  Sólo la porción query de la URI. Este campo tiene el tipo `<uri>`, según lo define la especificación W3C.

Crear Identificadores de Campos Personalizados
También podemos crear campos definidos por el usuario para incluirlos en un fichero de log de accesos HTTP que use el formato extendido. Para crear un campo personalizado identificamos el campo en el fichero de log **ELF** usando la directiva **Fields** y luego creamos una clase Java que genere la salida deseada. Podemos crear una clase separada para cada campo, o la clase Java puede sacar varios campos. Puedes encontrar un ejemplo de clase Java en Clase Java para Crear un Campo ELF Personalizado.
Para crear un campo personalizado:

1. Incluimos el nombre de campo en la directiva `Fields`, usando la forma:
2. `X-myCustomField.`

   Donde `myCustomField` es el nombre totalmente cualificado de la clase.

3. Creamos una clase Java con el mismo nombre totalmente cualificado de la clase y personalizamos el campo que hemos definido en la directiva `Fields` (por ejemplo, `myCustomField`). Esta clase define la información que queremos almacenar en nuestro campo personalizado. La clase Java debe implementar el siguiente interfase:
4. `weblogic.servlet.logging.CustomELFLogger`

En nuestra clase Java, debemos implementar el método `logField()`, que toma un objeto `HttpAccountingInfo` y un objeto `FormatStringBuffer` como sus argumentos:

   - Usamos el objeto `HttpAccountingInfo` para acceder a los datos de la solicitud y respuesta HTTP que queremos mostrar en el campo personalizado. Los métodos **Getter** se proporcionan para acceder a esta información.
   - Usamos la clase `FormatStringBuffer` para crear los contenidos de nuestro campo personalizado. Los métodos se proporcionan para crear salidas adecuadas.
5. Compilamos la clase Java y añadimos la clase a la sentencia **CLASSPATH** usada para arrancar WebLogic Server. Probablemente necesitaremos modificar las sentencias **CLASSPATH** en los scripts que usamos para arrancar WebLogic Server.

Nota:
No debemos situar esta clase dentro de una aplicación Web o Enterprise en formato expandido o jar.

6. Configuramos WebLogic Server para usar el formato de log extendido.

Nota:
Cuando escribamos la clase Java que define nuestro campo personalizado, no deberíamos ejecutar ningún código que pueda relentizar el sistema (por ejemplo, acceder a bases de datos o ejecutar I/O importantes o llamadas a red). Recuerda que se guarda una entrada en el fichero log de accesos HTTP por cada petición HTTP.

Nota:
Si queremos sacar más de un campo, delimitamos los campos con un caracter tab. Para más información sobre los delimitadores de campos y otros problemas de formateo ELF, puedes visitar http://www.w3.org/TR/WD-logfile-960221.html.

Ejemplo de clase:

```
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"));
    }
}
```

Evitar Ataques de Denegación de Servicio

Un ataque de Denegación-de-Servicio es un intento malicioso de sobrecargar un servidor con peticiones. Un tipo común de ataques es enviar enormes cantidades de datos en un método **HTTP POST**. Podemos seleccionar tres atributos en WebLogic Server para evitar este tipo de ataques. Estos atributos se seleccionan desde la Consola de Administración, bajo **Servers** o **Virtual Hosts**. Si definimos estos atributos para un host virtual, los valores configurados para el host virtual sobrescriben a los configurados para el servidor.

- `PostTimeoutSecs`
  Podemos limitar la cantidad de tiempo que WebLogic Server espera entre bloques de datos recibidos en un POST HTTP.
- `MaxPostTimeSecs`
  Limita la cantidad total de tiempo que gasta un WebLogic Server recibiendo datos POST. Si este limite se dispara, se lanza una `PostTimeoutException` y aparecerá el siguiente mensaje en el log del servidor:
- `Post time exceeded MaxPostTimeSecs.`
- `MaxPostSize`
  Limita el número de bytes de datos recibidos en un POST desde una sola petición. Si se alcanza este límite, se lanzará una excepción `MaxPostSizeExceeded` y se enviará el siguiente mensaje al log del servidor:
- `POST size exceeded the parameter MaxPostSize.`

Se envía un código de error HTTP 413 (Request Entity Too Large) al cliente.
Si el cliente está en modo escucha, obtiene estos mensajes. Si no lo está, se rompe la conexión.

Configurar un Servidor WebLogic para Tunneling HTTP

El tunneling HTTP proporciona una forma de simular una conexión socket con estado entre WebLogic Server y un cliente Java cuando nuestra única opción es usar el protocolo HTTP. Generalmente se usa para enrutar a través de un puerto HTTP en un firewall de seguridad. HTTP es un protocolo sin estado, pero WebLogic Server proporciona tunneling funcionalmente para hacer que la conexión parezca ser una **T3Connection** normal. Sin embargo, podemos esperar alguna perdida de rendimiento en comparación con una conexión de socket normal.

Configurar la Conexión de Tunneling HTTP

Bajo el protocolo HTTP, un cliente sólo podría hacer una petición, y luego aceptar una respuesta del servidor. El servidor podría no comunicar voluntariamente con el cliente, y el protocolo es sin estado, lo que significa que no es posible una conexión continua en los dos sentidos.
El tunneling HTTP de WebLogic simula una **T3Connection** mediante el protocolo, HTTP, evitando estas limitaciones. Hay dos atributos que podemos configurar en la Consola de Administración. Podemos acceder a estos atributos en la sección **Servers**, bajo la pestaña **Tuning** situada debajo de la pestaña **Configuration**. Se aconseja que los dejemos en sus valores por defecto a menos que experimentemos problemas de conexión. Estas propiedades las usan los servidores para determinar si la conexión del cliente es todavía válida, o si el cliente todavía está vivo.

- `Enable Tunneling`
  Activa o desactiva el tunneling HTTP. Por defecto está desactivado.
- `Tunneling Ping`
  Cuando se configura una conexión tunnel HTTP, automáticamente el cliente envía una petición al servidor, por eso el servidor podría ofrecer una respuesta al cliente. El cliente también podría incluir instrucciones en una petición, pero este comportamiento sucede sin importar si la aplicación cliente necesita comunicar con el servidor. Si el servidor no responde (como parte del código de la aplicación) a la petición del cliente dentro del número de segundos configurado en este atributo, lo hace de cualquier modo. El cliente acepta la respuesta y automáticamente envía otra petición inmediatamente.

Por defecto son 45 segundos; el rango válido va de 20 a 900 segundos.

- `Tunneling Timeout`
  Si ha pasado el número de segundos configurado en este atributo desde la última petición enviada por el cliente al servidor (en respuesta a una respuesta), entonces el servidor decide que el cliente está muerto, y termina la conexión tunnel HTTP. El servidor chequea el tiempo en el intervalo especificado por este atributo, cuando de otra forma debería responder a la petición del cliente.

Por defecto son 45 segundos; el rango válido va de 20 a 900 segundos.

Conectar con un Servidor WebLogic desde el Cliente

Cuando nuestro cliente solicita una conexión con un WebLogic Server, todo lo que necesitamos hacer para usar el tunneling HTTP es especificar el protocolo HTTP en la URL. Por ejemplo:

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

Desde el lado del cliente, se añade una etiqueta especial al protocolo http, para que el Servidor WebLogic sepa que ésta es una conexión tunneling, en lugar de una petición HTTP normal. Nuestro código de aplicación no necesita hacer ningún trabajo extra para hacer que esto suceda.
El cliente debe especificar el puerto en la URL, incluso si el puerto es 80. Podemos configurar nuestro Servidor WebLogic para que escuche peticiones HTTP en cualquier puerto, aunque la elección más común es el puerto 80 porque normalmente tiene permitido el paso por los firewalls.
Especificamos el puerto de escucha para el Servidor WebLogic en la Consola de Administración bajo el nodo **"Servers"**, en la pestaña **"Network"**.

Usar I/O Nativa para Servir Ficheros Estáticos (Sólo Windows)

Cuando se ejecuta WebLogic Server sobre Windows NT/2000 podemos especificar que WebLogic Server use la operación **TransmitFile** del sistema en lugar de usar los métodos Java para servir ficheros estáticos como ficheros HTML, los ficheros de texto, y ficheros de imágenes. Usando I/O nativa podemos proporcionar mejoras de rendimiento cuando se sirven grandes ficheros estáticos.
Para usar I/O nativa, añadimos dos parámetros al descriptor de despliegue **web.xml** de una aplicación Web que contenga los ficheros a servir usando I/O nativa. El primer parámetro, `weblogic.http.nativeIOEnabled` debería configurarse como **TRUE** para permitir el servicio de ficheros por I/O nativa. El segundo parámetro `weblogic.http.minimumNativeFileSize` configura el tamaño de fichero para usar I/O nativa. Si el fichero a servir es mayor que este tamaño, se usara la I/O nativa. Si no especificamos este parámetro, se usa un valor de 400 bytes.
Generalmente, la I/O nativa proporciona una mayor ganancia de rendimiento cuando sirve ficheros mayores; sin embargo, cuando se incrementa la carga de una máquina que ejecuta WebLogic Server, esta ganancia disminuye. Podríamos necesitar experimentar para encontrar el valor correcto de `weblogic.http.minimumNativeFileSize`.
El siguiente ejemplo, muestra las entradas completas que deberíamos añadir al descriptor de desarrollo **web.xml**. Estas entradas deben situarse en el fichero **web.xml** después del elemento `<distributable>` y antes del elemento `<servlet>`.

```
<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>
```

Desplegar y Configurar Aplicaciones Web

Introducción

Un Aplicación Web contiene recursos de aplicación como Servlets, JavaServer Pages (JSP), librerías de etiquetas JSP, y recursos estáticos como páginas HTML y ficheros de imágenes. Una aplicación Web también puede definir enlaces a recursos fuera de la aplicación como Enterprise JavaBeans (EJB). Las aplicaciones Web usan un descriptor de despliegue estándar J2EE en conjunción con un descriptor de despliegue específico de WebLogic para definir los recursos y otros parámetros de operación.
Las páginas JSP y los servlets HTTP pueden acceder a todos los servicios y APIs disponibles en el Servidor WebLogic. Estos servicios incluyen EJBs, conexiones a bases de datos mediante JDBC, JavaMessaging Service (JMS), XML, y más.
Las aplicaciones Web usan una estructura de directorio estándar definida en la especificación J2EE y pueden desplegarse como una colección de ficheros que usan esta estructura de directorio (este tipo de despliegue se llama formato de directorio expandido) o como un fichero individual llamado un fichero **.war**. El despliegue de aplicaciones Web en formato de directorio expandido se recomienda principalmente mientras desarrollamos nuestra aplicación. Mientras que en entornos de producción se recomienda desplegar una aplicación Web usando un fichero **.war**.

Pasos para Desplegar una Aplicación Web

Para desplegar una Aplicación Web:

1. Distribuimos los recursos (servlets, JSPs, ficheros estáticos y descriptores de despliegue) en el formato de directorio prescrito. Para más información puedes ver Estructura de Directorio.
2. Escribimos el descriptor de despliegue de la aplicación Web (web.xml). En este paso registramos los servlets, definimos los parámetros de inicialización de los servlets, registramos las librerías de etiquetas JSP, definimos las restricciones de seguridad, y definimos otros parámetros de la Aplicación Web.
3. Creamos el descriptor de despliegue específico de WebLogic (weblogic.xml). En este paso definimos las propiedades JSP, los mapeos JNDI, los mapeos de los roles de seguridad y los parámetros de sesión HTTP. Si no necesitamos definir ninguno de los atributos de este fichero, no necesitamos crearlo.
4. Almacenamos los ficheros de la estructura de directorios anterior en un fichero **.war**. Sólo usamos el archivado cuando la Aplicación Web está preparada para desplegarse en un entorno de producción. (Durante el desarrollo podríamos encontrar más conveniente para actualizar los componentes individuales de nuestra aplicación Web, desplegarla en el formato de directorio extendido). Para crear un archivo **.war**, usamos esta línea de comandos desde el directorio raíz que contiene nuestra aplicación Web:
5.
```
jar cv0f myWebApp.war .
```

   Este comando crea un fichero de Aplicación Web llamado `myWebApp.war`.

6. Desplegar la Aplicación Web sobre el Servidor WebLogic de una de estas dos formas: usando la Consola de Administración o copiando la Aplicación Web dentro del directorio de aplicaciones de nuestro dominio.

Para desplegar una Aplicación Web en formato **war** usando la Consola de Administración (no podemos desplegar una Aplicación Web en formato de directorio extendido usando este procedimiento):

- Seleccionamos el nodo **Web Applications** en el panel izquierdo.
- Pulsamos **Install a New Web Application**.
- Navegamos a la localización del fichero **.war** en nuestro sistema de ficheros.
- Pulsamos **Upload**.

Este procedimiento crea una nueva entrada en el fichero **config.xml** que contiene la configuración para nuestra Aplicación Web y copia nuestra Aplicación a una localización interna.
Para desplegar una Aplicación Web (en formato archivado o expandido) copiando:

- Copiamos un fichero **.war** o un directorio de más alto nivel que contiene una Aplicación Web en un formato de directorio extendido en el directorio `mydomain/config/applications` de nuestra distribución WebLogic Server. (Donde `mydomain` es el nombre de nuestro dominio). Tan pronto como se complete la copia, WebLogic Server despliega automáticamente la Aplicación Web.
- (Opcional) Usamos la Consola de Administración para configurar la Aplicación Web. Una vez que cambiamos cualquier atributo (ver paso 6) de la Aplicación Web, la configuración es escrita en el fichero **config.xml** y la Aplicación Web será desplegada estáticamente la próxima vez que reiniciemos el Servidor WebLogic. Si no usamos la Consola de Administración, nuestra Aplicación Web todavía es desplegada automáticamente cada vez que arrancamos el Servidor WebLogic, incluso aunque la información de configuración no se haya grabado en el fichero **config.xml**.

Nota:
Si despliegas tu Aplicación Web en formato expandido, debes leer Modificar Componentes de una Aplicación Web.

Nota:
*Si modificamos cualquier componente de un fichero **.war** en su localización original en nuestro sistema de ficheros, debemos re-desplegar nuestro fichero **.war** subiéndolo de nuevo desde la Consola de Administración.*

7. Asignamos atributos de despliegue a nuestra Aplicación Web:
   - Abrimos la Consola de Administración.
   - Seleccionamos el nodo **Web Applications**.
   - Seleccionamos nuestra Aplicación Web.
   - Asignamos nuestra Aplicación Web a un Servidor WebLogic, a un Cluster o a un Host Virtual.
   - Seleccionamos la pestaña **File** y definimos los atributos apropiados.

Estructura de Directorio

Desarrollamos nuestra Aplicación Web dentro de una estructura de directorio específica para que pueda ser archivada o desplegada sobre un Servidor WebLogic, u otro servidor compatible con Servlet 2.2. Todos los servlets, clases, ficheros estáticos y otros recursos que pertenecen a una Aplicación Web están organizados bajo un árbol de directorios. La raíz de este árbol define el documento raíz de nuestra Aplicación Web. Todos los ficheros bajo el directorio raíz pueden servirse a los clientes, excepto los ficheros bajo los directorios especiales **WEB-INF** y **META-INF** localizados bajo el directorio raíz. Nombramos el directorio raíz con el nombre de nuestra Aplicación Web. Este nombre se usará para resolver las solicitudes de componentes de la Aplicación Web.
Los ficheros privados deben situarse en el directorio **WEB-INF**, bajo el directorio raíz. Todos los ficheros que haya en este directorio son privados, y no serán servidos a los clientes.

- `WebApplicationName/`
  Situamos nuestros ficheros estáticos, como ficheros HTML o JSP en este directorio (o un subdirectorio). Este directorio es el documento raíz de nuestra Aplicación Web.
  - `/WEB-INF/web.xml`
    El descriptor de despliegue de la Aplicación Web que la configura.
  - `/WEB-INF/weblogic.xml`
    El descriptor de despliegue específico de WebLogic que define cómo se mapen los recursos nombrados en el fichero **web.xml** a los recursos residentes en algún lugar del Servidor webLogic. Este fichero también se usa para definir atributos de sesiones JSP y HTTP.
  - `/WEB-INF/classes`
    Contiene las clases del lado del servidor como servlets HTTP y clases de utilidad.
  - `/WEB-INF/lib`
    Contiene los ficheros **.jar** usados por la Aplicación Web.

Desplegar y Re-Desplegar Aplicaciones Web

El procedimiento que usamos para desplegar y re-desplegar una Aplicación Web depende de si la vamos a desplegar en formato expandido o comprimido. Cuando modificamos un componente de una Aplicación Web también debemos re-desplegarla sobre el Servidor WebLogic para servir el componente modificado. Estos procedimientos se describen en esta sección.

Modificar Componentes de una Aplicación Web

Cuando modificamos cualquier componente de una Aplicación Web (como una clase Servlet, un fichero HTML o JSP, o uno de los descriptores de despliegue), la nueva versión del componente no puede ser servida por el Servidor WebLogic hasta que hayamos re-desplegado la Aplicación Web. El procedimiento usado para re-desplegar depende de si la desplegamos usando el formato comprimido (**.war**) o expandido.
Componentes en formato .war
Cuando editamos un componente de una Aplicación Web que se ha desplegado en un fichero war, necesitamos **re-jar** (re-empaquetar) el archivo y luego subir el fichero **.war** usando uno de los procedimientos descritos en el paso 5 de Pasos para Desplegar una Aplicación Web.
Componentes en Formato de Directorio Expandido
Cuando editamos un componente de una Aplicación Web que se ha desplegado en formato de directorio expandido, debemos tener en mente que WebLogic actualiza los componentes de forma diferente:

- **Ficheros JSP**
  Los ficheros JSP se re-despliegan basándose en la selección del atributo `pageCheckSeconds` que definimos en el descriptor de despliegue específico de WebLogic, **weblogic.xml**, de nuestra Aplicación Web. Este atributo define el intervalo de tiempo en el cual el Servidor WebLogic chequea los ficheros JSP

para ver si han sido modificados. Si se selecciona a 0 (cero), las páginas son chequeadas en cada petición. Si selecciona a -1, se desactiva el chequeo y recompilación de las páginas.

Nota:
Los ficheros JSP se re-despliegan automáticamente sólo en el Servidor de Administración. Si queremos que los JSPs sean re-desplegados en el servidores controlados dirigidos por la Aplicación Web, debemos re-desplegar la Aplicación Web (ver más abajo).

- **Servlets**
  Los servlets JSP se re-despliegan basándose en la selección del atributo `Reload Period` que definimos en el descriptor de despliegue específico de WebLogic, **weblogic.xml**, de nuestra Aplicación Web. Este atributo define el intervalo de tiempo en el cual el Servidor WebLogic chequea las clases Servlets para ver si han sido modificadas. Si se selecciona a 0 (cero), las clases son chequeadas en cada petición. Si selecciona a -1, el Servidor WebLogic no chequeará para ver si las clases han sido modificadas.
- **HTML y otros ficheros estáticos**
  Si modificamos un HTML u otro fichero estático, como un fichero de imagen o de texto, debemos re-desplegar la Aplicación Web para que el Servidor WebLogic pueda reconocer los cambios.

Re-Desplegar una Aplicación Web

Usamos uno de estos tres procedimientos para re-desplegar una Aplicación Web:

- Usando la Consola de Administración:
  1. Seleccionamos el nodo **Web Application**.
  2. Des-seleccionamos la caja **Deployed** en el panel derecho.
  3. Pulsamos **Apply**.
  4. Seleccionamos la caja **Deployed** en el panel derecho.
  5. Pulsamos **Apply**.
- Modificando el fichero **REDEPLOY**:
  1. Creamos un subdirectorio llamado **WEB-INF**, bajo el directorio raíz de nuestra Aplicación Web.
  2. Creamos un fichero vacío llamado **REDEPLOY** y lo grabamos en el directorio **WEB-INF**.
  3. Para re-desplegar la Aplicación Web, modificamos el fichero **REDEPLOY** abriéndolo, modificando los contenidos (añadir un espacio es la forma más fácil de hacer esto), y luego grabando el fichero. De forma alternativa, en máquinas UNIX, podemos usar el comando `touch` sobre el fichero **REDEPLOY**.
- Copiando de nuevo el fichero war en el directorio **applications** (sólo se aplica al despliegue automático).

Nota:
Re-desplegar una Aplicación Web, también la re-despliega a todos los servidores controlados dirigidos por esta Aplicación Web.

Desplegar Aplicaciones Web como parte de una Aplicación Enterprise

Podemos desplegar una Aplicación Web como parte de una Aplicación Enterprise. Una Aplicación Enterprise es una unidad de despliegue J2EE que empaqueta juntas Aplicaciones Web, EJBs y Adaptadores de Recursos en una sola unidad desplegable. Si desplegamos una Aplicación Web como parte de una Aplicación Enterprise, podemos especificar un string que se usa en lugar del nombre real de la Aplicación Web cuando el Servidor WebLogic sirve una petición para la Aplicación Web. Especificamos el nuevo nombre con el elemento `<context-root>` en el descriptor de despliegue **application.xml** para la Aplicación Enterprise.
Por ejemplo, para una Aplicación Web llamada **oranges**, normalmente solicitaríamos un recurso de esa Aplicación Web con una URL como esta:
`http://host:port/oranges/catalog.jsp`
Si la Aplicación Web **oranges** se empaqueta en una Aplicación Enterprise, podemos especificar un valor para el `<context-root>` como se ve en el siguiente ejemplo:

```
<module>
<wedestacar>
<web-uri>oranges.war</web-uri>
<context-root>fruit</context-root>
</wedestacar>
</module>
```

Entonces usaríamos la siguiente URL para acceder al mismo recurso de la Aplicación Web **oranges**:
`http://host:port/fruit/catalog.jsp`

URIs y Aplicaciones Web

Construiremos la URL usada para acceder a una Aplicación Web desde un cliente usando el siguiente patrón:
`http://hoststring/ContextPath/servletPath/pathInfo`
donde:

- `hoststring`
  es el nombre del host que es mapeado a un host virtual o `hostname:portNumber`
- `ContextPath`
  es el nombre de nuestra Aplicación Web.
- `servletPath`
  es un servlet que está mapeado a `servletPath`.
- `pathInfo`
  es la porción que resta de la URL, normalmente un nombre de fichero.

Si estamos usando hosting virtual, podemos sustituir el nombre del host virtual por la porción `hoststring` de la URL.

Configurar Servlets

Los Servlets se registran o configuran como una parte de una Aplicación Web. Para registrar un servlet, añadimos varias entradas al descriptor de despliegue de la Aplicación Web. La primera entrada bajo, el elemento `<servlet>` define el nombre del servlet y la clase compilada que ejecuta el servlet. Este elemento también contiene definiciones para los parámetros de inicialización y roles de seguridad del servlet. La segunda entrada, bajo el elemento `<servlet-mapping>` define el patrón URL que llama este servlet.

Mapeo de Servlets

El mapeo de Servlets controla el modo en que accedemos a un servlet. Los siguientes ejemplo, demuestran algunas de las formas que podemos usar para mapear servlets en nuestra aplicación Web.

```
<servlet>
  <servlet-name>watermelon</servlet-name>
  <servlet-class>myservlets.watermelon</servlet-class>
</servlet>

<servlet>
  <servlet-name>garden</servlet-name>
  <servlet-class>myservlets.garden</servlet-class>
</servlet>

<servlet>
  <servlet-name>list</servlet-name>
  <servlet-class>myservlets.list</servlet-class>
</servlet>

<servlet>
  <servlet-name>kiwi</servlet-name>
  <servlet-class>myservlets.kiwi</servlet-class>
</servlet>

<servlet-mapping>
  <servlet-name>watermelon</servlet-name>
  <url-pattern>/fruit/summer/*</url-pattern>
</servlet-mapping>

<servlet-mapping>
  <servlet-name>garden</servlet-name>
  <url-pattern>/seeds/*</url-pattern>
</servlet-mapping>

<servlet-mapping>
  <servlet-name>list</servlet-name>
  <url-pattern>/seedlist</url-pattern>
</servlet-mapping>

<servlet-mapping>
  <servlet-name>kiwi</servlet-name>
  <url-pattern>*.abc</url-pattern>
</servlet-mapping>
```

URL	**Servlet invocado**
http://host:port/mywebapp/fruit/summer/index.html	watermelon
http://host:port/mywebapp/fruit/summer/index.abc	watermelon
http://host:port/mywebapp/seedlist	list
http://host:port/mywebapp/seedlist/index.html	El Servlet por defecto, si está configurado, o un mensaje de error **HTTP 404 file not found**. Si el mapeo para el servlet list había sido /seedlist*, se llamará al servlet list.
http://host:port/mywebapp/seedlist/pear.abc	kiwi Si el mapeo para el servlet list había sido /seedlist*, se llamará al servlet list.
http://host:port/mywebapp/seeds	Garden
http://host:port/mywebapp/seeds/index.html	Garden
http://host:port/mywebapp/index.abc	Kiwi

Parámetros de Inicialización de Servlets

Definimos los parámetros de inicialización para los servlets en el descriptor de despliegue de la Aplicación Web, en el elemento `<init-param>` del elemento `<servlet>`, usando etiquetas `<param-name>` y `<param-value>`. Por ejemplo, el siguiente listado de configuración de los parámetros de inicialización de Servlets:

```
<servlet>
  <servlet-name>HelloWorld2</servlet-name>
  <servlet-class>examples.servlets.HelloWorld2</servlet-class>

  <init-param>
    <param-name>greeting</param-name>
    <param-value>Welcome</param-value>
  </init-param>

  <init-param>
    <param-name>person</param-name>
    <param-value>WebLogic Developer</param-value>
  </init-param>
</servlet>
```

Configurar JSP

Desplegamos los ficheros JSP situándolos en la raíz (o en un subdirectorio bajo la raíz) de una Aplicación Web. Los parámetros de configuración adicionales para JSP están definidos en el elemento `<jsp-descriptor>` del descriptor de despliegue específico de WebLogic, **weblogic.xml**. Estos parámetros definen las siguientes funcionalidades:

- Opciones del compilador JSP.
- Depurado.
- La frecuencia del chequeo que el Servidor WebLogic hace buscando JSPs actualizados que necesitan ser recompilados.
- Codificación de caracteres

Configurar Librerías de Etiquetas JSP

Weblogic Server, en concordancia con la especificación Servlet 2.2 proporciona la habilidad de crear y usar etiquetas JSP personalizadas. Estas etiquetas son clases Java que pueden ser llamadas desde dentro de un página JSP. Para crear etiquetas JSP personalizadas, las situamos en una librería de etiquetas y definimos su comportamiento en un descriptor de librería de etiquetas (TLD). Este TLD debe estar disponible para la Aplicación Web que contiene el JSP definiéndolo en el descriptor de despliegue de la Aplicación Web. Es una buena idea situar el fichero TLD en el directorio **WEB-INF** de nuestra Aplicación Web, porque este directorio nunca está disponible públicamente.
En el descriptor de despliegue de la Aplicación Web, definimos un patrón URI para la librería de etiquetas. Este patrón URI debe corresponder con al valor de la directiva `taglib` en nuestras páginas JSP. También definimos la localización del TLD. Por ejemplo, si la directiva `taglib` en la página JSP es:

```
<%@ taglib uri="myTaglib" prefix="taglib" %>
```

y el TLD está localizado en el directorio **WEB-INF** de nuestra Aplicación Web, podríamos crear la siguiente entrada en el descriptor de despliegue de la Aplicación Web:

```
<taglidestacar>
  <taglib-uri>myTaglib</taglib-uri>
  <tablig-location>WEB-INF/myTLD.tld</taglib-location>
</taglidestacar>
```

WebLogic Server también incluye muchas etiquetas JSP personalizadas que podemos usar en nuestras aplicaciones. Estas etiquetas realizan cacheo, facilitan la consulta de control de flujo basado en parámetros, y facilitan la iteración sobre conjuntos de objetos.

Configurar Páginas de Bienvenida

WebLogic Server nos permite seleccionar una página que es servida por defecto si la URL solicitada es un directorio. Esta característica puede hacer nuestro sitio más fácil de usar, porque el usuario puede teclear una URL sin dar un nombre de fichero específico.
Las páginas de bienvenida se definen a nivel de Aplicación Web. Si nuestro servidor está hospedando varias Aplicaciones Web, necesitamos definir páginas de bienvenida separadas para cada Aplicación Web.
Para definir las páginas de bienvenida, editamos el descriptor de despliegue de la Aplicación Web, **web.xml**. Si no definimos nuestra página de bienvenida, el Servidor WebLogic buscará los siguientes ficheros en el siguiente orden y servirá el primero que encuentre:

1. index.html
2. index.htm
3. index.jsp

Seleccionar un Servlet por Defecto

Cada Aplicación Web tiene un Servlet por defecto. Este servlet puede ser un servlet que especifiquemos nosotros, o, si no lo especificamos, el Servidor WebLogic usa un servlet interno llamado **FileServlet** como servlet por defecto. Para más información puedes ver Como Resuelve las Peticiones HTTP el Servidor WebLogic.
Podemos registrar cualquier servlet como servlet por defecto. Escribir nuestro propio servlet por defecto nos permite usar nuestra propia lógica para decidir cómo manejar una petición que cae dentro del servlet por defecto.
Seleccionar un servlet por defecto reemplaza el `FileServlet` y debería hacerse cuidadosamente, porque `FileServlet` se usa para servir la mayoría de los ficheros, como ficheros de texto, HTML, imágenes, etc. Si esperamos que nuestro servlet por defecto sirva dichos ficheros, necesitaremos escribir esa funcionalidad en nuestro servlet por defecto.
Para seleccionar un servlet por defecto definido por el usuario:

1. Definimos nuestro servlet como se describe en Configurar Servlets.
2. Mapeamos nuestro servlet por defecto con un patrón url de "/". Esto hará que nuestro servlet por defecto responda a todos los tipos de ficheros excepto aquellos con extensión ***.htm** o ***.html**, que son mapeados internamente a `FileServlet`.
   Si también queremos que nuestro servlet por defecto responda a los ficheros que terminan en ***.htm** o ***.html**, debemos mapear dichas extensiones hacia nuestro servlet por defecto, además de mapear "/". Para más información sobre el mapeo de servlets puedes ver Configurar Servlets.
3. Si todavía queremos que `FileServlet` sirva los ficheros con otras extensiones, mapeamos dichas extensiones hacia `FileServlet` (además de mapearlas hacia

nuestro servlet por defecto). Por ejemplo, si queremos que `FileServlet` sirva los ficheros **gif**, mapeamos ***.gif** hacia `FileServlet`.

Cómo Resuelve WebLogic Server Peticiones HTTP

Cuando WebLogic Server recibe una petición HTTP, la resuelve analizando las distintas partes de la URL y usando esta información para determinar qué aplicación Web o servidor debería manejarla. Abajo tenemos varios ejemplos de combinaciones de peticiones para Aplicaciones Web, host virtuales, servlets, JSPs y ficheros estáticos y el respuesta resultante.

Nota:
Si empaquetamos nuestra Aplicación Web como parte de una Aplicación Enterprise podemos proporcionar un nombre alternativo para una Aplicación Web que se usa para resolver la petición hacia la Aplicación Web. Puedes encontrar más información en Desplegar Aplicaciones Web como parte de una Aplicación Enterprise.

La siguiente tabla proporciona algunos ejemplos de URLs y los ficheros servidos por el Servidor WebLogic. La columna "Index Directories Checked " se refiere al atributo `Index Directories` que controla si se ofrece un listado de directorio si no se solicita ningún fichero específicamente. Seleccionamos `Index Directories` usando la Consola de Administración, sobre el nodo **Web Applications**, bajo la pestaña **Configuration/Files**.

URL	Index Directories Checked?	Fichero servidor en la respuesta
http://host:port/apples	no	Fichero de bienvenida definido en la aplicación Web apples
http://host:port/apples	si	Listado del directorio de nivel superior de la aplicación Web apples.
http://host:port/oranges/naval	no importa	El Servlet mapeado con <url-pattern> de /naval en la aplicación web oranges
http://host:port/naval	ni importa	El Servlet mapeado con <url-pattern> de /naval en la aplicación web oranges y esta aplicación está definida como aplicación web por defecto.
http://host:port/apples/pie.jsp	no importa	pie.jsp, del directorio de más alto nivel de la aplicación apples.
http://host:port	si	Listado del directorio de nivel superior de la aplicación web por defecto.
http://host:port	no	Fichero de bienvenida de la aplicación web por defecto.
http://host:port/apples/myfile.html	no importa	myfile.html, del directorio de nivel superior de la aplicación web apples.
http://host:port/myfile.html	no importa	myfile.html, del directorio de nivel superior de la aplicación web por defecto.
http://host:port/apples/images/red.gif	no importa	red.gif, del subdirectorio images del directorio de más alto nivel de la aplicación web apples
http://host:port/myFile.html donde myfile.html no existe en la aplicación web apples y no se ha definido un servlet por defecto	no importa	Error 404
http://www.fruit.com/	no	Fichero de bienvenida de la aplicación web por defecto para un host virtual con el nombre de host www.fruit.com.
http://www.fruit.com/	si	Listado del directorio del nivel superior de la aplicación web por defecto de un host virtual con el nombre www.fruit.com.
http://www.fruit.com/oranges/myfile.html	no importa	myfile.html, de la aplicación web oranges que está dirigida a un host virtual con el nombre www.fruit.com.

Personalizar Respuestas de Error HTTP

Podemos configurar WebLogic Server para responder con nuestras páginas Web personalizadas u otros recursos HTTP cuando ocurre un error HTTP o una excepción Java particular, en lugar de responder con las páginas de respuesta de error estándar de WebLogic Server.
Definimos las páginas de error personalizadas en el elemento `<error-page>` del descriptor de despliegue de la Aplicación Web (**web.xml**).

Usar CGI con WebLogic Server

WebLogic Server proporciona funcionalidades para soportar nuestros scripts Common Gateway Interface (CGI). Para nuevos proyectos, sugerimos usar Servlets HTTP o JavaServer Pages.
WebLogic Server soporta todos los CGI mediante un servlet interno de WebLogic llamado `CGIServlet.TouseCGI`, registrando el `CGIServlet` en el descriptor de despliegue de la Aplicación Web.

Configurar WebLogic Server para usar CGI

Para configurar CGI en WebLogic Server:

1. Declaramos `CGIServlet` en nuestra Aplicación Web usando los elementos `<servlet>` y <servlet-mapping>. el nombre de la clase para el `CGIServlet` es **weblogic.servlet.CGIServlet**.
2. Registramos los siguientes parámetros de inicialización para `CGIServlet` definiendo los siguientes elementos `<init-param>`:
   - `cgiDir`
     El path al directorio que contiene nuestros scripts CGI. Podemos especificar varios directorios, separado por un “;” (Windows) o un “:” (Unix). Si no especificamos `cgiDir`, el directorio por defecto será un directorio llamado **cgi-bin** bajo la raíz de la Aplicación Web.
   - `extension mapping`
     Mapea una extensión de fichero hacia el intérprete o ejecutable que ejecuta el script. Si el script no requiere un ejecutable, se podría omitir este parámetro de inicialización.
     Los mapeos de extensiones de `<param-name>` deben empezar con un asterisco seguido por un punto, seguido por la extensión de fichero, por ejemplo: ***.pl**.
     El `<param-value>` contiene el path del intérprete o ejecutable que ejecuta el script. Podemos crear varios mapeos creando elementos `<init-param>` para cada uno.

Entradas de ejemplo incluidas en el descriptor de despliegue de la aplicación Web cuando se registra `CGIServlet`:

```
<servlet>
  <servlet-name>CGIServlet</servlet-name>
  <servlet-class>weblogic.servlet.CGIServlet</servlet-class>

  <init-param>
    <param-name>cgiDir</param-name>
    <param-value>
      /bea/wlserver6.0/config/mydomain/applications/myWebApp/cgi-bin
    </param-value>
  </init-param>

  <init-param>
    <param-name>*.pl</param-name>
    <param-value>/bin/perl.exe</param-value>
  </init-param>
</servlet>
...
<servlet-mapping>
  <servlet-name>CGIServlet</servlet-name>
  <url-pattern>/cgi-bin/*</url-pattern>
</servlet-mapping>
```

Solicitar un Script CGI

La URL usada para solicitar un script perl debe seguir el patrón:
`http://host:port/myWebApp/cgi-bin/myscript.pl`
donde:

- `host:port`
  es el nombre del host y el número de puerto del Servidor WebLogic
- `cgi-bin`
  es el nombre del patrón url mapeado al `CGIServlet`,
- `myWebApp`
  es el nombre de nuestra aplicación Web
- `myscript.pl`
  es el nombre del script Perl que está localizado en el directorio especificado por el parámetro de inicialización `cgiDir`.

Servir Recursos del Classpath con el ClasspathServlet

Si necesitamos servir clases u otros recursos desde el CLASSPATH del sistema, o desde el directorio `WEB-INF/classes` de la aplicación Web, podemos usar un servlet especial llamado `ClasspathServlet`. Este servlet es útil para aplicaciones que usan applets o clientes RMI y requieren acceder a clases del lado del servidor. `ClasspathServlet` está registrado implícitamente y está disponible desde cualquier aplicación.
Hay dos formas en las que podemos usar `ClasspathServlet`:

- Para servir un recurso desde el CLASSPATH del sistema, llamamos al recurso con una URL como esta:
- `http://server:port/classes/my/resource/myClass.class`
- Para servir un recurso desde el directorio `WEB-INF/classes` de una aplicación Web, llamamos al recurso con una URL como esta:
- `http://server:port/myWebApp/classes/my/resource/myClass.class`

En este caso, el recurso está localizado en el siguiente directorio, en relación a la raíz de la aplicación Web:

```
WEB-INF/classes/my/resource/myClass.class
```

Aviso:
Como `ClasspathServlet` sirve cualquier recurso localizado en el CLASSPATH del sistema, no deberíamos situar recursos que no deberían estar disponibles públicamente en el CLASSPATH del sistema.

Pasar Peticiones a otro Servidor WebLogic

Cuando usamos WebLogic Server como nuestro servidor Web principal, también podríamos querer configurarlo para pasar (actuar como proxy) ciertas peticiones a un servidor HTTP secundario, como Netscape Enterprise Server, Apache, o Microsoft Internet Information Server. Cualquier petición proxy es redirigida a una URL específica. Incluso podemos pasarla a otro servidor Web en una máquina diferente. Nuestras peticiones proxy se basan en la URL de la petición entrante.
El `HttpProxyServlet` (proporcionado como parte de la distribución) toma una petición HTTP, la redirige a la URL proxy, y envía la respuesta de vuelta al cliente a través del Servidor WebLogic. Para usar el proxy, debemos configurarlo en una Aplicación Web y desplegar esa Aplicación Web sobre el Servidor WebLogic que está redirigiendo las peticiones.

Seleccionar un Proxy a un Servidor HTTP Secundario

Para seleccionar un proxy a un servidor HTTP secundario:

1. Registramos el servlet proxy en nuestro descriptor de despliegue de Aplicación Web (ver Ejemplo de web.xml para usar con ProxyServlet). La aplicación Web debe ser la Aplicación Web por defecto del servidor que está respondiendo a las peticiones. El nombre de la clase para el servlet proxy es `weblogic.t3.srvr.HttpProxyServlet`.
2. Definimos un parámetro de inicialización para el `ProxyServlet` con un `<param-name>` de `redirectURL` y un `<param-value>` que contenga la URL del servidor al que se deberían pasar las peticiones.
3. Mapeamos el `ProxyServlet` a un `<url-pattern>`. Específicamente, mapeamos las extensiones de ficheros que deseamos pasar (proxy), por ejemplo ***.jsp**, o ***.html**. Usamos el elemento `<servlet-mapping>` en el descriptor de despliegue de la Aplicación Web.

Si seleccionamos `<url-pattern>` a "/", entonces cualquier petición que no pueda ser resuelta por el Servidor WebLogic será pasara al servidor remoto. Sin embargo, también debemos mapear específicamente las extensiones ***.jsp**, ***.html**, y ***.html** si queremos pasar los ficheros que terminan en estas extensiones.

4. Desplegamos la Aplicación en el Servidor WebLogic que redirige las peticiones entrantes.

Ejemplo de web.xml para usar con ProxyServlet

Lo siguiente es un ejemplo de un descriptor de despliegue de una Aplicación Web para usar el ProxyServlet:

```
<!-- 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.HttpProxyServlet</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>
```

Pasar Peticiones (proxy) a un Cluster WebLogic

El `HttpClusterServlet` (proporcionado con la distribución de WebLogic Server) pasa peticiones desde un servidor WebLogic a otros servidores WebLogic en un Cluster WebLogic. `HttpClusterServlet` proporciona balance de carga y protección de fallos para las peticiones HTTP pasadas.

Configurar el HttpClusterServlet

Para configurar el `HttpClusterServlet`:

1. Configuramos el ejemplar de WebLogic Server que pasará las peticiones a un cluster de servidores WebLogic. Usamos la Consola de Administración:
    - Creamos una nueva Aplicación Web en nuestro dominio.
    - Creamos un nuevo Servidor en el dominio, o usamos el servidor por defecto.
    - Asignamos la aplicación Web que hemos creado como la aplicación Web por defecto para el servidor que acabamos de crear.
2. Registramos el `HttpClusterServlet` en el descriptor de despliegue de la aplicación Web que hemos creado (puedes ver un ejemplo en Ejemplo de Descriptor de Despliegue para el HttpClusterServlet). La Aplicación Web debe ser la aplicación web por defecto del servidor que está respondiendo a las peticiones. Puedes encontrar más información en Designar una Aplicación Web por Defecto.

El nombre de la clase para `HttpClusterServlet` es `weblogic.servlet.internal.HttpClusterServlet`.

3. Definimos los parámetros de inicialización apropiados para `HttpClusterServlet`. Los definimos con el elemento `<init-param>` en el descriptor de despliegue de la aplicación Web. Debemos definir el parámetro `defaultServers`, y, cuando sea apropiado, los parámetros adicionales según se definen en la siguiente tabla.
4. Mapeamos el Proxy Servlet a un `<url-pattern>`. Específicamente, mapeamos las extensiones de ficheros que deseamos pasar (proxy), por ejemplo ***.jsp**, o ***.html**. Usamos el elemento `<servlet-mapping>` en el descriptor de despliegue de la Aplicación Web.

Si seleccionamos `<url-pattern>` a "/", entonces cualquier petición que no pueda ser resuelta por el Servidor WebLogic será pasada al servidor remoto. Sin embargo, también debemos mapear específicamente las extensiones ***.jsp**, ***.html**, y ***.html** si queremos pasar los ficheros que terminan en estas extensiones.

Otra forma de configurar el `url-pattern` es mapear un `url-pattern` como **/foo** y luego configurar el parámetro `pathTrim` como `foo`, lo que elimina `foo` de la URL pasada (proxy).

<param-name>	**<param-value>**	**Valor por Defecto**
defaultServers	(Requerido) Una lista de nombres de host y números de puertos de los servidores a los que estamos pasando las peticiones en la forma: host1:HTTP_Port:HTTPS_Port| host2:HTTP_Port:HTTPS_Port (donde host1 y host2 son los nombres de host de los servidores en el cluster, HTTP_Port es el puerto donde el host está escuchando peticiones HTTP, y HTTPS_Port es el puerto donde el host escucha peticiones HTTP SSL). Separando cada host con un carácter |. Si seleccionamos el parámetro secureProxy a ON, el puerto HTTPS usa SSL entre el Servidor WebLogic que ejecuta HttpClusterServlet y los servidores WebLogic del Cluster. Siempre debemos definir un puerto HTTPS, incluso si tenemos secureProxy a OFF.	Ninguno
secureProxy	ON/OFF. Si selecciona a ON, permite SSL entre el HttpClusterServlet y los miembros de un Cluster de Servidores WebLoigc.	OFF
DebugConfigInfo	ON/OFF. Si se selecciona a ON, podemos consultar información de depuración del HttpClusterServlet añadiendo un parámetro ?_WebLogicBridgeConfig a cualquier petición. Por razones de seguridad, se recomienda tener este parámetro a OFF en entornos de producción.	OFF
connectionTimeout	La cantidad de tiempo, en segundos, que un socket espera entre lecturas de bloques de datos. Si el tiempo expira, se lanza una java.io.InterruptedIOException.	0 = infinito
numOfRetries	Número de intentos que hace HttpClusterServlet para recuperar una conexión fallida.	5
pathTrim	String a recortar del principio de la URI original.	ninguna
trimExt	La extensión de fichero a recortar del final de la URI.	Ninguna
pathPrepend	String a añadir al principio de la URL original, después de que se haya recortado pathTrim, y antes de que la petición sea reenviada al miembro del cluster de servidores WebLogic.	Ninguna

Ejemplo de Descriptor de Despliegue para HttpClusterServlet

Lo siguiente es un ejemplo de un descriptor de despliegue de una Aplicación Web para usar el `HttpClusterServlet`:

```
<!-- 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>
```

Configura la Seguridad en Aplicaciones Web

Podemos asegurar una Aplicación Web usando autentificación, mediante la restricción de acceso a ciertos recursos en la Aplicación Web, o usando llamadas de seguridad a nuestro código Servlet. Se pueden usar varios reinos de seguridad.

Configurar la Autentificación para Aplicaciones Web

Definimos la autentificación para Aplicaciones Web en el descriptor de despliegue de la Aplicación Web, usando el elemento `<login-config>`. En este elemento definimos los reinos de seguridad conteniendo credenciales de usuario, el método de autentificación, y la localización de los recursos para autentificación.
Para configurar la autentificación para Aplicaciones Web:

1. Elegimos el método de autentificación. Las opciones disponibles son:
   - `BASIC`
     Esta autentificación usa el navegador web para mostrar una caja de diálogo **username/password**. Estos valores se autentifican contra el reino.
   - `FORM`
     La autentificación basada en formulario requiere que devolvamos un formulario HTML que contenga el nombre de usuario y la password. Los campos devueltos por los elementos del formulario deben ser: `j_username` y `j_password`, y el atributo `action` debe ser `j_security_check`. Aquí hay un ejemplo de código HTML para usar la autentificación **FORM**:

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

El recurso usado para generar el formulario HTML podría ser una página HTML, un JSP o un Servlet. Definimos este recurso con el elemento `<form-login-page>`.
El objeto sesión HTTP se crea cuando se sirve la página loggin. Por lo tanto, el método `session.isNew()` devolverá FALSE cuando se llame desde páginas servidas después de que la autentificación tenga éxito.

   - `CLIENT-CERT`
     Usa certificados de cliente para autentificar la petición.
2. Si elegimos la autentificación `FORM`, también definimos la localización de los recursos usados para generar una página HTML y un recurso que responda a una autentificación fallida.
3. Definimos el reino usado para autentificación. Si no especificamos un reino particular, se usará el reino definido con el campo `Auth Realm Name` en la pestaña **Web Application --> Configuration --> Other** de la Consola de Administración.

Múltiples Aplicaciones Web, Cookies y Autentificación

Por defecto, WebLogic Server asigna el mismo nombre de cookie (JSESSIONID) para todas las aplicaciones Web. Cuando usamos cualquier tipo de autentificación, todas las aplicaciones Web que usan el mismo nombre de cookie usan una sola firma para autentificación. Una vez que un usuario se ha autentificado, esa autentificación será válida para peticiones de cualquier aplicación Web que use el mismo nombre de cookie. No se le pedirá al usuario que vuelva a autentificarse.
Si queremos requerir autentificación separada para una Aplicación Web, podemos especificar un nombre de cookie distinto para la Aplicación Web. Podemos hacer esto usando el parámetro `CookieName`, definido en el descriptor de despliegue específico de WebLogic, **weblogic.xml**, en el elemento `<session-descriptor>`.

Restringir el Acceso a Recursos de una Aplicación Web

Podemos aplicar restricciones a recursos específicos (servlets, JSPs, o páginas HTML) de nuestra aplicación Web. Para aplicar restricciones de seguridad:

1. Definimos un rol que es mapeado a uno o más principales en un reino de seguridad. Definimos los roles con el elemento `<security-role>` en el descriptor de despliegue de la Aplicación Web. Entonces mapeamos esos roles a los principales en nuestro reino con el elemento `<security-role-assignment>` en el descriptor de despliegue específico de WebLogic.
2. Definimos a qué recursos de la aplicación Web se aplican las restricciones usando el elemento `<url-pattern>` que está anidado dentro del elemento `<web-resource-collection>`. El `<url-pattern>` puede referirse a un directorio, un nombre de fichero o a un `<servlet-mapping>`.
   Para aplicar las restricciones de seguridad a toda la aplicación Web, usamos el siguiente `<url-pattern>`:
3.
4. `  <url-pattern>/*</url-pattern>`
5.
6. Definimos el método HTTP (GET o POST) a los que se aplican las restricciones usando el elemento `<http-method>` que está anidado dentro del elemento `<web-resource-collection>`.
7. Definimos si se debería usar o no SSL para las comunicaciones entre el cliente y el servidor usando el elemento `<transport-guarantee>` que está anidado dentro del método `<user-data-constraint>`.

Código de ejemplo:

```
Entradas en web.xml:

<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>east</role-name>
    <role-name>manager</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>east</role-name>
</security-role>

<security-role>
  <description>managers</description>
  <role-name>manager</role-name>
</security-role>

Entradas en weblogic.xml:

<security-role-assignment>
  <role-name>east</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>
```

Usar Usuarios y Roles Programáticamente en Servlets

Podemos escribir nuestros servlets para acceder programáticamente a usuarios y roles en nuestro código de servlet usando el método `javax.servlet.http.HttpServletRequest.isUserInRole(String role)`.
El string **role** es mapeado al nombre suministrado en el elemento `<role-name>` que está anidado dentro del elemento `<security-role-ref>` de una declaración `<servlet>` en el descriptor de despliegue de la aplicación. El elemento `<role-link>` se mapea a un `<role-name>` definido en el elemento `<security-role>` del descriptor de despliegue de la aplicación Web.

Por ejemplo:

Código Servlet:
isUserInRole("manager");

Entradas en web.xml:

```
<servlet>
...
  <role-name>manager</role-name>
  <role-link>mgr</role-link>
...
</servlet>

<security-role>
  <role-name>mgr</role-name>
</security-role>
```

```
Entradas en weblogic.xml:

<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>
```

Configurar Recursos Externos de una Aplicación Web

Cuando accedemos a recursos externos, como una DataSource desde una Aplicación Web mediante JNDI, podemos mapear el nombre JNDI que buscamos en nuestro código al nombre JNDI real unido en el árbol JNDI. Este mapeo se hace usando los dos descriptores de despliegue **web.xml** y **weblogic.xml** y nos permite cambiar estos recursos sin tener que modificar el código de nuestra aplicación. Proporcionamos un nombre que se usa en nuestro código Java, el nombre del recurso que está unido al árbol JNDI, y el tipo Java del recurso, e indicamos si la seguridad del recurso se maneja programáticamente por el servlet o desde las credenciales asociadas con la petición HTTP.
Para configurar recursos externos:

1. Introducimos el nombre del recurso en el descriptor de despliegue según lo usamos en nuestro cogido, el tipo Java, y el tipo de autorización de seguridad.
2. Mapeamos el nombre del recurso al nombre JNDI.

El siguiente ejemplo asume que hemos definido una fuente de datos llamada **accountDataSource**:

```
Código del Servlet:

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

Entradas en web.xml:

<resource-ref>
  ...
  <res-ref-name>myDataSource</res-ref-name>
  <res-type>javax.sql.DataSource</res-type>
  <res-auth>CONTAINER</res-auth>
  ...
</resource-ref>

Entradas en weblogic.xml:

<resource-description>
  <res-ref-name>myDataSource</res-ref-name>
  <jndi-name>accountDataSource</jndi-name>
</resource-description>
```

Referenciar EJBs en una Aplicación Web

Referenciamos EJBs en una aplicación Web dándoles un nombre en el descriptor de despliegue de la aplicación Web que es mapeado al nombre JNDI para el EJB que está definido en el fichero descriptor de despliegue **weblogic-ejb-jar.xml**.
Para configurar EJBs para usarlos en una Aplicación Web:

1. Introducimos el nombre de referencia del EJB que usamos para buscar el EJB en nuestro código, el nombre de la clase Java de los interfaces home y remoto del EJB en el elemento `<ejb-ref>` del descriptor de despliegue de la Aplicación Web.
2. Mapeamos el nombre de referencia en el elemento `<ejb-reference-description>` del descriptor de despliegue específico de WebLogic, **weblogic.xml**, al nombre JNDI definido en el fichero **weblogic-ejb-jar.xml**.

Si la Aplicación Web es parte de un Enterprise Application Archive (fichero ***.ear**), podemos referenciar un EJB por el nombre usado en el **.ear** con el elemento `<ejb-link>`.

Configurar el Manejo de Sesión

WebLogic Server está configurado para manejar el seguimiento de sesión por defecto. No necesitamos configurar ninguna de estas propiedades para usar seguimiento de sesión. Sin embargo, configurar el modo en que WebLogic Server maneja las sesiones es una parte importante del ajuste de nuestra aplicación para un mejor rendimiento.
El ajuste depende de factores como:

- Cuántos usuarios esperamos que usen el servlet.
- Cuantos usuarios concurrentes esperamos que usen el servlet.
- Cuánto durará cada sesión.
- Cuántos datos esperamos que se almacenen por cada usuario.

Propiedades de Sesión HTTP

Configuramos WebLogic Server para que haga seguimiento de sesión con las propiedades del descriptor de despliegue específico de WebLogic, **weblogic.xml**.
Puedes encontrar una lista completa de atributos de sesión en http://e-docs.bea.com/wls/docs60/programming/weblogic_xml.html#session-descriptor.

TimeOut de Sesión

Podemos especificar un intervalo de tiempo después del cual expiran las sesiones HTTP. Cuando una sesión expira, todos los datos almacenados en la sesión son descartados. Podemos seleccionar el intervalo de una de estas formas:

- Seleccionando el atributo `TimeoutSecs` del elemento `<session-descriptor>` de descriptor de despliegue específico de WebLogic. Este valor es en segundos.
- Seleccionando el elemento <session-timeout> en el descriptor de despliegue de la aplicación Web. Este valor se selecciona en segundos y sobrescribe cualquier valor que pongamos en el atributo `TimeoutSecs`.

Configurar Cookies de Sesión

WebLogic Server usa cookies para manejo de sesiones cuando lo soporta el navegador del cliente.
Las cookies que usa WebLogic Server para seguir la sesión son temporales por defecto y no sobreviven a la vida del navegador. Cuando un usuario cierra su navegador, las cookies se pierden y se acaba el tiempo de la sesión. Este comportamiento está en el espíritu del uso de la sesión y se recomienda que usemos las sesiones de esta forma.
Es posible configurar muchos aspectos de las cookies usadas para seguimiento de sesión con atributos definidos en el descriptor de despliegue específico de WebLogic. Puedes encontrar una lista completa de atributos relacionados con la sesión y las cookies en http://e-docs.bea.com/wls/docs60/programming/weblogic_xml.html#session-descriptor.

Usar Cookies de Larga Vida

Para datos de usuario de larga vida en el lado del cliente, nuestra aplicación debería crear y configurar sus propias cookies en el navegador mediante el API servlet HTTP, y no deberíamos intentar usar las cookies asociadas con la sesión HTTP. Nuestra aplicación podría usar cookies para auto-login un usuairo desde una máquina particular, en cuyo caso configuraríamos una nueva cookie con un largo tiempo de caducidad. Recuerda que la cookie sólo puede ser enviada desde la máquina del cliente. Nuestra aplicación debería almacenar los datos en el servidor si debe ser accedida por el usuario desde múltiples localizaciones.
No podemos conectar directamente la edad de un cookie de navegador con la longitud de una sesión. Si una cookie expira antes de su sesión asociada, esta sesión se vuelve huérfana. Si una sesión expira antes que su cookie asociada, el servler no podrá encontrar una sesión. En este punto, se asigna una sesión cuando se llama al método `getSession()`. Sólo deberíamos hacer un uso temporal de las sesiones.

Configurar la Persistencia de Sesión

Hay cuatro diferentes implementaciones de persistencia de sesión:

- Memoria (un-sólo-servidor, no-replicado)
- Persistencia de sistema de Ficheros
- Persistencia JDBC
- Replicación en memoria (en un cluster).

Las tres primeras se cubren aquí; la replicación en memoria se cubre en http://e-docs.bea.com/wls/docs60/cluster/servlet.html.
Para la replicación en memoria, fichero, y JDBC, necesitamos configurar atributos adicionales, incluyendo `PersistentStoreType`. Cada método tiene su propio conjunto de propiedades mostradas abajo:

Propiedades Comunes

Podemos configurar el número de sesiones que mantenemos en memoria configurando los siguientes atributos en el descriptor de despliegue específico de WebLogic. Estos atributos sólo son aplicables si estamos usando persistencia de sesión:

- `CacheSize`
  Limita el número de sesiones almacenadas que pueden estar activas en memoria en un momento dado. Si estamos esperando un alto volumen de sesiones simultáneas activas, no querremos que estas sesiones bloqueen la RAM de nuestro servidor ya que esto podría causar problemas de rendimiento con el intercambio en memoria virtual. Cuando el caché está lleno, las últimas sesiones más recientemente utilizadas se almacenan en el almacén persistente y son llamadas cuando se necesitan. Si no se usa persistencia, esta propiedad es

ignorada, y no hay límite software en el número de sesiones permitidas en la memoria principal. Por defecto, el número de sesiones cacheadas es 1024. El mínimo es 16, y el máximo es `Integer.MAX_VALUE`. Una sesión vacía ocupa menos de 100 bytes, pero crece según se van añadiendo datos.

- `SwapIntervalSecs`
  El intervalo de tiempo que espera el servidor entre la purga de las sesiones recientemente usadas del cache al almacenamiento persistente, cuando se ha alcanzado el límite `cacheEntries`.
  Si no se configura, esta propiedad tiene un valor por defecto de 10 segundos; el mínimo es 1 segundo, y el máximo 604800 (1 semana).
- `InvalidationIntervalSecs`
  Selecciona el tiempo, en segundos, que el Servidor WebLogic espera entre hacer el chequeo de limpieza de sesiones time-out o inválidas, y el borrado de las viejas sesiones y la liberación de memoria. Seleccionamos este parámetro a un valor menor que el valor del elemento `<session-timeout>`. Usamos este parámetro para ajustar el Servidor WebLogic para un mejor rendimiento en sites de gran tráfico.
  El valor mínimo es cada segundo (1). El valor máximo es una vez a la semana (604.800 segundos). Si no se configura, el valor por defecto es 60 segundos.

Usar Persistencia Basada en Memoria en un Sólo Servidor y no Replicada

Para usar este tipo de persistencia, seleccionamos la propiedad `PersistentStoreType` a `memory`. Cuando usamos el almacenamiento basado en memoria toda la información de sesión se almacena en memoria y se pierde cuando paramos e reiniciamos el Servidor WebLogic.

Usar Persistencia Basada en Fichero

Para el almacenamiento de sesiones basado en ficheros persistentes:

1. Seleccionamos `PersistentStoreType` a `file`.
2. Seleccionamos el directorio donde WebLogic Server almacena las sesiones.

Si no seleccionamos explícitamente un valor para este atributo, automáticamente se crea un directorio temporal en el Servidor WebLogic.
Si estamos usando la persistencia basada en ficheros en un cluster, debemos seleccionar explícitamente este parámetro a un directorio compartido que sea accesible por todos los servidores del cluster. Debemos crear el directorio nosotros mismos.

Usar Persistencia Basada en Bases de Datos

Para almacenamiento de persistencia de sesiones basado en JDBC:

1. Seleccionamos JDB como el método de almacenamiento de persistencia seleccionando el atributo `PersistentStoreType` a `jdbc`.
2. Seleccionamos el almacén de conexiones JDBC a utilizar para el almacenamiento persistente con el atributo `PersistentStorePool`. Usamos el nombre del almacén de conexiones que está definido en la Consola de Administración del Servidor WebLogic.

3. Seleccionamos un ACL para la conexión que corresponda con los usuarios que tienen permiso.
4. Configuramos una tabla de base de datos llamada **wl_servlet_sessions** para persistencia basada en JDBC. El almacén de conexiones que conecta con la base de datos necesita tener acceso de lectura/escritura para esta tabla. La siguiente tabla muestra los nombres de columnas y tipos de datos que deberíamos usar para crear esta tabla:

Nombre de Columna	Tipo
wl_id	**Alfanumérico de anchura variable**, hasta 100 caracteres, por ejemplo, Oracle VARCHAR2(100). La clave primaria debe configurarse como: wl_id + wl_context_path
wl_context_path	**Alfanumérica de anchura variable**, hasta 100 caracteres, por ejemplo, Oracle VARCHAR2(100). Esta columna se usa como parte de la clave primaria.
wl_is_new	**Single char**; Por ejemplo, Oracle CHAR(1)
wl_create_time	**Numérico**, 20 dígitos; Por ejemplo Oracle NUMBER(20)
wl_is_valid	**Single char**; por ejemplo, Oracle CHAR(1)
wl_session_values	**Large binary**; Por ejemplo, Oracle LONG RAW
wl_access_time	**Numérico**, 20 dígitos; por ejemplo, NUMBER(20)
wl_max_inactive_interval	**Integer**; por ejemplo, Oracle Integer. Número de sesiones entre peticiones de cliente antes de que la sesión sea invalidada. Un valor negativo indica que la sesión nunca será timeout.

Si estamos usando una base de datos Oracle, podemos usar la siguiente sentencia SQL para crear la tabla `wl_servlet_sessions`:

```
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) );
```

Podemos modificar la sentencia SQL anterior para usarla con nuestro motor de Base de Datos.

Nota:
Podemos configurar la duración máxima de la persistencia de sesión JDBC que debería esperar una conexión JDBC del almacén de conexiones antes de fallar al cargar los datos de la sesión con el atributo `JDBCConnectionTimeoutSecs`.

Usar Reescritura de URL

En algunas ocasiones, un navegador podría no aceptar cookies, lo que hace imposible el seguimiento de sesión utilizando cookies. La reescritura de URL es una solución a esta situación que puede ser sustituida automáticamente cuando el Servidor WebLogic detecta que el navegador no soporta cookies. La reescritura de URL implica la codificación de la ID de la sesión en los hiper-enlaces de las páginas web que nuestro servlet devuelve al navegador. Cuando luego el usuario pulsa sobre estos enlaces, el Servidor WebLogic extrae la ID de la dirección URL y encuentra la `HttpSession` apropiada cuando nuestro servlet llama al método `getSession()`.

Para activar la reescritura de URLs en el Servidor WebLogic, seleccionamos el atributo `URLRewritingEnabled` en el descriptor de despliegue específico de WebLogic, bajo el elemento `<session-descriptor>` a TRUE. (El valor por defecto de este atributo es TRUE).

Guías de Codificación para Reescritura de URLs

Hay algunas guías generales para el modo en que nuestro código debería manejar las URLs para soportar la reescritura de URLs:

- Evitar reescribir una URL directamente en el stream de salida, como se muestra aquí:

- out.println("catalog");
  En su lugar, usamos el método `HttpServletResponse.encodeURL()`, por ejemplo:

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

  Llamar al método `encodeURL()` determina si la URL necesita ser rescrita, y si es así, la reescribe, incluyendo el ID de sesión en la URL. El ID de sesión se añade a la URL y empieza con un punto y coma.

- Además de las URLs que son devueltas como una respuesta al Servidor WebLogic, también codificamos las URLs que envían redirecciones. Por ejemplo:

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

  WebLogic Server usa reescritura de URL cuando una sesión es nueva, incluso si el navegador acepta cookies, porque el servidor no puede decir si el navegador acepta cookies en la primera visita de una sesión.

- Nuestro servlet puede determinar si una ID de sesión dada fue recibida desde una cookie chequeando el `Boolean` devuelto por el método `HttpServletRequest.isRequestedSessionIdFromCookie()`. Nuestra aplicación podría responder apropiadamente, o simplemente relegar en la reescritura de URL del Servidor WebLogic.

Reescritura de URL y Wireless Access Protocol (WAP)

Si estamos escribiendo una aplicación Wap, debemos usar la reescritura de URL porque el protocolo WAP no soporta cookies. Además, algunos dispositivos WAP tienen un límite de 128 caracteres en la longitud de la URL (incluyendo parámetros), lo que limita la cantidad de datos que se pueden transmitir usando reescritura de URL. Para permitir más espacio para los parámetros, podemos limitar el tamaño de la ID de sesión que genera aleatoriamente el Servidor WebLogic especificando el número de límites con el atributo **IDLength**

Usar Conjuntos de Caracteres y Datos POST

Podemos seleccionar el conjunto de caracteres que se usa para procesar los datos desde un formulario que usa el método POST. Para informar a la aplicación que procesa los datos del formulario que se está usando un conjunto de caracteres particular debemos añadir caracteres específicos "señales" a la URL usada para procesar los datos del formulario (especificado con el atributo `action` de las etiquetas `<form>`) y luego mapear dichos caracteres a una codificación en el descriptor de despliegue de la Aplicación Web. Los datos POST normalmente se leen usando ASCII a menos que especifiquemos lo contrario usando el siguiente procedimiento.
Para procesar los datos POST en un conjunto de caracteres no ASCII:

1. Creamos una nueva entrada en el descriptor de despliegue de la Aplicación Web: **web.xml**, dentro de un elemento `<context-param>`. Está entrada debería ir después del elemento `<distributable>` y antes del elemento `<servlet>` en el fichero **web.xml**. En esta entrada, el `<param-name>` siempre incluye el nombre de la clase `weblogic.httpd.inputCharset`, seguido por un punto, seguido por el string de señal. El `<param-value>` contiene el nombre del conjunto de caracteres HTTP. En el siguiente ejemplo, el string **/rus/jo*** se mapea al conjunto de caracteres **windows-1251**:

```
2.
3.   <context-param>
4.   <param-name>weblogic.httpd.inputCharset./rus/jo*</param-name>
5.   <param-value>windows-1251</param-value>
6.   </context-param>
7.
```

8. Codificamos el formulario HTML para usar el string de señal cuando envíe los datos del formulario. Por ejemplo:

```
9.
10.  <form action="http://some.host.com/myWebApp/rus/jo/index.html">
11.  ...
12.  </form>
```

Situamos el string de señal después del nombre de la aplicación Web (también llamado path de contexto —myWebApp— en este caso) y antes de la porción de renombrado de la URL.

Instalar y Configurar el Plug-In para el Servidor Apache HTTP

Introducción

El Plug-in para el Servidor Apache HTTP, permite que las peticiones sean pasadas (proxy) desde un Servidor Apache hacia un Servidor WebLogic. El Plug-in amplía una Instalación de Apache permitiendo que WebLogic manejar aquellas peticiones que requieren las funcionalidades dinámicas del Servidor WebLogic.
El plug-in se ha pensado para un entorno en el que un Servidor Apache sirve páginas estáticas, y otra parte del árbol de documentos (páginas dinámicas mejor generadas por Servlets HTTP o JavaServer Pages) se delega en el Servidor WebLogic, que podría estar operando en un proceso diferente, posiblemente en un host diferente. En el usuario final -- el navegador -- las peticiones HTTP delegadas al Servidor WebLogic parecerán que vienen de la misma fuente.
El tunneling HTTP también puede operar a través del plug.in, proporcionando acceso a los servicios WebLogic a los clientes que no sean navegadores.
El Apache HTTP Server Plug-In opera como un módulo de Apache en un Servidor Apache. El servidor Apache carga los módulos en la arrancada, y luego ciertas peticiones HTTP se delegan hacía ellos. Los módulos Apache son similares a los Servlets HTTP, excepto que un módulo Apache se escribe en código nativo de la plataforma.

Mantener Vivas las Conexiones Apache

El Plug-In Apache HTTP Server crea un socket por cada petición y cierra el socket después de haber leído la respuesta. Como el Servidor Apache es multi-procesador, el almacén de conexiones y las conexiones vivas entre el Servidor WebLogic y el Plug-in del Servidor Apache no pueden soportarse.

Peticiones Proxy

El plug-in pasa (hace de proxy) peticiones al Servidor WebLogic basándose en una configuración que nosotros especificamos. Podemos pasar peticiones basándonos en la URL de las peticiones (o en una porción de la URL). Esto se llama proxy por path. También podemos pasar peticiones basándonos en el tipo MIME del fichero solicitado. También podemos usar una combinación de ambos métodos. Si una petición cumple los dos criterios, se pasa por path. También podemos especificar parámetros adicionales para aquellos tipos de peticiones que definen un comportamiento adicional del plug-in.

Plataformas Soportadas

El Plug-In Apache HTTP Server soporta las plataformas Linux, Solaris y HPUX11. Para más información sobre el soporte de las versiones específicas de Apache puedes ver la página http://e-docs.bea.com/wls/platforms/index.html#apach.

Instalar el Plug-in para Apache Web Server

Instalamos el Plun-In para Apache HTTP Server como un módulo más de Apache en nuestra instalación. Para instalar el Plun-in:

1. Localizamos el fichero de objeto compartido para nuestra plataforma.

El Plug-in de apache se distribuye como un objeto compartido (.so) para usarlo en las plataformas Solaris, Linux, y HPUX11. El fichero de objeto compartido se distribuye en versiones separadas, dependiendo de la plataforma, de si se va usar SSL entre el cliente y Apache y la longitud de la encriptación para SSL (normal o 128 bits).

Los ficheros de objetos compartidos se localizan en los siguientes directorios de nuestra instalación WebLogic Server:

- Solaris
  lib/solaris
- Linux
  lib/linux
- HPUX11
  lib/hpux11

Elegimos el objeto compartido apropiado según la siguiente tabla:

Versión de Apache	**Longitud de Encriptación Normal**	**Encriptación de 128 bits**
Apache, Version 1.x	`mod_wl.so`	`mod_wl128.so`
Apache w/SSL/EAPI Version 1.x (Stronghold, modssl etc).	`mod_wl_ssl.so`	`mod_wl128_ssl.so`
Apache + Raven Version 1.x Requerido porque Raven aplica patches frontpage que hacen el plug-in incompatible con el objeto compartido estándar.	`mod_wl_ssl_raven.so`	`mod_wl128_ssl_raven.so`

2. Activar el Objeto compartido.

El Plug-in de Apache HTTP Server se instalará como un Apache Dynamic Shared Object (DSO). El DSO soportado en Apache está basado en un módulo llamado `mod_so.c` que debe estar activado antes de cargar `mod_wl.so`. Si instalamos Apache usando el Script suministrado, `mod_so.c` debería estar activado. Para verificar que lo está, ejecutamos el siguiente comando:

```
APACHE_HOME/bin/httpd -l
```

(donde **APACHE_HOME** es el directorio que contiene la instalación de nuestro Apache HTTP Server.) Este comando lista todos los módulos activados. Si `mod_so.c` no está listado, debemos reconstruir nuestro Apache HTTP Server desde el código fuente, asegurándonos de que están configuradas las siguientes opciones:

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

3. Instalamos el Plug-In de Apache HTTP Server con un programa de soporte llamado **apxs** (APache eXtenSion) que construye módulos basados en DSO fuera del árbol fuente de Apache, y añadimos la siguiente línea al fichero **httpd.conf**:

```
4.  AddModule mod_so.c
```

En nuestra instalación de WebLogic Server, usamos un shell de línea de comandos para navegar al directorio que contiene el objeto compartido de nuestra plataforma y activamos el `weblogic_module` lanzando este comando (observa que debemos tener Perl instalado para ejecutar este script):

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

Este comando copia el fichero `mod_wl.so` al directorio `APACHE_HOME/libexec`. También añade dos líneas de instrucciones para `weblogic_module` en el fichero **httpd.conf** y activa el módulo. Debemos asegurarnos de que se han añadido estas dos líneas al fichero **APACHE_HOME/conf/httpd.conf**:

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

5. Verificamos la sintaxis del fichero **APACHE_HOME/conf/httpd.conf** con el siguiente comando:

```
6.  APACHE_HOME/bin/apachect1 configtest
```

   La salida de este comando indica cualquier error que haya en nuestro fichero **httpd.conf**.

7. Configuramos cualquier parámetro adicional en el fichero de configuración de Apache **httpd.conf** como se describe en la sección Configurar el Plug-In para Apache HTTP Server. El fichero **httpd.conf** nos permite personalizar el comportamiento del Plug-In Apache HTTP Server.
8. Arrancamos Weblogic Server.
9. Arrancamos (o reiniciamos si hemos cambiado la configuración) el Servidor Apache HTTP.
10. Probamos el Plug-in de Apache abriendo un navegador y seleccionando la URL hacia el servidor Apache + "/weblogic/", que nos debería mostrar la página HTML por defecto del servidor WebLogic, el fichero de bienvenida, o el servlet por defecto, según se haya definido en la Aplicación Web sobre el Servidor WebLogic. Por ejemplo:

```
11.  http://myApacheserver.com/weblogic/
```

Configurar el Plug-In para Apache HTTP Server

Después de instalar el plug-in, editamos el fichero **httpd.conf** para configurarlo. Editar el fichero **httpd.conf** informa al servidor Apache que debería cargar la librería nativa del Plug-in como un módulo Apache y también describe qué peticiones debería manejar el módulo.

Editar el Fichero httpd.conf

Para editar el fichero **httpd.conf** para configurar el Plug-In Apache HTTP Server:

1. Abrimos el fichero **httpd.conf**. Este fichero está localizado en **APACHE_HOME/conf/httpd.conf** (donde `APACHE_HOME` es el directorio raíz de nuestra instalación de Apache).
2. Verificamos que se han añadido estas dos líneas después de ejecutar la utilidad **apxs**:

```
3.  LoadModule weblogic_module libexec/mod_wl.so
4.  AddModule mod_weblogic.c
```

5. Añadimos un bloque **IfModule** que defina uno de:
   - Para un Servidor WebLogic sin clusters:
     Los parámetros **WebLogicHost** y **WebLogicPort**.
   - Para un cluster de Servidores WebLogic:
     El parámetro **WebLogicCluster**.

Por ejemplo:

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

6. Si queremos pasar (proxy) peticiones por su tipo MIME, también añadimos una línea `MatchExpression` al bloque `IfModule`. (También podemos pasar peticiones por path. El paso por path tiene precedencia sobre el paso por tipo MIME, si sólo queremos pasar peticiones por path, saltamos al próximo paso).

Por ejemplo, el siguiente bloque **IfModule** es para un servidor WebLogic sin clusters, especifica que todos los ficheros con el tipo MIME **.jsp** sean pasados:

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

También podemos usar múltiples `MatchExpressions`, por ejemplo:

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

Si estamos pasando peticiones por tipo MIME a un cluster de Servidores WebLogic, usamos el parámetro `WebLogicCluster` en lugar de los parámetros `WebLogicHost` y `WebLogicPort`. Por ejemplo:

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

7. Si queremos pasar peticiones por path, usamos el bloque `Location` y la sentencia `SetHandler`. `SetHandler` especifica el manejador para el modulo Plug-in de Apache HTTP Server. Por ejemplo, el siguiente bloque `Location` pasa todas las peticiones que contengan **/weblogic** en la URL:

```
8.
9.   <Location /weblogic>
10.      SetHandler weblogic-handler
11.  </Location>
12.
```

13. Definimos cualquier parámetro adicional para el Plug-In de Apache HTTP Server.

El Plug-In de Apache HTTP Server reconoce los parámetros listados en Parámetros Generales para Plug-Ins de Servidores Web. Para modificar el comportamiento de nuestro Plug-In de Apache HTTP Server, definimos estos parámetros:

- En un bloque `Location`, para parámetros que se aplican al paso (proxy) por path, o
- en un bloque `IfModule`, para parámetros que se aplican al paso (proxy) por tipo MIME.

Notas sobre la Edición del Fichero httpd.conf

- Como una alternativa al procedimiento descrito en Editar el Fichero httpd.conf, podemos definir los parámetros en un fichero separado llamado **weblogic.conf** que es incluido en el bloque `IfModule`. Usar este fichero incluido podría ayudarnos a modularizar la configuración. Por ejemplo:

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

Nota:
Definir los parámetros en un fichero incluido no está soportado cuando se usa SSL entre el Plug-In de Apache HTTP Server y el Servidor WebLogic.

- Cada parámetro debe introducirse en una nueva línea. No debemos poner un '=' entre el parámetro y su valor. Por ejemplo:

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

- Si una petición corresponde tanto con un tipo MIME especificado en una `MatchExpression` en un bloque `IfModule` como con un path especificado en un bloque `Location`, tiene preferencia el comportamiento especificado en lo bloque `Location`.
- Si definimos el parámetro `CookieName`, debemos hacerlo en un bloque `IfModule`.

Usar SSL con el Plug-In de Apache

Podemos usar el protocolo Secure Sockets Layer (SSL) para proteger la conexión entre el Plug-In de Apache HTTP Server y el Servidor WebLogic. El protocolo SSL proporciona confidencialidad e integridad de los datos pasados entre el Plug-In de Apache HTTP Server y el Servidor WebLogic. Además, el protocolo SSL permite al plug-in autenticarse a si mismo en el Servidor WebLogic para asegurar que la información es pasada a un principal seguro.
El Plug-In de Apache HTTP Server no usa el protocolo de transporte (http o https) especificado en la petición HTTP (normalmente por el navegador) para determinar si se usa o no el protocolo SSL para proteger los datos de la conexión entre el Plug-In de Apache HTTP Server y el Servidor WebLogic.

Configurar SSL entre el Plug-In de Apache HTTP Server y el Servidor WebLogic

Para usar el protocolo SSL entre el Plug-In de Apache HTTP Server y el Servidor WebLogic:

1. Configuramos el Servidor WebLogic para SSL. (Ver Configurar el Protocolo SSL).
2. Configuramos el puerto de escucha SSL del Servidor WebLogic. (Ver Configurar el Puerto de Escucha).
3. Seleccionamos el parámetro `WebLogicPort` en el fichero **httpd.conf** al puerto de escucha configurado en el paso 2.
4. Seleccionamos el parámetro `SecureProxy` en el fichero **httpd.conf** a ON.
5. Seleccionamos cualquier parámetro adicional en el fichero **httpd.conf** que defina información sobre la conexión SSL. Puedes ver una lista completa de parámetros en Parámetros SSL para Plug-Ins de Servidores Web.

Problemas de Configuración SSL-Apache

Cuando configuramos el Plug-In Apache para usar SSL hay dos problemas conocidos:

- El parámetro `PathTrim` debe configurarse dentro de la etiqueta <Location>. La siguiente configuración es incorrecta:

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

  <IfModule mod_weblogic.c>
      WebLogicHost localhost
      WebLogicPort 7001
      PathTrim /weblogic
  </IfModule>
```

Esta es la configuración correcta:

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

- La directiva `Include` no funciona con Apache SSL. Debemos configurar todos los parámetros directamente en el fichero **httpd.conf**. No debemos usar la siguiente configuración cuando usemos SSL:

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

Errores de Conexión y Control de Fallos en Clustering

Cuando el Plug-In de Apache HTTP Server intenta conectar con el Servidor WebLogic, usa varios parámetros de configuración para determinar cuánto tiempo esperar las conexiones con el host del Servidor WebLogic y, después de establecida la conexión, cuánto esperar por una respuesta. Si el plug-in no puede conectar o no recibe una respuesta, intentará conectar y enviar la petición a otro Servidor WebLogic del Cluster. Si la conexión falla o no hay respuesta de ningún servidor WebLogic del cluster, se envía un mensaje de error.

Fallos de Conexión

El fallo de un host al responder a una petición de conexión podría indicar posibles problemas con la máquina host, problemas de red, u otros varios fallos de servidor.
El fallo de un Servidor WebLogic al responder, podría indicar que WebLogic no se está ejecutando que no está disponible, un cuelgue de servidor, un problema de base de datos, u otro fallo de aplicación.

Control de Fallos con un Sólo Servidor (sin cluster)

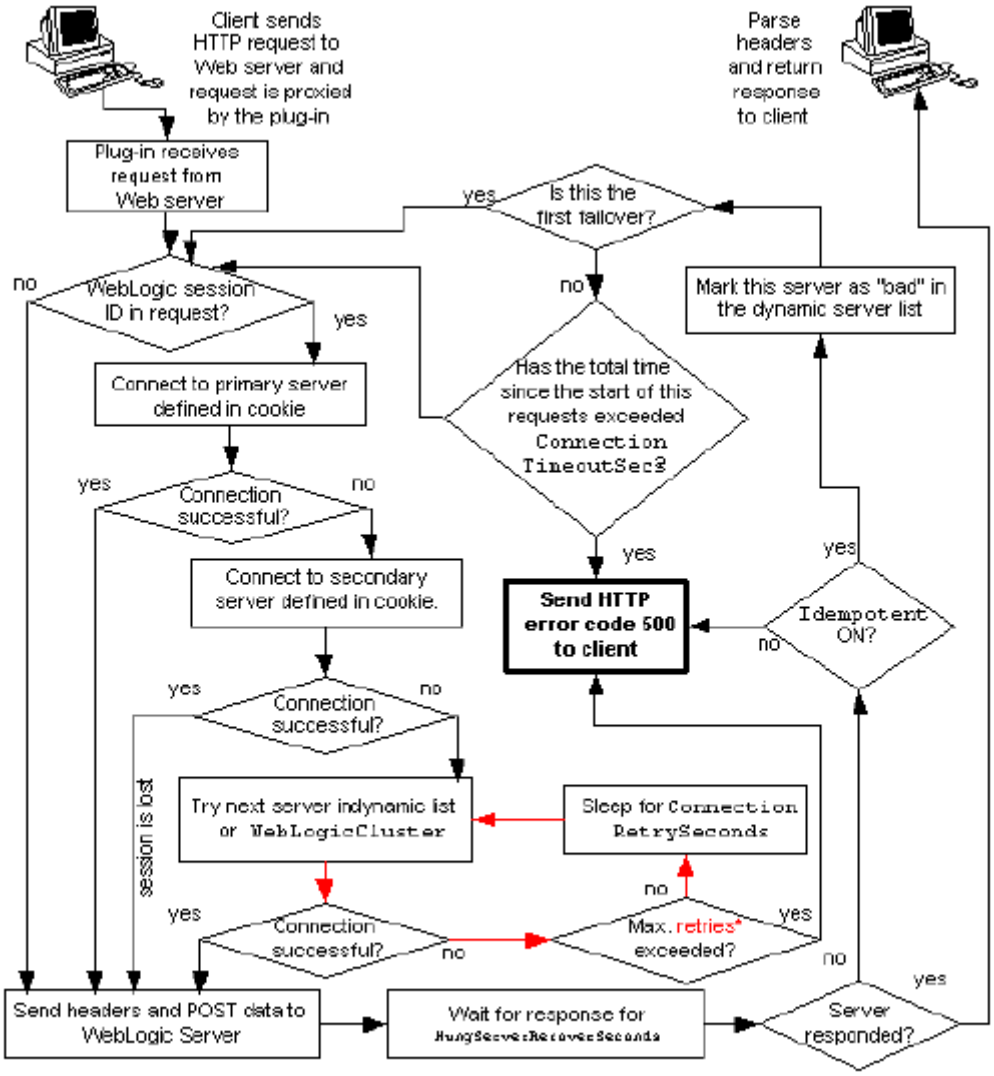
Si estamos ejecutando un sólo Servidor WebLogic se aplica la misma lógica descrita aquí, excepto en que el plug-in sólo intenta conectar con el servidor definido en el parámetro `WebLogicHost`. Si el intento falla, se devuelve un mensaje de error **HTTP 503**. El plug-in continúa intentando conectar con el Servidor WebLogic hasta que se excede el tiempo `ConnectTimeoutSecs`.

La Lista de Servidores Dinámica

Cuando especificamos una lista de servidores WebLogic en el parámetro `WebLogicCluster`, el plug-in usa esa lista como punto de entrada para el balance de carga entre los miembros del cluster. Después de que se haya enrutado la primera petición a uno de esos servidores, se devuelve una lista dinámica que contiene una lista actualizada con los servidores que hay en el cluster. La lista actualizada añade cualquier nuevo servidor en el cluster y borra cualquier otro que haya dejado de formar parte de él, o que haya fallado al responder peticiones. Esta lista se actualiza automáticamene con la respuesta HTTP cuando ocurre un cambio en el cluster.

Control de Fallos, Cookies y Sesiones HTTP

Cuando una petición contiene una información de sesión almacénada en un cookie, en los datos POST, o codificando la URL, la ID de la sesión contiene una referencia al servidor específico en que se estableció originalmente la sesión (llamado servidor primario) y una referencia a un servidor adicional donde se ha replicado la sesión original (llamado servidor secundario). Una petición que contiene una cookie intenta conectar con el servidor primario, si el intento falla, la petición se enruta hacia el servidor secundario. Si ambos servidores fallan, la sesión se pierde y el plug-in intenta hacer una nueva conexión con otro servidor de la lista dinámica del cluster. Puedes ver más información en la siguiente figura:



*The Maximum number of retries allowed in the red loop is equal to `ConnectTimeoutSecs ÷ ConnectRetrySecs`.

Plantilla del Fichero http.conf

Esta sección contiene un ejemplo del fichero **http.conf**. Podemos usar este ejemplo como una plantilla que podemos modificar para adaptarla a nuestro entorno y servidor. Las líneas que empiezan con # son comentarios. Observa que el Servidor Apache HTTP no es sensible a las mayúsculas, y que las líneas `LoadModule` y `AddModule` las añade automáticamente la utilidad **apxs**:

```
##################################################
APACHE-HOME/conf/httpd.conf file
##################################################
LoadModule weblogic_module libexec/mod_wl.so
AddModule mod_weblogic.c

<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>
```

Ejemplos de Ficheros de Configuración

En lugar de definir parámetros en el bloque `Location` de nuestro fichero **httpd.conf**, si lo preferimos, podemos usar un fichero **weblogic.conf** que carga `IfModule` en el fichero **httpd.conf**. Los siguientes ejemplos se podrían usar como plantillas que podemos modificar para adaptarlas a nuestro entorno y servidor. Las líneas que empiezan con # son comentarios

Ejemplo Usando Clusters WebLogic

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

Ejemplo Usando Varios Clusters

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

<IfModule mod_weblogic.c>
    MatchExpression *.jsp
WebLogicHost=myHost|WebLogicPort=7001|Debug=ON
    MatchExpression *.html
WebLogicCluster=myHost1:7282,myHost2:7283|ErrorPage=
    http://www.xyz.com/error.html
</IfModule>
```

Ejemplo Sin Clusters

```
# 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>
```

Ejemplo de Configuración de Hosting Virtual basado en IP

```
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>
```

Ejemplo de Configuración de Hosting Virtual Basado en Nombre

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

<VirtualHost 162.99.55.208>
    ServerName myserver.mydomain.com
        <Location / >
            SetHandler weblogic-handler
            WebLogicHost russell
            WebLogicPort 7001
            Debug ON
            DebugConfigInfo ON
        </Location>
</VirtualHost>
```

Instalar y Configurar el Plug-In para el Microsoft Internet Information Server (ISAPI)

Introducción al Microsoft Internet Information Server (ISAPI)

El Plug-In para Microsoft Internet Information Server permite que las peticiones sean pasadas (proxy) desde un Microsoft Internet Information Server (IIS) a un Servidor WebLogic. El Plug-In mejora la instalación de un IIS permitiendo que el Servidor WebLogic maneje aquellas peticiones que requieren las funcionalidades dinámicas de WebLogic.
El plug-in se ha pensado para un entorno en el que Microsoft Internet Information Server sirve páginas estáticas, y otra parte del árbol de documentos (páginas dinámicas mejor generadas por Servlets HTTP o JavaServer Pages) se delega en el Servidor WebLogic, que podría estar operando en un proceso diferente, posiblemente en un host diferente. En el usuario final -- el navegador -- las peticiones HTTP delegadas al Servidor WebLogic parecerán que vienen de la misma fuente. La funcionalidad del tunneling HTTP del protocolo cliente-servidor de WebLogic también puede operar a través del Plug-In, proporcionando acceso a todos los servicios del Servidor WebLogic.

Almacén de Conexiones y Keep-Alive

El Plug-In para Microsoft Internet Information Server mejora el rendimiento usando un almacén de conexiones re-utilizables desde el plug-in hacia el Servidor WebLogic. El Plug-In implementa conexiones HTTP 1.1 keep-alive entre el plug-in y el Servidor WebLogic re-utilizando la misma conexión en el almacén para subsecuentes respuestas desde el mismo cliente. Si la conexión está inactiva por más de 30 segundos, (u otra cantidad de tiempo definida por el usuario) se cierra y se devuelve al almacén.

Peticiones Proxy

El plug-in pasa (hace de proxy) peticiones al Servidor WebLogic basándose en una configuración que nosotros especificamos. Podemos pasar peticiones basándonos en la URL de la peticiones (o en una porción de la URL). Esto se llama proxy por path. También podemos pasar peticiones basándonos en el tipo MIME del fichero solicitado. También podemos usar una combinación de ámbos métodos. Si una petición cumple los dos criterios, se pasa por path. También podemos especificar parámetros adicionales para aquellos tipos de peticiones que definen un comportamiento adicional del plug-in.

Plataformas Soportadas

Para obtener la última información sobre los sistemas operativos y la compatibilidad de las versiones IIS con el Plug-In de Microsoft Internet Information Server, puedes ver la página http://e-docs.bea.com/wls/platforms/index.html#iis.

Instalar el Plug-In para Microsoft Internet Information Server

Para instalar el Plug-In de Microsoft Internet Information Server:

1. Copiamos el fichero **iisproxy.dll** desde el directorio **/bin** de nuestra instalación de WebLogic Server dentro de un directorio de conveniencia que sea accesible por IIS. Este directorio también debe contener el fichero **iisproxy.ini**.
2. Arrancamos el IIS Internet Service Manager, seleccionándolo desde el menú de Inicio de Microsoft IIS.
3. En la panel izquierdo del Service Manager, seleccionamos nuestra website (por defecto es “Default Web Site”).
4. Pulsamos en la flecha **“Play”** de la barra de herramientas para arrancar.

5. Abrimos las propiedades para la website seleccionada pulsando con el botón derecho del ratón sobre la selección de webiste en el panel izquierdo.
6. En el panel **Properties**, seleccionamos la pestaña **Home Directory**, y pulsamos el botón **Configuration** en la sección **Applications Settings**.
7. Configuramos el tipo de fichero a pasar (proxy):
    - Sobre la pestaña **App Mappings**, pulsamos el botón **Add** para añadir los tipos de ficheros y configurarlos para ser pasaddos a WebLogic Server.
    - En la caja de diálogo, navegamos hasta encontrar el fichero **iisproxy.dll**.
    - Seleccionamos la **Extension** del tipo de fichero que queremos pasar a WebLogic Server.
    - Marcamos el checkbox **Script engine**.
    - Desmarcamos el checkbox **Check that file exists**.
    - Seleccionamos los métodos de exclusión según sea necesario para crear una instalación segura.
    - Cuando finalizamos pulsamos el botón **OK** para grabar la configuración. Repetimos este proceso por cada tipo de fichero que queramos pasar a WebLogic.
    - Cuando hayamos terminado de configurar los tipos de ficheros, pulsamos el botón **OK** para cerrar el panel **Properties**.

Nota:
Cualquier información de path que añadamos a la URL después del servidor y del puerto se pasa directamente a WebLogic Server. Por ejemplo, si solicitamos un fichero desde IIS con la URL:
`http://myiis.com/jspfiles/myfile.jsp`
es pasada a WebLogic Server con una URL como:
`http://mywebLogic:7001/jspfiles/myfile.jsp`

8. Creamos el fichero **iisproxy.ini**.

El fichero **iisproxy.ini** contiene parejas `nombre=valor` que definen los parámetros de configuración para el plug-in. Los parámetros se listan en Parámetros Generales para Plug-Ins de Servidores Web.

Nota:
*Los cambios en los parámetros no tendrán efecto hasta que no reiniciemos el "IIS Admin Service" (bajo **services**, en el panel de control).*

BEA recomienda que situemos el fichero **iisproxy.ini** en el mismo directorio que contiene el fichero **iisproxy.dll**. También podemos usar otras localizaciones. Si situamos el fichero en otro lugar, observa que WebLogic Server busca **iisproxy.ini** en los siguientes directorios, y en el siguiente orden:

- En el mismo directorio donde está **iisproxy.dll**.
- El directorio home de la versión más reciente de WebLogic Server que esté referenciada en el Registro de Windows. Si WebLogic Server no encuentra el fichero **iisproxy.ini** aquí, continúa buscando en el registro de Windows viejas versiones de WebLogic Server y busca el fichero **iisproxy.ini** en los directorios home de dichas instalaciones.
- `c:\weblogic`

9. Definimos el nombre de host y el puerto del Servidor WebLogic al que Microsoft Internet Information Server pasará las peticiones. Dependiendo de nuestra configuración, hay dos formas de definir el host y el puerto:
    - Si estamos pasando peticiones a un sólo servidor WebLoigc, definimos los parámetros `WebLogicHost` y `WebLogicPort` en el fichero **iisproxy.ini**. Por ejemplo:
    - `WebLogicHost=localhost`
    - `WebLogicPort=7001`
    - Si estamos pasando peticiones a un cluster de servidores WebLogic, definimos el parámetro `WebLogicCluster` en el fichero **iisproxy.ini**. Por ejemplo:
    - `WebLogicCluster=myweblogic.com:7001,yourweblogic.com:7001`
10. Configuramos el paso por path. Además de pasar por el tipo de fichero podemos configurar el Plug-In Microsoft Internet Information Server para que sirva ficheros basándose en su path especificando algunos parámetros adicionales en el fichero **iisproxy.ini**. Pasar por path tiene preferencia sobre el paso por tipo MIME.

Para configurar el paso (proxy) por path:

- Situamos el fichero **iisforward.dll** en el mismo directorio que el fichero **iisproxy.dll** y añadimos el fichero **iisforward.dll** como un filtro de servicio en IIS (WebSite Properties ?ISAPI Filters tab ?Add el iisforward dll).
- Registramos `.wlforward` como un tipo especial de fichero a manejar por **iisproxy.dll**.
- Definimos la propiedad `WlForwardPath` en **iisproxy.ini**. `WlForwardPath` define el path pasado al Servidor WebLogic, por ejemplo:
- `WlForwardPath=/weblogic.`
- Seleccionamos el parámetro `PathTrim` para cortar el `WlForwardPath` cuando sea necesario. Por ejemplo, usando:
- `WlForwardPath=/weblogic`
- `PathTrim=/weblogic`

cortamos una petición desde IIS a Weblogic Server. Por lo tanto, `/weblogic/session` es cambiado por `/session`.

- Si queremos que las peticiones no contengan información extra de path (en otras palabras, que las peticiones sólo contengan el nombre del host), seleccionamos el parámetro `DefaultFileName` con el nombre de la página de bienvenida de la Aplicación Web a la que se le está pasando la petición. El valor de este parámetro se añade a la URL.
- Si necesitamos depurar nuestra aplicación, seleccionamos el parámetro `Debug=ON` en el fichero **iisproxy.ini**. Se genera un fichero **c:\tmp\iisforward.log** que contiene un diario de la actividad del plug-in que podemos usar para depurar.

11. Seleccionamos cualquier parámetro adicional del fichero **iisproxy.ini**. Los parámetros se listan en Parámetros Generales para Plug-Ins de Servidores Web.
12. Si estamos pasando servlets desde IIS al Servidor WebLogic y no estamos pasando por path, debemos leer la sección Pasar Servlets desde IIS al Servidor WebLogic.

Crear ACLs através de IIS

Los ACLs no funcionan a través del Plug-In de Microsoft Internet Information Server si el IIS no pasa la cabecera de autorización. Usamos la siguiente información para asegurarnos de que la cabecera de Autorización es pasada.
Cuando usamos Autenticación Básica, el usuario entra con derechos de log local. Para permitir el uso de Autenticación básica, concedemos a cada cuenta de usuario los derechos de usuario **Log On Locally** en el servidor IIS. Observa que podrían resultar dos problemas del uso de logon local de la Autenticación básica:

- Si el usuario no tiene permisos de logon local, la Autenticación básica no funcionará incluso si las configuraciones de FrontPage, IIS, y Windows NT parecen ser correctas.
- Un usuario que tenga permisos de logon local y que pueda obtener acceso físico al ordenador host que ejecuta IIS podría arrancar una sesión interactiva en la consola.

Para permitir la Autenticación Básica, en el pestaña **Directory Security** de la consola, nos aseguramos que que la opción `Allow Anonymous` está a "on" y todas las demás opciones están a "off".

Fichero iisproxy.ini de Ejemplo

Aquí tenemos un fichero **iisproxy.ini** de ejemplo, para usarlo con un sólo servidor WebLogic (sin clusters). Las línea de comentarios empiezan con el carácter "#":

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

Aquí tenemos un fichero **iisproxy.ini** de ejemplo con un cluster de Servidores WebLogic:

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

Nota:
Si estamos usando SSL entre el Plug-In y el servidor WebLogic el número de puerto debería está definido como el puerto de escucha SSL.

Usar SSL con el Plug-In de Microsoft Internet Information Server

Podemos usar el protocolo Secure Sockets Layer (SSL) para proteger la conexión entre el Plug-In de Microsoft Internet Information Server y el Servidor WebLogic. El protocolo SSL proporciona confidencialidad e integridad de los datos pasados entre el Plug-In de Microsoft Internet Information Server y el Servidor WebLogic. Además, el protocolo SSL permite al plug-in autenticarse a si mismo en el Servidor WebLogic para asegurar que la información es pasada a un principal seguro.
El Plug-In de Microsoft Internet Information Server no usa el protocolo de transporte (http o https) para determinar si se usará o no el protocolo SSL para proteger la conexión entre el plug-in y el Microsoft Internet Information Server. Para usar el protocolo SSL con el Plug-In de Microsoft Internet Information Server, configuramos el Servidor WebLogic que recibe las peticiones pasadas para usar el protocolo SSL. Se usa el puerto de este servidor que está configurado para comunicación segura SSL para comunicar con el Microsoft Internet Information Server.
Para usar el protocolo SSL entre el Plug-In de Microsoft Internet Information Server y el Servidor WebLogic:

1. Configuramos el Servidor WebLogic para SSL. Para más información puedes ver Configurar el Protocolo SSL.
2. Configuramos el puerto de escucha SSL del Servidor WebLogic. (Ver Configurar el Puerto de Escucha).
3. Seleccionamos el parámetro `WebLogicPort` en el fichero **iisproxy.ini** al puerto de escucha configurado en el paso 2.
4. Seleccionamos el parámetro `SecureProxy` en el fichero **iisproxy.ini** a ON.
5. Seleccionamos cualquier parámetro adicional en el fichero **iisproxy.ini** que defina información sobre la conexión SSL. Puedes ver una lista completa de parámetros en Parámetros SSL para Plug-Ins de Servidores Web.

Por ejemplo:

```
WebLogicHost=myweblogic.com
WebLogicPort=7002
SecureProxy=ON
```

Pasar Servlets desde IIS al Servidor WebLogic

Los Servlets se pueden pasar por path si la **iisforward.dll** está registrada como un filtro. Deberíamos llamar a nuestro servelt con una URL similar a esta:

```
http://weblogic:7001/weblogic/myServlet
```

Para pasar servlets si **iisforward.dll** no está registrada como un filtro, debemos configurar el paso por tipo de fichero. Para pasar Servlets por tipo de fichero:

1. Registramos un tipo de fichero arbitrario (extensión) con IIS para pasar la petición al Servidor WebLogic, como se describe en el paso 7 de Instalar el Plug-In de Microsoft Internet Information Server.
2. Registramos nuestro servlets en la Aplicación Web apropiada.
3. Llamamos a nuestro servlet con una URL formada siguiendo este patrón:
4. `http://www.myserver.com/virtualName/anyfile.ext`

donde `virtualName` es el patrón de URL definido en el elemento `<servlet-mapping>` del descriptor de despliegue de la aplicación Web para este servlet, y `ext` es un tipo de fichero (extensión) registrada con IIS para ser pasada para el Servidor WebLogic. la parte `anyfile` de la URL se ignora en este contexto.

Nota:

- *Si los enlaces de imagenes llamados desde el servlet son parte de la aplicación Web, también debemos pasar las peticiones de las imágenes al sevidor WebLogic registrando el tipo de fichero apropiado (probablemente .gif y .jpg) con IIS. Sin embargo, podemos elegir servir estas imágenes directamente desde IIS si se desea*
- *Si el servlet que está siendo pasado tiene enlaces que llaman a otros servlets, estos enlaces también deben ser pasados al servidor WebLogic, conforme al patrón mostrado arriba.*

Probar la Instalación

Después de instalar y configurar el Plug-In de Microsoft Internet Information Server, seguimos los siguientes pasos para desplegarla y probarla:

1. Nos aseguramos de que el Servidor WebLogic e IIS está ejecutándose.
2. Grabamos un fichero JSP dentro del documento raíz de la Aplicación Web por defecto.
3. Abrimos un navegador y seleccionamos la URL con el IIS más **filename.jsp** como se muestra en este ejemplo:
4. `http://myii.server.com/filename.jsp`

   Si **filename.jsp** se ve en nuestro navegador, el plug-in está funcionando.

Errores de Conexión y Control de Fallos en Clustering

Cuando el Plug-In de Microsoft Internet Information Server intenta conectar con el Servidor WebLogic, usa varios parámetros de configuración para determinar cuánto tiempo esperar las conexiones con el host del Servidor WebLogic y, después de establecida la conexión, cuánto esperar por una respuesta. Si el plug-in no puede conectar o no recibe una respuesta, intentará conectar y enviar la petición a otro Servidor WebLogic del Cluster. Si la conexión falla o no hay respuesta de ningún servidor WebLogic del cluster, se envía un mensaje de error.

Fallos de Conexión

El fallo de un host al responder a una petición de conexión podría indicar posibles problemas con la máquina host, problemas de red, u otros varios fallos de servidor.
El fallo de un Servidor WebLogic al responder, podría indicar que WebLoigc no se está ejecutando que no está disponible, un cuelgue de servidor, un problema de base de datos, u otro fallo de aplicación.

Control de Fallos con un Sólo Servidor (sin cluster)

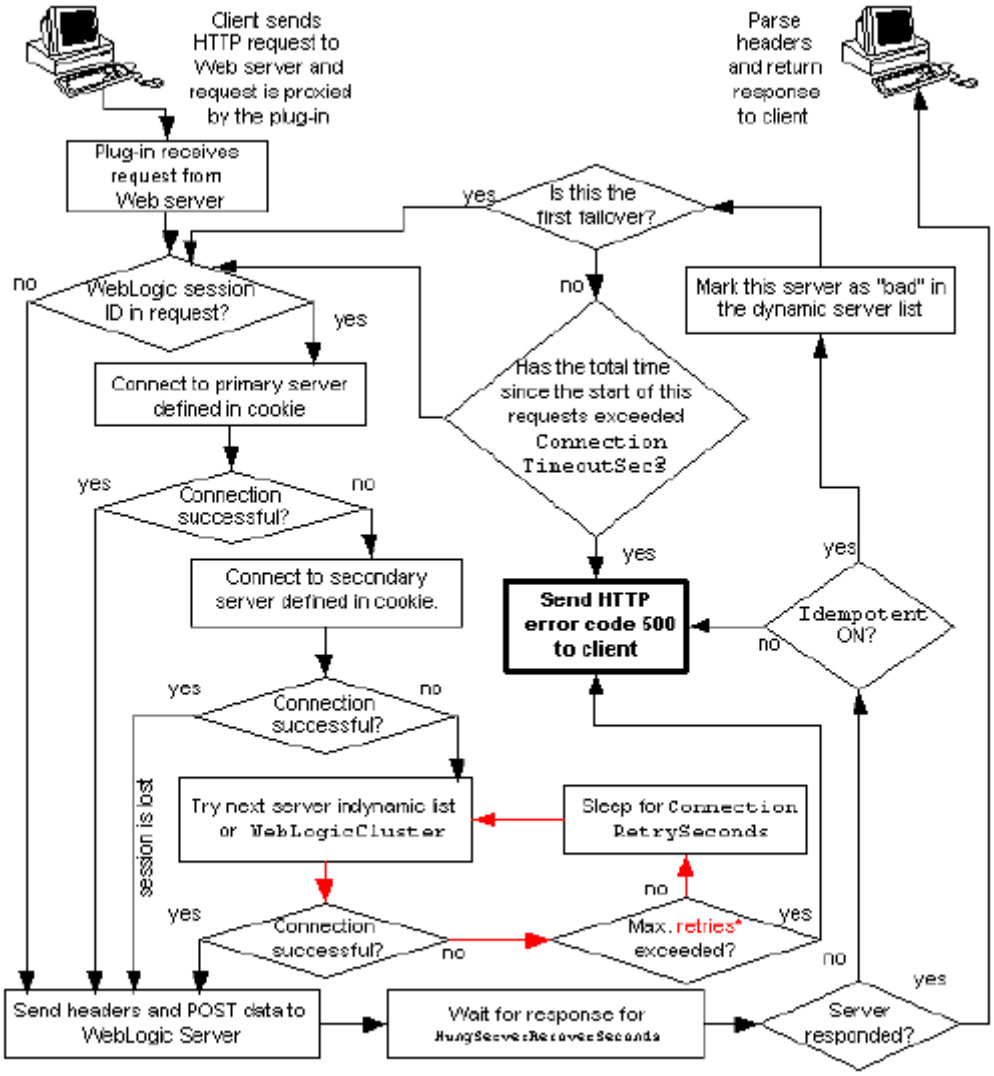
Si estamos ejecutando un sólo Servidor WebLogic se aplica la misma lógica descrita aquí, excepto en que el plug-in sólo intenta conectar con el servidor definido en el parámetro `WebLogicHost`. Si el intento falla, se devuelve un mensaje de error **HTTP 503**. El plug-in continúa intentando conectar con el Servidor WebLogic hasta que se excede el tiempo `ConnectTimeoutSecs`.

La Lista de Servidores Dinámica

Cuando especificamos una lista de servidores WebLogic en el parámetro `WebLogicCluster`, el plug-in usa esa lista como punto de entrada para el balance de carga entre los miembros del cluster. Después de que se haya enrutado la primera petición a uno de esos servidores, se devuelve una lista dinámica que contiene una lista actualizada con los servidores que hay en el cluster. La lista actualizada añade cualquier nuevo servidor en el cluster y borra cualquier otro que haya dejado de formar parte de él, o que haya fallado al responder peticiones. Esta lista se actualiza automáticamente con la respuesta HTTP cuando ocurre un cambio en el cluster.

Control de Fallos, Cookies y Sesiones HTTP

Cuando una petición contiene una información de sesión almacenada en un cookie, en los datos POST, o codificando la URL, la ID de la sesión contiene una referencia al servidor específico en que se estableció originalmente la sesión (llamado servidor primario) y una referencia a un servidor adicional donde se ha replicado la sesión original (llamado servidor secundario). Una petición que contiene una cookie intenta conectar con el servidor primario, si el intento falla, la petición se enruta hacia el servidor secundario. Si ambos servidores fallan, la sesión se pierde y el plug-in intenta hacer una nueva conexión con otro servidor de la lista dinámica del cluster. Puedes ver más información en la siguiente figura:



*The Maximum number of retries allowed in the red loop is equal to ConnectTimeoutSecs ÷ ConnectRetrySecs.

Instalar y Configurar el Plug-In para el Netscape Enterprise Server (NSAPI)

Introducción al Plug-In para el Netscape Enterprise Server (NSAPI)

El Plug-In de Netscape Enterprise Server permite que las peticiones sean pasadas desde Netscape Enterprise Server (NES, también llamado iPlanet) hacia el servidor WebLogic.
El plug-in se ha pensado para un entorno en el que un Servidor Netscape Enterprise sirve páginas estáticas, y otra parte del árbol de documentos (páginas dinámicas mejor generadas por Servlets HTTP o JavaServer Pages) se delega en el Servidor WebLogic, que podría estar operando en un proceso diferente, posiblemente en un host diferente. Las conexiones entre el Servidor WebLogic y el Plug-In de Netscape Enterprise Server se realizan usando texto claro o Secure Sockets Layer (SSL). En el usuario final -- el navegador -- las peticiones HTTP delegadas al Servidor WebLogic parecerán que vienen de la misma fuente.
El tunneling HTTP también puede operar a través del plug-in, proporcionando acceso a los servicios WebLogic a los clientes que no sean navegadores.
El Plug-In Netscape Enterprise Server Plug-In opera como un módulo NSAPI (ver http://home.netscape.com/servers/index.html) dentro de un servidor Netscape Enterprise Server. El módulo NSAPI lo carga NES en la arrancada, y luego se le delegan ciertas peticiones. NSAPI es similar a un servlet HTTP (Java), excepto en que NSAPI está escrito en el código nativo de la plataforma.
Para más información sobre las versiones de los servidores Netscape Enterprise Server e iPlanet soportadas, puedes ver la página http://e-docs.bea.com/wls/platforms/index.html#plugin.

Almacén de Conexiones y Keep-Alive

El Plug-In WebLogic Server NSAPI proporciona eficiencia de rendimiento usando un almacén de conexiones re-utilizables desde el plug-in hacia el servidor WebLogic. El plug-in NSAPI implementa automáticamente conexiones keep-alive entre el plug-in y el servidor WebLogic. Si una conexión está inactiva durante más de 30 segundo, o un valor definido por el usuario, la conexión se cierra.

Pasar (proxy) Peticiones

El Plug-In pasa peticiones al servidor WebLogic basándose en una configuración que nosotros especificamos. Podemos pasar peticiones basándonos en la URL de las peticiones (o en una porción de la URL). Esto se llama proxy por path. También podemos pasar peticiones basándonos en el tipo MIME del fichero solicitado. También podemos usar una combinación de ambos métodos. Si una petición cumple los dos criterios, se pasa por path. También podemos especificar parámetros adicionales para aquellos tipos de peticiones que definen un comportamiento adicional del plug-in.

Instalar el Plug-In para Netscape Enterprise Server

Para instalar y configurar el Plug-In de Netscape Enterprise Server:

1. Copiar la librería:

El módulo plug-in NSAPI WebLogic se distribuye como un objeto compartido (.so) sobre plataformas UNIX y como una librería de enlace dinámico (.dll) sobre Windows. Estos ficheros se localizan respectivamente en los directorios **/lib** o **/bin** de nuestra dirección del servidor WebLogic.
Elegimos el fichero de librería apropiada para nuestro entorno desde la página http://e-docs.bea.com/wls/platforms/index.html#plugin y copiamos ese fichero al sistema de ficheros donde está localizado NES.

2. Modificamos el fichero **obj.conf**. Este fichero define qué peticiones son pasadas al servidor WebLogic y otra información de configuración.
3. Si estamos pasando peticiones por tipo MIME:
   - Añadimos las líneas apropiadas al fichero **obj.conf**.
   - Añadimos cualquier nuevo tipo MIME referenciado en el fichero **obj.conf** al fichero **MIME.types**. Podemos añadir tipos MIME usando la consola del servidor Netscape o editando directamente este fichero.

Para editar directamente el fichero **MIME.types**, abrimos el fichero para editarlo y tecleamos la siguiente línea:

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

Nota:
Para NES 4.0 (iPlanet), en lugar de añadir el tipo MIME para JSPs, cambiamos el tipo MIME existente de:
`magnus-internal/jsp`
a
`text/jsp`

Para usar la consola de Netscape, seleccionamos **Manage Preferences --> Mime Types**, y hacemos las adiciones o ediciones.

4. Desplegamos y probamos el Plug-In de Netscape Enterprise Server:
   - Arrancamos WebLogic Server.
   - Arrancamos Netscape Enterprise Server. Si NES ya está ejecutándose, debemos reiniciar o aplicar las nuevas selecciones desde la consola para que tengan efecto.
   - Para probar el Plug-In de Netscape Enterprise Server, abrimos un navegador y seleccionamos la URL hacia el Enterprise Server + /weblogic/, lo que nos debería traer la página HTML por defecto del servidor WebLogic, la página de bienvenida o el servlet por defecto, según lo definido en la aplicación Web por defecto del Servidor WebLogic, como se muestra en este ejemplo:
   - `http://myenterprise.server.com/weblogic/`

Modificar el Fichero obj.conf

Para usar el Plug-In de Netscape Enterprise Server, debemos hacer varias modificaciones en el fichero **obj.conf** de NES. Estas modificaciones especifican cómo se pasan las peticiones al servidor WebLogic. Podemos pasar peticiones por URL o por tipo MIME. El procedimiento para cada uno se describe más adelante en esta sección. El fichero **obj.conf** de Netscape es muy estricto en cuanto a la situación del texto. Para evitar problemas, observaremos estos cuidados en el objeto **obj.conf** :
Eliminar los espacios en blanco al principio y al final. Los espacios en blanco extras pueden hacer que falle nuestro Servidor Netscape. Si debemos introducir más caracteres de los que caben en una línea, situamos una barra invertida (\) al final de la línea y continuamos tecleado en la siguiente. La barra invertida añade directamente el fin de la primera línea al inicio de la segunda, debemos asegurarnos de no usar ningún espacio, ni al final de la primera línea (antes de la barra invertida), ni al inicio de la segunda línea. No debemos dividir atributos en varias líneas. (por ejemplo, todos los servidores de un cluster deben listarse en la misma línea, siguiendo a WebLogicCluster). Si no existe en la configuración un parámetro requerido, cuando el objeto sea invocado lanzará un error HTML que avisa del parámetro omitido en la configuración.
Para configurar el fichero **obj.conf**:

1. Localizamos y abrimos **obj.conf**.

Este fichero está en la siguiente localización:

```
NETSCAPE_HOME/https-INSTANCE_NAME/config/obj.conf
```

Donde `NETSCAPE_HOME` es el directorio raíz de la instalación de NES, y `INSTANCE_NAME` es el ejemplar o configuración particular del servidor que estamos usando. Por ejemplo, en una máquina UNIX, llamada **myunixmachine**, el fichero **obj.conf** se encontraría en:

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

2. Instruimos a NES para que cargue la librería nativa como un módulo NSAPI.

Añadimos las siguientes líneas al principio del fichero **obj.conf**. Estas líneas instruyen a NES para que cargue la librería nativa (el fichero .so o .dll) como un modulo NSAPI:

```
Init fn="load-modules" funcs="wl_proxy,wl_init"\
shlib=/usr/local/netscape/plugins/SHARED_LIBRARY
Init fn="wl_init"
```

donde `SHARED_LIBRARY` es el objeto compartido o dll (por ejemplo **libproxy.so**) que hemos instalado en el paso 1 de Instalar y Configurar el Plug-In de Netscape Enterprise Server. La función `load-modules` etiqueta la librería compartida para cargarla cuando arranque NES. Los valores `wl_proxy` y `wl_init` identifican las funciones que ejecuta el Plug-In de Netscape Enterprise Server.

3. Si queremos pasar peticiones por URL, (también llamado paso por path), creamos una etiqueta `<Object>` separada por cada URL que queramos pasar y definimos el parámetro `PathTrim`. Pasar por path tiene precedencia sobre el paso por tipo MIME. Lo siguiente es un ejemplo de un etiqueta `<Object>` que pasa peticiones que contengan el string `*/weblogic/*`:

```
4.  <Object name="weblogic" ppath="*/weblogic/*">
5.  Service fn=wl_proxy WebLogicHost=myserver.com\
6.  WebLogicPort=7001 PathTrim="/weblogic"
7.  </Object>
```

Para crear una etiqueta `<Object>` para pasar peticiones por URL:

- Especificamos un nombre para este objeto (opcional) dentro de la etiqueta `<Object>` abierta usando el nombre del atributo. Este nombre es sólo informativo y no se usa en el Plug-In de Netscape Enterprise Server. Por ejemplo:

```
<Object name=myObject ...>
```

- Especificamos la URL a pasar dentro de la etiqueta `<Object>`, usando el atributo `ppath`. Por ejemplo:

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

El valor del atributo `ppath` puede ser cualquier string que identifique peticiones para el Servidor WebLogic. Cuando usamos `ppath`, cada petición que contenga ese path será redirigida. Por ejemplo, un `ppath` de ***/weblogic/*** redirige todas las peticiones que empiecen con **http://enterprise.com/weblogic** al Plug-In de Netscape Enterprise Server, que envía la petición al host o cluster del WebLogic especificado.

- Añadimos la directiva `Service` entre las etiquetas `<Object>` y `</Object>`. En esta directiva podemos especificar cualquier parámetro válido como parejas nombre=valor. Separamos los distintos parámetros con un y solo un espacio en blanco. Por ejemplo:

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

Debemos especificar los siguientes parámetros:

- Para un servidor WebLogic sin Cluster: `WebLogicHost` y `WebLogicPort`.
- Para un servidor WebLogic con Cluster: `WebLogicCluster`.

La directiva `Service` siempre debe empezar con `Service fn=wl_proxy`, seguido por parejas de parámetros nombre=valor válidas.
Aquí tenemos un ejemplo de definiciones de objetos para dos `ppaths` distintos que identifican peticiones que deben ser enviadas a diferentes ejemplares de WebLogic Server:

```
<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>
```

Nota:
Los parámetros que no son obligatorios, como `PathTrim`, se pueden usar para configurar la forma en que `ppath` se pasa a través del Plug-In de Netscape Enterprise Server.

8. Si queremos pasar por tipo MIME, el tipo MIME debe estar listado en el fichero **MIME.types**. Para ver instrucciones sobre como modificar este fichero puedes ir al paso 3 de Instalar y Configurar el Plug-In de Netscape Enterprise Server.

Todas las peticiones con una extensión del tipo MIME designado (por ejemplo, .jsp) pueden pasarse al Servidor WebLogic sin importar la URL.
Para pasar todas las peticiones de un cierto tipo de fichero al Servidor WebLogic:

- Añadimos una directiva `Service` a la definición del objeto por defecto (`<Object name=default ...>`). Por ejemplo, para pasar todas las JSPs a un Servidor WebLogic, deberíamos añadir la siguiente directiva `Service` después de la última línea que empieza con:
- `NameTrans fn=....`

y antes de la línea que empieza con:

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

Esta directiva pasa todos los ficheros con extensión .jsp al servidor WebLogic designado, donde son servidas con una URL como esta:

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

El valor del parámetro `PathPrepend` debería corresponder al contexto raíz de una aplicación Web que está desplegada en el Servidor o Cluster WebLogic al que pasamos las peticiones.
Después de añadir las entradas para el Plug-in del Netscape Enterprise Server, la definición del objeto por defecto debería ser similar a la del siguiente ejemplo:

```
<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>
```

- Añadimos una sentencia `Service` a la definición del objeto por defecto para todos los otros tipos MIME que queramos pasar al Servidor WebLogic.

9. Si queremos permitir el tunneling-HHTP (opcional):

Añadimos la siguiente definición de objeto al fichero **obj.conf**, sustituyendo el nombre del host o cluster WebLogic y el número de puerto por el que deseemos nosotros:

```
<Object name="tunnel" ppath="*/HTTPClnt*">
Service fn=wl_proxy WebLogicHost=192.192.1.4\
WebLogicPort=7001
</Object>
```

Usar SSL con el Plug-In de NSAPI

Podemos usar el protocolo Secure Sockets Layer (SSL) para proteger la conexión entre el Plug-In de Netscape Enterprise Server, y el Servidor WebLogic. El protocolo SSL proporciona confidencialidad e integridad de los datos pasados entre el Plug-In de Netscape Enterprise Server y el Servidor WebLogic. Además, el protocolo SSL permite al plug-in autenticarse a si mismo en el Servidor WebLogic para asegurar que la información es pasada a un principal seguro.
El Plug-In de Netscape Enterprise Server no usa el protocolo de transporte (http o https) especificado en la petición HTTP (normalmente por el navegador) para determinar si se usa o no el protocolo SSL para proteger los datos de la conexión entre el Plug-In de Netscape Enterprise Server y el Servidor WebLogic.
Para usar el protocolo SSL entre el Plug-In de Netscape Enterprise Server y el Servidor WebLogic:

1. Configuramos el Servidor WebLogic para SSL. (Ver Configurar el Protocolo SSL).
2. Configuramos el puerto de escucha SSL del Servidor WebLogic. (Ver Configurar el Puerto de Escucha).
3. Seleccionamos el parámetro `WebLogicPort` en la directiva **Service** al puerto de escucha configurado en el paso 2.
4. Seleccionamos el parámetro `SecureProxy` en la directiva **Service** a ON.
5. Seleccionamos cualquier parámetro adicional en la directiva **Service** que defina información sobre la conexión SSL. Puedes ver una lista completa de parámetros en Parámetros SSL para Plug-Ins de Servidores Web.

Errores de Conexión y Control de Fallos en Clustering

Cuando el Plug-In de Netscape Enterprise Server intenta conectar con el Servidor WebLogic, usa varios parámetros de configuración para determinar cuánto tiempo esperar las conexiones con el host del Servidor WebLogic y, después de establecida la conexión, cuánto esperar por una respuesta. Si el plug-in no puede conectar o no recibe una respuesta, intentará conectar y enviar la petición a otro Servidor WebLogic del Cluster. Si la conexión falla o no hay respuesta de ningún servidor WebLogic del cluster, se envía un mensaje de error.

Fallos de Conexión

El fallo de un host al responder a una petición de conexión podría indicar posibles problemas con la máquina host, problemas de red, u otros varios fallos de servidor.
El fallo de un Servidor WebLogic al responder, podría indicar que WebLogic no se está ejecutando que no está disponible, un cuelgue de servidor, un problema de base de datos, u otro fallo de aplicación.

Control de Fallos con un Sólo Servidor (sin cluster)

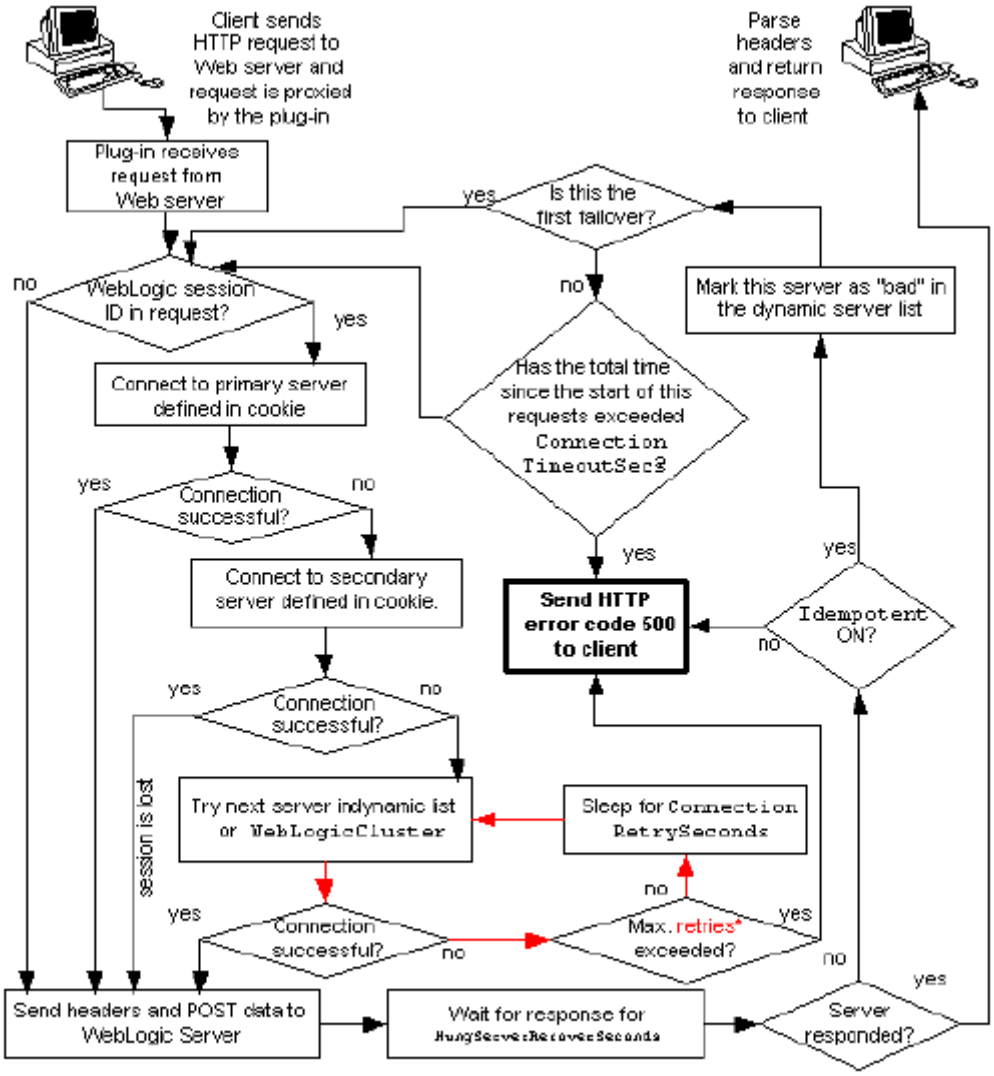
Si estamos ejecutando un sólo Servidor WebLogic se aplica la misma lógica descrita aquí, excepto en que el plug-in sólo intenta conectar con el servidor definido en el parámetro `WebLogicHost`. Si el intento falla, se devuelve un mensaje de error **HTTP 503**. El plug-in continúa intentando conectar con el Servidor WebLogic hasta que se excede el tiempo `ConnectTimeoutSecs`.

La Lista de Servidores Dinámica

Cuando especificamos una lista de servidores WebLogic en el parámetro `WebLogicCluster`, el plug-in usa esa lista como punto de entrada para el balance de carga entre los miembros del cluster. Después de que se haya enrutado la primera petición a uno de esos servidores, se devuelve una lista dinámica que contiene una lista actualizada con los servidores que hay en el cluster. La lista actualizada añade cualquier nuevo servidor en el cluster y borra cualquier otro que haya dejado de formar parte de él, o que haya fallado al responder peticiones. Esta lista se actualiza automáticamente con la respuesta HTTP cuando ocurre un cambio en el cluster.

Control de Fallos, Cookies y Sesiones HTTP

Cuando una petición contiene una información de sesión almacenada en un cookie, en los datos POST, o codificando la URL, la ID de la sesión contiene una referencia al servidor específico en que se estableció originalmente la sesión (llamado servidor primario) y una referencia a un servidor adicional donde se ha replicado la sesión original (llamado servidor secundario). Una petición que contiene una cookie intenta conectar con el servidor primario, si el intento falla, la petición se enruta hacia el servidor secundario. Si ambos servidores fallan, la sesión se pierde y el plug-in intenta hacer una nueva conexión con otro servidor de la lista dinámica del cluster. Puedes ver más información en la siguiente figura:



*The Maximum number of retries allowed in the red loop is equal to ConnectTimeoutSecs ÷ ConnectRetrySecs.

Comportamiento Contra Fallos Cuando se usan Firewalls y Directores de Carga

En la mayoría de las configuraciones el Plug-In de Netscape Enterprise Server envía peticiones a un ejemplar principal de un cluster. Cuando ese ejemplar no está disponible, la petición cae sobre el ejemplar secundario. Sin embargo, en algunas configuraciones que usan combinaciones de firewalls y directores de carga, cualquiera de los servidores (firewalls o director de carga) puede aceptar la petición y devolver una conexión con éxito mientras el ejemplar principal del cluster WebLogic no está disponible. Después de intentar dirigir la petición al ejemplar principal del Servidor WebLogic (que no está disponible), la petición es devuelta al plug-in como un “connection reset”.
La peticiones que se ejecutan a través de combinaciones de firewalls (con o sin directores de carga) son manejadas por el Servidor WebLogic. En otras palabras, responde a una condición de **connection reset** sobre un ejemplar secundario del Servidor WebLogic.

Ejemplo de fichero obj.conf (sin usar cluster WebLogic)

Abajo hay un ejemplo de las líneas que se deberían añadir al fichero **obj.conf** si no estamos usando un cluster. Podemos usar este ejemplo como una plantilla que podemos modificar para que se adapte a nuestro entorno y servidor. Las líneas que empiezan con '#' son comentarios.

Nota:
*Debemos asegurarnos de que no incluimos ningún espacio en blanco extraño en el fichero **obj.conf**. Al copiar/pegar desde los ejemplos de abajo algunas veces se añaden espacios en blanco extras que pueden crear problemas cuando se lee el fichero.*

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

Ejemplo de fichero obj.conf (usando un cluster WebLogic)

Abajo hay un ejemplo de las líneas que se deberían añadir al fichero **obj.conf** si estamos usando un cluster. Podemos usar este ejemplo como una plantilla que podemos modificar para que se adapte a nuestro entorno y servidor. Las líneas que empiezan con '#' son comentarios.

Nota:
Debemos asegurarnos de que no incluimos ningún espacio en blanco extraño en el fichero ***obj.conf****. Al copiar/pegar desde los ejemplos de abajo algunas veces se añaden espacios en blanco extras que pueden crear problemas cuando se lee el fichero.*

```
## ------------- BEGIN SAMPLE OBJ.CONF CONFIGURATION ---------
# (using a WebLogic 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 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 ---------
```

Manejar la Seguridad

Introducción al Control de la Seguridad

La implementación de la Seguridad en un Servidor WebLogic ampliamente desplegado consiste en campos de configuración que definen la política de seguridad para ese despliegue. El Servidor WebLogic proporciona una Consola de Administración para ayudarnos a definir la política de seguridad. Usando la consola, especificamos valores específicos de seguridad para los siguientes elementos de nuestro servidor:

- Reinos
- Usuarios y Grupos
- Listas de Control de Acceso (ACLs) y permisos para recursos del Servidor WebLogic
- Protocolo SSL
- autentificación mutua
- Auditoria de proveedores
- Filtros Personalizados
- Propagación del contexto de seguridad

Como las características de seguridad están muy relacionadas, es dificil determinar dónde empezar cuando se configura la seguridad. De hecho, definir la seguridad para nuestro despliegue de Servidor WebLogic podría ser un proceso iterativo. Aunque más de una secuencia de pasos podría funcionar, recomendamos el siguiente procedimiento:

1. Cambiar la password del sistema para proteger nuestro Servidor WebLogic.
2. Especificar un reino de seguridad. Por defecto, el Servidor WebLogic está instalado con el reino **File**. Sin embargo, podríamos preferir un reino de seguridad alternativo o un reino de seguridad personalizado.
3. Definir usuarios para el reino de seguridad. Podemos organizar posteriormente los usuarios implementando grupos en el reino de seguridad.
4. Definir ACLs y permisos para los recursos de nuestro Servidor WebLogic.
5. Proteger la conexión de red entre los clientes y el Servidor WebLogic usando el protocolo SSL. Cuando se implementa SSL, el Servidor WebLogic usa su certificado digital, enviado por una autoridad de certificación creíble, para autentificar la os clientes. Este paso es opcional pero lo recomendamos.
6. Mejorar la protección de nuestro Servidor WebLogic implementando autentificación mutua. Cuando ésta se implementa, el propio Servidor WebLogic debe autentificarse ante el cliente y luego el cliente se debe autentificar ante el Servidor WebLogic. De nuevo este paso es opcional, pero los recomendamos.

Esta sección describe estos pasos de configuración y los campos que configuramos en la Consola de Administración.

Configurar el Controlador de Seguridad Java

Cuando ejecutamos WebLogic Server bajo Java 2 (JDK 1.3), WebLogic Server usa el "Java Security Manager" para controlar los recursos del Servidor WebLogic. El controlador de seguridad Java necesita incluir un fichero de política de seguridad para configurar los permisos. La distribución de WebLogic Server incluye un fichero de política de seguridad llamado **weblogic.policy** que contiene un conjunto de permisos por defecto. Con este fichero, podemos arrancar WebLogic Server sin crear primero nuestro propio fichero de política de seguridad.
Editar las siguientes líneas en el fichero **weblogic.policy**, reemplazamos la localización del directorio en que instalamos WebLogic Server.

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

Una vez realizados estos pasos, recomendamos realizar los siguientes pasos:

- Hacer una copia de seguridad del fichero **weblogic.policy** y ponerla en un sitio seguro.
- Seleccionar los permisos sobre las protecciones del fichero **weblogic.policy** para que sólo el administrador del sistema donde está desplegado el Servidor WebLogic tenga privilegios de lectura y escritura.

Configuramos las propiedades `java.security.manager` y `java.security.policy` en la línea de comandos Java cuando arrancamos WebLogic Server. Estas propiedades realizan las siguientes funciones:

- La propiedad `java.security.manager` especifica que la Máquina Virtual Java (JVM) usará una política de seguridad. No necesitamos especificar ningún argumento para esta propiedad.
- La propiedad `java.security.policy` especifica la localización del fichero de política de seguridad a utilizar por la JVM. El argumento de esta propiedad es el nombre totalmente cualificado del fichero **weblogic.policy**. Por ejemplo:
- `$ java ... -Djava.security.manager\`
- `-Djava.security.policy==c:/weblogic/weblogic.policy`

Cambiar la Password del Sistema

Durante la instalación especificamos una password para el usuario **system**. La password especificada se asocia con el usuario **system** en el Servidor WebLogic y es almacenada en el fichero `fileRealm.properties` en el directorio **\wlserver6.0\config\mydomain**.
La password especificada corresponde al Servidor de Administración para el dominio y para todos los servidores controlados asociado con ese Servidor de Administración.
La password está encriptada y además se protege cuando el Servidor WebLogic le aplica un poco de basura. Para mejorar la seguridad, recomendamos cambiarla frecuentemente.
Para cambiar la password de **system**, hacemos lo siguiente:

1. Abrimos la ventana **Users** en la Consola de Administración.
2. Introducimos **system** en el campo **User**.
3. Introducimos una nueva password en el campo **Password**.
4. Confirmamos la password.

Cuando usamos un Servidor de Administración y servidores controlados en un dominio, los servidores controlados siempre deben usar la password del Servidor de Administración del dominio. Siempre debemos cambiar la password desde la Consola de Administración, ya que así, la nueva password será propagada a todos los servidores controlados del dominio. Recuerda que la password de **system** debe ser la misma para todos los servidores de un dominio.

Nota:
*Los dominios **Petstore** y **ExampleServer** todavía almacenan la password de **system** en un fichero **password.ini**. Cuando usemos estos dominios, debemos modificar la password de **system** modificando la información de la password en ese fichero.*

Especificar un Reino de Seguridad

Por defecto el Servidor WebLogic se instala con un reino de seguridad **File**. Antes de usar el reino **File**, necesitamos definir varios campos que gobiernan el uso de este reino. Podemos configurar estos campos en la pestaña **Filerealm** de la ventana **Security** de la Consola de Administración.
La siguiente tabla describe cada campo de la pestaña **Filerealm**:

Campo	Descripción
`Max Users`	Especifica el número máximo de usuarios a usar con el reino **File**. Este reino está pensado para usarse con menos de 10000 usuarios. El valor mínimo para este campo es 1 y el máximo es 10000. El valor por defecto es 1000.
`Max Groups`	Especifica el número máximo de grupos a usar con el reino **File**. El valor mínimo para este campo es 1 y el máximo es 10000. El valor por defecto es 1000.
`Max ACLs`	Especifica el número máximo de ACLs a usar con el reino **File**. El valor mínimo para este campo es 1 y el máximo es 10000. El valor por defecto es 1000.

Si por alguna razón se corrompe o destruye el fichero `fileRealm.properties`, deberemos reconfigurar toda la información de seguridad del Servidor WebLogic. Por lo tanto, recomendamos seguir los siguientes pasos:

1. Hacer una copia de seguridad del fichero **fileRealm.properties** y ponerla en un sitio seguro.
2. Seleccionar los permisos de protección del fichero **fileRealm.properties** para que sólo el administrador del Servidor WebLogic tenga privilegios de lectura y escritura y que ningún otro usuario tenga privilegios.

Nota:
*También deberíamos hacer una copia de seguridad del fichero **SerializedSystemIni.dat** para el reino **File**. Para más información sobre el fichero **SerializedSystemIni.dat**, puedes ir a la sección Proteger Passwords.*

Si en lugar del reino **File**, queremos usar uno de los reinos de seguridad alternativos proporcionados por el Servidor WebLogic o un reino de seguridad personalizado, configuramos los campos del reino deseado y reiniciamos el Servidor WebLogic.

Configurar el Reino Caching

Nota:
Todas las instrucciones de configuración están basadas en el uso de la Consola de Administración.

El reino **Caching** trabaja con el reino **File**, reinos de seguridad alternativos o reinos personalizados para satisfacer las peticiones de los clientes con la autentificación y autorización apropiadas. El reino **Caching** almacena los resultados tanto con éxito como sin éxito de las búsquedas del reino. Maneja cachés separados para Usuarios, Grupos, permisos, ACLs y peticiones de autentificación. El reino **Caching** Mejora el rendimiento del Servidor WebLogic cacheando las búsquedas, reduciendo el número de llamadas a otros reinos de seguridad.
El reino **Caching** se instala automáticamente cuando instalamos el Servidor WebLogic: el caché se configura para delegar a otros reinos de seguridad pero no está activado. Tenemos que activar el caché desde la Consola de Administración.
Cuando activamos el caché, el reino **Caching** graba los resultados de una búsqueda de reino en su caché. Los resultados de la búsqueda permanecen en el cache durante el número de segundos especificados definidos por los campos time-to-live (TTL) o el caché se ha llenado. Cuando el caché se llena, los nuevos resultados de búsquedas reemplazan a los resultados más antiguos. Los campos TTL determinan el tiempo que un objeto del caché es válido. Cuando más alto seleccionemos estos campos, con menos frecuencia el reino **Caching** llamará al reino de seguridad secundario. Al reducir la frecuencia de esas llamadas se mejora el rendimiento. La pega es que los cambios en el reino de seguridad subyacente no se reconocen hasta que el objeto del caché haya expirado.

Nota:
Cuando obtenemos un objeto desde un reino de seguridad, el objeto refleja una "foto" de objeto. Para actualizar el objeto, debemos llamar de nuevo al método `get()` *del objeto. Por ejemplo, los miembros de un grupo se seleccionan cuando el grupo es recuperado del reino de seguridad con una llamada al método* ***getgroup()****. Para actualizar los miembros del grupo, debemos llamar de nuevo al método getgroup().*

Por defecto, el reino **Caching** opera sobre la suposición de que el reino secundario es sensible a las mayúsculas. En el caso de un reino de seguridad sensible a las mayúsculas, los propietarios de los nombres de usuario **bill** y **Bill**, por ejemplo, son tratados como usuarios distintos. Los reinos de seguridad **Windows NT** y **LDAP** son ejemplos de reinos de seguridad insensibles a las mayúsculas. Si estamos usando un reino de seguridad que no es sensible a las mayúsculas, debemos desactivar el campo `CacheCaseSensitive`. Cuando se selecciona este campo, el reino **Caching** convierte los nombres de usuarios a minúsculas para que el Servidor WebLogic ofrezca los resultados correctos cuando realiza comparaciones sensibles a las mayúsculas. Cuando definimos o referenciamos Usuarios o Grupos en un reino de seguridad sensible a las mayúsculas, debemos teclearlos en minúsculas.
Configurar el reino **Caching** implica activar varios tipos de cachés (como ACL, Authentication, Group, y Permission) y definir cómo opera cada uno de ellos. Para hacer estas tareas, definimos valores para los campos mostrados en la pestaña **General** de la ventana **Caching Realm Configuration**. Para grabar nuestros cambios, pulsamos sobre el botón **Apply**. Cuando hayamos finalizado de definir los campos, reiniciamos el Servidor WebLogic. La siguiente tabla describe cada uno de los campos de la pestaña **General**:

Campo	Descripción
Name	Muestra el reino de seguridad activo. Este campo no se puede cambiar.
Basic Realm	El nombre de la clase para el reino de seguridad alternativo o el reino de seguridad personalizado que está siendo usado con el reino **Caching**.
Case Sensitive Cache	Define si el reino de seguridad especificado es sensible a las mayúsculas. Por defecto, este campo está activado: el reino es sensible a las mayúsculas. Para usar un reino que lo sea (como los reinos **Windows NT** y **LDAP**), debemos desactivar este campo.

Para activar y configurar el caché ACL, definimos valores para los campos mostrados en la pestaña **ACL** de la ventana **Caching Realm Configuration**. Para grabar nuestros cambios, pulsamos sobre el botón **Apply**. Cuando hayamos finalizado de definir los campos, reiniciamos el Servidor WebLogic. La siguiente tabla describe cada uno de los campos de la pestaña **ACL**:

Campo	Descripción
Enable ACL Cache	Opción para activar el caché ACL
ACL Cache Size	El número máximo de búsquedas ACL a almacenar. El valor por defecto es 211. Este campo debería ser un número primo para un mejor rendimiento de las búsquedas.
ACL Cache Positive TTL	El número de segundos para retener los resultados de una búsqueda exitosa.
ACL Cache Negative TTL	El número de segundos para retener los resultados de una búsqueda sin éxito. El valor por defecto es 10 segundos.

Para activar y configurar el caché Authentication, definimos valores para los campos mostrados en la pestaña **Authentication** de la ventana **Caching Realm Configuration**. Para grabar nuestros cambios, pulsamos sobre el botón **Apply**. Cuando hayamos finalizado de definir los campos, reiniciamos el Servidor WebLogic. La siguiente tabla describe cada uno de los campos de la pestaña **Authentication**:

Campo	Descripción
Enable Authentication Cache	Opción para activar el caché Authentication.
Authentication Cache Size	El número máximo de peticiones `Authenticate` a almacenar en el caché. El valor por defecto es 211. Este campo debería ser un número primo para un mejor rendimiento de la búsqueda.
Authentication Cache TTLPositive	El número de segundos para retener los resultados de una búsqueda exitosa.
Authentication Cache TTLNegative	El número de segundos para retener los resultados de una búsqueda sin éxito. El valor por defecto es 10 segundos.

Para activar y configurar el caché Group, definimos valores para los campos mostrados en la pestaña **Group** de la ventana **Caching Realm Configuration**. Para grabar nuestros cambios, pulsamos sobre el botón **Apply**. Cuando hayamos finalizado de definir los campos, reiniciamos el Servidor WebLogic. La siguiente tabla describe cada uno de los campos de la pestaña **Group**:

Campo	Descripción
Group Cache Enable	Opción para activar el caché Group.
Group Cache Size	El número máximo de búsquedas de `Group` a almacenar en el caché. El valor por defecto es 211. Este campo debería ser un número primo para un mejor rendimiento de la búsqueda.
Group Cache TTLPositive	El número de segundos para retener los resultados de una búsqueda exitosa. El valor por defecto es 60 segundos.
Group Cache TTLNegative	El número de segundos para retener los resultados de una búsqueda sin éxito. El valor por defecto es 10 segundos.
Group Membership Cache TTL	El número de segundos a almacenar los miembros de un grupo antes de actualizarlo. El valor por defecto es 10 segundos.

Para activar y configurar el caché User, definimos valores para los campos mostrados en la pestaña **User** de la ventana **Caching Realm Configuration**. Para grabar nuestros cambios, pulsamos sobre el botón **Apply**. Cuando hayamos finalizado de definir los campos, reiniciamos el Servidor WebLogic. La siguiente tabla describe cada uno de los campos de la pestaña **User**:

Campo	Descripción
Enable User Cache	Opción para activar el caché User.
User Cache Size	El número máximo de búsquedas de `User` a almacenar en el caché. El valor por defecto es 211. Este campo debería ser un número primo para un mejor rendimiento de la búsqueda.
User Cache TTLPositive	El número de segundos para retener los resultados de una búsqueda exitosa. El valor por defecto es 60 segundos.
User Cache TTLNegative	El número de segundos para retener los resultados de una búsqueda sin éxito. El valor por defecto es 10 segundos.

Para activar y configurar el caché Permission, definimos valores para los campos mostrados en la pestaña **Permission** de la ventana **Caching Realm Configuration**. Para grabar nuestros cambios, pulsamos sobre el botón **Apply**. Cuando hayamos finalizado de definir los campos, reiniciamos el Servidor WebLogic. La siguiente tabla describe cada uno de los campos de la pestaña **Permission**:

Campo	Descripción
Enable Permission Cache	Opción para activar el caché Permission.
Permission Cache Size	El número máximo de búsquedas de `Permission` a almacenar en el caché. El valor por defecto es 211. Este campo debería ser un número primo para un mejor rendimiento de la búsqueda.
Permission Cache TTLPositive	El número de segundos para retener los resultados de una búsqueda exitosa. El valor por defecto es 60 segundos.
Permission Cache TTLNegative	El número de segundos para retener los resultados de una búsqueda sin éxito. El valor por defecto es 10 segundos.

Configurar el Reino LDAP

Nota:
*El reino de seguridad **LDAP** ha sido reescrito para mejorar el rendimiento y la configurabilidad de BEA recomienda actualizar nuestra instalación WebLogic Server 6.0 al Service Pack 1.0 para aprovechar esta funcionalidad. El WebLogic Server 6.0 Service Pack 1.0 está disponible para descarga en la página de BEA Systems.*

El reino de seguridad **LDAP** proporciona autentificación a través de un servidor Lightweight Directory Access Protocol (LDAP). Este servidor nos permite manejar todos los usuarios de nuestra organización en un sólo lugar, el directorio LDAP. El reino de seguridad **LDAP** actualmente soporta Netscape Directory Server, Microsoft Site Server, y Novell NDS.
Configurar el reino de seguridad **LDAP** implica definir campos que permitan al reino de seguridad **LDAP** en el Servidor WebLogic comunicarse con el servidor LDAP y campos que describan cómo se almacenan los usuarios y grupos en el directorio LDAP.
Antes de poder usar el reino de seguridad **LDAP**, necesitamos activar el reino **Caching** e introducir el nombre de la clase del reino de seguridad **LDAP** en el campo **Basic Realm**.
Para usar el reino de seguridad **LDAP** en lugar del reino **File**, vamos al nodo **Security --> Realms** en el panel izquierdo de la Consola de Administración. En el panel derecho, pulsamos el enlace **Create a New LDAP Realm**.
Para especificar el nombre del reino de seguridad **LDAP** y el nombre de la clase que contiene ese reino de seguridad definimos los valores de los campos mostrados en la pestaña **General** de la ventana **LDAP Realm Create**. Para grabar nuestros cambios, pulsamos el botón **Apply**. Cuando hayamos terminado de

definir los campos, reiniciamos el Servidor WebLogic. La siguiente tabla describe todos los campos de la pestaña **General**:

Campo	Descripción
Name	El nombre del reino de seguridad **LDAP** como `AccountingRealm`.
Realm Class Name	El nombre de la clase Java que contiene el reino de seguridad **LDAP**. La clase Java debería estar incluida en el CLASSPATH del Servidor WebLogic.

Para permitir la comunicación entre el servidor LDAP y el Servidor WebLogic definimos valores para los campos mostrados en la pestaña **LDAP** en la ventana **LDAP Realm Create**. Para grabar nuestros cambios, pulsamos el botón **Apply**. Cuando hayamos terminado de definir los campos, reiniciamos el Servidor WebLogic. La siguiente tabla describe todos los campos de la pestaña **LDAP**:

Campo	Descripción
LDAPURL	La localización del servidor LDAP. Cambiamos la URL al nombre del ordenador en el que se esté ejecutando el servidor LDAP y el número de puerto en el que esté escuchando. Si queremos que el Servidor WebLogic se conecte al servidor LDAP usando el protocolo SSL, usamos el número de puerto SSL del servidor LDAP en la URL.
Principal	El nombre distinguido (DN) del Usuario LDAP usado por el Servidor WebLogic para conectarse con el servidor LDAP. Este usuario debe poder listar Usuarios y Grupos LDAP.
Credential	La password que autentifica al usuario LDAP, según se define en el campo `Principal`.
Enable SSL	Opción para activar el uso del protocolo SSL para proteger las comunicaciones entre el Servidor LDAP y el Servidor WebLogic. Debemos tener en mente los siguientes puntos: • Desactivar este campo si el servidor LDAP no está configurado para usar el protocolo SSL. • Si seleccionamos el campo `UserAuthentication` a **external**, este campo debe estar activado.
AuthProtocol	El tipo de autentificación usada para autentificar el servidor LDAP. Seleccionamos este campo con uno de los siguientes valores: • `None` para no usar autenticación. • `Simple` para autentificación por password. • `CRAM-MD5` para autentificación certificada. Netscape Directory Server soporta CRAM-MD5. Microsoft Site Server y Novell NDS soportan autentificación Simple.

Para especificar cómo se almacenan los usuarios en el directorio LDAP definimos los campos mostrados en le pestaña **Users** en la ventana **LDAP Realm Create**. Para grabar nuestros cambios, pulsamos el botón **Apply**. Cuando hayamos terminado de definir los campos, reiniciamos el Servidor WebLogic. La siguiente tabla describe todos los campos de la pestaña **Users**:

Campo	Descripción
User Authentication	Determina el método para autentificar usuarios. Seleccionamos este campo a uno de los siguientes valores: • `Local` especifica que el reino de seguridad **LDAP** recupera datos de usuario, incluyendo la password desde el servidor de directorio LDAP, y chequea las password en el Servidor WebLogic. Esta

selección es apropiada para Netscape Directory Server y Microsoft Site Server.

- `External` especifica que el reino de seguridad **LDAP** autentifica un usuario intentando unir el servidor de directorio LDAP con el nombre de usuario y la password suministrados por el cliente del Servidor WebLogic. Si elegimos esta selección, también debemos usar el protocolo SSL. Esta selección es apropiada para Novell NDS.
- Bind

User Password Attribute	La password del usuario LDAP.
User DN	Una lista de atributos que, cuando se combinan con los atributos del campo **User Name Attribute**, identifican únicamente a un usuario LDAP.
User Name Attribute	El nombre de login del usuario LDAP. El valor de este campo puede ser el nombre común de un usuario LDAP pero normalmente es un string abreviado, como el ID de usuario.

Para especificar cómo se almacenan los Grupos en el directorio LDAP, asignamos valores a los campos mostrados en la pestaña **Groups** en la ventana **LDAP Realm Create**. Para grabar nuestros cambios, pulsamos el botón **Apply**. Cuando hayamos terminado de definir los campos, reiniciamos el Servidor WebLogic. La siguiente tabla describe todos los campos de la pestaña **Groups**:

Campo	Descripción
Group DN	La lista de atributos que, combinada con el campo **Group Name Attribute**, idéntica únicamente a un grupo en el directorio LDAP.
Group Name Attribute	El nombre de un Grupo en el directorio LDAP. Normalmente es un nombre común.
Group Is Context	Este checkbox booleano especifica cómo se graban los miembros del Grupo en el directorio LDAP: • Marcamos este checkbox si cada entrada del Grupo contiene un usuario. Por defecto, este campo está activado. • Desmarcamos este checkbox si hay una entrada Grupo que contiene un atributo por cada miembro del grupo.
Group Username Attribute	El nombre del atributo LDAP que contiene un miembro del Grupo en una entrada de Grupo.

Si hemos activado el caché, el reino **Caching** almacena los Usuarios y los Grupos internamente para evitar las búsquedas frecuentes en el directorio LDAP. Todo objeto en el caché de Usuarios y Grupos tiene un campo TTL que puede configurarse en el reino **Caching**. Si hacemos cambios en el directorio LDAP, dichos cambios no se verán reflejados en el reino de seguridad **LDAP** hasta que expiren los objetos almacenados en el caché o sean sacados de él. El TTL por defecto es 60 segundos para búsquedas sin éxito y 10 segundos para búsquedas con éxito. A menos que cambiemos los campos TTL para los cachés de Usuarios y Grupos, los cambios en el directorio LDAP deberían verse reflejados en el reino de seguridad **LDAP** en 60 segundos.
Su algún código del lado del servido ha realizado alguna búsqueda en el reino de seguridad LDAP, como una llamada a `getUser()` sobre el reino de seguridad **LDAP**, el objeto devuelto por el reino no puede liberarse hasta que el código lo libere. Por lo tanto, un Usuario autentificado por el Servidor WebLogic permanece válido mientras que persista la conexión. incluso si borramos al usuario del directorio LDAP.

Configurar un Reino de Seguridad de Windows NT

El reino de seguridad de Windows NT usa información de cuenta definida en un dominio Windows NT para autentificar Usuarios y Grupos. Podemos ver los Usuarios y Grupos en el reino de seguridad Windows NT a través de la Consola de Administración, pero debemos manejar los Usuarios y Grupos a través de las facilidades proporcionadas por Windows NT.
El reino de seguridad de Windows NT proporciona autentificación (Usuarios y Grupos) pero no autorización (ACLs). El usuario **system** definido en el Servidor WebLogic también debe ser declarado en el dominio Windows NT. Sobre una plataforma Windows NT, el Servidor WebLogic se debe ejecutar bajo la cuenta de usuario **system**, y los clientes deben suministrar la password del usuario **system** para autentificarse satisfactoriamente. Cuando definamos la cuenta **system** en Windows NT, nos debemos asegurar de que el propietario de la cuenta tiene permisos administrativos y que puede leer información relacionada con la seguridad desde el controlador del domino Windows NT.
Para usar el reino de seguridad Windows NT, debemos ejecutar el Servidor WebLogic como un servicio Windows NT sobre un ordenador del dominio Windows NT. No tenemos porque ejecutar el Servidor WebLogic sobre un controlador de dominio.
Como el Servidor WebLogic lee ACLs desde el fichero **fileRealm.properties** durante la arrancada, debemos reiniciar el Servidor WebLogic después de cambiar un ACL. Si usamos Grupos con ACLs, podemos reducir la frecuencia con la que deberemos reiniciar el Servidor WebLogic. Cambiar los miembros de un Grupo Windows NT nos permite manejar dinámicamente los accesos de Usuarios individuales a los recursos del Servidor WebLogic.
Antes de poder usar el reino de seguridad **Windows NT**, necesitamos activar el reino **Caching** e introducir el nombre de la clase del Reino de seguridad de **Windows NT** en el campo **Basic Realm**.
Para usar el reino de seguridad de **Windows NT** en vez de usar el reino **File**, vamos al nodo **Security -> Realms** en el panel izquierdo de la Consola de Administración. En el panel derecho pulsamos sobre el enlace **Create a New NT Realm**.
Configurar el reino de Seguridad de **Windows NT** implica configurar campos que definen un nombre para el reino y el ordenador sobre el que se está ejecutando el dominio Windows NT. Para especificar un nombre de reino, debemos definir valores para los campos mostrados en la ventana **NT Realm Create** de la Consola de Administración. Para grabar nuestros cambios, pulsamos el botón **Apply**. Cuando hayamos terminado de definir los campos, reiniciamos el Servidor WebLogic. La siguiente tabla describe todos los campos de la ventana **NT Realm Configuration**:

Campo	Descripción
Name	El nombre del reino de Seguridad Windows NT, como, `AccountingRealm`.
Realm Class Name	El nombre de la clase Java que implementa el reino de seguridad Windows NT. La clase Java necesita estar en el CLASSPATH del Servidor WebLogic.
Primary Domain	El host y número de puerto del ordenador donde están definidos los Usuarios y Grupos para el dominio Windows NT. Si introducimos varios host y números de puertos, debemos usar una lista separada por comas.

Una vez que tengamos configurado el reino de seguridad **Windows NT** en la Consola de Administración. necesitamos definir el usuario **system** en Windows NT:

1. Usamos la cuenta **Administrator** para entrar en el dominio Windows NT que estamos usando con el Servidor WebLogic.
2. Vamos a **Programs -> Administrative Tools**.
3. Seleccionamos **User Manager**.
4. Definimos el usuario **system**.
5. Marcamos la opción **Show Advanced User Rights**.
6. Seleccionamos el **Act** como parte de la opción de sistema operativo desde el menú desplegable **Rights**.
7. Pulsamos el botón **Add**.
8. Nos aseguramos de la variable de entorno PATH de Windows NT incluye el directorio **\wlserver6.0\bin**. (WebLogic Server carga la librería **W1ntrealm.dll** desde este directorio.)

Configurar un Reino de Seguridad UNIX

El reino de seguridad **UNIX** es un pequeño programa nativo, **wlauth**, para buscar Usuarios y Grupos y para autentificar Usuarios sobre las bases de nombres de login UNIX y sus passwords. Sobre algunas plataformas, **wlauth** usa PAM (Pluggable Authentication Modules) que nos permite configurar los servicios de autentificación en el sistema operativo sin alterar las aplicaciones que usan el servicio. Sobre plataformas para las que PAM no está disponible, **wlauth** usa un mecanismo de login estándar, que incluye passwords sombreadas, cuando son soportadas.
Como WebLogic Server lee ACLs desde el fichero **fileRealm.properties** durante la arrancada, debemos reiniciar el Servidor WebLogic después de cambiar un ACL. Si usamos Grupos con ACLs, podemos reducir la frecuencia con la que deberemos reiniciar el Servidor WebLogic. Cambiar los miembros de un Grupo UNIX nos permite manejar dinámicamente los accesos de Usuarios individuales a los recursos del Servidor WebLogic.
El programa **wlauth** ejecuta setuid root. Necesitamos permisos de `root` para modificar los atributos del fichero del programa **wlauth** y seleccionar el fichero de configuración de PAM para **wlauth**.
Realizamos los siguientes pasos para configurar el reino de seguridad **UNIX**:

1. Si el Servidor WebLogic está instalado en un disco de red, copiamos el fichero **wlauth** a un sistema de ficheros en el ordenador que ejecute el Servidor WebLogic, por ejemplo, el directorio **/usr/sbin**. El fichero **wlauth** está en el directorio **weblogic/lib/arch**, donde `arch` es el nombre de nuestra plataforma.
2. Como usuario **root**, ejecutamos los siguientes comandos para cambiar al propietario y los permisos del fichero **wlauth**:
3. `# chown root wlauth`
4. `# chmod +xs wlauth`
5. Sobre plataformas PAM (Solaris y Linux), seleccionamos la configuración PAM para **wlauth**.
    - Solaris—Añadimos las siguientes líneas a nuestro fichero **/etc/pam.conf**:
    - `# 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`
    - Linux—Creamos un fichero llamado **/etc/pam.d/wlauth** que contenga lo siguiente:
    - `#%PAM-1.0`
    - `#`
    - `# File name:`

```
o     # /etc/pam.d/wlauth
o     #
o     # If you do not use shadow passwords, delete "shadow".
o     auth required /lib/security/pam_pwdb.so shadow
o     account required /lib/security/pam_pwdb.so
```

Para usar el reino de seguridad de **UNIX** en vez de usar el reino **File**, vamos al nodo **Security -> Realms** en el panel izquierdo de la Consola de Administración. En el panel derecho pulsamos sobre el enlace **Create a New UNIX Realm**.
Antes de poder usar el reino de seguridad **UNIX**, necesitamos activar el reino **Caching** e introducir el nombre de la clase del reino de seguridad **UNIX** en el campo **Basic Realm**.
Configurar el reino de Seguridad de **UNIX** implica configurar campos que definen un nombre para el reino y el ordenador sobre el que se está ejecutando el dominio UNIX. Para especificar un nombre de reino, debemos definir valores para los campos mostrados en la ventana **UNIX Realm Create** de la Consola de Administración. Para grabar nuestros cambios, pulsamos el botón **Apply**. Cuando hayamos terminado de definir los campos, reiniciamos el Servidor WebLogic. La siguiente tabla describe todos los campos de la ventana **UNIX Realm Configuration**:

Campo	Descripción
Name	El nombre del reino de Seguridad UNIX, como, `AccountingRealm`.
Realm Class Name	El nombre de la clase Java que implementa el reino de seguridad UNIX. La clase Java necesita estar en el CLASSPATH del Servidor WebLogic.
AuthProgram	El nombre del programa usado para autentificar usuarios en el reino de seguridad **UNIX**. En la mayoría de los casos, el nombre del programa es **wlauth**.

Si **wlauth** no está en el path de clases del Servidor WebLogic o si le hemos dado al programa un nombre distinto a **wlauth**, debemos añadir una propiedad en la línea de comandos Java cuando iniciemos el Servidor WebLogic. Editamos el script que usamos para arrancar el Servidor WebLogic y añadimos la siguiente opción después del comando `java`:

```
-Dweblogic.security.unixrealm.authProgram=wlauth_prog
```

Reemplazamos **wlauth_prog** con el nombre del programa **wlauth**, incluyendo el path completo si no está en el path de búsqueda. Arrancamos el Servidor WebLogic. Si el programa **wlauth** está en el path del Servidor WebLogic y se llama **wlauth**, este paso no es necesario.

Configurar un Reino de Seguridad RDBMS

El reino de seguridad **RDBMS** es un reino de seguridad personalizado proporcionado por BEA que almacena Usuarios, Grupos y ACLs en un base de datos relacional. Este reino puede ser manejada a través de la Consola de Administración.
Para usar el reino de seguridad **RDBMS** en lugar del reino **File**, vamos al nodo **Security -> Realms** en el panel izquierdo de la Consola de Administración. En el panel derecho, pulsamos sobre el enlace **Create a New RDBMS Realm**.
Antes de poder usar el reino de seguridad **RDBMS**, necesitamos activar el reino **Caching** e introducir el nombre de la clase del reino de seguridad **RDBMS** en el campo **Basic Realm**.
Configurar el reino de Seguridad **RDBMS** implica configurar campos que definen el driver JDBC que se va a utilizar para conectar con la base de datos y definir el esquema usado para almacenar Usuarios, Grupos y ACLs en la base de datos. Para definir estos campos, debemos definir valores para los campos mostrados en las tres pestañas de la ventana **RDBMS Realm Create**: la pestaña **General**, la pestaña **Database** y la pestaña **Schema**. La siguiente tabla describe todos los campos de la pestaña **General**:

Campo	Descripción
Name	El nombre del reino de Seguridad RDBMS, como, `AccountingRealm`.
Realm Class Name	El nombre de la clase WebLogic que implementa el reino de seguridad RDBMS. La clase Java necesita estar en el CLASSPATH del Servidor WebLogic.

La siguiente tabla describe todos los campos de la pestaña **Database**:

Campo	Descripción
Driver	El nombre completo del driver JDBC. Esta clase debe estar en el CLASSPATH del Servidor WebLogic.
URL	La URL para la base de datos que estamos usando con el reino RDBMS, según lo especifica la documentación de nuestro driver JDBC.
User Name	El nombre de usuario por defecto para la base de datos.
Password	La password del usuario por defecto de la base de datos.

Las propiedades **Schema** usadas para definir Usuarios, Grupos y ACLs almacenados en la base de datos se listan en la pestaña **Schema**. Cuando finalicemos de definir los valores para los campos necesarios en cada una de las tres pestañas, grabamos los cambios pulsando el botón **Apply**. Luego reiniciamos el Servidor WebLogic.

Configurar un Reino de Seguridad Personalizado

Podemos crear un reino de seguridad personalizado que provenga de un almacén existente de usuarios como un servidor de directorios en la red. Para usar un reino de seguridad personalizado, creamos una implementación de los interfaces `weblogic.security.acl.AbstractListableRealm` o `weblogic.security.acl.AbstractManageableRealm` y luego usamos la Consola de Administración para instalar nuestra implementación.
Para instalar un reino de seguridad personalizado, vamos al nodo **Security -> Realms** en el panel izquierdo de la Consola de Administración. En el panel derecho, pulsamos sobre el enlace **Create a New Custom Realm**.
Antes de poder usar el reino de seguridad **RDBMS**, necesitamos activar el reino **Caching** e introducir el nombre de la clase del reino de seguridad **RDBMS** en el campo **Basic Realm**.
Personalizar un reino de seguridad personalizado implica configurar campos que definen el nombre del reino y el interfase que lo implementa, y especificar información que define cómo se almacenará los Usuarios, los Grupos y los ACLs en el reino de seguridad personalizado. Para definir esta información, debemos definir valores para los campos de la ventana **Custom Realm Create** de la Consola de Administración. Para grabar nuestros cambios, pulsamos el botón **Apply**. Cuando hayamos terminado de definir los campos, reiniciamos el Servidor WebLogic. La siguiente tabla describe todos los campos de la ventana **Custom Realm Configuration**:

Campo	Descripción
Name	El nombre del reino de Seguridad Personalizado, como, `AccountingRealm`.
Realm Class Name	El nombre de la clase WebLogic que implementa el reino de seguridad Personalizado. La clase Java necesita estar en el CLASSPATH del Servidor WebLogic.
Configuration Data	La información necesaria para conectar con el almacén de seguridad.

Probar un Reino de Seguridad Alternativo o Personalizado

Si hemos arrancado WebLogic Server con un reino de seguridad alternativo o personalizado, realizamos los siguientes pasos para asegurarnos de que el reino está funcionando apropiadamente:

1. Arrancamos la Consola de Administración. La Consola muestra todos los Usuarios, Grupos y ACLs conocidos en el reino de seguridad.
2. Usamos la Consola de Administración para añadir un ACL para el servlet de ejemplo **HelloWorld**. Damos acceso a este servlet a un Usuario y a un Grupo de nuestro reino de seguridad. Seleccionamos un Grupo que no incluya el usuario especificado.
3. Reiniciamos WebLogic Server y cargamos el servlet **HelloWorld** con un ACL usando la siguiente URL:
4. `http://localhost:portnumber/helloWorld`

Intentamos introducir un nombre de usuario y una password para un Usuario que no esté incluido en el ACL que hemos añadido para el Servlet. Deberíamos obtener un mensaje diciendo que no estamos autorizados para hacer eso.
Intentamos introducir el nombre de usuario y la password para el Usuario que incluimos en el ACL, o como un usuario individual o como miembro del Grupo. El servlet se debería cargar y mostrar el mensaje "Hello World".

Migrar Reinos de Seguridad

WebLogic Server 6.0 proporciona una nueva arquitectura de control para reinos de seguridad. La arquitectura implementada a través de **MBeans** nos permite manejar los reinos de seguridad a través de la Consola de Administración. Si tenemos un reino de seguridad de una versión anterior de WebLogic Server, debemos usar la siguiente información para migrar a la nueva arquitectura:

- Si estamos usando los reinos de seguridad Windows NT, UNIX, o LDAP, usamos la opción **Convert WebLogic Properties** de la Consola de Administración para convertir el reino de seguridad a la nueva arquitectura.. Observaremos que podemos ver los Usuarios, Grupos y ACLs en la Consola de Administración, sin embargo, todavía necesitamos usar las herramientas proporcionadas por los distintos entornos para manejar los Usuarios y Grupos.
- Si estamos usando un reino de seguridad personalizado, seguimos los pasos de Instalar un Reino de Seguridad Personalizado para especificar información sobre los Usuarios, Grupos y opcionalmente ACLs que están almacenados en el reino de seguridad.
- El reino de seguridad **Delegating** ya no se soporta en WebLogic Server 6.0. Si lo estamos usando, tendremos que usar otro tipo de reino de seguridad para almacenar Usuarios, Grupos y ACLs.
- Si estamos usando el reino de seguridad **RDBMS**, usamos una de las siguientes opciones para convertirlo:
    - Si no cambiamos la fuente del reino de seguridad **RDBMS**, seguimos los pasos de Configurar un Reinado de Seguridad RDBMS para ejemplarizar una nueva clase para nuestro reino de seguridad RDBMS existente y definimos la información sobre el driver JDBC que se utiliza para conectar con la base de datos y el esquema utilizado. En este caso, estamos creando un **MBean** en WebLogic Server 6.0 para el reino de seguridad **RDBMS**.

- Si personalizamos el reino de seguridad RDBMS, convertimos nuestra fuente para usar los **MBeans**. Usamos el ejemplo de código del directorio `\samples\examples\security\rdbmsrealm` como guía para convertir nuestro reino RDBMS. Una vez hayamos convertido nuestro reino de seguridad RDBMS a MBeans, seguimos los pasos de Configurar un Reinado de Seguridad RDBMS para ejemplarizar una nueva clase para nuestro reino de seguridad RDBMS existente y definimos la información sobre el driver JDBC que se utiliza para conectar con la base de datos y el esquema utilizado.

Definir Usuarios

Nota:
*Esta sección explica cómo añadir Usuarios al reino **File**. Si estamos usando un reino alternativo, debemos usar las herramientas de control proporcionadas por ese reino para definir un Usuario.*

Los Usuarios son entidades que pueden ser autentificadas en un reino de seguridad del Servidor WebLogic. Un Usuario puede ser una persona o una entidad de software, como un cliente Java. A cada usuario se le dá una identidad única dentro de un reino de seguridad del Servidor WebLogic. Como administradores del sistema deberemos garantizar que no hay dos usuarios idénticos en el mismo reino de seguridad.
Definir usuarios en un reino de seguridad implica especificar un nombre único y una password para cada Usuario que tenga acceso a los recursos del Servidor WebLogic en la ventana **Users** de la Consola de Administración. La siguiente tabla describe los campos de la ventana **Users**:

Campo	Descripción
Name	El nombre de un Usuario, es decir, una entidad que accederá a los recursos del Servidor WebLogic. Los nombres son sensibles a las mayúsculas.
Password	La password para el Usuario. La password debe ser de 8 caracteres como mínimo y son sensibles a las mayúsculas.

El reino **File** tiene dos usuarios especiales, **system** y **guest**:

- El Usuario **system** es el usuario administrativo que controla las operaciones del Servidor WebLogic a nivel de sistema, como parar y arrancar servidores, y bloquear o desbloquear recursos. El Usuario **system** se define durante el procedimiento de instalación de WebLogic Server.
- WebLogic Server proporciona automáticamente el Usuario **guest**. Cuando no se requiere autorización alguna, el Servidor WebLogic asigna la identidad **guest** a un cliente dándole acceso a cualquier recurso que esté disponible para el usuario **guest**. Un cliente puede entrar como el Usuario **guest** como nombre de usuario y **guest** como password.

Los usuarios **system** y **guest** son como los demás usuarios en un reino de seguridad del Servidor WebLogic:

- Para acceder a recursos del Servidor WebLogic, deben tener los ACLs apropiados.
- Para ejecutar una operación sobre un recurso del Servidor WebLogic, deben proporcionar un nombre de usuario y una password (o un certificado digital).

Para mejorar la seguridad de nuestro despliegue del Servidor WebLogic, recomendamos deshabilitar el usuario **guest**. Para hacer esto, marcamos la opción **Guest Disabled** sobre la pestaña **General** en la ventana **Security** de la Consola de Administración. Cuando desactivamos al usuario **guest**, no se borra, sólo se convierte en inaccesible para que nadie pueda entrar con ese nombre de usuario.
Para borrar Usuarios, introducimos el nombre del usuario en la caja de lista **Remove These Users** y pulsamos **Remove**.

Definir Grupos

Nota:
*Esta sección explica cómo añadir Grupos al reino **File**. Si estamos usando un reino alternativo, debemos usar las herramientas de control proporcionadas por ese reino para definir un Grupo.*

Un Grupo representa un conjunto de Usuarios que normalmente hacen algo en común, como trabajar en el mismo departamento de la compañía. Los Grupos se usan principalmente para manejar un número de Usuarios de una forma eficiente. Cuando a un Grupo se le concede algún permiso en un ACL, todos los miembros de ese grupo reciben el mismo permiso.
Podemos registrar un Grupo con el reino de seguridad del Servidor WebLogic realizando los siguientes pasos:

1. Seleccionar el campo **Name** en la ventana **Groups** de la Consola de Administración.
2. Pulsar el botón **Create**.
3. Introducir los Usuarios en el campo **Add User**.
4. Pulsar el botón **Update Group** cuando hayamos terminado de añadir usuarios.

El reino **File** tiene uno Grupo interno: **everyone**. Todos los usuarios definidos en el reino de seguridad definido automáticamente son miembros del Grupo **everyone**.
Para borrar Grupos, introducimos el nombre del Grupo en la caja de listas **Remove These Groups** y pulsamos **Remove**.

Definir un Grupo para un Host Virtual

En el Servidor WebLogic, los host virtuales que requieren autentificación están representados en un reino de seguridad como un Grupo. Todos los usuarios de un host virtual primero se definen como usuarios del reino de seguridad para un Servidor WebLogic particular y luego son definidos como miembros del grupo que representa el host virtual.

Definir ACLs

Los Usuarios acceden a recursos en un reino de seguridad del Servidor WebLogic. Si un usuario puede o no acceder a un recurso lo determina la lista de control de acceso (ACL) para ese recurso. Una ACL define los permisos por los que un usuario puede interactuar con el recurso. Para definir ACLs, creamos una ACL para un recurso, especificamos los permisos para el recurso y luego concedemos los permisos a un conjunto especificado de Usuarios y Grupos.

Todos los recursos de un Servidor WebLogic tienen uno o más permisos que pueden concederse. La siguiente tabla suma las funciones de varios recursos del Servidor WebLogic para los que se pueden restringir permisos con un ACL:

Para este recurso de WebLogic Server...	**Este ACL...**	**Concede permisos para estas funciones...**
WebLogic Servers	`weblogic.server` `weblogic.server.servername`	boot
Command-line Administration Tools	`weblogic.admin`	shutdown, lockserver, unlockserver
WebLogic Events	`weblogic.servlet.topicName`	send receive
WebLogic servlets	`weblogic.servlet.servletName`	execute
WebLogic JDBC connection pools	`weblogic.jdbc.connectionPool.poolname`	reserve reset shrink
WebLogic Passwords	`weblogic.passwordpolicy`	unlockuser
WebLogic JMS destinations	`weblogic.jms.topic.topicName` `weblogic.jms.queue.queueName`	send, receive
WebLogic JNDI contexts	`weblogic.jndi.path`	lookup modify list

Para crear ACLs para un recurso del Servidor WebLogic, abrimos la Consola de Administración y realizamos los siguientes pasos:

1. Especificamos el nombre del recurso que queremos proteger con un ACL.

Por ejemplo, creamos un ACL para un almacén de conexiones JDBC llamado **demopool**.

2. Especificamos un permiso para el recurso.

Podemos crear ACLs separados por cada permiso disponible para un recurso o un ACL que conceda todos los permisos para un recurso. Por ejemplo, podremos crear tres ACLS para el almacén de conexiones JDBC, **demopool**: uno con el permiso `reserve` otro con el permiso `reset`, y otro con el permiso `shrink`. O podemos crear un ACL con los permisos `reserve`, `reset`, y `shrink`.

3. Especificamos Usuarios o Grupos que tienen los permisos especificados para el recurso.

Cuando se crean ACLs para recursos de un Servidor WebLogic, necesitamos usar la sintaxis de la tabla mostrada en Definir ACLs. Por ejemplo, el almacén de conexiones JDBC llamado **demopool** se especificaría como `weblogic.jdbc.connectionPool.demopool`.
Antes de poder reiniciar un Servidor WebLogic, necesitamos dar permiso para hacerlo a un conjunto de usuarios. Esta seguridad evita que usuarios no autorizados reinicien el Servidor WebLogic.

Configurar el Protocolo SSL

El protocolo Secure Sockets Layer (SSL) proporciona conexiones seguras permitiendo que dos aplicaciones que se conectan a través de la red se autentifiquen la una a la otra y encriptando los datos intercambiados entre las aplicaciones. El protocolo SSL proporciona autentificación de servidor y opcionalmente del cliente, confidencialidad e integridad de datos.
Para configurar el protocolo SSL, realizamos los siguientes pasos:

1. Obtenemos una clave privada y un certificado digital para el Servidor WebLogic. Necesitamos un certificado digital y una clave privada por cada Servidor WebLogic que use el protocolo SSL.
2. Almacenamos la clave privada y el certificado digital para el Servidor WebLogic.
3. A través de la Consola de Administración, configuramos los campos del protocolo SSL y la clave privada, el certificado digital, y las autoridades de certificación creíbles para el Servidor WebLogic. Estos campos están definidos de una forma básica; debemos definirlos para cualquier Servidor WebLogic que use el protocolo SSL.

Solicitar la Clave Privada y el Certificado Digital

Para adquirir un certificado digital desde una autoridad de certificación, necesitamos enviar nuestra petición en un formato particular llamado "Certificate Signature Request" (CSR). WebLogic Server incluye un servlet **Certificate Request Generator** que crea un CSR. Este servlet recolecta información sobre nosotros y genera un fichero de clave privada y un fichero de petición de certificado. Luego podemos enviar el CSR a una autoridad de certificación como **VeriSign** o **Entrust.net**. Antes de poder usar el servlet **Certificate Request Generator**, el Servidor WebLogic debe estar instalado y en ejecución.
Para generar un CSR, realizamos los siguientes pasos:

1. Arrancar el servlet Certificate Request Generator. El fichero `.war` para el servlet está localizado en el directorio `\wlserver6.0\config\mydomain\applications`. El fichero `.war` se instala automáticamente cuando arrancamos el Servidor WebLogic.
2. En un Navegador Web, introducimos la URL del servlet Certificate Request Generator de esta forma:
3. `https://hostname:port/Certificate`

Los componentes de esta URL se definen de esta forma:

- `hostname` es el nombre DNS de la máquina que está ejecutando el Servidor WebLogic.
- `port` es el número de puerto por el que el Servidor WebLogic escucha conexiones SSL. El valor por defecto es 7002.

Por ejemplo, si el Servidor WebLogic se está ejecutando sobre una máquina llamada **ogre** y está configurado para escuchar comunicaciones SSL en el puerto por defecto, para ejecutar el servlet Certificate Request Generator, debemos introducir la siguiente URL en nuestro navegador Web:

```
https://ogre:7002/certificate
```

4. El servlet Certificate Request Generator carga un formulario en nuestro navegador Web.

Completamos el formulario mostrado en el navegador, usando la información de la siguiente tabla:

Campo	Descripción
Country code	El código ISO de dos letras de nuestro país. El código para los Estados Unidos es US.
Organizational unit name	El nombre de nuestra división, departamento, u otra unidad operacional de nuestra organización.
Organization name	El nombre de nuestra organización. La autoridad de certificación podría requerir que cualquier nombre de host introducido en este campo pertenezca a un dominio registrado en esta organización.
E-mail address	La dirección e-mail del administrador. El certificado digital es enviado a esta dirección de correo.
Full host name	El nombre totalmente cualificado del Servidor WebLogic en el que será instalado el Certificado Digital. Este nombre es el usado para las búsquedas DNS del Servidor WebLogic, por ejemplo, **node.mydomain.com**. Los navegadores Web comparan el nombre de Host de la URL con el nombre del certificado digital. Si posteriormente cambiamos el nombre del host, debemos solicitar un nuevo certificado digital.
Locality name (city)	El nombre de nuestra ciudad. Si operamos con un licencia concedida por una ciudad, este campo es obligatorio; debemos introducir el nombre de la ciudad que nos ha concedido la licencia.
State name	El nombre del Estado o Provincia en el que opera nuestra organización si nuestra organización está en Estados Unidos o Canadá, respectivamente. No debemos abreviar.
Private Key Password	La password usada para encriptar la clave privada. Introducimos una password en este campo si queremos usar una clave protegida con el Servidor WebLogic. Si elegimos usar una clave protegida, se nos pedirá esta password siempre que se use la clave. Si especificamos una password, obtenemos una clave privada encriptada PKCS-8. BEA recomienda el uso de una password para proteger las claves privadas. Si no queremos usar un clave protegida, dejamos este campo en blanco. Para usar claves privadas protegidas, activamos el campo **Use Encrytped Keys** sobre la pestaña **SSL** de la ventana **Server** en la Consola de Administración.
RandomString	Un string de caracteres a utilizar para el algoritmo de encriptación. No tenemos que recordar este string en el futuro. Añade un factor externo al algoritmo de encriptación, haciendo más difícil que alguien rompa la encriptación. Por esta razón, introducimos un string que no vaya a ser recordado. BEA recomienda utilizar un string largo con una buena mezcla de letras mayúsculas y minúsculas, dígitos, espacios y signos de puntuación; estas cadenas largas y mezcladas contribuyen a una encriptación más segura.
Strength	La longitud (en bits) de las claves a generar. Cuando más larga sea la clave, más difícil será romper la encriptación. Si tenemos una versión doméstica del Servidor WebLogic, podemos elegir entre claves de 512-, 768-, o 1024-bits. Recomendamos la clave de 1024-bits.

5. Pulsamos el botón **Generate Request**.

El servlet Certificate Request Generator muestra un mensaje informándonos de cualquier campo que esté vacío o si cualquiera ellos contiene valores nulos. Pulsamos el botón **Back** en nuestro navegador y corregimos los errores.
Cuando todos los campos hayan sido aceptados, el servlet Certificate Request Generator genera los siguientes ficheros en el directorio de arrancada de nuestro Servidor WebLogic:

   - `www_mydomain_com-key.der`— El fichero de la clave privada. El nombre de este fichero debería ir dentro del campo **Server Key File Name** de la pestaña **SSL** en la Consola de Administración.
   - `www_mydomain_com-request.dem`— El fichero de petición de certificado, en formato binario.
   - `www_mydomain_com-request.pem`— El fichero CSR que enviamos a la autoridad de certificación. Contiene los mismos datos que el fichero `.dem` pero codificado en ASCII por lo que podemos copiarlo dentro del correo o pegarlo dentro de un formulario Web.
6. Seleccionamos una autoridad de certificación y seguimos las instrucciones de la página web de esa autoridad para comprar un certificado digital.
   - **VeriSign, Inc.** ofrece dos opciones para WebLogic Server: Global Site Services con características de encriptación fuerte de 128-bits para navegadores Web y export, y Secure Site Services, que ofrece encriptación de 128-btis para navegadores web domésticos y encriptación de 40-btis para navegadores web de exportación.
   - **Entrust.net digital certificates** ofrece encriptación de encriptación de 128-btis para navegadores web domésticos y encriptación de 40-btis para navegadores web de exportación.
7. Cuando se nos pida que seleccionemos el tipo de servidor, elegimos **BEA WebLogic Server** para asegurarnos de que recibimos un certificado digital que sea compatible con el Servidor WebLogic.
8. Cuando recibamos nuestro certificado digital de la autoridad de certificación, necesitamos almacenarlo en el directorio **\wlserver6.0\config\mydomain**.

Nota:
*Si obtenemos un fichero de clave privada de una fuente distinta al servlet Certificate Request Generator, debemos verificar que el fichero está en formato **PKCS#5/PKCS#8 PEM.***

9. Configuramos el Servidor WebLogic para usar el protocolo SSL, necesitamos introducir la siguiente información en la pestaña **SSL** de la ventana **Server Configuration**:
   - En el campo **Server Certificate File Name**, introducimos la localización completa del directorio y el nombre del certificado digital para el Servidor WebLogic.
   - En el campo **Trusted CA File Name**, introducimos la localización completa del directorio y el nombre del certificado digital de la autoridad de certificación que firmó el certificado digital del Servidor WebLogic.
   - En el campo **Server Key File Name**, introducimos la localización completa del directorio y el nombre del fichero de clave privada para el Servidor WebLogic.
10. Usamos la siguiente opción de la línea de comandos para arrancar el Servidor WebLogic:

11. `-Dweblogic.management.pkpassword=password`

donde `password` es la password definida cuando se solicitó el certificado digital.

Almacenar las Claves Privadas y los Certificados Digitales

Una vez que tenemos una clave privada y un certificado digital, copiamos el fichero de la clave privada generado por el servlet Certificate Request Generator y el certificado digital recibido de la autoridad de certificación en el directorio `\wlserver6.0\config\mydomain`.
Los ficheros de claves privadas y certificados digitales están generados en los formatos **PEM** o **Definite Encoding Rules** (DER). La extensión del fichero identifica el formato del fichero del certificado digital.
Un fichero de clave privada PEM(.pem) empieza y termina con las siguientes líneas respectivamente:

```
-----BEGIN ENCRYPTED PRIVATE KEY-----

-----END ENCRYPTED PRIVATE KEY-----
```

Un certificado digital PEM(.pem) empieza y termina con las siguientes líneas, respectivamente:

```
-----BEGIN CERTIFICATE-----

-----END CERTIFICATE-----
```

Nota:
Nuestro certificado digital podría ser uno de los varios certificados digitales en en el fichero, cada uno limitado por las líneas `BEGIN CERTIFICATE` y `END CERTIFICATE`. Normalmente el certificado digital para un Servidor WebLogic está en un fichero, con una extensión `.pem` o `.der`, y la cadena de certificados del servidor en otro fichero. Se usan dos ficheros porque diferentes Servidores WebLogic podrían usar la misma cadena de certificados.
El primer certificado digital en el fichero de la autoridad de certificación es el primer certificado digital en la cadena de certificados del Servidor WebLogic. Los siguientes certificados del fichero son los siguientes de la cadena. El último certificado de la cadena es un certificado digital auto-firmado que termina la cadena de certificados.

Un fichero de formato DER(.der) contiene datos binarios. El Servidor WebLogic requiere que la extensión del fichero corresponda con el contenido del fichero de certificado por ese debemos asegurarnos de grabar el fichero que recibimos de nuestra autoridad de certificación con la extensión de fichero correcta.
Asignamos protecciones al fichero de la clave privada y a los certificados digitales para que sólo el usuario **system** del Servidor WebLogic tenga privilegios de lectura y ningún otro usuario tenga privilegios de acceso al fichero de clave primaria o al certificado digital. Si estamos creando un fichero con certificados digitales de varias autoridades de certificación o un fichero que contiene una cadena de certificados, debemos usar el formato PEM. El Servidor WebLogic proporciona una herramienta para convertir ficheros en formato DER a formato PEM, y vise-versa.

Definir Autoridades de Certificación

Cuando establecemos una conexión SSL, el Servidor WebLogic chequea la identidad de la autoridad de certificación contra una lista de autoridades de certificación creíbles para asegurarse de que la autoridad que se está utilizando actualmente es verdadera.
Copiamos el certificado raíz de la autoridad de certificación en el directorio `\wlserver6.0\config\mydomain` de nuestro Servidor WebLogic y seleccionamos los campos descritos en Definir Campos para el Protocolo SSL.
Si queremos usar una cadena de certificados, añadimos el certificado digital codificado en PEM al certificado digital de la autoridad de certificación que envía el certificado digital para el Servidor WebLogic. El último certificado digital del fichero debería ser un certificado digital auto-firmado (es decir, el certificado **rootCA**).
Si queremos usar autentificación mutua, tomamos los certificados raíces de las autoridades de autentificación que queremos aceptar y los incluimos en el fichero CA creíble (trusted CA).

Definir Campos para SSL

Para definir campos para el protocolo SSL, realizamos los siguientes pasos:

1. Abrimos la Consola de Administración.
2. Abrimos la ventana **Administration**.
3. Seleccionamos la pestaña **SSL**. Definimos los campos de esta pestaña introduciendo valores y marcando los chekboxes necesarios. (Para más detalles, ver la siguiente tabla).
4. Pulsamos el botón **Apply** para grabar los cambios.
5. Reiniciamos el Servidor WebLogic.

La siguiente tabla describe todos los campos de la pestaña **SSL** de la ventana **Server Configuration**:

Nota:
Recuerda que si estamos usando una clave privada protegida PKCS-8, necesitamos especificar la password para la clave privada en la línea de comandos cuando arrancamos el Servidor WebLogic.

Campo	Descripción
Enabled	Checkbox que activa el uso del protocolo SSL. Por defecto, este campo está activado.
SSL Listen Port	El número de puerto dedicado en el que el Servidor WebLogic escucha conexiones SSL. Por defecto es 7002.
Server Key File Name	El directorio de la localización completa y el nombre del fichero de clave privada del Servidor WebLogic. La extensión del fichero (.DER o .PEM) indica el método que debería usar el Servidor WebLogic para leer los contenidos del fichero.
Server Certificate File Name	El directorio de la localización completa y el nombre del fichero del certificado digital del Servidor WebLogic. La extensión del fichero (.DER o .PEM) indica el método que debería usar el Servidor WebLogic para leer los contenidos del fichero.
Server Certificate Chain File Name	El directorio de la localización completa y el nombre del resto de los certificados digitales del Servidor WebLogic. La extensión del fichero (.DER o .PEM) indica el método que debería usar el Servidor WebLogic para leer los contenidos del fichero.
Client Certificate Enforced	Checkbox que activa la autentificación mutua.
Trusted CA File Name	El nombre del fichero que contiene el certificado digital de la autoridad(es) de certificación creíbles por el Servidor WebLogic. Este fichero puede contener un sólo certificado digital o varios de ellos. La extensión del fichero (.DER o .PEM) le dice al Servidor WebLogic cómo leer los contenidos del fichero.
CertAuthenticator	El nombre de la clase Java que implementa el interfase `CertAuthenticator`.
Use Java	Checkbox que activa el uso de librerías nativas de Java. WebLogic Server proporciona una implementación pura-Java del protocolo SSL: las librerías nativas de Java mejoran el rendimiento de las operaciones SSL sobre plataformas Solaris, Windows NT, IBM AIX. Por defecto, este campo no está activado.
Use Encrypted Keys	Campo que especifica que la clave privada del Servidor WebLogic ha sido encriptada con un password. Por defecto es **false**.
Handler Enabled	Campo que especifica si el Servidor WebLogic rechaza o no conexiones SSL que fallan en la autentificación del cliente por una de las siguientes razones: • El certificado digital del cliente solicitado no está completo. • El cliente no envió un certificado digital. • El certificado digital del cliente no fue enviado por una

	autoridad de certificación especificada en el campo **Trusted CA Filename**.
	Por defecto, el **SSL Handler** permite a un Servidor WebLogic hacer conexiones SSL salientes a otro Servidor WebLogic. Por ejemplo un EJB en un Servidor WebLogic podría abrir un stream HTTPS sobre otro Servidor Web. Con el campo **Handler Enabled** activado, el Servidor WebLogic actúa como un cliente en una conexión SSL. Por defecto este campo está activado. Sólo debemos desactivar este campo si queremos proporcionar nuestra propia implementación de conexiones SSL salientes. **Nota:** El **SSL Handler** no tiene efecto en la habilidad del Servidor WebLogic para manejar conexiones SSL entrantes.
Export Key Lifespan	El número de veces que el Servidor WebLogic usa una clave exportable entre un servidor doméstico y un cliente exportable antes de generar una nueva. Cuando más seguro queramos que sea el Servidor WebLogic menor número de veces se debería usar la clave antes de generar una nueva. El valor por defecto es usarla 500 veces.
Login Timeout Millis	El número de milisegundos que el Servidor WebLogic debería esperar una conexión SSL antes de marcar un error de time-out. El valor por defecto es 25000 milisegundos. Las conexiones SSL tardan mucho más tiempo en negociarse que las conexiones normales. Si los clientes se conectan a través de Internet, debemos subir el valor por defecto para acomodarnos a la latencia adicional de la red.
Certificate Cache Size	El número de certificados digitales que son divididos y almacenados por el Servidor WebLogic. El valor por defecto es 3.

Configurar la Autentificación Mutua

Cuando el Servidor WebLogic está configurado para autentificación mutua, se requiere que los clientes presenten sus certificados digitales al Servidor WebLogic que valida los certificados digitales contra una lista de autoridades de certificación creíbles.
Para configurar nuestro Servidor WebLogic para el protocolo SSL y la autentificación de certificados, debemos completar los procedimientos de Configurar el Protocolo SSL.
Copiamos los certificados raíces de las autoridades de certificación a utilizar por el Servidor WebLogic en el directorio `\wlserver6.0\config\mydomain`. Durante la autentificación mutua, se requiere que los clientes presenten sus certificados digitales enviados por una de estas autoridades de certificación creíbles.
Para configurar la autentificación mutua, pulsamos la opción **Client Certificate Enforced** sobre la pestaña **SSL** en la ventana **Server Configuration** de la Consola de Administración. Por defecto, esta opción no está activada.

Configurar RMI sobre IIOP sobre SSL

El protocolo SSL puede usarse para proteger conexiones IIOP sobre objetos remotos RMI. El protocolo SSL asegura las conexiones mediante autentificación y encriptación de los datos intercambiados entre objetos. Para usar el protocolo SSL para proteger IIOP sobre conexiones RMI, hacemos los siguiente:

1. Configuramos el Servidor WebLogic para usar el protocolo SSL. Puedes encontrar más información en Configurar el Protocolo SSL.
2. Configuramos el cliente Object Request Broker (ORB) para usar el protocolo SSL. Nos debemos referir a la documentación de nuestro cliente ORB para buscar la información sobre cómo configurar el protocolo SSL.
3. Usar la utilidad **host2ior** para imprimir el WebLogic Server IOR en la consola. Esta utilidad imprime dos versiones del IOR, una para conexiones SSL y otra para conexiones no SSL. La cabecera del IOR específica si puede usarse o no en conexiones SSL.
4. Usar el IOR SSL cuando obtengamos la referencia inicial al servicio **CosNaming** que accede al árbol JNDI del Servidor WebLogic.

Proteger Passwords

Es importante proteger las password que estamos usando para acceder a los recursos de un Servidor WebLogic. En el pasado, los nombres de usuarios y las passwords se almacenaban en texto normal en un reino de seguridad. Ahora el Servidor WebLogic emborrona todas las passwords. Cuando el Servidor WebLogic recibe una petición de un cliente, la password presentada por el siente se emborrona y el Servidor WebLogic la compara con la password ya emborronada que tiene almacenada.
Todo fichero **filerealm.properties** tiene asociado un fichero **SerializedSystemIni.dat** que se usa para emborronar las passwords. Durante la instalación el fichero **SerializedSystemIni.dat** se pone en el directorio `\wlserver6.0\config\mydomain`. Si por cualquier razón este fichero se corrompe o se destruye, debemos reconfigurar el Servidor WebLogic.
Recomendamos seguir estos pasos:

- Hacer una copia de seguridad del fichero **SerializedSystemIni.dat** y ponerla en el mismo sitio que la copia del fichero **filerealm.properties** asociado.
- Configurar los permisos del fichero **SerializedSystemIni.dat** para que el administrador del Servidor WebLogic tenga privilegios de lectura y escritura y los otros usuarios no tengan ningún privilegio.

Si ya tenemos un fichero **weblogic.properties** y queremos emborronar las passwords en el fichero, usamos la opción **Convert weblogic.properties** de la ventana principal de la Consola de Administración para convertir el fichero **weblogic.properties** en un fichero **config.xml**. Una vez convertido este fichero, todas las passwords existentes están protegidas.
La adivinación de passwords es un tipo común de ataque de seguridad. En este tipo de ataque, un hacker intenta entrar en un ordenador usando varias combinaciones de nombres de usuario y password. El Servidor WebLogic refuerza su protección contra este tipo de ataques proporcionando un conjunto de campos diseñados para proteger passwords.
Para proteger passwords en nuestro Servidor WebLogic, debemos realizar los siguientes pasos:

1. Abrir la Consola de Administración.
2. Abrir la ventana **Security Configuration**.
3. Seleccionar la pestaña **Passwords**. Definimos los campos deseados en esta pestaña introduciendo valores en los prompts apropiados y marcando los checkboxes necesarios (para más detalles ver la siguiente tabla.)
4. Pulsamos el botón **Apply** para grabar nuestras elecciones.

5. Reiniciamos el Servidor WebLogic.

La siguiente tabla describe todos los campos de la pestaña **Passwords** de la ventana **Security Configuration**:

Campo	Descripción
Minimum Password Length	Número de caracteres requeridos en una password. Las passwords deben ser de como mínimo 8 caracteres. El valor por defecto es 8.
Lockout Enabled	Checkbox que solicita el bloqueo de una cuenta de usuario cuando se hace un intento sin éxito de entrar en esa cuenta. Por defecto este campo está activado.
Lockout Threshold	El número de introducciones de passwords falladas que un usuario puede intentar antes de que la cuenta sea bloqueada. Cualquier intento subsecuente de acceder a la cuenta (incluso si la combinación nombre de usuario/password es correcta) lanza una excepción de seguridad; la cuenta permanece bloqueada hasta que la desbloquee explícitamente el administrador del sistema o se intente otro login después de que termine el periodo de duración del bloqueo. Observamos que los intentos de login fallidos deben hacerse dentro del tiempo definido en el campo **Lockout Reset Duration**. El valor por defecto es 5.
Lockout Duration	El número de minutos que una cuenta de usuario permanece bloqueada en respuesta a varios intentos de login fallidos dentro de una cantidad de tiempo especificada por el campo **Lockout Reset Duration**. El valor por defecto es 30 minutos. Para desbloquear una cuenta de usuario, necesitamos tener el permiso `unlockuser` para la política `weblogic.password`.
Lockout Reset Duration	El número de minutos dentro del cual debe ocurrir un intento de login fallido para bloquear la cuenta de usuario. Una cuenta se bloquea si se suceden el número intentos de loggins fallidos definidos en el campo **Lockout Threshold** dentro del tiempo definido en este campo. Por ejemplo, si el valor de este campo es cinco minutos y se hacen tres intentos de login fallidos dentro de un intervalo de seis minutos, la cuenta no se bloqueará. Si hacen cinco intentos fallidos dentro del periodo de cinco minutos, la cuenta será bloqueada. El valor por defecto es 5 minutos.
Lockout Cache Size	Especifica el tamaño de caché de intentos de logín sin utilizar y fallidos. El valor por defecto es 5.

Instalar un Proveedor de Auditoria

WebLogic Server nos permite crear un proveedor de auditoria para recibir y procesar notificaciones de eventos de seguridad, como peticiones de autentificación, fallidas o con éxito, intentos de autorización, y recepción de certificados digitales no válidos

Para usar un proveedor de auditoria, creamos una implementación del interfase `weblogic.security.audit.AuditProvider`. Luego usamos la Consola de Administración para instalar y activar nuestra implementación.

Para instalar un proveedor de auditoria, introducimos el nombre de nuestra implementación de la clase `AuditProvider` en el campo **Audit Provider Class** de la ventana **Security Configuration**. Reiniciamos el Servidor WebLogic.

Podemos encontrar un ejemplo de creación de un proveedor de auditoria `LogAuditProvider` en el directorio `\samples\examples\security` de nuestra instalación del Servidor WebLogic.

Instalar un Filtro de Conexión

Podemos crear filtros de conexiones que nos permitan rechazar o aceptar conexiones de clientes basándose en el origen y el protocolo del cliente. Después de que el cliente se conecte, y antes de que se realice ningún trabajo en su cuenta, el Servidor WebLogic pasa el número IP del cliente y el puerto, el protocolo (HTTP, HTTPS, T3, T3S, o IIOP), y el número de puerto del Servidor WebLogic al filtro de conexión. Examinando esta información, podemos elegir permitir la conexión o lanzar una `FilterException` para terminarla.
Para usar un filtro de conexión, primero debemos crear una implementación del interfase `weblogic.security.net.ConnectionFilter`. Luego usamos la Consola de Administración para instalar nuestra implementación.
Para instalar un filtro de conexión, introducimos el nombre de nuestra implementación del interfase `weblogic.security.net.ConnectionFilter`, en el campo **Connection Filter** de la pestaña **General** de la ventana **Security Configuration** de la Consola de Administración y reiniciamos el Servidor WebLogic.
Podemos encontrar un ejemplo de creación de un filtro de conexión `SimpleConnectionFilter` en el directorio `\samples\examples\security` de nuestra instalación del Servidor WebLogic.

Configurar la Propagación del Contexto de Seguridad

La propagación del contexto de seguridad permite a las aplicaciones que se están ejecutando en un entorno WebLogic Server acceder a objetos y operaciones en dominios BEA WebLogic Enterprise (WLE). El componente BEA WebLogic Enterprise Connectivity (WLEC) de un Servidor WebLogic proporciona la capacidad de propagación del contexto de seguridad.
Cuando se usa la propagación del contexto de seguridad, la identidad de seguridad de un Usuario definido en un reino de seguridad WebLogic Server se propaga como parte del contexto de servicio de una petición Internet Inter-ORB Protocol (IIOP) enviada al dominio WLE sobre una conexión de red que es parte del almacén de conexiones WLEC. Toda conexión de red del almacén de conexiones WLEC ha sido autentificada usando una identidad de Usuario definida.
Para usar la propagación del contexto de seguridad, creamos un almacén de conexiones WLEC por cada dominio WLE al que queramos acceder desde el Servidor WebLogic. WebLogic Server rellena cada almacén de conexiones WLEC con conexiones IIOP. Las aplicaciones Java en un entorno WebLogic Server obtienen conexiones desde un almacén de conexiones WLEC y usa esas conexiones para llamar a objetos e invocar operaciones en los dominios WLE.
Antes de usar la propagación del contexto de seguridad, añadimos `WLE_HOME/lib/wleorb.jar` y `WLE_HOME/lib/wlepool.jar` a la variable **CLASSPATH** en los ficheros **startAdminWebLogic.sh** o **startAdminWebLogic.cmd**.

Aquí están los pasos para la implementación de la propagación del contexto de seguridad:

1. Creamos un nuevo almacén de conexiones WLEC para el propósito de la propagación del contexto de seguridad. Para crearlo vamos al nodo **Services --> WLEC** en el panel izquierdo de la Consola de Administración. En el panel derecho, pulsamos el enlace `Create a new WLEC Connection Pool` y definimos los campos descritos en la siguiente tabla:

Campo	Descripción
Name	El nombre del almacén de conexiones WLEC. El nombre debe ser único para cada almacén de conexiones WLEC.

Primary Addresses	Una lista de direcciones para **Listener/Handlers IIOP** que pueden usarse para establecer conexiones entre el almacén de conexiones y el dominio WLE. El formato de cada dirección es `//hostname:port`.
	Las direcciones deben corresponder con las direcciones ISL definidas en el fichero **UBBCONFIG**. Las distintas direcciones están separadas por puntos y coma. Por ejemplo: `//main1.com:1024; //main2.com:1044`.
	Para configurar el almacén de conexiones WLEC usando el protocolo SSL, usamos el prefijo `corbalocs` con las direcciones IIOP. Por ejemplo: `corbalocs://hostname:port`.

Failover Addresses	Una lista de direcciones de `Listener/Handlers IIOP` que se usan si las conexiones no se pueden establecer con las direcciones definidas en el campo **Primary Addresses**. Múltiples direcciones irán separadas por puntos y coma. Este campo es opcional.
Domain	El nombre del dominio WLE al que se conecta este almacén de conexiones WLEC. Sólo podemos tener un almacén de conexiones WLEC por cada dominio WLE. El nombre de domino debe corresponder con el parámetro **domainid** de la sección **RESOURCES** del fichero **UBBCONFIG** del dominio WLE.
Minimum Pool Size	El número de conexiones IIOP a añadir al almacén de conexiones WLEC cuando arranca el Servidor WebLogic. El valor por defecto es 1.
Maximum Pool Size	El número máximo de conexiones IIOP que se pueden hacer desde el almacén de conexiones WLEC. El valor por defecto es 1.

2. Pulsamos el botón **Create**.
3. Propagamos el contexto de seguridad para un Usuario en un reino de seguridad de un Servidor WebLogic para un dominio WLE. Para hacer esto, definimos los campos de la pestaña **Security** en la ventana **Connection Pool Configuration**.

La siguiente tabla muestra estos campos:

Campo	Descripción
User Name	Un nombre de usuario WLE. Este campo sólo se requiere cuando el nivel de seguridad del dominio WLE es `USER_AUTH`, `ACL` o `MANDATORY_ACL`.
User Password	La password del usuario definido en el campo **User Name**. Este campo es requerido sólo cuando definimos el campo **User Name**.
User Role	El role del usuario WLE. Este campo sólo es necesario cuando el nivel de seguridad en el dominio WLE es `APP_PW`, `USER_AUTH`, `ACL`, o `MANDATORY_ACL`.
Application Password	La password de la aplicación WLE. Este campo sólo es necesario cuando el nivel de seguridad en el dominio WLE es `APP_PW`, `USER_AUTH`, `ACL`, o `MANDATORY_ACL`.
Minimum Encryption Level	El nivel de encriptación SSL mínimo usado entre el dominio WLE y el Servidor WebLogic. Los valores posibles son 0, 40, 56, y 128. El valor por defecto es 40. Cero (0) indica que los datos están firmados pero no sellados. 40, 56, y 128 especifican la longitud, en bits, de la clave de encriptación. Si no se alcanza este valor mínimo de encriptación, la conexión SSL entre el WLE y el Servidor WebLogic fallará.
Maximum Encryption Level	El nivel máximo de encriptación usado entre el dominio WLE y el Servidor WebLogic. Los posibles valores son 0, 40, 56, y 128. El valor por defecto es el máximo nivel permitido por el kit de licencia del paquete de encriptación.
Enable Certificate Authentication	Checkbox que activa el uso de certificado de autentificación. Por defecto el certificado de autentificación está desactivado.
Enable Security Context	Marcamos este checkbox para pasar el contexto de seguridad del usuario del Servidor WebLogic pasado al dominio WLE. Por defecto, el contexto de seguridad está desactivado.

4. Para grabar nuestros cambios, pulsamos el botón **Apply** y reiniciamos el Servidor WebLogic.
5. Ejecutamos el comando `tpusradd` para definir el Usuario del Servidor WebLogic como un Usuario autorizado en el dominio WebLogic Enterprise.
6. Seleccionamos la opción `-E` del comando `ISL` para configurar el **Listener/Handler IIOP** para detectar y utilizar el contexto de seguridad propagado desde el reino de Servidor WebLogic. Esta opción requiere que especifiquemos un nombre principal. Este nombre define el principal usado por el almacén de conexiones WLEC para entrar en el dominio WebLogic Enterprise. El nombre principal debería corresponder con el nombre definido en el campo **User Name** cuando se creó el almacén de conexiones WLEC.

Usar certificados de autentificación entre el entorno WebLogic Server y el entorno WebLogic Enterprise implica realizar una nueva negociación SSL cuando se establece la conexión desde el entorno WebLogic Server a un objeto CORBA, RMI, o un EJB en un entorno WebLogic Enterprise. Para soportar peticiones de varios clientes sobre la misma conexión SSL, debemos configurar el certificado de autentificación para que opere de la siguiente forma:

1. Obtener un certificado digital del principal y poner la clave privada en el directorio **TUXDIR/udataobj/security/keys** del WebLogic Enterprise.
2. Usar el comando `tpusradd` para definir el principal como un usuario de WebLogic Enterprise.

3. Definir el Listener/Handler IIOP en el fichero **UBBCONFIG** con la opción `-E` para indicar que se va a utilizar el principal para autentificación.
4. Definir el nombre principal en el campo **User Name** cuando creamos un almacén de conexiones WLEC en la Consola de Administración del Servidor WebLogic.
5. Obtener un certificado digital para el Listener/Handler IIOP.
6. Especificar el certificado digital en la opción **SEC_PRINCIPAL_NAME** del comando ISL y usamos la opción `-S` para indicar que se debería usar un puerto seguro para la comunicación entre el dominio WebLogic Enterprise y el reino de seguridad del Servidor WebLogic.

Manejar Transacciones

Esta sección proporciona guías para configurar y manejar transacciones a través de la Consola de Administración. Para información sobre la configuración de los almacenes de conexiones JDBC para permitir que los drivers JDBC participen en las transacciones distribuidas, puedes ver Manejar Conectividad JDBC.

Introducción

La Consola de Administración proporciona una interfase para las herramientas que nos permiten configurar las características del Servidor WebLogic, incluyendo el API JavaTransaction (JTA).
El proceso de configuración implica especificar valores para atributos. Estros atributos definen varios aspectos del entorno de transacciones, incluyendo:

- Time-outs de transacciones y límites.
- Control de comportamiento de transacciones.
- Prefijo del fichero log de transacciones.

Antes de configurar nuestro entorno de transacciones, deberíamos familiarizarnos con los componentes J2EE que pueden participar en las transacciones, como EJBs, JDBC, y JMS.

- EJBs (Enterprise JavaBeans) usan JTA para soporte de transacciones. Varios descriptores de despliegue se relacionan con el manejo de transacciones. Para más información sobre programación con EJBs y JTA, podrás ver próximamente Programar WebLogic Enterprise JavaBeans.
- JDBC (Java Database Connectivity) proporciona interfaces estándar para acceder a sistemas de bases de datos relacionales desde Java. JTA proporciona soporte de transacciones sobre conexiones recuperadas usando un driver JDBC y la fuente de datos de la transacción. Para más información podrás ver próximamente Programar WebLogic JDBC.
- JMS (Java Messaging Service) usa JTA para soportar transacciones entre múltiples fuentes de datos. WebLogic JMS es un controlador de recursos compatible XA. Para más información sobe programación de JMS y JTA, podrás ver próximamente Programar WebLogic JMS.

Configurar Transacciones

La Consola de Administración proporciona valores por defecto para todos los atributos de configuración JTA. Si especificamos un valor inválido para cualquier atributo de configuración, el Servidor WebLogic no arrancará.
Las selecciones de configuración para JTA son aplicables a nivel de dominio. Esto significa que los atributos de configuración se aplican a todos los servidores del dominio. La tareas de monitorizado y de log para JTA se realizan a nivel de servidor.
Una vez que hayamos configurado WebLogic JTA y cualquier transacción de partida, el sistema puede realizar transacciones usando el API JTA y las extensiones WebLogic JTA.
Podemos configurar cualquier atributo de transacción antes de ejecutar las aplicaciones (configuración estática) o, con una excepción, en el momento de ejecución de la aplicación (configuración dinámica). El atributo `TransactionLogFilePrefix` debe configurarse antes de ejecutar las aplicaciones.
Para configurar los atributos de transación, completamos el siguiente procedimiento:

1. Arrancamos la Consola de Administración.
2. Seleccionamos el nodo del dominio en el panel izquierdo. Por defecto se muestra la pestaña **Configuration** del dominio.

3. Pulsamos sobre la pestaña **JTA**.
4. Por cada atributo, especificamos un valor, o si está disponible, aceptamos el valor por defecto.
5. Pulsamos el botón **Apply** para almacenar los nuevos valores de atributos.
6. Nos aseguramos de que el atributo **Transaction Log File Prefix** está seleccionado cuando configuramos el servidor. Para más información sobre el atributo loggin, puedes ver la sección Monitorizar y hacer Logs de Transacciones.

La siguiente tabla describe brevemente los atributos de transacciones disponibles en el Servidor WebLogic:

Atributo	Descripción
`Timeout Seconds`	El tiempo, en segundos, que una transacción podría estar activa antes de que el sistema fuerce a terminarla.
`Abandon Timeout Seconds`	El tiempo máximo, en segundos, que un coordinador de transacción persiste en intentar completar una transacción.
`Before Completion Iteration Limit`	El número de retro-llamadas `beforeCompletion` que se procesan antes de que el sistema fuerce su terminación
`Max Transactions`	El número máximo de transacciones que podrían estar activas en un servidor particular en un momento dado.
`Max Unique Name Statistics`	El número máximo de nombres de transacciones únicas que podrían ser seguras por un servidor en un momento dado.
`Forget Heuristics`	Un valor booleano especificando si el controlador de transacciones debería instruir a un recurso para que olvide cualquier transacción con un salida heurística.

Monitorizar y Hacer Logs de Transacciones

La Consola de Administración nos permite monitorizar las transacciones y especificar el prefijo del fichero de log de transacciones. Las tareas de monitorización y loggin se realizan a nivel de servidor. Las estadísticas de transacciones se muestran para un servidor específico y cada servidor tiene un fichero log de transacciones.
Para mostrar las estadísticas de transacciones y configurar el prefijo para los ficheros de log de transacciones, completamos el siguiente procedimiento:

1. Arrancamos la Consola de Administración.
2. Pulsamos en el nodo **server** en el panel izquierdo.
3. Seleccionamos un servidor específico en el panel izquierdo.
4. Pulsamos en la pestaña **Monitoring**.
5. Pulsamos la pestaña **JTA**. Los totales para las estadísticas de transacciones se muestran en el diálogo JTA.

(También podemos pulsar los enlaces de texto de monitorización para monitorizar las transacciones por recurso, o por nombre, o para monitorizar todas las transacciones activas.)

6. Pulsamos la pestaña **Logging**.
7. Pulsamos la pestaña JTA.
8. Introducimos un prefijo para el fichero log de transacciones y pulsamos sobre **Apply** para grabar la selección de atributos.

Mover un Servidor a otra Máquina

Cuando un servidor de aplicaciones se mueve a otra máquina, debe poder localizar los ficheros de log de transacciones en el nuevo disco. Por esta razón, recomendamos mover este fichero a la nueva máquina antes de arrancar el servidor allí. Haciendo esto podemos asegurarnos de que **recovery** funciona correctamente. Si el path es diferente en la nueva máquina, actualizamos el atributo **TransactionLogFilePrefix** con el nuevo path antes de arrancar el servidor.
Cuando migramos logs de transacciones después de un fallo de servidor, hacemos que todos los fichero de log de transacciones estén disponibles en la nueva máquina antes de arrancar el servidor allí. Podemos conseguir esto almacenando los ficheros de log de transacciones en un disco de doble puerto disponible en ambas máquinas. Como el caso de una migración planeada, actualizamos el atributo **TransactionLogFilePrefix** con el nuevo path antes de arrancar el servidor si el path es diferente en la nueva máquina. De otro modo, las transacciones que estuvieran en el proceso de ser enviadas en el momento del crash podrían no resolverse correctamente, resultando en una inconsistencia de datos.

Manejo de Conectividad JDBC

Introducción al Control JDBC

La Consola de Administración proporciona un interfase a las herramientas que nos permiten configurar y manejar características del Servidor WebLogic, incluyendo JDBC (conectividad de bases de datos con Java). Para la mayoría de las funciones administrativas de JDBC, que incluyen creación, manejo y monitorización de conectividad, los administradores de sistemas usan la Consola Administrativa o el interfase de la línea de comandos. Los desarrolladores de aplicaciones podrían querer usar el API JDBC.
Las tareas realizadas más frecuentemente para configurar y manejar la conectividad incluyen:

- Definir los atributos que gobiernan la conectividad JDBC entre el Servidor WebLogic y nuestro sistema de control de la base de datos.
- Manejar la conectividad establecida.
- Monitorizar la conectividad establecida.

Sobre la Consola Administrativa

Nuestra forma principal de configurar y manejar la conectividad JDBC es a través de la Consola de Administración. Usando la Consola de Administración, configuramos la conectividad estáticamente antes de arrancar el servidor. Para más información, puedes ver Arrancar la Consola de Administración.
Además de seleccionar la conectividad, la Consola de Administración nos permite manejar y monitorizar la conectividad establecida.

Sobre el Interface de la Línea de Comandos

El interfase de la línea de comandos proporciona una forma de crear y manejar dinámicamente Almacenes de Conexiones. Puedes encontrar más información sobre cómo usar la interfase de la línea de comandos en Referencia de Interface de la Línea de Comandos para el Servidor WebLogic.

Sobre la Conectividad JDBC

Para información sobre la configuración y el control de la conectividad programáticamente, puedes ir a Programas WebLogic JDBC.

Información Relacionada

Los drivers JDBC, usados localmente y en transacciones distribuidas, interfieren con muchos componentes del Servidor WebLogic y la información aparece en varios documentos. Por ejemplo, la información sobre los drivers JDBC está incluida en los conjuntos de documentación para JDBC, JTA y jDrivers WebLogic.

Componentes JDBC - Almacenes de Conexiones, Fuentes de Datos y MultiPools

Almacenes de Conexiones

Un Almacén de Conexiones contiene grupos nombrados de conexiones JDBC que son creadas cuando se registra el Almacén de Conexiones, normalmente cuando arranca el Servidor WebLogic. Nuestra aplicación toma una conexión del almacén, la usa, y luego la devuelve al almacén cerrándola.
Todas las configuraciones que hacemos en la Consola de Administración son estáticas, es decir, todas se hacen antes de que arranque el Servidor WebLogic. Podemos crear Almacenes de Conexiones dinámicamente--después de que el servidor arranque--usando la línea de comandos o programáticamente, usando el API (Programar WebLogic JDBC).

MultiPools

Usados en transacciones locales (no distribuidas) o en configuraciones de un sólo Servidor WebLogic, los MultiPools ayudan en:

- El balance de carga-- los almacenes se añaden sin ningún orden adjunto y son accedidos usando un esquema circular. Cuando se necesitan conexiones, se selecciona el Almacén de Conexiones que hay después del último accedido.
- Alta disponibilidad-- configurar los almacenes como una lista ordenada determina el orden en el que ocurren los intercambios de conexiones. Por ejemplo, primero se selecciona el primer almacén de la lista, luego el segundo, etc.

Todas las conexiones de un Almacén de Conexiones particular son idénticas, es decir, están adjuntas a una misma base de datos. Sin embargo, los almacenes de conexiones dentro de un **MultiPool** podrían estar asociados con diferentes DBMS. Puedes leer más sobre la programación de **MultiPools** en Programar WebLogic JDBC.

Fuentes de Datos

Un objeto `Data Source` permite a los clientes JDBC obtener una conexión DBMS. Cada objeto `Data Source` apunta a un Almacén de Conexiones o un `MultiPool`. Los objetos `Data Source` pueden definirse con o sin JTA, lo que proporciona soporte para transacciones distribuidas. Puedes leer más sobre la programación de Fuentes de Datos en Programar WebLogic JDBC.

Nota:
Los `Data Sources Tx` no pueden apuntar a MultiPools, sólo a Connection Pools, porque los MultiPools no están soportados en las transacciones distribuidas.

Guías de Configuración JDBC, para Almacenes de Conexiones, Fuentes de Datos y MultiPools

Introducción a la Configuración JDBC

Para Configurar la conectividad JDBCm configuramos **Connection Pools**, objetos **Data Source** (siempre recomendados, pero opcionales en algunos casos) y **MultiPools** (opcionales) para definir atributos en la Consola de Administración y, para definir almacenes de conexiones dinámicamente, en la línea de comandos. Hay tres tipos de transacciones:

- **Local** transacción no distribuida.
- **Distribuida con Driver XA** negociación de dos fases.
- **Distribuida sin Driver XA** controlador de un sólo recurso y un sólo ejemplar de base de datos.

La siguiente tabla describe cómo usar estos objetos en transacciones locales y distribuidas:

Descripción/ Objeto	**Transacciones Locales**	**Transacciones Distribuidas con Driver XA**	**Transacciones Distribuidas sin Driver XA**
JDBC driver	• WebLogic jDriver para Oracle, Microsoft SQL Server, e Informix. • Drivers de terceras partes compatibles.	• jDriver WebLogic para Oracle/XA. • Drivers de terceras partes compatibles.	• jDriver WebLogic para Oracle, Microsoft SQL Server, e Informix. • Drivers de terceras partes compatibles.
Data Source	Objeto Data Source recomendado. (Si no hay Data Source, usar el API JDBC).	Tx Data Source requerido.	Tx Data Source requerido. Seleccionamos two-phase commit=true si hay más de un recurso.
Connection Pool	Requiere un objeto Data Source cuando se configura en la Consola de Administración.	Requiere TX Data Source.	Requiere TX Data Source.
MultiPool	Se requieren Connection Pool y Data Source Sólo se usa en configuraciones de un sólo servidor.	No soportado en transacciones distribuidas.	No soportado en transacciones distribuidas.

Nota:
Las transacciones distribuidas usan el jDriver WebLogic para Oracle/XA, el modo de transacción para el jDriver WebLogic para Oracle.

Drivers Soportados para Transacciones Locales:

- Los drivers JDBC 2.0 que soportan el JDBC Core 2.0 API (java.sql), incluyendo los jDrivers de WebLogic para Oracle, Microsoft SQL Server, e Informix. El API nos permite crear objetos de clases necesarios para establecer una conexión con una fuente de datos, enviar consultas y sentencias de actualización a la fuente de datos, y procesar los resultados.

Drivers Soportados para Transacciones Distribuidas:

- Cualquier driver JDBC 2.0 que soporte los interfaces estándar de transacciones distribuidas JDBC 2.0 (`javax.sql.XADataSource`, `javax.sql.XAConnection`, `javax.transaction.xa.XAResource`), incluyendo el jDriver de WebLogic para Oracle/XA.
- Cualquier driver JDBC que soporta el JDBC 2.0 Core API pero no soporta los interfaces estándar de transacciones distribuidas JDBC 2.0. Sólo un driver JDBC no-XA en un momento dado puede participar en una transacción distribuida.

Configurar Drivers JDBC

Configurar Drivers JDBC para Transacciones Locales
Para configurar los drivers JDBC para transacciones locales, configuramos el **Connection Pool JDBC** de esta forma:

- Especificamos el atributo `Driver Classname` como el nombre de la clase que soporta el interfase `java.sql.Driver`.
- Especificamos las propiedades de datos. Estas propiedades se pasan al driver específico como propiedades del driver.

La siguiente tabla muestra un ejemplo de configuración de **Connection Pool** usando el jDriver de WebLogic para Oracle:

Nombre de Atributo	Valor del Atributo
Name	`myConnectionPool`
Targets	`myserver`
DriverClassname	`weblogic.jdbc.oci.Driver`
Initial Capacity	`0`
MaxCapacity	`5`
CapacityIncrement	`1`
Properties	`user=scott;server=localdb`

La siguiente tabla muestra un ejemplo de configuración de **Data Source** usando el jDriver de WebLogic para Oracle:

Nombre de Atributo	Valor del Atributo
Name	`myDataSource`
Targets	`myserver`
JNDIName	`myconnection`
PoolName	`myConnectionPool`

La siguiente tabla muestra un ejemplo de configuración de **Connection Pool** usando el jDriver de WebLogic para Microsoft SQL Server:

Nombre de Atributo	Valor del Atributo
Name	`myConnectionPool`
Targets	`myserver`
DriverClassname	`weblogic.jdbc.mssqlsvr4.Driver`
Initial Capacity	`0`
MaxCapacity	`5`
CapacityIncrement	`1`

Properties	`user=sa;password=secret;db=pubs;`
	`server=myHost:1433;appname=MyApplication;`
	`hostn ame=myhostName`

La siguiente tabla muestra un ejemplo de configuración de **Data Source** usando el jDriver de WebLogic para Microsoft SQL Server:

Nombre de Atributo	Valor del Atributo
Name	`myDataSource`
Targets	`myserver`
JNDIName	`myconnection`
PoolName	`myConnectionPool`

La siguiente tabla muestra un ejemplo de configuración de **Connection Pool** usando el jDriver de WebLogic para Informix:

Nombre de Atributo	Valor del Atributo
Name	`myConnectionPool`
Targets	`myserver`
DriverClassname	`weblogic.jdbc.informix4.Driver`
Initial Capacity	`0`
MaxCapacity	`5`
CapacityIncrement	`1`

Properties	`user=informix;password=secret;`
	`server=myDBHost;port=1493;`
	`db=myDB`

La siguiente tabla muestra un ejemplo de configuración de **Data Source** usando el jDriver de WebLogic para Informix:

Nombre de Atributo	Valor del Atributo
Name	`myDataSource`
Targets	`myserver`
JNDIName	`myconnection`
PoolName	`myConnectionPool`

Configurar los Drivers XA JDBC para Transacciones Distribuidas
Para permitir que los drivers XA JDBC participen en transacciones distribuidas, configuramos el **JDBC Connection Pool** de esta forma:

- Especificamos el atributo `Driver Classname` como el nombre de la clase que soporta la interfase `javax.sql.XADataSource interfase`.
- Nos aseguramos de que se han especificado las propiedades de la base de datos. Estas propiedades se pasan al `XADataSource` especificado como propiedades de la fuente de datos.

Los siguientes atributos son un ejemplo de una configuración **JDBC Connection Pool** usando el jDriver de WebLogic para Oracle en modo XA:

Nombre de Atributo	Valor del Atributo
Name	`fundsXferAppPool`
Targets	`myserver`
DriverClassname	`weblogic.jdbc.oci.xa.XADataSource`
Initial Capacity	`0`
MaxCapacity	`5`
CapacityIncrement	`1`
Properties	`user=scott;password=tiger;server=localdb`

Los siguientes atributos son un ejemplo de una configuración **Tx Data Source** usando el jDriver de WebLogic para Oracle en modo XA:

Nombre de Atributo	Valor del Atributo
Name	`fundsXferData Source`
Targets	`myserver`
JNDIName	`myapp.fundsXfer`
PoolName	`fundsXferAppPool`

También podemos configurar el **JDBC Connection Pool** para usar un driver de terceras partes en modo XA. En dichos casos, las propiedades de la fuente de datos se seleccionan mediante reflection sobre el ejemplar `XADataSource` usando el patrón de diseño de los `JavaBeans`. En otras palabras, para la propiedad **abc**, el ejemplar `XADataSource` debe soportar los métodos `get` y `set` con los nombres `getAbc` y `setAbc`, respectivamente.

Los siguientes atributos son un ejemplo de una configuración **JDBC Connection Pool** usando el Driver Thin para Oracle:

Nombre de Atributo	Valor del Atributo
Name	`jtaXAPool`
Targets	`myserver,server1`
DriverClassname	`oracle.jdbc.xa.client.OracleXADataSource`
Initial Capacity	`1`
MaxCapacity	`20`
CapacityIncrement	`2`
Properties	`user=scott;password=tiger;` `url=jdbc:oracle:thin:@baybridge:1521:bay817`

Los siguientes atributos son un ejemplo de una configuración **Tx Data Source** usando el Driver Thin para Oracle:

Nombre de Atributo	Valor del Atributo
Name	`jtaXADS`
Targets	`myserver,server1`
JNDIName	`jtaXADS`
PoolName	`jtaXAPool`

Configuramos el **JDBC Connection Pool** para usarlo con un driver **Cloudscape** de esta forma:

Nombre de Atributo	Valor del Atributo
Name	`jtaXAPool`
Targets	`myserver,server1`
DriverClassname	`COM.cloudscape.core.XADataSource`
Initial Capacity	`1`
MaxCapacity	`10`
CapacityIncrement	`2`
Properties	`databaseName=CloudscapeDB`
SupportsLocalTransaction	`true`

Configuramos el **Tx Data Source** para usarlo con un driver **Cloudscape** de esta forma:

Nombre de Atributo	Valor del Atributo
Name	`jtaZADS`
Targets	`myserver,myserver1`
JNDIName	`JTAXADS`
PoolName	`jtaXAPool`

WebLogic jDriver para Oracle/XA Data Source Properties

La siguiente tabla lista las propiedades de fuente de datos soportadas por WebLogic jDriver para Oracle. La columna JDBC 2.0 indica si una propiedad específica de fuente de datos es una propiedad estándar JDBC 2.0 (Y) o un extensión del Servidor WebLogic para JDBC (N).

La columna **Opcional** indica si una propiedad de fuente de datos particular es opcional o no. Las propiedades marcadas con **Y*** son mapeadas a sus correspondientes campos en el string `xa_open` de Oracle (valor de la propiedad `openString`). Si no se especifican, sus valores por defecto se toman de la propiedad `openString`. Si se especifican, sus valores por defecto deberían corresponder con los especificados en la propiedades `openString`.

Si las propiedades no corresponden, se lanza una `SQLException` cuando intentamos una conexión XA.

Las propiedades obligatorias están marcadas con **N*** y también son mapeadas a sus correspondientes campos del string `xa-open` de Oracle. Especificamos estas propiedades cuando las especifica el string `xa-open` de Oracle. Si no se especifican, o se especifican pero no corresponden, se lanzará una `SQLException` cuando intentemos una conexión XA.

Los nombres de propiedades marcados con **** son soportados, por no usados, por el Servidor WebLogic.

Nombre de Propiedad	Tipo	Descripción	JDBC 2.0	Opcional	Valor por Defecto
databaseName**	String	Nombre de una base de datos particular en el servidor	Y	Y	Ninguno
dataSourceName	String	Un nombre de fuente de datos; usado para nombrar un XADataSource subyacente.	Y	Y	Nombre del Connection Pool
description	String	Descripción de esta fuente de datos.	Y	Y	Ninguno
networkProtocol**	String	Protocolo de red usado para comunicar con el servidor.	Y	Y	Ninguno
password	String	Una password de base de datos.	Y	N*	Ninguno
portNumber**	Int	Número de puerto en el que el servidor escucha peticiones.	Y	Y	Ninguno
roleName**	String	El nombre del rol SQL inicial	Y	Y	Ninguno
serverName	String	Nombre del Servidor de bases de datos.	Y	Y*	Ninguno
user	String	Nombre de la cuenta de usuario.	Y	N*	Ninguno
openString	String	String XA abierto de Oracle.	N	Y	Ninguno
oracleXATrace	String	Indica si está activa la salida XA tracing. Si está activa (true), se situará un fichero con el formato xa_poolnamedate.trc en el directorio donde se arrancó el servidor.	N	Y	true

La siguiente tabla lista el mapeo entre los campos string `xa_open` de Oracle y las propiedades de la fuente de datos:

Nombre de campo String xa_open Oracle	Atributo de fuente de datos JDBC 2.0	Opcional
acc	user, password	N
sqlnet	ServerName	

Observa que también los usuarios podrían especificar `Threads=true` en un string xa_open de Oracle. Para una completa descripción de los campos xa_open puedes consular la documentación de Oracle.

Configurar Drivers JDBC No-XA para Transacciones Distribuidas

Cuando configuramos el **JDBC Connection Pool** para permitir que los drivers JDBC no-XA participen con otros recursos en transacciones distribuidas, especificamos el atributo `Enable Two-Phase Commit` para el **JDBC Tx Data Source**. (Este parámetro es ignorado por recursos que soportan el interfase `XAResource`.) Observa que sólo un almacén de conexiones no-XA puede participar en una transacción distribuida.

No-XA Driver/Single Resource

Si estamos usando un sólo driver no-XA y es el único recurso de la transacción, dejamos la opción `Enable Two-Phase Commit` sin marcar en la Consola (aceptamos el valor por defecto `enableTwoPhaseCommit = false`). En este caso, el **Transaction Manager** realiza una optimización de una fase.

No-XA Driver/Multiple Resources

Si estamos usando un driver JDBC no-XA con otros recursos XA, marcamos `Enable Two-Phase Commit` en la Consola (`enableTwoPhaseCommit = true`).

Cuando `enableTwoPhaseCommit` se selecciona a true, los recursos JDBC no-XA siempre devuelven **XA-OK** durante la llamada al método `XAResource.prepare()`. El recurso intenta negociar o volver a su transacción local en respuesta a las subsecuentes llamadas a `XAResource.commit()` o `XAResource.rollback()`. Si la negociación o el retorno falla, resulta un error heurístico. Los datos de la aplicación se podrían quedar en un estado inconsistente como resultado de un fallo heurístico.

Cuando `Enable Two-Phase Commit` no se selecciona en la Consola (`enableTwoPhaseCommit=false`), el recurso JDBC no-XA hace que `XAResource.prepare()` falle. Este mecanismo asegura que sólo hay un participante en la transacción, ya que `commit()` lanza una `SystemException` en este caso.

Cuando sólo hay un recurso participando en una transacción, la única fase de optimización pasa de `XAResource.prepare()`, y la transacción se negocia con éxito en la mayoría de los ejemplares.

Aquí tenemos la configuración de los atributos para un ejemplo de **JDBC Connection Pool** usando un driver JDBC no-XA:

Nombre de Atributo	Valor del Atributo
Name	`fundsXferAppPool`
Targets	`myserver`
DriverClassname	`weblogic.jdbc.oci.Driver`
Initial Capacity	`0`
MaxCapacity	`5`
CapacityIncrement	`1`
Properties	`jdbc:weblogic:oracle`

Aquí tenemos la configuración de los atributos para un ejemplo de **Tx Data Source** usando un driver JDBC no-XA:

Nombre de Atributo	Valor del Atributo
Name	`fundsXferDataSource`
Targets	`myserver,server1`
JNDIName	`myapp.fundsXfer`
PoolName	`fundsXferAppPool`
EnableTwoPhaseCommit	`true`

Seleccionar y Manejar Almacenes de Conexiones, Fuentes de Datos y MultiPools

Configuración y Asignación JDBC

Usando las Consola de Administración, podemos configurar la conectividad estáticamente especificando atributos y propiedades de bases de datos para los componentes JDBC —Connection Pools, Data Sources, y MultiPools.
Los **Data Sources** están asociados con los **Connection Pools** o **MultiPools** ("pool")—toda **Data Source** está comúnmente asociado con un almacén específico. La **Data Source** y el almacén son asignados a la misma fuente—al mismo servidor o server/cluster relacionado. No podemos asignar una **Data Source** a un servidor, y luego el **Connection Pool** a otro. En la siguiente tabla encontraremos más información:

Escenario Nº:	**Asociación...**	**Asigna...**	**Descripción Fuente...**
1	Data Source A con Connection Pool A	1. La Data Source A para el servidor controlado 1, y 2. El Connection Pool para el servidor controlado 1	La **Data Source** y el **Connection Pool** asignados a la misma fuente.
2	Data Source B con Connection Pool B	• La Data Source B al Cluster X, entonces • El Connection Pool B al servidor Controlado 2 en el Cluster X.	La **Data Source** y el **Connection** asignado a las fuentes server/cluster relacionados.
3	Data Source C con Connection Pool C	• La Data Source A y el Connection Pool A al servidor controlado 1 - Y - • La Data Source al Cluster X; leugo asignamos el Connection Pool A al servidor controlad 2 en el cluster X.	La **Data Source** y el **Connection Pool** asignados como una unidad a dos fuentes diferentes.

(Podemos asignar más de una **Data Source** a un almacén, pero no hay un propósito práctico para esto). Podemos asignar estas combinaciones de **Data Source/pool** a más de un servidor o cluster, pero deben asignarse combinadas. Por ejemplo, no podemos asignar un **Data Source** a un Servidor Controlado A si está asociado a un **Connection Pool** que está asignado sólo a un Servidor B.
Podemos configurar dinámicamente los **Connection Pools** (después de arrancar el servidor) usando el interfase de la línea de comandos. También podemos configurarlos programáticamente usando el API.
Configuraciones JDBC para Servidores o Clusters
Una vez que hemos configurado y asociado la **Data Source** y el **Connection Pool** (o **MultiPool**), entonces asignamos cada objeto al mismo servidor o servidor/cluster. Aquí podemos ver algunos escenarios comunes:

- En un cluster, asignamos la **Data Source** al cluster, y asignamos el **Connection Pool** asociado a cada servidor controlador en el cluster.
- En una configuración de un sólo servidor, asignamos cada **Data Source** y su **Connection Pool** asociado al servidor.
- Si estamos usando un **MultiPool**, asignamos los **Connection Pools** al **MultiPool**; luego asignamos la **Data Source** a todos los **Connection Pools** y el **MultiPool** al servidor.

Configurar la Conectividad JDBC Usando la Consola de Administración

La Consola de Administración nos permite configurar, manejar y monitorizar la conectividad JDBC. Para mostrar las pestañas que usaremos para realizar estas tareas, completamos el siguiente procedimiento:

1. Arrancamos la Consola de Administración.
2. Localizamos el nodo **Services** en el panel izquierdo, luego expandimos el nodo **JDBC**.
3. Seleccionamos la pestaña específica del componente que queremos configurar o manejar — **Connection Pools**, **MultiPools**, **Data Source**, o **Tx Data Source**.
4. Seguimos las instrucciones de la Ayuda en Línea...

Configurar la Conectividad JDBC Usando el Interface de la Línea de Comandos
Puedes encontrar más información sobre cómo usar el interfase de la línea de comandos en Referencia de Interface de la Línea de Comandos para el Servidor WebLogic.

Manejar JMS

Configurar JMS

Usando la Consola de Administración, definimos los atributos de configuración para:

- Activar JMS.
- Crear servidores de JMS
- Crear y/o personalizar valores de servidores JMS, factorías de conexiones, destinos (colas y tópicos), plantillas de destinos, claves de destino, almacenes de sesiones y consumidores de conexiones.
- Configurar aplicaciones JMS personalizadas.
- Definir umbrales y cuotas.
- Activar las características JMS deseadas, como cluster de servidores (ver la siguiente sección), proceso de mensajes concurrentes, ordenación de destinos, y mensajería persistente.

WebLogic JMS proporciona valores por defecto para algunos atributos de configuración; nosotros debemos proporcionar valores para los otros. Si especificamos un valor no válido para cualquier atributo de configuración, o si no especificamos un valor para un atributo para el que no existe valor or defecto, el Servidor WebLogic no iniciará JMS cuando lo arranquemos. Con el producto se proporciona un ejemplo de configuración JMS.
Cuando migramos desde una versión anterior, la información de configuración se convertirá automáticamente, según se describe en Migrar Aplicaciones Existentes.
Para configurar los atributos de WebLogic JMS, realizamos los siguientes pasos:

1. Arrancamos la Consola de Administración.
2. Seleccionamos el botón **JMS** bajo **Services** en el panel izquierdo para expandir la lista.
3. Seguimos los procedimientos descritos en las siguientes secciones...

Una vez configurado WebLogic JMS, las aplicaciones podrán enviar y recibir mensajes usando el API JMS.

Configurar Factorías de Conexiones

Las factorías de conexiones permiten a los clientes JMS crear conexiones JMS. Configuramos las factorías para crear conexiones con atributos predefinidos.
Para configurar las factorías de conexiones, usamos el nodo **Connection Factories** en la Consola de Administración para definir lo siguiente:

- Configurar atributos, incluyendo:
  - Nombre de la factoría de conexiones.
  - Nombre para acceder a la factoría de conexiones desde dentro del espacio de nombres JNDI.
  - Identificador de Cliente (client ID) que puede ser usado por los clientes con suscriptores durables.
  - Atributos de envío de mensajes por defecto (es decir, prioridad, tiempo-de-vida, y modo).

    - Máximo número de mensajes en espera que podrían existir en una sesión asíncrona y la política de sobrecarga, (es decir, la acción a tomar, en sesiones multicast, cuando se alcanza el máximo).
    - Si está permitido o no que el método `close()` sea llamado desde el método `onMessage()`.
    - Atributos de Transacción (time-out de transacción y si están permitidas las transacciones de un usuario JTA).
- Fuentes (WebLogic Servers) que están asociadas con una factoría de conexiones para soporte de clustering. Las Targets nos permiten limitar el uso de servidores, grupos, y/o clusters en los que se podría desplegar la factoría de conexiones.

WebLogic JMS define una factoría de conexiones, por defecto:
`weblogic.jms.ConnectionFactory`. Todos los atributos de configuración están seleccionados a sus valores por defecto en esta factoría.
Si la definición de la factoría de conexiones por defecto es apropiada para nuestra aplicación, no necesitamos configurar ninguna otra factoría adicional.

Nota:
Usando la factoría de conexiones por defecto, no tenemos control sobre el servidor JMS en el que la factoría podría estar desplegada. Si queremos dirigirnos a un servidor JMS particular, creamos una nueva factoría de conexiones y especificamos el servidor (o servidores) JMS fuente apropiados.

Algunos atributos de la factoría de conexiones son configurables dinámicamente. Cuando los atributos dinámicos se modifican durante la ejecución, los nuevos valores sólo son efectivos para las nuevas conexiones, y no afectan al comportamiento de las conexiones existentes.

Configurar Plantillas

Las plantillas proporcionan una forma eficiente de definir múltiples destinos con selecciones de atributos similares. Las plantillas ofrecen los siguientes beneficios:

- No necesitamos re-introducir cada selección de atributo cada vez que definamos un nuevo destino; podemos usar la plantilla y sobrescribir cualquier selección a la que queramos asignar un nuevo valor.
- Podemos modificar dinámicamente las selecciones de atributos compartidos simplemente modificando la plantilla.

Para definir los atributos de configuración de la plantilla de destino, usamos el nodo **Templates** de la Consola de Administración. Los atributos configurables para una plantilla de destino son los mismos que los configurados para un destino. Estos atributos de configuración se heredan por los destinos que usan, con las siguientes excepciones:

- Si el destino que está usando una plantilla especifica un valor que sobrescribe un atributo, se usa el valor sobrescrito.
- El atributo `Name` no es heredado por el destino. Este nombre sólo es válido para la plantilla. Todos los destinos deben definir explícitamente un nombre único.
- Los atributos `JNDI Name`, `Enable Store`, y `Template` no están definidos para plantillas de destino.
- Los atributos `Multicast` no están definidos para plantillas de destino porque sólo se aplican a tópicos.

Cualquier atributo que no se haya definido explícitamente para un destino es asignado a su valor por defecto. Si no existe valor por defecto, debemos asegurarnos de especificar un valor dentro de la plantilla

de destino o como un atributo de destino sobrescrito. Si no hacemos esto, la información quedará incompleta, fallará la configuración de WebLogic JMS, y WebLogic no se iniciará.

Configurar Claves de Destino

Las claves de destino se usan para definir el orden de ordenación para un destino específico.
Para crear una clave de destino, usamos el nodo **Destination Keys** de la Consola de Administración y definimos los siguientes atributos de configuración:

- Nombre de la clave de destino.
- Nombre de la propiedad por la que ordenar.
- Tipo de clave esperada.
- Dirección en la que ordenar (ascendente o descendente).

Configurar Stores

El **Store** consiste en un fichero o base de datos que se usa para mensajería persistente.
A través del uso del JDBC, JMS nos permite almacenar los mensajes existentes en una base de datos, a la que accederemos a través del almacén de conexiones JDBC designado. La base de datos JMS puede ser cualquier base de datos que sea accesible a través de un driver JDBC. WebLogic soporta y proporciona drivers para las siguientes bases de datos:

- Cloudscape
- Informix
- Microsoft SQL (MSSQL) Server (Versiones 6.5 y 7)
- Oracle (Versión 8.1.6)
- Sybase (Versión 12)

Opcionalmente, podemos restringir la lista de control de acceso (ACL) para el almacén de conexiones JDBC. Si restringimos esta ACL, deberemos incluir al usuario **system** de WebLogic y a cualquier usuario que envíe mensajes JMS en la lista.

Nota:
Los ejemplos JMS están configurados para trabajar con la base de datos Cloudscape de Java. Con WebLogic Server se incluye una versión de evaluación de Cloudscape y una base de datos de ejemplo: ***demoPool****.*

Para crear un fichero o base de datos, usamos el nodo **Stores** de la Consola de Administración y definimos los siguientes atributos:

- Nombre del **store**.
- (Para un fichero **store**) Directorio en que se grabó el fichero.
- (Para un **store** de base de datos JDBC) el almacén de conexiones JDBC y el nombre de la tabla de la base de datos para usar con múltiples ejemplares.

Aviso:
No podemos configurar un almacén de conexiones XA con un **store** de base de datos JDBC.

Nota:
Los ***stores*** *JMS pueden incrementar la cantidad de memoria requerida durante la inicialización de un Servidor WebLogic según se incrementa el número de mensajes almacenados. Si la inicialización falla debido a la memoria insuficiente cuando estamos reiniciando un Servidor WebLogic, debemos incrementar el tamaño de la pila de la Máquina Virtual Java (JVM) proporcionalmente al número de mensajes que haya almacenados en el* ***store*** *JMS. E intentamos reiniciar otra vez.*

Sobre los Stores JMS
La base de datos JMS contiene dos tablas de sistema que JMS genera y utiliza automáticamente:

- `<prefix>JMSStore`
- `<prefix>JMSState`

El nombre `prefix` identifica de forma única tablas JMS en el **store**. Especificar un prefijo único permite que existan varios **stores** en la misma base de datos. El prefijo se configura mediante la Consola de Administración cuando configuramos el **store JDBC**. Un prefijo se adhiere al nombre de la tabla cuando el DBMS requiere un nombre totalmente cualificado, o cuando necesitamos diferenciar entre tablas JMS para dos servidores WebLogic, permitiendo que puedan almacenarse distintas tablas en una sola DBMS.
El prefijo debería especificarse usando el siguiente formato, lo que resultará en un nombre de tabla válido que será añadido al nombre de la tabla JMS:
`[[catalog.]schema.]prefix`

Aviso:
No se debería permitir que dos **stores JMS** usen las mismas tablas de la base de datos, ya que esto resultaría en una corrupción de los datos.

Selecciones de JDBC Connection Pool para Stores JMS
Para implementaciones que usan un **store JDBC**, si el DMS debería desconectarse y ponerse en línea de nuevo, WebLogic JMS no podrá acceder al **store** hasta que se reinicie el Servidor WebLogic. Para evitar esta situación, deberíamos configurar los siguientes atributos en el **JDBC Connection Pool** asociado con el **store JMS**:
TestConnectionsOnReserve=”true” TestTableName=”[[[catalog.]schema.]prefix]JMSState”
De otra forma, si los recursos JDBC se desconectan y vuelven a ponerse en línea, el JMS no podrá reutilizarlos hasta que el Servidor WebLogis sea reiniciado.

Configurar Servidores JMS

Un Servidor JMS maneja conexiones y peticiones de mensajes por cuenta de los clientes.
Para crear un servidor JMS, usamos el nodo **Servers** de la Consola de Administración y definimos lo siguiente:

- Configuración de atributos incluyendo:
  - Nombre del servidor JMS.
  - **Store** (fichero o base de datos JDBC) requerido para mensajes persistentes. Si no asignamos un **store** para un servidor JMS, la mensajería persistente no estará soportada en ese servidor.
  - Plantilla usada para crear todos los destinos temporales, incluyendo colas temporales y tópicos temporales.
  - Umbrales y cuotas para mensajes y bytes (número máximo y umbrales alto y bajo).
- Fuentes (WebLogic Servers) que están asociados con un servidor JMS para soportar clustering. Las fuentes nos permiten limitar el conjunto de servidores, grupos y/o clusters en lo que so podría desplegar un servidor JMS.

Nota:
El despliegue de un servidor JMS es diferente del de una factoría de conexiones o una plantilla. Un servidor JMS se despliega en un sólo servidor. Una factoría de conexiones o una plantilla puede ser ejemplarizada en múltiples servidores simultáneamente.

Configurar Destinos

Un destino identifica una cola o tópico. Una vez que hemos definido un Servidor JMS, podemos configurar sus destinos. Podemos configurar uno o más destinos por cada servidor JMS.
Podemos configurar destinos explícitamente o configurando una plantilla de destino que pueda ser usada por múltiples destinos con similares selecciones de atributos, como se describe en Configurar Plantillas.
Para configurar destinos explícitamente, usamos el nodo **Destinations** de la Consola de Administración, y definimos los siguientes atributos de configuración:

- Nombre y tipo (cola o tópico) de destino.
- Nombre para acceder al destino dentro del espacio de nombres JNDI.
- Si hay disponible o no un **Store** para almacenar mensajes persistentes.
- Plantilla usada para crear destinos.
- Claves usadas para definir la ordenación para un destino específico.
- Umbrales y cuotas para mensajes y bytes (número máximo, y umbrales superior e inferior),
- Atributos de mensaje que pueden ser sobrescritos (como prioridad, tiempo-de-vida, y modo de entrega).
- Atributo de Multicasting, incluyendo dirección de multicast, puerto y tiempo-de-vida (sólo para tópicos).

Algunos atributos de destino son configurables dinámicamente. Cuando los atributos se configuran en tiempo de ejecución, sólo se ven afectados los mensajes entrantes, los mensajes almacenados no se ven afectados.

Configurar Almacenes de Sesión

Los almacenes de sesión de servidor hacen posible que una aplicación procese mensajes de forma concurrente. Una vez que hemos definido un servidor JMS, tenemos la opción de configurar sus almacenes de sesión de servidor.
Podemos configurar uno o más almacenes de sesión por cada servidor JMS.
Para configurar los almacenes de sesión, usamos el nodo **Session Pools** de la Consola de Administración y definimos los siguientes atributos de configuración:

- Nombre del almacén de sesión del servidor.
- Factoría de conexión con la que está asociada el almacén de sesión de servidor y que es utilizada para crear sesiones.
- Clase oyente de `Message` usada para recibir y procesar mensajes concurrentemente.
- Atributos de transacción (modo de reconocimiento y si el almacén de sesiones crea sesiones transaccionales).
- Número máximo de sesiones concurrente.

Algunos atributos de almacenes de sesión se pueden configurar dinámicamente, pero los nuevos valores no tendrán efecto hasta que se reinicien los almacenes de sesiones.

Configurar Clientes de Conexiones

Los consumidores de conexiones recuperan sesiones de servidor y procesan mensajes. Una vez hemos definido un almacén de sesión, tenemos la opción de configurar sus consumidores de conexiones.
Podemos configurar uno o más consumidores de conexión por cada almacén de sesión.
Para configurar consumidores de conexión, usamos el nodo **Session Pools** en la Consola de Administración para definir los siguientes atributos de configuración:

- Nombre del consumidor de conexión.
- Número máximos de mensajes que puede acumular el consumidor de conexión.
- Selector de expresión JMS usado para filtrar mensajes.
- Destino sobre el que escuchará el consumidor de conexión.

Monitorizar JMS

Se proporcionan estadísticas para los siguientes objetos JMS: servidores JMS, conexiones, sesiones, destinos, productores y consumidores de mensajes, y almacenes de sesión de servidor. Podemos monitorizar estas estadísticas usando la Consola de Administración.
Las estadísticas JMS continúan incrementándose mientras se esté ejecutando el servidor. Solo se pueden resetear cuando se reinicia el servidor.
Para ver la información de monitorización JMS, seguimos los siguientes pasos:

1. Arrancamos la Consola de Administración.
2. Seleccionamos el nodo **JMS** bajo **Services**, en el panel izquierdo, para expandir la lista de servicios JMS.
3. Seleccionamos el nodo **JMS Server** bajo **JMS** en el panel izquierdo.

La información sobre los servidores JMS se muestra en el panel derecho.

4. Seleccionamos el servidor JMS que queremos monitorizar en la lista de servidores JMS o, desde los Servidores JMS mostrados en el panel de la derecha.

La información sobre los servidores JMS se muestra en el panel derecho.

5. Seleccionamos la pestaña **Monitoring** para ver los datos monitorizados.

Recuperar después de un Fallo del Servidor WebLogic

Las siguientes secciones describen como reiniciar o reemplazar un Servidor WebLogic en el evento de un fallo del sistema, y proporcionan consideraciones de programación para una terminación amistosa de una aplicación cuando sucede dicho evento.

Re-arrancar o Reiniciar el Servidor WebLogic

En el caso de que falle un Servidor WebLogic, hay tres métodos que podemos usar para realizar una recuperación del sistema:

- Reiniciar el servidor que ha fallado.
- Arrancar un nuevo servidor usando la misma dirección IP que el servidor que ha fallado.
- Arrancar un nuevo servidor usando una dirección IP diferente a la del servidor que ha fallado.

Para reiniciar el servidor que ha fallado o arrancar un nuevo servidor usando la misma dirección IP, reiniciamos el servidor y arrancamos los procesos de servidor.
Para arrancar un nuevo servidor usando una dirección IP diferente:

1. Actualizamos el **Domain Name Service** (DNS) para que el alias del servidor reconozca la nueva dirección IP.
2. Reiniciamos el servidor y arrancamos los procesos de servidor.
3. Opcionalmente realizamos una o más de las siguientes tareas:

Si nuestra aplicación usa...	**Realizamos las tareas...**
Persistent messaging—JDBC Store	o Si el **store JDBC** existe físicamente en el servidor que ha fallado, migramos la base de datos al nuevo servidor para asegurarnos de que el atributo URL del almacén de conexiones JDBC refleja la localización apropiada. o Si la base de datos JDBC no existe físicamente, los accesos a la base de datos no se han visto impactados, y no es necesario ningún cambio.
Persistent messaging—File Store	Migramos el fichero al nuevo servidor, asegurándonos de que el path dentro del directorio home del Servidor WebLogic es el mismo que en el servidor original.
Transactions	Migramos el log de transacciones al nuevo servidor copiando todos los ficheros llamados **<servername>*.tlog**. Esto se puede realizar almacenando los ficheros de log de transacciones en un disco con doble puerto que puede montarse en cualquier máquina, o copiando los ficheros manualmente. Si los ficheros están localizados en un directorio diferente en el nuevo servidor, actualizamos la configuración del atributo `TransactionLogFilePrefix` antes de arrancarlo. **Nota:** Si la migración sigue a un crash del sistema, es muy importante que los ficheros de log de transacciones estén disponibles cuando se re-arranque el servidor en su nueva localización. De otra forma, las transacciones en proceso de ser enviadas en el momento del crash podrían no resolverse correctamente, resultando en una inconsistencia de datos. Todas las transacciones no enviadas serán deshechas.

Consideraciones de Programación

Podríamos querer programar que nuestra aplicación termine de forma amistosa en el caso de un fallo de un Servidor WebLogic. Por ejemplo:

Si un Servidor WebLogic falla y ...	entonces...
Estamos conectados al Servidor WebLogic que ha fallado.	Se le enviará una `JMSException` al oyente de excepciones de conexión. Debemos re-arrancar la aplicación una vez que el servidor se haya reiniciado o se reemplazado.
No estamos conectados al Servidor WebLogic que ha fallado.	Debemos re-establecer todo una vez que el servidor se haya reiniciado o reemplazado.
Un Servidor JMS está apuntando al Servidor WebLogic que ha fallado.	Se enviará una `ConsumerClosedException` al oyente de excepciones de sesión. Debemos re-establecer cualquier consumidor de mensajes que se haya perdido.

Manejar JNDI

Cargar Objetos en el Árbol JNDI

Antes de poder acceder a un objeto usando WebLogic JNDI, debemos cargar el objeto en el árbol **WebLogic Server JNDI**. La Consola de Administración del Servidor WebLogic nos permite cargar servicios J2EE en el árbol JNDI.
Para cargar un objeto en el árbol JNDI, elegimos un nombre bajo el que queremos sea cargado en el árbol JNDI y lo introducimos en el campo **JNDI Name** cuando creamos el objeto.

Ver el Árbol JNDI

De vez en cuanto podemos encontrar útil ver los objetos en el árbol JNDI del Servidor WebLogic. Para ver el árbol JNDI, pulsamos el enlace **JNDI Tree** en la pestaña **Monitoring** de nuestro despliegue de Servidor WebLogic.

Manejar Licencias de Servidor WebLogic

Instalar una Licencia de Evaluación

Una copia de evaluación de 30 días de WebLogic Server está activa 30 días para que podamos usarlo inmediatamente. Para usar WebLogic Server más allá del periodo de evaluación de 30 días, necesitamos contactar con nuestro vendedor para pedirle otro periodo de evaluación o comprar una licencia por cada dirección IP en la que pensemos usar WebLogic Server. Todos los productos de evaluación de WebLogic Server están licenciados para usarse en un sólo servidor con acceso sólo desde 3 direcciones IP de clientes.
Si descargamos WebLogic Server desde la site de BEA, nuestra licencia de evaluación está incluida en la distribución. El programa de instalación de WebLogic Server nos permite especificar la localización del directorio home de BEA, e instalar un fichero de licencia BEA, **license.bea**, en ese directorio.

Actualizar una Licencia

Necesitaremos actualizar el fichero de licencia BEA en alguno de estos casos:

- Hemos comprado software de BEA adicional.
- Obtenemos una nueva distribución que incluye nuevos productos.
- Hemos aplicado una extensión de nuestra licencia de evaluación de 30 días.

En cualquiera de esos casos, recibiremos un fichero de actualización de licencia por e-mail como un attachment. Para actualizar el fichero de licencia BEA, hacemos lo siguiente:

1. Grabamos el fichero de licencia actualizado con un nombre distinto a **license.bea** en el directorio home de BEA.
2. Nos aseguramos de que **java** (Java 2) está en nuestro path. Para añadir el JDK a nuestro path, introducimos uno de los siguientes comandos:
3. `set PATH=.\jdk130\bin;%PATH% (sistemas Windows)`
4. `set PATH=./jdk130/bin:$PATH (sistemas UNIX)`
5. En un shell de comandos, introducimos el siguiente comando en el directorio home de BEA:
6. `UpdateLicense license_update_file`

donde `license_update_file` es el nombre con el que grabamos el fichero de licencia actualizado recibido por e-mail. Al ejecutar este comando se actualiza el fichero **license.bea**.

7. Grabamos una copia de nuestro fichero **license.bea** en un lugar seguro fuera de la distribución WebLogic. Aunque nadie puede usar nuestro fichero de licencia, deberíamos guardar esta información en un lugar protegido de posibles acciones malintencionadas o inocentes.

Usar las Utilidades Java del Servidor WebLogic

WebLogic Server proporciona varios programas Java que simplifican las tareas de instalación y configuración, proporcionan servicios, y ofrecen atajos convenientes. Esta página describe las utilidades Java proporcionadas por WebLogic Server. Para todas las utilidades se especifica la sintaxis de la línea de comandos, y para algunas, se proporcionan algunos ejemplos.

AppletArchiver

La utilidad **AppletArchiver** ejecuta un applet en un marco separado, mantiene un registro de todas las clases descargadas y los recursos usados por el applet, y los paquetes en los que están bien un fichero **.jar** o un fichero **.cab**. (La utilidad **cabarc** está disponible por Microsoft.)

Sintaxis de AppletArchiver

```
$ java utils.applet.archiver.AppletArchiver URL filename
```

Argumento	Definición
URL	URL del applet
filename	Nombre de fichero local que es el destino para el archivo **.jar/.cab**.

der2pem

La utilidad **der2pem** convierte un certificado X509 del formato DER al formato PEM. El fichero **.pem** se escribe en el mismo directorio que el fichero **.der** fuente.

Sintaxis de der2pem

```
$ java utils.der2pem derFile [headerFile] [footerFile]
```

Argumento	Definición
derFile	El nombre del fichero a convertir. El nombre debe terminar con una extension **.der**, y debe contener un certificado válido en el formato **.der**.
headerFile	La cabecera a colocar en el fichero PEM. La cabecera por defecto es "`-----BEGIN CERTIFICATE-----`" Usamos una cabecera de fichero si el fichero DER que está siendo convertido es una fichero de clave privada, y creamos la cabecera conteniendo uno de los siguientes: • "`-----BEGIN RSA PRIVATE KEY-----`" para clave privada sin encriptar. • "`-----BEGIN ENCRYPTED PRIVATE KEY-----`" para clave privada encriptada. **Nota:** Debe haber una nueva línea al final de la línea de cabecera en el fichero.
footerFile	El pie a colocar en el fichero PEM. El valor por defecto es "`-----END CERTIFICATE-----`". Usamos un pie de fichero si el fichero DER que está siendo convertido es un fichero de clave privada, y creamos el pie de fichero conteniendo uno de los siguientes: • "`-----END RSA PRIVATE KEY-----`" para clave privada sin encriptar. • "`-----END ENCRYPTED PRIVATE KEY-----`" para clave privada encriptada. **Nota:** Debe haber una nueva línea al final de la línea de pie en el fichero.

Ejemplo de der2pem

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

dbping

La utilidad de línea de comandos **dbping** prueba la conexión entre una DBMS y nuestra máquina cliente mediante un jDriver WebLogic de dos capas.

Sintaxis de dbping

```
$ java -Dbea.home=WebLogicHome utils.dbping DBMS user password DB
```

Argumento	Definición
WebLogicHome	El directorio que contiene nuestra instalación de WebLogic Server. Por ejemplo `d:\beaHome\wlserver6.1`. Requerido sólo si usamos un driver JDBC suministrado por BEA.
DBMS	Elegimos uno de los siguientes para nuestro driver JDBC: • WebLogic jDriver para Microsoft SQL Server: **MSSQLSERVER4** • WebLogic jDriver para Oracle: **ORACLE** • WebLogic jDriver para Informix: **INFORMIX4** • Oracle Thin Driver: **ORACLE_THIN** • Sybase JConnect driver: **JCONNECT**
user	Un nombre válido de usuario para login. Usamos los mismos valores que usamos con **isql** o **sqlplus**.
password	Una password válida para el usuario. Usamos los mismos valores que usamos con **isql** o **sqlplus**.
DB	Nombre de la base de datos. Usamos los siguientes formatos, dependieno del driver JDBC que usemos: • WebLogic jDriver para Microsoft SQL Server: **DBNAME@HOST:PORT** • WebLogic jDriver para Oracle: **DBNAME** • WebLogic jDriver para Informix: **DBNAME@HOST:PORT** • Oracle Thin Driver: **HOST:PORT:DBNAME** • Sybase JConnect driver: **JCONNECT:HOST:PORT:DBNAME** donde: o HOST es el nombre de la máquina que hospeda la DBMS. o PORT es el puerto donde el host de la base de datos escucha conexiones, y o DBNAME es el nombre de la base de datos en el DBMS.

deploy

La utilidad **deploy** obtiene una aplicación J2EE desde un fichero de archivo ((.jar, .war, o .ear) y despliega la aplicación J2EE en un Servidor WebLogic en ejecución.

Sintaxis de Deploy

```
$ java weblogic.deploy [options] [list|deploy|undeploy|update]
password {application} {source}
```

Argumentos

Argumento	Definición
applications	Obligatorio. Identifica el nombre de la aplicación. Este nombre puede especificarse en el momento del despliegue, con las utilidades de despliegue o con la consola.
deploy	Opcional. Despliega un fichero de aplicación J2EE (.jar, .war, o .ear) en el servidor especificado.
list	Opcional. Lista todas las aplicaciones desplegadas en el Servidor WebLogic Especificado.
password	Obligatorio. Especifica la password del usuario **system** del Servidor WebLogic
source	Obligatorio. Especifica la localización exacta del fichero de archivo de la aplicación (.jar, .war, o .ear) , o el path al directorio de más alto nivel de la aplicación.
undeploy	Opcional. Elimina una aplicación existente del servidor especificado.
update	Opcional. Re-despliega una aplicación en el servidor especificado.

Opciones de deploy

Opción	Definición
`-component componentname:target1,target2`	Los componentes a desplegar en varios destinos, deben especificarse como: `componentname:target1,target2` donde `componentname` es el nombre del fichero .jar o .war sin la extensión. Esta opción puede especificarse varias veces para cualquier número de componentes (.jar o .war). Un fichero .ear no puede ser desplegado. Cada uno de sus componentes debe ser desplegado separadamente usando esta opción.
`-debug`	Imprime información de depuración detallada en **stdout** durante el proceso de despliegue.
`-help`	Imprime una lista de todas las opciones disponibles en la utilidad **deploy**.
`-host host`	Especifica el nombre de host del servidor WebLogic a usar para desplegar la aplicación J2EE. Si no especificamos esta opción, la utilidad intenta conectar usando el nombre de host **localhost**.
`-port port`	Especifica el número de puerto del servidor WebLogic usado para desplegar la aplicación J2EE. Si no especificamos la opción `-port`, **deploy** se contacta usando el puerto por defecto 7001.
`-url url`	Especifica la URL de un servidor WebLogic. el valor por defecto es `localhost:7001`.
`-username username`	Nombre del usuario con el que se hará la conexión. El valor por defecto es `system`.
`-version`	Imprime la versión de la utilidad **deploy**

Ejemplos de deploy

Ver una Aplicación J2EE Desplegada
Para ver una aplicación que está desplegada en un Servidor WebLogic local, introducimos el siguiente comando:

```
% java weblogic.deploy list password
```

El valor de `password` es la password de la cuenta `system` en el servidor WebLogic.
Para listar una aplicación desplegada en un servidor remoto, especificamos las opciones `port` y `host`, de esta forma:

```
% java weblogic.deploy -port port_number -host host_name list password
```

Desplegar una Nueva Aplicación J2EE
Para desplegar un fichero de aplicación J2EE (.jar, .war, o .ear) o un directorio de aplicación que no está desplegado en el Servidor WebLogic, introducimos el siguiente comando:

```
% java weblogic.deploy -port port_number -host host_name
    deploy password application source
```

Los valores son los siguientes:

- `application` es el string que queremos asignar a esta Aplicación.
- `source` es el path completo del fichero de la aplicación J2EE (.jar,.war,.ear) que queremos desplegar, o el path completo del directorio de la aplicación.

Por ejemplo:

```
% java weblogic.deploy -port 7001 -host localhost deploy weblogicpwd
Basic_example
    c:\mysamples\ejb\basic\BasicStatefulTraderBean.jar
```

Nota:
El fichero de aplicación J2EE (.jar,.war,.ear) copiado al directorio de aplicaciones del Servidor de Administración es renombrado con el nombre de la aplicación. Por lo tanto, en el ejemplo anterior, el nombre del directorio de archivo de aplicación. `. ./config/mydomain/applications` *se cambia de* ***BasicStatefulTraderBean.jar*** *a* ***Basic_example.jar****.*

Eliminar un Aplicación J2EE Desplegada.
Para eliminar una aplicación J2EE desplegada, sólo necesitamos la referencia asignada al nombre de la aplicación, como se muestra en el siguiente ejemplo:

```
% java weblogic.deploy -port 7001 -host localhost undeploy
    weblogicpwd Basic_example
```

Nota:
Eliminar una aplicación J2EE no elimina la aplicación del Servidor WebLogic. No podemos re-utilizar el nombre de la aplicación con la utilidad ***deploy****. Podemos usarlo para actualizar el despliegue como se describe en la siguiente sección.*

Actualizar una Aplicación J2EE Desplegada
Para actualizar una aplicación J2EE, usando el argumento `update` y especificamos el nombre de la aplicación J2EE activa, de esta forma:

```
% java weblogic.deploy -port 7001 -host localhost update
    weblogicpwd Basic_example
    c:\updatesample\ejb\basic\BasicStatefulTraderBean.jar
```

Para actualizar un componente específico en uno o más servidores, introducimos el siguiente comando:

```
% java weblogic.deploy -port 7001 -host localhost –component
    BasicStatefulTraderBean.jar:sampleserver,exampleserver update
    weblogicpwd Basic_example
    c:\updatesample\ejb\basic\BasicStatefulTraderBean.jar
```

getProperty

La utilidad nos ofrece detalles sobre nuestra configuración Java y nuestro sistema. No tiene argumentos.

Sintaxis de getProperty

```
$ java utils.getProperty
```

Ejemplo

```
$ 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

La utilidad **logToZip** busca un fichero de log de servidor HTTP en un formato de log común, encuentra las clases Java cargadas en él por el servidor, y crea un fichero .zip descomprimido que contiene dichas clases Java. Se ejecuta desde el directorio raíz del documento de nuestro servidor HTTP.
Para usar esta utilidad, debemos tener acceso a los ficheros de logs creados por el servidor HTTP.

Sintaxis de logToZip

```
$ java utils.logToZip logfile codebase zipfile
```

Argumento	Definición
`logfile`	Obligatorio. El path totalmente cualificado del fichero log.
`codebase`	Obligatorio. El code base para el applet, o "" si no hay code base. Concatenando el code base con el nombre completo del paquete del applet, obtenemos el path completo al applet (en relación al documento raíz HTTP).
zipfile	Obligatorio. Nombre del fichero .zip a crear. El fichero zip resultante se crea en el directorio en el que ejecutamos el programa, El path para el fichero especificado puede ser relativo o absoluto. En los ejemplos, se da un path relativo, por eso el fichero zip se crea en el directorio actual.

Ejemplos de logToZip

El siguiente ejemplo muestra como se crea un fichero zip para un applet que reside en el propio documento raíz, es decir, sin code base:

```
$ cd /HTTP/Serv/docs
$ java utils.logToZip /HTTP/Serv/logs/access "" app2.zip
```

El siguiente ejemplo muestra cómo se crea un fichero zip para un applet que reside en un subdirectorio del documento raíz:

```
C:\>cd \HTTP\Serv
C:\HTTP\Serv>java utils.logToZip \logs\applets\classes app3.zip
```

MulticastTest

La utilidad **MulticastTest** nos ayuda a depurar problemas cuando configuramos un cluster WebLogic. La utilidad envía paquetes `multicast` y devuelve información sobre la efectividad del trabajo `multicast` en nuestra red. Específicamente, **MulticastTest** muestra los siguientes tipos de información en la salida estándar:

1. Una confirmación y un ID de secuencia por cada mensaje enviado por este servidor.
2. La secuencia y el ID del remitente para el mensaje recibido desde cualquier servidor del cluster, incluyendo este servidor.
3. Un aviso de secuencia perdida cuando se recibe un mensaje fuera de secuencia.
4. Un aviso de mensaje perdido cuando no se recibe un mensaje esperado.

Para usar `MulticastTest`, arrancamos una copia de la utilidad en cada nodo sobre el que queramos comprobar el tráfico multicast.

Aviso:
No debemos ejecutar la utilidad **MulticastTest** especificando la misma dirección **multicast** (el parámetro -a) que el cluster WebLogic que se está ejecutando actualmente. La utilidad está pensada para verificar que el multicast funciona correctamente antes de arrancar los Servidores WebLogic del Cluster.

Sintaxis de MulticastTest

```
$ java utils.MulticastTest -n name -a address [-p portnumber]
[-t timeout] [-s send]
```

Argumento	Definición
`-n name`	Obligatorio. Un nombre que identifica el remitente de los mensajes secuenciados. Usamos un nombre diferente por cada proceso de prueba que arranquemos.
`-a address`	Obligatorio. La dirección multicast sobre la que: (a) se deberían enviar los mensajes secuenciados, y (b) los servidores del cluster se están comunicando unos con otros. (El valor por defecto para cualquier cluster cuya dirección multicast no esté seleccionada, es 237.0.0.1.)
`-p portnumber`	Opcional. El puerto multicast por el que están comunicando todos los servidores del cluster. (El puerto multicast es el mismo puerto de escucha que el servidor WebLogic, que por defecto es 7001 si no se selecciona).
`-t timeout`	Opcional. El tiempo muerto, en segundos, si no se reciben mensajes multicast. Si no se selecciona, el valor por defecto es 600 segundos (10 minutos). Si se excede este tiempo, se envía una confirmación positiva de timeout a `stdout`.
`-s send`	Opcional. Intervalo, en segundos, entre envíos. Si no se selecciona, el valor por defecto es 2 segundos. Se envía una confirmación positiva de que se ha enviado cada mensaje a `stdout`.

Ejemplo de MulticastTest

```
$ 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

La utilidad **myip** devuelve la dirección IP del host.

Sintaxis de myip

```
$ java utils.myip
```

Ejemplo de myip

```
$ java utils.myip
Host toyboat.toybox.com is assigned IP address: 192.0.0.1
```

pem2der

La utilidad **pem2der** convierte un certificado X509 del formato PEM al formato DER. El fichero .der se escribe en el mismo directorio que el fichero .pem fuente.

Sintaxis de pem2per

```
$ java utils.pem2der pemFile
```

Argumento	Definición
`pemFile`	El nombre del fichero a convertir. El nombre del fichero debe terminar en una extensión **.pem**, y debe contener un certificado válido en formado PEM.

Ejemplo de pem2der

```
$ java utils.pem2der graceland_org.pem
Decoding
.....................................................................
.....................................................................
.....................................................................
.....................................................................
.....................................................................
```

Schema

La utilidad **Schema** nos permite cargar sentencias SQL en una base de datos usando los drivers JDBC de WebLogic.

Sintaxis de Schema

```
$ java utils.Schema driverURL driverClass [-u username]
[-p password] [-verbose SQLfile]
```

Argumento	Definición
`driverURL`	Obligatorio. La URL para la clase del driver JDBC.
`driverClass`	Obligatorio. El path para la clase del driver JDBC.
`-u username`	Opcional. Nombre de usuario válido.
`-p password`	Opcional. La password válida del usuario.
`-verbose`	Opcional. Imprime las sentencias SQL y los mensajes de la base de datos.
`SQLfile`	Obligatorio cuando se usa el argumento `-verbose`. Fichero de texto con sentencias SQL.

Ejemplo de Schema

El siguiente código muestra un ejemplo de línea de comandos para **Schema**:

```
$ java utils.Schema "jdbc:cloudscape:demo;create=true"
   COM.cloudscape.core.JDBCDriver
   -verbose examples/utils/ddl/demo.ddl
```

El siguiente código muestra un ejemplo de fichero **.ddl**:

```
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

La utilidad **showLicenses** muestra información sobre los productos BEA instalados en esta máquina.

Sintaxis de showLicenses

```
$ java utils.showLicenses
```

system

La utilidad **system** muestra información básica sobre el entorno operativo de nuestro ordenador, incluyendo el fabricante y versión de nuestro JDK, nuestro CLASSPATH, y detalles sobre el sistema operativo

Sintaxis de system

```
$ java utils.system
```

Ejemplo de system

```
$ 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

La utilidad **t3dbping** prueba una conexión JDBC WebLogic a una DBMS mediante un driver JDBC de dos capas. Debemos tener acceso al servidor WebLogic y a la DBMS para usar esta utilidad.

Sintaxis de t2dbping

```
$ java utils.t3dbping WebLogicURL username password DBMS
   driverClass driverURL
```

Argumento	Definición
`WebLogicURL`	Obligatorio. La URL del Servidor WebLogic.
`username`	Obligatorio. Nombre de usuario válido para la DBMS.
`password`	Obligatorio. Password válida para el usuario de la DBMS.
`DBMS`	Obligatorio. Nombre de la base de datos.
`driverClass`	Obligatorio. Nombre de paquete completo del driver de dos capas del Servidor WebLogic.
`driverURL`	Obligatorio. URL del driver de dos capas del Servidor WebLogic.

verboseToZip

Cuando se ejecuta desde el directorio el documento raíz de nuestro servidor HTP, **verboseToZip** toma la salida estándar de una aplicación Java que se está ejecutando en modo `verbose`, encuentra las clases Java referenciadas, y crea un fichero zip descomprimido que contiene dichas clases.

Sintaxis de verboseToZip

```
$ java utils.verboseToZip inputFile zipFileToCreate
```

Argumento	Definición
`inputFile`	Obligatorio. Fichero temporal que contiene la salida de la aplicación que se ejecuta en modo `verbose`.
`zipFileToCreate`	Obligatorio. Nombre del fichero zip a crear. Este fichero se crea en el directorio en el que ejecutamos la aplicación.

Ejemplo UNIX de verboseToZip

```
$ java -verbose myapplication > & classList.tmp
$ java utils.verboseToZip classList.tmp app2.zip
```

Ejemplo NT de verboseToZip

```
$ java -verbose myapplication > classList.tmp
$ java utils.verboseToZip classList.tmp app3.zip
```

version

La utilidad **version** muestra información de versión sobre nuestro servidor WebLogic instalado por `stdout`.

Sintaxis de version

```
$ java weblogic.version
```

Ejemplo de version

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

writeLicense

La utilidad **writeLicense** escribe información sobre las licencias WebLogic en un fichero llamado **writeLicense.txt**, localizado en el directorio actual. Este fichero puede ser enviado por e-mail, por ejemplo, al soporte técnico de WebLogic.

Sintaxis de writeLicense

```
$ java utils.writeLicense -nowrite -Dweblogic.system.home=path
```

Argumento	Definición
`-nowrite`	Obligatorio. Envía la salida a `stdout` en lugar de a `writeLicense.txt`.
`-Dweblogic.system.home`	Obligatorio. Selecciona el home del sistema de WebLogic (el directorio raíz de la instalación de WebLogic Server). Este argumento es necesario, a menos que estemos ejecutando **writeLicense** desde el directorio `home` de nuestro sistema WebLogic.

Ejemplos de writeLicense

```
$ java utils.writeLicense -nowrite
```

Ejemplo de salida UNIX:

```
* * * * * * 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
...
```

Ejemplo de salida Windows NT:

```
* * * * * * * 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
```

Referencia del Interface de Línea de Comandos del Servidor WebLogic

Sobre el Interface de Línea de Comandos

Como alternativa a la Consola de Administración, WebLogic Server ofrece una interfase de línea de comandos a sus herramientas de administración, así como muchas propiedades de configuración **Mbean** de tiempo de ejecución.
Usamos la interfase de la línea de comandos si:

- Queremos crear scripts para una administración y control más eficiente.
- No podemos acceder a la Consola de Administración a través de un navegador.
- Preferimos usar la interfase de la línea de comandos mejor que la interfase gráfica de usuario.

Antes de Empezar

Los ejemplos de este documento están basados en las siguientes suposiciones:

- WebLogic Server está instalado en el directorio **c:/weblogic**.
- El JDK está alcalizado en el directorio **c:/java**.
- Hemos arrancado WebLogic Server desde el directorio en el que fue instalado.

Antes de poder ejecutar los comandos de WebLogic Server, debemos hacer lo siguiente:

1. Instalar y configurar el software WebLogic Server, como se describe en Guía de Instalación de WebLogic Server.
2. Seleccionar el CLASSPATH correctamente.
3. Activar la interfase de la línea de comandos realizando uno de los siguientes pasos:
    - Arrancar el servidor desde el directorio en el que fue instalado.
    - Si no hemos arrancado el servidor desde su directorio de instalación, introducir el siguiente comando, reemplazando `c:/weblogic` con el nombre del directorio en que está instalado el Software WebLogic Server:
    - `-Dweblogic.system.home=c:/weblogic`

Usar Comandos del Servidor WebLogic

Esta sección presenta la Sintaxis y los argumentos necesarios para usar los comandos del Servidor WebLogic. Estos comandos no son sensibles a las mayúsculas.

Sintaxis

```
java weblogic.Admin [–url URL] [-username username]
[-password password] COMMAND arguments
```

Argumento

Los siguientes argumentos son necesarios para muchos de los comandos del Servidor WebLogic.

Argumento	Definición
URL	La URL del host del Servidor WebLogic, incluyendo el número de puerto TCP en el que el servidor escucha peticiones de clientes. El formato es `hostname: port`. El valor por defecto es **localhost: 7001**. **Nota:** La URL usada con un comando de servidor siempre se refiere al Servidor WebLogic, mientras que la URL usada con los comandos de tiempo de ejecución y los comandos de configuración **Mbean** siempre se refieren a un Servidor de Administración específico.
`username`	Opcional. Nombre de usuario a ser autentificado para que los comandos puedan ser ejecutados. El valor por defecto es **guest**.
password	Opcional. La password para autentificar para que los comandos puedan ser ejecutados. El valor por defecto es **guest**.

Un administrador debe tener los permisos de control de acceso apropiados para ejecutar los comandos usados para controlar los **Mbeans** en tiempo de ejecución.

Referencia de Comandos de Administración del Servidor WebLogic

CANCEL_SHUTDOWN

El comando **CANCEL_SHUTDOWN** cancela el comando **SHUTDOWN** para un Servidor WebLogic especificado.
Cuando usamos el comando **SHUTDOWN**, podemos especificar un retardo (en segundos).
Un administrador podría cancelar el comando `shutdown` durante el tiempo de retardo. Hay que tener cuidado ya que el comando **SHUTDOWN** desactiva los loggins, y permanecen desactivados incluso después de cancelar el shutdown. Usamos el comando **UNLOCK** para re-activar los loggins.
Puedes ver los comandos SHUTDOWN y UNLOCK
Sintaxis

```
java weblogic.Admin [-url URL] [-username username]
    [-password password] CANCEL_SHUTDOWN
```

Ejemplo
En el siguiente ejemplo, un usuario del sistema llamado `system` con la password `gumby1234` pide que se cancela el `shutdown` del servidor WebLogic que está escuchando el puerto 7001 de la máquina `localhost`:

```
java weblogic.Admin -url t3://localhost:7001 -username system
    -password gumby1234 CANCEL_SHUTDOWN
```

CONNECT

Hace el número de conexiones especificadas con el Servidor WebLogic y devuelve dos números representando el tiempo total empleado en cada viaje y la media de tiempo (en milisegundos) que se mantiene cada conexión.
Sintaxis

```
java weblogic.Admin [-url URL] [-username username]
    [-password password] CONNECT count
```

Argumento	Definición
count	Número de conexiones a realizar.

Ejemplo
En el siguiente ejemplo, un usuario con el nombre `adminuser` y la password `gumby1234` ejecuta el comando **CONNECT** para establecer 25 conexiones a un servidor llamado `localhost` y devuelve información sobre dichas conexiones:

```
java weblogic.Admin -url localhost:7001 -username adminuser
    -password gumby1234 CONNECT 25
```

HELP

Proporciona la Sintaxis e información de utilización de todos los comandos del Servidor WebLogic (por defecto) o para un sólo comando si se especifica un comando en la línea de comandos de **HELP**.
Sintaxis

```
java weblogic.Admin HELP [COMMAND]
```

Ejemplo
En el siguiente ejemplo, se pide información sobre el comando PING:

```
java weblogic.Admin HELP PING
```

El comando **HELP** devuelve la siguiente salida en `stdout`:

```
Usage: weblogic.Admin [-url url] [-username username]
    [-password password] <COMMAND> <ARGUMENTS>
    PING <count> <bytes>
```

LICENSES

Lista las licencias de todos los ejemplares de WebLogic Server instaladas en el servidor especificado.
Sintaxis:

```
java weblogic.Admin [-url URL] [-username username]
    [-password password] LICENSES
```

Ejemplo
En el siguiente ejemplo, un administrador usando el nombre de usuario por defecto (`guest`) y la password por defecto (`guest`) pide información sobre las licencias de un Servidor WebLogic que se está ejecutando en el puerto 7001 de la máquina `localhost`:

```
java weblogic.Admin -url localhost:7001 -username guest
    -password guest LICENSES
```

LIST

Lista las uniones de un nodo en el árbol de nombres JNDI.
Sintaxis:

```
java weblogic.Admin [-username username] [-password password]
    LIST context
```

Argumento Definición

context	Obligatorio. El contexto JNDI para la búsqueda, por ejemplo weblogic, weblogic.ejb, javax.

Ejemplo
En este ejemplo, el usuario `adminuser`, con la password `gumby1234`, pide una lista de uniones de nodos en `weblogic.ejb`:

```
java weblogic.Admin -username adminuser -password gumby1234
    LIST weblogic.ejb
```

LOCK

Bloquea un Servidor WebLogic contra loggins no-privilegiados. Cualquier intento posterior de login inicia una excepción de seguridad que podría contener un string como mensaje.

Nota:
Este comando es privilegiado. Requiere la password para el usuario administrador del Servidor WebLogic.

Sintaxis

```
java weblogic.Admin [-url URL] [-username username]
    [-password password] LOCK “string_message”
```

Argumento Definición

“string_message”	Opcional. Mensaje, entre comillas, para ser suministrado en la excepción de seguridad que se lanza si un usuario no-privilegiado intenta hacer un login mientras el servidor WebLogic está bloqueado.

Ejemplo
En el siguiente ejemplo se bloquea un Servidor WebLogic:

```
java weblogic.Admin -url localhost:7001 -username adminuser
    -password gumby1234
    LOCK "Sorry, WebLogic Server is temporarily out of service."
```

Cualquier aplicación que intente entrar en el servidor bloqueado con un nombre de usuario no privilegiado recibirá el mensaje especificado: `Sorry, WebLogic Server is temporarily out of service`.

PING

Envía un mensaje para verificar que un Servidor WebLogic está escuchando en un puerto, y que está listo para aceptar peticiones de clientes.
Sintaxis

```
java weblogic.Admin [-url URL] [-username username]
    [-password password] PING [round_trips] [message_length]
```

Argumento	Definición
round_trips	Opcional. Número de pings.
message_length	Opcional. Tamaño de los paquetes a enviar en cada ping. Las peticiones de pings con paquetes mayores de 10MB lanzan excepciones.

Ejemplo
En el siguiente ejemplo, el comando chequea un servidor WebLogic que se está ejecutando en el puerto 7001 de la máquina `localhost` diez (10) veces:

```
java weblogic.Admin -url localhost:7001 -username adminuser
    -password gumby1234 PING 10
```

SERVERLOG

Muestra el fichero log generado sobre un servidor específico.

- Si no especificamos una URL, se muestra por defecto el log del Servidor de Administración.
- Si especificamos una URL de servidor, podemos recuperar un log de un Servidor que no sea de Administración.
- Si omitimos los argumentos `starttime` y `endtime`, se empieza a mostrar todo el contenido del log del servidor desde el inicio del comando.

Sintaxis

```
java.weblogic.Admin [-url URL] [-username username]
    [-password password] SERVERLOG [[starttime]|[endtime]]
```

Argumento	Definición
starttime	Opcional. Primer momento en que se empiezan a mostrar los mensajes. Si no se especifica, los mensajes empiezan a mostrarse, por defecto cuando se ejecuta el comando **SERVERLOG**. El formato de la fecha es `yyyy/mm/dd`. La hora se indica usando un reloj de 24-horas. La fecha y hora de inicio deben introducirse entre comillas, en el siguiente formato: "`yyyy/mm/dd hh:mm`".
endtime	Opcional. Hora del último mensaje a mostrar. Si no se especifica, el valor por defecto es cuando se ejecuta el comando **SERVERLOG**. El formato de la fecha es `yyyy/mm/dd`. La hora se indica usando un reloj de 24-horas. La fecha y hora de inicio deben introducirse entre comillas, en el siguiente formato: "`yyyy/mm/dd hh:mm`".

Ejemplo
En el siguiente ejemplo, se hace una petición para ver el log del servidor que está escuchando en el puerto 7001 de la máquina `localhost`.

```
java weblogic.Admin -url localhost:7001
   SERVERLOG “2001/12/01 14:00” “2001/12/01 16:00”
```

SHUTDOWN

Apaga el Servidor WebLogic que se especifica en la URL.
Sintaxis

```
java weblogic.Admin [-url URL] [-username username]
    [-password password] SHUTDOWN [seconds] [“lockMessage”]
```

Argumento	Definición
seconds	Opcional. Número de segundos permitidos para esperar entre la invocación de este comando y el cierre del servidor.
“lockMessage”	Opcional. Mensaje, entre comillas, a suministrar en el mensaje enviado a los usuarios que están intentando entrar mientras el Servidor WebLogic está bloqueado.

Ejemplo
En el siguiente ejemplo, un usuario con el nombre de usuario `username` y la password administrativa `gumby1234` cierra un servidor WebLogic que está escuchando en el puerto 7001 de la máquina `localhost`:

```
java weblogic.Admin -url localhost:7001 -username adminuser
    -password gumby1234 SHUTDOWN 300 “Server localhost is shutting
down.”
```

Después de haber enviado el comando, se espera un intervalo de cinco minutos (300 segundos). Luego el comando cierra el servidor especificado y envía el siguiente mensaje a `stdout`:

```
Server localhost is shutting down.
```

THREAD_DUMP

Proporciona una imagen en tiempo real de los threads que se están ejecutando concurrentemente en el Servidor WebLogic.
Sintaxis

```
java weblogic.Admin [-url URL] [-username username]
    [-password password] THREAD_DUMP
```

UNLOCK

Desbloquea el Servidor WebLogic especificado después de una operación LOCK.
Sintaxis

```
java weblogic.Admin [-url URL] [-username username]
    [-password password] UNLOCK
```

Argumento	Definición
username	Obligatorio. Se debe suministrar un nombre de usuario administrador válido para usar este comando.
password	Obligatorio. Se debe suministrar una password de usuario administrador válido para usar este comando.

Ejemplo
En el siguiente ejemplo, un administrador llamado `adminuser` con la password `gumby1234` pide que se desbloquee el servidor WebLogic que está escuchando en el puerto 7001 de la máquina `localhost`:

```
java weblogic.Admin -url localhost:7001 -username adminuser
    -password gumby1234 UNLOCK
```

VERSION

Muestra la versión del Software de WebLogic Server que está ejecutando en la máquina especificada por la URL.
Sintaxis

```
java weblogic.Admin -url URL -username username
    -password password VERSION
```

Ejemplo
En el siguiente ejemplo, un usuario pide la versión del Servidor WebLogic que se está ejecutando en el puerto 7001 de la máquina `localhost`:

```
java weblogic.Admin -url localhost:7001 -username guest
    -password guest VERSION
```

Nota:
En este ejemplo, el valor por defecto para los argumentos `username` *y* `password` *es* `guest`.

Referencia de Comandos para la Administración de Almacénes de Conexiones

CREATE_POOL

Nos permite la creación de un almacén de conexiones mientras se está ejecutando el Servidor WebLogic.
Sintaxis

```
java weblogic.Admin [-url URL] [-username username]
    [-password password] CREATE_POOL poolName aclName=aclX,
    props=myProps,initialCapacity=1,maxCapacity=1,
    capacityIncrement=1,allowShrinking=true,shrinkPeriodMins=15,
    driver=myDriver,url=myURL
```

Argumento	Definición
poolName	Obligatorio, Nombre único para el almacén.
aclName	Obligatorio. Identifica las diferentes listas de acceso dentro de **fileRealm.properties** en el directorio **config** del servidor. El nombre emparejado debe ser dynaPool.
props	Propiedades de la conexión a la base de datos; normalmente en el formato "database login name; database password; server network id".
initialCapacity	Número inicial de conexiones del almacén. Si esta propiedad se define como un número positivo mayor que cero, el Servidor WebLogic crea esas conexiones en el momento del arranque. Por defecto, es 1; no puede exceder el valor de `maxCapacity`.
maxCapacity	Número máximo de conexiones permitidas en el almacén. El valor por defecto es 1; si se define debe ser mayor o igual que 1.
capacityIncrement	Número de conexiones que se pueden añadir de una vez. El valor por defecto es 1.
allowShrinking	Indica si el almacén puede reducir las conexiones que se ha detectado que no están en uso. Por defecto es true.
shrinkPeriodMins	Obligatorio. Intervalo entre reducciones. Unidades en minutos. El valor mínimo es 1. Si `allowShrinking = True`, entonces el valor por defecto es 15 minutos.
driver	Obligatorio. el nombre del driver JDBC. Sólo pueden participar drivers locales (no-XA).
url	Obligatorio. La URL del driver JDBC.
testConnsOnReserve	Indica las conexiones de prueba reservadas. Por defecto false.
testConnsOnRelease	Indica las conexiones de prueba que son liberadas. Por defecto false.
testTableName	Tabla de la base de datos usada cuando se prueban las conexiones, debe estar presente para que la prueba tenga éxito. Requerido si se han definido `testConnOnReserve` o `testConOnRelease`.
refreshPeriod	Selecciona el intervalo de refresco de conexiones. Toda conexión no usada será comprobada usando `TestTableName`. Las conexiones que no pasen el text serán cerradas y reabiertas en un intento de re-establecer una conexión físicamente válida con la base de datos. Si no se selecciona `TestTableName` no se realizará el test.
`loginDelaySecs`	El número de segundos de retardo antes de la creación de la conexión física con la base de datos. Este retardo tiene lugar tanto durante la creación inicial del almacén como durante su tiempo de vida si crea una nueva conexión. Algunos servidores de bases de datos no pueden manejar múltiples peticiones de conexiones en una sucesión rápida. Esta propiedad nos permite construir un pequeño retardo para dejar que la base de datos los capture.

Ejemplo
En el siguiente ejemplo, un usuario con el nombre `adminuser` y la password `gumby1234` ejecuta **CREATE_POOL** para crear dinámicamente un almacén de conexiones:

```
java weblogic.Admin -url localhost:7001 -username adminuser
    -password gumby1234 CREATE_POOL myPool

java weblogic.Admin -url t3://forest:7901 -username system
    -password gumby1234 CREATE_POOL dynapool6 "aclName=someAcl,
    allowShrinking=true,shrinkPeriodMins=10,
    url=jdbc:weblogic:oracle,driver=weblogic.jdbc.oci.Driver,
    initialCapacity=2,maxCapacity=8,
    props=user=SCOTT;password=tiger;server=bay816"
```

DESTROY_POOL

Las conexiones son cerradas y eliminadas del almacén y el almacén muere cuando no le quedan conexiones. Sólo el usuario "`system`"o los usuarios con permisos concedidos como "`admin`" en la ACL asociada con el almacén de conexiones pueden destruir el almacén.
Sintaxis

```
java weblogic.Admin [-url URL] [-username username]
    [-password password] DESTROY_POOL poolName [true|false]
```

Argumento	Definición
poolName	Obligatorio. Nombre único del almacén
false (soft shutdown)	Shutdown blando esperando que las conexiones sean devueltas al almacén antes de cerrarlas.
true (por defecto—hard shutdown)	Shutdown duro, mata todas las conexiones inmediatamente. Los clientes que están usando conexiones del almacén obtienen excepciones si intentan usarla después de que se haya cerrado

Ejemplo
En el siguiente ejemplo, un usuario con el nombre `adminuser` y la password `gumby1234` ejecuta el comando **DESTROY_POOL** para congelar temporalmente el almacén de conexiones activo:

```
java weblogic.Admin -url localhost:7001 -username adminuser
    -password gumby1234 DESTROY_POOL myPool false
```

DISABLE_POOL

Podemos desactivar temporalmente un almacén de conexiones, evitando que cualquier cliente obtenga una conexión desde el almacén. Sólo el usuario "`system`"o los usuarios con permisos concedidos como "`admin`" en la ACL asociada con el almacén de conexiones pueden activar o desactivar el almacén.
Tenemos dos opciones para desactivar las conexiones. 1) Congelando las conexiones de un almacén de conexiones que luego hemos planeado volver a activar, y 2) destruir las conexiones.
Sintaxis

```
java weblogic.Admin [-url URL] [-username username]
    [-password password] DISABLE_POOL poolName [true|false]
```

Argumento	Definición
poolName	Nombre del almacén de conexiones.
false (desactiva y suspende)	Desactiva el almacén de conexiones, y suspende a los clientes que actualmente tienen una conexión. Los intentos de comunicación con la base de datos lanzarán una excepción. Sin embargo, los clientes pueden cerrar sus conexiones mientras el almacén está desactivado; las conexiones son devueltas al almacén y no puede ser reservada por otro cliente hasta que se active de nuevo el almacén.
true (Por defecto—desactiva y destruye)	Desactiva el almacén de conexiones, y destruye las conexiones JDBC de los clientes. Cualquier transacción sobre la conexión es deshecha y la conexión se devuelve al almacén de conexiones.

Ejemplo
En el siguiente ejemplo, un usuario con el nombre `adminuser` y la password `gumby1234` ejecuta el comando **DISABLE_POOL** para congelar temporalmente el almacén de conexiones que se va a activar más tarde:

```
java weblogic.Admin -url localhost:7001 -username adminuser
    -password gumby1234 DISABLE_POOL myPool false
```

ENABLE_POOL

Cuando se activa un almacén, los estados de las conexiones JDBC por cada conexión en uso son exactamente los mismos que cuando se desactivó la conexión, los clientes pueden continuar sus operaciones JDBC exactamente donde las dejaron.
Sintaxis

```
java weblogic.Admin [-url URL] [-username username]
    [-password password] ENABLE_POOL poolName
```

Argumento	Definición
poolName	Obligatorio. Nombre único del almacén

Ejemplo
En el siguiente ejemplo, un usuario con el nombre `adminuser` y la password `gumby1234` ejecuta el comando **ENABLE_POOL** para re-establecer las conexiones que se han sido congeladas:

```
java weblogic.Admin -url localhost:7001 -username adminuser
-password gumby1234 ENABLE_POOL myPool
```

EXISTS_POOL

Comprueba si existe un almacén de conexiones con el nombre especificado en el Servidor WebLogic.
Podemos usar este método para determinar si un almacén de conexiones dinámico ya está creado para asegurarnos de que seleccionamos un nombre único para el almacén de conexiones que queremos crear.
Sintaxis

```
java weblogic.Admin [-url URL] [-username username]
    [-password password] EXISTS_POOL poolName
```

Argumento	Definición
poolName	Obligatorio. Nombre único del almacén

Ejemplo
En el siguiente ejemplo, un usuario con el nombre `adminuser` y la password `gumby1234` ejecuta el comando **EXISTS_POOL** para determina si existe o no un almacén con el nombre especificado:

```
java weblogic.Admin -url localhost:7001 -username adminuser
    -password gumby1234 EXISTS_POOL myPool
```

RESET_POOL

Este comando resetea las conexiones de un almacén de conexiones registradas.
Este es un comando privilegiado. Debemos suministrar la password del administrativo del Servidor WebLogic para usar este comando. Debemos conocer el nombre del almacén de conexiones, que es una entrada en el fichero **config.xml**.
Sintaxis

```
java weblogic.Admin URL RESET_POOL poolName system password
```

Argumento	Definición
URL	La URL del host del Servidor WebLogic y el número de puerto TCP en el que está escuchando peticiones de clientes; por ejemplo "`t3://host:port.`"
poolName	Obligatorio. Nombre del almacén registrado en el fichero **config.xml** del Servidor WebLogic.
password	La password administrativa del usuario "`system`". Debemos suministrar el nombre de usuario "`system`" y su password administrativa para usar este comando.

Ejemplo
Este comando refresca el almacén de conexiones registrado como "eng" en el Servidor WebLogic que está escuchando en el puerto 7001 del host `xyz.com`;

```
java weblogic.Admin t3://xyz.com:7001 RESET_POOL eng system gumby
```

Referencia de Comandos de Control de Mbean

CREATE

Crea y ejemplariza un **Mbean** de configuración. Devuelve OK en `stdout` cuando tiene éxito.
Este comando no puede usarse para Mbeans de tiempo de ejecución. El ejemplar de Mbean se graba en el fichero **config.xml** o en el reino de seguridad, dependiendo de los cambios que hayamos hecho.

> **Nota:**
> Cuando creamos Mbeans, también se crean los objetos de configuración.

Sintaxis

```
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
```

Argumento	Definición
name	Obligatorio. El nombre que hemos elegido para el Mbean que estamos creando.
mbean_type	Obligatorio. Cuando se crean atributos para varios objetos del mismo tipo.
mbean_name	Obligatorio. Nombre totalmente cualificado del Mbean, en el siguiente formato: “`domain:Type=type,Name=name`” `Type` especifica una agrupación de objetos y `Name` especifica el nombre del Mbean.
domain_name	Opcional. Nombre del dominio, por ejemplo, `mydomain`. Si no se especifica, se utiliza el valor de dominio por defecto.

Ejemplo

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

DELETE

Borra un `Mbean` de configuración. Devuelve OK en `stdout` cuando tiene éxito.
Este comando no puede usarse para Mbeans de tiempo de ejecución.

> ***Nota:***
> *Cuando borramos Mbeans, también se borran los objetos de configuración.*

Sintaxis
```
java weblogic.Admin [–url URL] [-username username] [-password
    password] DELETE {–type mbean_type|–mbean mbean_name}
```

Argumento	Definición
mbean_type	Obligatorio. Cuando se borran atributos para varios objetos del mismo tipo.
mbean_name	Obligatorio. Nombre totalmente cualificado del Mbean, en el siguiente formato: "`domain:Type=type,Name=name`" `Type` especifica una agrupación de objeto y `Name` especifica el nombre del Mbean.

Ejemplo
```
java weblogic.Admin -url localhost:7001 -username adminuser
    -password gumby1234 DELETE -mbean
    "mydomain:Type:Server,Name=AcctServer"
```

GET

Muestra los atributos de los Mbeans de tiempo de ejecución. Podemos solicitar una lista de atributos para varios objetos del mismo tipo solicitando atributos para lo siguiente:

- Todos los Mbeans que pertenecen al mismo tipo de Mbean:
- `GET {-pretty} -type mbean_type`
- Un Mbean específico:
- `GET {-pretty} -mbean mbean_name`

El nombre de cada uno de los Mbean especificados se incluye en la salida. Si se especifica `-pretty`, cada pareja de atributos nombre-valor se muestra en una nueva línea.
El comando GET sólo puede llamar a Mbeans de tiempo de ejecución.
La pareja nombre.valor por cada atributo se especifica entre corchetes. Este formato facilita el scripting simplificando el análisis de la salida.
El nombre del Mbean se incluye en la salida de esta forma:
```
{mbeanname mbean_name {property1 value} {property2 value}. . .}
{mbeanname mbean_name {property1 value} {property2 value} . . .}
. . .
```
Si se especifcia `-pretty`:
```
mbeanname: mbean_name
property1: value
property2: value
.
.
.
mbeanname: mbean_name
property1: value
property2: value
```
Sintaxis
```
java weblogic.Admin [–url URL] [-username username] [-password
    password] GET {-pretty} {–type mbean_type|–mbean mbean_name}
    [–property property1] [–property property2]...
```

Argumento	Definición
mbean_type	Obligatorio. Cuando se borran atributos para varios objetos del mismo tipo.
mbean_name	Obligatorio. Nombre totalmente cualificado del Mbean, en el siguiente formato: "`domain:Type=type,Location:location,Name=name`" `Type` especifica una agrupación de objeto, `Location` especifica la localización del Mbean y `Name` especifica el nombre del Mbean.
pretty	Opcional. Produce una salida bien formateada.
property	Opcional. El nombre del atributo o atributos del Mbean a listar. **Nota:** si no se especifica ningún atributo usando este argumento, se listan todos los atributos.

Ejemplo
En el siguiente ejemplo, un usuario solicita una listado de los atributos Mbean de un servidor llamado `localhost`, que está escuchando en el puerto 7001:

```
java weblogic.Admin -url localhost:7001 GET -pretty -type Server
```

INVOKE

Llama al método apropiado (incluyendo argumentos) del Mbean especificado. Este comando sólo puede llamar a Mbeans de tiempo de ejecución. Usamos este comando para llamar a métodos que no tienen atributos get o set de Mbean.
Sintaxis

```
java weblogic.Admin [–url URL] [-username username] [-password
    password] INVOKE {–type mbean_type|–mbean mbean_name} –method
    methodname [argument . . .]
```

Argumento	Definición
mbean_type	Obligatorio. Cuando se borran atributos para varios objetos del mismo tipo.
mbean_name	Obligatorio. Nombre totalmente cualificado del Mbean, en el siguiente formato: "`domain:Type=type,Location:location,Name=name`" `Type` especifica una agrupación de objeto, `Location` especifica la localiación del Mbean y `Name` especifica el nombre del Mbean.
methodname	Obligatorio. Nombre del método a llamar. Siguiendo al nombre del método, el usuario puede especificar los argumentos pasados a la llamada del método, de esta forma: "`domain:Name=name,Type=type`"

Ejemplo
El siguiente ejemplo llama a un mBean de administración llamado `admin_one` usando el método `getAttributeStringValue`:

```
java weblogic.Admin -username system -password gumby1234 INVOKE
    -mbean mydomain:Name=admin_one,Type=Administrator
    -method getAttributeStringValue PhoneNumber
```

SET

Selecciona los valores de un atributo especificado para el Mbean de configuración nombrado. Devuelve OK por stdout si tiene éxito. Este comando no puede usarse con Mbeans de tiempo de ejecución.
Los nuevos valores se graban en el fichero **config.xml** o en el reino de seguridad, dependiendo de donde hayamos modificados los nuevos valores.
Sintaxis

```
java weblogic.Admin [–url URL] [-username username]
    [-password password] SET {–type mbean_type|–mbean mbean_name}
    –property property1 property1_value
    [-property property2 property2_value] . . .
```

Argumento	Definición
mbean_type	Obligatorio. Cuando se borran atributos para varios objetos del mismo tipo.
mbean_name	Obligatorio. Nombre totalmente cualificado del Mbean, en el siguiente formato: "`domain:Type=type,Location:location,Name=name`" `Type` especifica una agrupación de objeto, `Location` especifica la localiación del Mbean y `Name` especifica el nombre del Mbean.
property	Obligatorio. el nombre del atributo de propiedad a seleccionar.
property _value	Obligatorio. El valor a seleccionar en la propiedad.

Parámetros para Plug-ins de Servidores Web

Introducción

Introducimos los parámetros para cada Plug-in de Servidor Web en ficheros de configuración Especiales. Cada Servidor Web tiene un nombre diferente para su fichero de configuración y diferentes reglas para su formateo. Puedes visitar las siguientes secciones para encontrar más detalles sobre cada Plug-in:

- Instalar y Configurar el Plug-In para Apache HTTP Server
- Instalar y Configurar el Plug-In para Microsoft Internet Information Server (ISAPI)
- Instalar y Configurar el Plug-In para Netscape Enterprise Server (NSAPI)

Parámetros Generales para Plug-Ins de Servidores Web

***Nota:** Los parámetros son sensibles a las mayúsculas:*

Parámetro	Valor por Defecto	Descripción
WebLogicHost (Requerido cuando pasamos [proxy] a un sólo servidor WebLogic).	ninguno	El host WebLogic Server (o nombre de host virtual según se define en un Servidor Web ejecutándose en WebLogic Server) al que se deben reenviar las solicitudes HTTP. Si estamos usando un cluster WebLogic, usamos el parámetro WebLogicCluster en lugar de WebLogicHost.
WebLogicPort (Requerido cuando pasamos [proxy] a un sólo servidor WebLogic).	ninguno	Puerto en el que el host WebLogic Server está escuchando solicitudes de conexiones WebLogic. (Si estamos usando SSL entre el plug-in y el servidor WebLogic, seleccionamos este parámetro al puerto de escucha SSL (ver Configurar el puerto de Escucha) y seleccionamos el parámetro SecureProxy a ON). Si estamos usando un cluster WebLogic, usamos el parámetro WebLogicCluster en lugar de WebLogicPort.
WebLogicCluster (Requerido cuando pasamos [proxy] a un cluster WebLogic).	ninguno	Lista de Servidores WebLogic que pueden usarse en un cluster para balance de carga. El cluster es una lista delimitada por comas de entradas host:port. Por ejemplo: WebLogicCluster myweblogic.com:7001, yourweblogic.com:7001,theirweblogic.com:7001. Si estamos usando SSL entre el plug-in y el servidor WebLogic, seleccionamos este parámetro al puerto de escucha SSL (ver Configurar el puerto de Escucha) y seleccionamos el parámetro SecureProxy a ON. Usamos WebLogicCluster en lugar de los parámetros WebLogicHost y WebLogicPort. WebLogic Server primero busca el parámetro WebLogicCluster. Si no lo encuentra, busca y usa WebLogicHost y WebLogicPort. El plug-in da una simple vuelta entre todos los miembros disponibles del cluster. La lista de cluster especificada en esta propiedad es un punto de entrada para la lista de cluster dinámica que mantienen el servidor y el plug-in. WebLogic Server y el plug-in funcionan juntos para actualizar automáticamente la lista de clusters con los miembros del cluster nuevos, fallados y recuperados. Podemos desactivar el uso de la lista dinámica de cluster seleccionando el parámetro DynamicServerList a OFF (sólo Microsoft Internet Information Server). El plug-in dirige las solicitudes HTTP que contienen un cookie, una sección URL codificada, o una sesión almacenada en los datos POST al servidor del cluster que creo originalmente la cookie.
PathTrim	null	String recortada por el plug-in desde el principio de la URL original, antes de que la solicitud sea re-enviada al Servidor WebLlogic. Por ejemplo, si la URL http://myWeb.server.com/weblogic/foo se pasa al plug-in para dividirla y PathTrim ha sido seleccionado para dividir a partir de /weblogic antes de manejar la URL hacia WebLogic Server, la URL re-enviada al Servidor WebLogic es: http://myweblogic.server.com:7001/foo
PathPrepend	null	String que el plug-in añade al principio de la URL original, después de que la haya recortado PathTrim, y antes de re-enviarla al Servidor WebLogic.
ConnectTimeoutSecs	10	Tiempo máximo en segundos que el plug-in debería intentar conectar con el servidor WebLogic. Debemos hacer es valor mayor que ConnectRetrySecs. Si ConnectTimeoutSecs expira sin una conexión exitosa, incluso después de los intentos apropiados (ver ConnectRetrySecs), se enviará al cliente una respuesta "HTTP 503/Service Unavailable". Podemos personalizar la respuesta de error usando el parámetro ErrorPage.
ConnectRetrySecs	2	Intervalo, en segundos que el plug-in debería esperar entre intentos de conexión con el Servidor WebLogic (o todos los servidores en el cluster). Debemos hacer este número menor que ConnectTimeoutSecs. El número de veces que el plug-in intenta conectar antes de devolver al cliente una respuesta "HTTP 503/Service Unavailable" se calcula dividiendo ConnectTimeoutSecs entre ConnectRetrySecs. Para no especificar entradas, seleccionamos ConnectRetrySecs igual a ConnectTimeoutSecs. Sin embargo, el plug-in intenta conectarse al menos dos veces. Podemos personalizar la respuesta de error usando el parámetro ErrorPage.
Debug	OFF	Seleccionamos el tipo de log realizado para operaciones de depurado. No es recomendable activar estas opciones de depuración en sistemas de producción. La información de depuración se escribe en el fichero **/tmp/wlproxy.log** en sistemas UNIX y en **c:\tmp\wlproxy.log** sobre sistemas Windows NT/2000. Podemos seleccionar cualquiera de las siguientes opciones de log (las opciones HFC,HTW,HFW, y HTC podrían seleccionarse en combinaiones introduciéndolas separadas por comas, por ejemplo “HFC,HTW”):

- **ON**

El plug-in guarda solo mensajes informativos y de error.

- **OFF**

No se guarda información de depuración.

- **HFC**

El plug-in guarda cabeceras del cliente, y mensajes informativos y de error.

- **HTW**

El plug-in guarda cabeceras enviadas al servidor WebLogic, mensajes informativos y mensajes de error.

- **HFW**

El plug-in guarda cabeceras enviadas desde el servidor WebLogic, mensajes informativos y mensajes de error.

- **HTC**

El plug-in guarda cabeceras enviadas al cliente, mensajes informativos y mensajes de error.

- **ALL**

El plug-in guarda cabeceras enviadas al y desde el cliente, cabeceras enviadas al y desde el servidor WebLogic, mensajes informativos y mensajes de error.

DebugConfigInfo	OFF	Activa el parámetro especial de consulta "__WebLogicBridgeConfig". Usado para obtener detalles sobre los parámetros de configuración desde el plug-in. Por ejemplo, si activamos "__WebLogicBridgeConfig" seleccionando DebugConfigInfo y luego enviamos una solicitud que incluye el string de consulta ?__WebLogicBridgeConfig, el plug-in obtiene la información de configuración es estadísticas en tiempo de ejecución y devuelve la información al navegador. El plug-in no conecta con el Servidor WebLogic en este caso. Este parámetro es estrictamente para depuración y el formato de los mensajes de salidas puede cambiar con las versiones. Por razones de seguridad, mantenemos este parámetro desactivado en sistemas de producción.
StatPath (No disponible en el Plug-in de Microsoft Internet Information Server)	false	Si se selecciona a true, el plug-in chequea la existencia y los permisos de path traducido ("Proxy-Path-Translated") de la solicitud antes de enviarla al Servidor WebLogic. Si el fichero no existe, se devuelve al cliente una respuesta "HTTP 404 File Not Found". Si el fichero existe pero no es legible por todo el mundo, se le devuelve al cliente una respuesta "HTTP 403/Forbidden". En cualquier caso, el mecanismo por defecto para que el Servidor Web maneje estas respuestas rellena el cuerpo de la respuesta. Esta opción es útil si tanto la aplicación Web del Servidor WebLogic como el servidor Web tiene el mismo documento raíz. Podemos personalizar la respuesta de error usando el parámetro ErrorPage.
ErrorPage	ninguno	Podemos crear nuestra propia página de error que se muestra cuando nuestro servidor Web no peude re-enviar solicitudes al Servidor WebLogic. Podemos seleccionar este parámetro de dos formas: • Como una URL relativa (nombre de fichero). Dependiendo de cómo configuremos el proxy (por tipo MIME o por path) la página de error podría enviarse al Servidor WebLogic que no está respondiendo. Por esta razón probablemente sea más útil especificar una URL absoluta. • Como una URL absoluta (recomendado). Usando una URL absoluta a la página de error siempre pasará las solicitudes al recurso correcto de nuestro servidor Web u otro Servidor WebLogic. Por ejemplo:

		• http://host:port/myWebApp/ErrorPage.html.
HungServerRecoverSecs	300	Define la cantidad de tiempo que el plug-in espera una respuesta a una petición desde el Servidor WebLogic. El plug-in espera durante HungServerRecoverSecs a que el servidor responda y luego declara el servidor como muerto, y pasa al próximo servidor. El valor debería seleccionarse a un valor muy grande. Si el valor es menor que el tiempo que tardan en procesarse los servlets, podríamos ver resultados inesperados. Valor mínimo: 10
Idempotent	ON	Cuando se selecciona a ON si el servidor no responde en HungServerRecoverSecs, el plug-in pasa al siguiente servidor. Si se selecciona a OFF el plug-no pasa a otro servidor. Si estamos usando el Netscape Enterprise Server Plug-In, o Apache HTTP Server podemos seleccionar este parámetro de forma diferente para diferentes URLs o tipos MIME.
CookieName	JSESSIONID	Si cambiamos el nombre del cookie de sesión del Servidor WebLogic en la Aplicación Web del Servidor WebLogic, necesitamos cambiar el parámetro CookieName en el plug-in al mismo valor. El nombre del cookie de sesión WebLogic se selecciona en el descriptor de despliegue específico de WebLogic, en el elemento <session-descriptor>.
DefaultFileName	ninguno	El plug-in realiza los siguientes pasos: 1. Recorta el path especificado con el parámetro PathTrim. 2. Si la URI es "/" el plug-in le añade el valor de DefaultFileName. 3. Añade el valor especificado con PathPrepend. Este procedimiento evita redirecciones desde el Servidor WebLogic. Seleccionamos DefaultFileName a la página de bienvenida por defecto de la Aplicación Web en el Servidor WebLogic al que se pasan las peticiones. Por ejemplo, si DefaultFileName se selecciona a welcome.html, y PathTrim se selecciona a /weblogic, una solicitud como: "http://somehost/weblogic" se convierte en "http://somehost/welcome.html". Para que este parámetro funcione, se debe especificar el mismo fichero de bienvenida en todas las Aplicaciones Web a las que se les redirigen solicitudes. **Nota:** para el plug-in de Apache usando Stronghold o Raven: No debemos definir el parámetro DefaultFileName en un bloque IFmodule. En su lugar, debemos definirlo en un bloque Location.
MaxPostSize	-1	Máximo tamaño permitido de los datos POST, en bytes. Si la longitud del contenido excede de MaxPostSize, el plug-in devuelve un mensaje se error. Si se selecciona a -1, no se chequea el tamaño de los datos POST. Esto es útil para evitar ataques de denegación-de-servicio que intentan sobrecargar el servidor con datos POST.
MatchExpression (sólo Apache HTTP Server)	ninguno	Cuando se pasa por tipo MIME, seleccionamos el patrón de nombre de fichero dentro de un bloque IfModule usando el parámetro MatchExpression. Ejemplo de cuando pasamos por tipo MIME: <IfModule mod_weblogic.c> MatchExpression *.jsp WebLogicHost=myHost|paramName=value </IfModule> Ejemplo de cuando se pasa por path: <IfModule mod_weblogic.c> MatchExpression /weblogic WebLogicHost=myHost|paramName=value </IfModule>
FileCaching	ON	Cuando se selecciona a ON, y el tamaño de los datos POST de una petición es mayor de 2048 bytes, los datos POST son almacenados en disco en un fichero temporal y reenviados al Servidor WebLogic en trozos de 8192 bytes. Sin embargo, seleccionar FileCaching a ON, puede causar un problema con la barra de progreso mostrada por el navegador que indica el progreso de la descarga. El navegador muestra que la descarga se ha completado incluso aunque el fichero se esté transfiriendo todavía. Cuando se selecciona a OFF y el tamaño de los datos POST es mayor de 2048 bytes, los datos POST son almacenados en memoria y enviados al Servidor WebLogic en trozos de 8192 bytes. Seleccionarla a OFF puede causar problemas si el servidor se apaga durante el procesamiento de la petición porque el plug-in no podrá corregir el fallo.
WlForwardPath (sólo Microsoft Internet Information Server)	null	Si WlForwardPath se selecciona a "/" todas las peticiones serán pasadas. Para reenviar cualquier petición que empiece con un string particular, seleccionamos WlForwardPath a ese string. Por ejemplo, seleccionado WlForwardPath a /weblogic renviará todas las

	peticiones que empiecen con /weblogic al Servidor WebLogic. Este parámetro es necesario si estamos pasando por path. Podemos seleccionar múltiples strings separándolas con comas. Por ejemplo: WlForwardPath=/weblogic,/bea.
MaxSkips (solo Microsoft Internet Information Server) 10	Válido sólo si DynamicServerList se selecciona a OFF. Si falla un Servidor WebLogic listado en el parámetro WebLogicCluster o en la lista dinámica de clusters devuelta por un Servidor WebLogic, ese servidor se marca como "bad" y el plug-in tratan de conectar con el siguiente servidor de la lista: MaxSkips selecciona el número de intentos después de que el plug-in reintente un servidor marcado como “bad”.
DynamicServerList (solo Microsoft Internet Information Server) ON	Cuando se selecciona a OFF, el plug-in ignora la lista dinámica de clusters usada para el balance de carga y sólo usa la lista estática especificada con el parámetro WebLogicCluster. Normalmente este parámetro debería permanecer a ON. Hay algunas implicaciones al seleccionar este parámetro a OFF: • Si fallan uno o más servidores de la lista estática, el plug-in puede gastar mucho tiempo intentando conectar con un servidor muerto, resultando en un decremento del rendimiento, • Si añadimos un nuevo servidor al cluster, el plug-in no podrá pasarle peticiones a menos que redefinamos estos parámetros. WebLogic Server añade automáticamente los nuevos servidores a la lista dinámica de servidores cuando se convierten en parte del cluster.

Parámetros SSL para Plug-Ins de Servidores Web

Nota: *Los parámetros son sensibles a las mayúsculas:*

Parámetro	Valor por Defecto	Descripción
`SecureProxy`	OFF	Seleccionamos este parámetro a ON para activar el uso del protocolo SSL para todas las comunicaciones entre el Plug-In y el Servidor WebLogic. Debemos recordar configurar un puerto en el correspondiente Servidor WebLogic para el protocolo SSL antes de definir este parámetro. Este parámetro podría seleccionarse a dos niveles: en la configuración para el servidor principal y, si hemos definido algún host virtual, en la configuración del host virtual. La configuración para el host virtual hereda la configuración SSL de la configuración del servidor principal si las configuraciones no se sobrescriben en la configuración del host virtual.
`TrustedCAFile`	ninguno	Nombre del fichero que contiene los certificados digitales de las autoridades de certificación creíbles para el plug-in del Servidor WebLogic. Este parámetro es necesario si el parámetro `SecureProxy` se selecciona a ON. El nombre del fichero debe incluir el path de directorios completo al fichero.
`RequireSSLHostMatch`	true	Determina si el nombre del host al que el Plug-in le pasa conexiones debe corresponder con el campo `Subject Distinguished Name` del certificado digital del servidor WebLogic al que se está conectando el plug-in.
`SSLHostMatchOID`	22	El ID del objeto ASN.1 (OID) que identifica qué campo del **Subject Distinguished Name** del certificado digital usar para realizar la comparación de correspondencia. El valor por defecto para este parámetro corresponde con el campo `CommonName` del **Subject Distinguished Name**. Los valores de OID comunes son: • Sur Name—23 • Common Name—22 • Email—13 • Organizational Unit—30 • Organization—29 • Locality—26